

National College of Ireland  
BSc in Computing  
2016/2017

Andrei Ivanov  
X13108824

FindSportClass.ie

Technical Report



# Contents

1	Executive Summary.....	4
1.1	Background.....	4
1.2	Aims .....	4
1.3	Structure .....	5
1.4	Definitions, Acronyms, and Abbreviations .....	5
2	System .....	7
2.1	Technologies .....	7
2.2	Requirements .....	7
2.2.1	Functional requirements.....	7
2.2.2	User requirements .....	7
2.2.3	Use case Diagram .....	8
2.2.4	Requirement 1 <User registration> .....	8
2.2.5	Requirement 2 <Sign in> .....	10
2.2.6	Requirement 3 <create sport class> .....	11
2.2.7	Requirement 5 <Searching for the suitable class>.....	12
2.2.8	Requirement 7 <Send Private Message> .....	13
2.2.9	Requirement 8 <Write feedback> .....	14
2.2.10	Requirement 10 <Delete account>.....	15
2.3	Usability Requirements .....	16
2.3.1	Performance/Response time requirement .....	16
2.3.2	Availability requirement .....	16
2.3.3	Security requirement .....	16
2.3.4	Portability requirement .....	16
2.3.5	Target Site Users.....	16
2.3.6	Scenario .....	17
2.4	System Architecture and Implementation. ....	18
2.4.1	Back end .....	19
2.4.2	API Documentation.....	21
2.5	Front end Angular 2.....	30
2.5.1	Database.....	34
2.6	Deployment.....	36

3	Testing.....	38
3.1	Functional Testing.....	38
3.2	User Testing (UX).....	50
3.2.1	Design & Navigation.....	50
3.2.2	URLs.....	50
3.2.3	404 & HTTP 301.....	51
3.2.4	Sitemap.....	51
3.2.5	Mobile & Tablet.....	51
3.2.6	Heuristic Evaluation.....	52
	Match between system and the real world.....	52
4	GUI.....	54
5	Evaluation.....	59
6	Conclusion.....	60
7	References.....	61
7.1	Tutorials used.....	61
7.2	References.....	61
8	Appendix.....	62
8.1	Project Proposal.....	62
8.1.1	Objectives.....	62
8.1.2	Background.....	62
8.2	Project Plan.....	63
8.3	Technical Details.....	64
8.4	Challenges and Considerations.....	64
8.5	Research and Interviews.....	65
8.5.1	Interviews.....	68
8.6	Monthly Journals.....	71

# **1 Executive Summary**

The main objective of this project is to provide people access to various sport classes online based in Ireland. Trainers/instructors have ability to advertise their regular sport classes and events. People have ability to search for these classes.

## **Introduction**

### ***1.1 Background***

Using search engines such as google and yahoo users can search for sport classes by type and location they require. For example, if I am in Swords, County Dublin and are looking for football class for my 5 years old son I can use google or yahoo search engines and they will find a dozen of sport classes available. Google provide links for different websites such as local community centres' webpages, trainer's outdated websites and so on. Also, I discover that lot of people advertise their classes on the board in the local shops and community centres. Finding sport class with all suitable criteria or even getting a list of available classes in your area is very time consuming. People have to ring or go personally to local community centres, visit all links that google search result provides and then manually filter available and suitable classes for them. "FindaSportClass.ie" web application will bring all trainers and their services into one place, make it easy to search for the end users and get more overall information about sport classes available in their area.

### ***1.2 Aims***

- To create web based application that allow trainers/instructors advertise their services
- To allow consumers to search for sport classes matching their criteria
- To allow consumers and trainers interact with each other directly.
- To ensure that the site is easy to navigate and sign-up process is as straightforward as possible.
- To allow consumer rate, leave and view for feedback

- To ensure that project is completed within specific timeline

### **1.3 Structure**

The first section is the overview and the summary of the project.

The second section details the background of the project, main objectives and technologies used in the project development.

The third section describe functional requirements, user requirements and use case diagram. It includes non-functional requirements and also details architecture of the system.

The fourth section contains the appendices to this document. Appendices include project proposal, interviews, research results and monthly reflective journals.

### **1.4 Definitions, Acronyms, and Abbreviations**

- Consumer

The consumer is an individual who is interested to participate in sport classes. He shall be able to search for sport classes available in the local area using search criteria.

- Provider

The provider is defined as an individual who will be able to advertise their sport classes for consumers

- User

Both consumer and provider

- Functional Requirements

Conditions that must be implemented to ensure that the basic functions of the system are met.

- Non-Functional Requirements

Conditions that doesn't prevent basic functionality, but still need to be implemented.

- GUI

Graphical User Interface – web application view for the user.

- TypeScript

Superset of JavaScript that provides classes and interfaces.

- Angular 2

Angular2 is platform for building responsive mobile and web applications. It much faster in rendering and re-rendering than angularJS. Angular 2 working together with typescript.

- JavaScript

High-level, dynamic programming language that is widely used for web development.

- NodeJS Express

Express is a NodeJS framework that provides all capabilities of NodeJS and allows to setup middleware to respond HTTP Requests.

- MongoDB

MongoDB is scalable, high performance open source NoSQL document type database.

- AWS

Amazon Web Services is a secure cloud services platform that offers computer power and database storage.

## **2 System**

### **2.1 Technologies**

The web application is developed using different web technologies. It is simple, responsive, single-page application (SPA) where the single page dynamically updates as the user interact with the application. The front-end application is developed using HTML, CSS, Bootstrap, JavaScript, Typescript. Angular 2 framework is used for frontend. MongoDB database is used to store and access the key records and information. Web application is deployed on Amazon Web Services (AWS) on EC2 server .

### **2.2 Requirements**

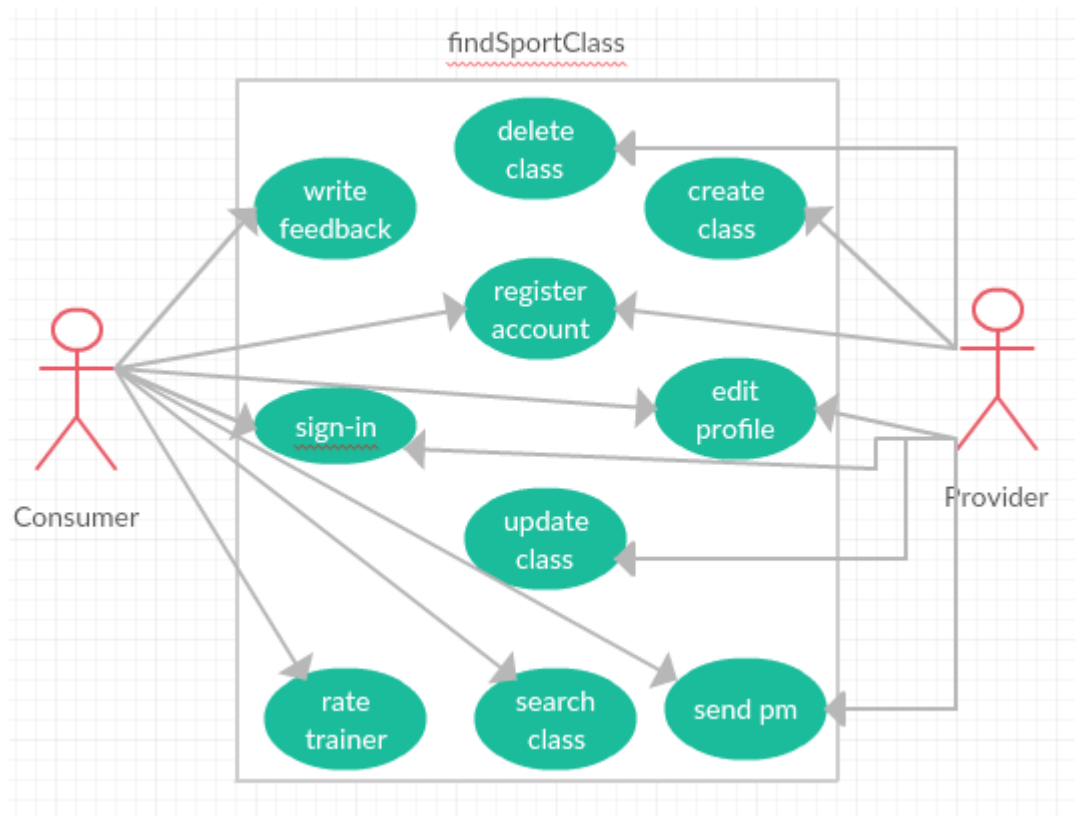
#### **2.2.1 Functional requirements**

- System shall allow user to register account
- System shall be able to grant access to the user after he provides username and password.
- System shall allow user to add, edit and delete sport class
- System shall display search results
- System shall provide private message service
- System shall allow user to write, save and view reviews
- System shall allow user to edit user profiles

#### **2.2.2 User requirements**

- The user shall be able to register an account
- The user shall be able to sign in
- The user shall be able to add, edit, delete their class
- The user shall be able to perform search
- The user shall be able to interact with other users
- The user shall be able to write reviews and rate sport trainers
- The user shall be able to personalize their profile

## 2.2.3 Use case Diagram



## 2.2.4 Requirement 1 <User registration>

### 2.2.4.1 Description & Priority

This requirement refers to an unregistered user who must create a new account on the site in order to get access to the site functionality such as sending private messages, booking events etc. After registration user get all privileges of “Registered user”

### 2.2.4.2 Use Case <Register account>

#### Description/Scope

The Register account use case allows the user to create a login and become a registered user

#### Actor

Consumer, Provider

#### Pre-Condition

User has no valid account, but can browse to the site



Activation	User clicks the registration button
<b>Flow Description</b>	
Main Flow	<ol style="list-style-type: none"> <li>1. The system prompts the user for registration information</li> <li>2. The user fills requested fields</li> <li>3. User click submit button</li> <li>4. System verifies information</li> <li>5. System creates account</li> </ol>
<b>Alternate Flow</b>	
Fields not completed	<ol style="list-style-type: none"> <li>1. Starts from Main Flow 3</li> <li>2. System displays information with appropriate message to complete the fields.</li> <li>3. System prompts user re-enter information</li> <li>4. User re-enters the required fields</li> <li>5. System validates user details</li> <li>6. System creates new account</li> </ol>
Username already	<ol style="list-style-type: none"> <li>1. Starts from Main flow 3</li> <li>2. System displays information that user name already exists</li> <li>3. System clear all the fields.</li> <li>4. System prompts user re-enter information</li> <li>5. User re-enters the required fields</li> <li>6. System validates user details</li> <li>7. System creates new account</li> </ol>
Exception Flow	User press "cancel" button at any time
Termination	<p>Success: Registration completed successfully</p> <p>Exception: User fails to register</p>
Post Condition	User redirected to the Home page

## 2.2.5 Requirement 2 <Sign in>

### 2.2.5.1 Description & Priority

This requirement relates to registered user who wants to sign in to his account. Signing in to the account gives users possibility to book events, send private messages, advertise classes etc.

### 2.2.5.2 Use Case <Sign in account>

Description/Scope	Allows a user to identify themselves on the site and access the required functionality.
Actor	Consumer, Provider
Pre-Condition	User has valid account, but has not yet logged in onto the system.
Activation	The use case starts when user browse to the site Home page
Flow Description	
Main Flow	<ol style="list-style-type: none"><li>1. System prompts user for username and password</li><li>2. User enters the required credentials</li><li>3. User clicks 'sign in' button</li><li>4. System validates user details</li><li>5. Users is signed in</li></ol>
Alternate Flow	
Invalid credentials	<ol style="list-style-type: none"><li>1. Starts from Main Flow 3</li><li>2. System displays information "username or password" is invalid.</li><li>3. System prompts user re-enter information</li><li>4. User re-enters the required fields</li><li>5. System validates user details</li><li>6. User is signed in</li></ol>
Exceptional Flow	User press "cancel" button at any time.
Termination	Success: User is signed in

Post Condition	Exception: User is not signed in User redirected to the Home page
----------------	--

## 2.2.6 Requirement 3 <create sport class>

### 2.2.6.1 Description & Priority

This requirement relates to the provider who wants advertise his sport class. Advertising sport class gives consumer opportunity to be found his by consumer via search.

### 2.2.6.2 Use case <create class>

Description/Scope	
A provider adds a sport class to the system.	
Actor	Provider
Pre-Condition	User is signed in to the system.
Activation	The use case starts when user clicks 'add class'
Flow Description	
Main Flow	<ol style="list-style-type: none"> <li>1. The system prompts for "new sport class" information</li> <li>2. The user fills appropriate fields</li> <li>3. User clicks add button</li> <li>4. System validates all details</li> <li>5. System creates new ad</li> </ol>
Alternate Flow	
Fields not completed	<ol style="list-style-type: none"> <li>1. Starts from Main Flow 3</li> <li>2. System displays information with appropriate message to complete the fields.</li> <li>3. User completes appropriate fields</li> <li>4. System validates details</li> <li>5. System creates add</li> </ol>

Exceptional Flow	User press “cancel” button at any time.
Termination	Main Flow: New sport class is added Exception: Fails to add new class
Post Condition	User redirected to the Home page

## 2.2.7 Requirement 5 <Searching for the suitable class>

### 2.2.7.1 Description & Priority

This requirement relates to the consumer who wants to perform search for suitable classes available.

### 2.2.7.2 Use case <Search Class>

Description/Scope	
The user perform search using search criteria	
Actor	Consumer
Pre-Condition	User is signed in to the system
Activation	The use case starts when user navigates to the search form
<b>Flow Description</b>	
Main Flow	<ol style="list-style-type: none"> <li>1. System prompts user for search criteria</li> <li>2. User choose search criteria and add the information to the appropriate fields</li> <li>3. User press “search” button</li> <li>4. System validates the search criteria</li> <li>5. System output the list of classes available according to user search criteria</li> </ol>
<b>Alternate Flow</b>	

Searching without search criteria	<ol style="list-style-type: none"> <li>1. System prompts user for search criteria</li> <li>2. User press “search” button</li> <li>3. System output the list of all sport classes</li> </ol>
-----------------------------------	---

Termination	Success: system outputs list of sport classes
Post Condition	User redirected to the search result

## 2.2.8 Requirement 7 <Send Private Message>

### 2.2.8.1 Description & Priority

This requirement relates to both consumer and provider. It is very important that users can interact with each other.

### 2.2.8.2 Use case <Send Private Message>

#### Description/Scope

User send private message to other user.

Actor	Consumer, Provider
Pre-Condition	User is signed in to the system
Activation	User press “Send PM” button
<b>Flow Description</b>	
Main Flow	<ol style="list-style-type: none"> <li>1. User presses “send pm” button</li> <li>2. System prompts user to fill appropriate fields</li> <li>3. User fill the fields</li> <li>4. System prompts user for receiver information</li> <li>5. User fills receiver information</li> <li>6. User click “send” button</li> <li>7. System validates information</li> <li>8. System confirms that message is sent</li> </ol>
Alternate Flow	<ol style="list-style-type: none"> <li>1. User navigates to receiver account</li> </ol>

	<ol style="list-style-type: none"> <li>2. User presses “send pm” button</li> <li>3. System prompts user to fill appropriate fields</li> <li>4. User fill the fields</li> <li>5. System validates information</li> <li>6. System confirms that message is sent</li> </ol>
Exceptional Flow	User can press cancel at any time
Termination	Success: PM is sent Exception: Fails to send pm
Post Condition	User is redirected to previous page

## 2.2.9 Requirement 8 <Write feedback>

### 2.2.9.1 Description & Priority

This requirement relates to consumer who want to write review about sport classes he have been participated.

### 2.2.9.2 Use case <write review>

Description/Scope	
User writes review about sport class he has been participated	
Actor	Consumer
Pre-Condition	User is signed in
Activation	User press “review” icon
Flow Description	
Main Flow	<ol style="list-style-type: none"> <li>1. User navigates to consumer account</li> <li>2. User press “write review” icon</li> <li>3. System prompts user for appropriate information</li> <li>4. User fills all fields</li> <li>5. User press “ok” button</li> <li>6. System validates information</li> <li>7. System confirms with appropriate message</li> </ol>

Exception Flow	User can press “cancel/back” button
Termination	Success: Review is posted successfully Exception: Fails to post review
Post Condition	User is redirected to the home page

## 2.2.10 Requirement 10 <Delete account>

### 2.2.10.1 Description & Priority

This requirement relates to both consumer and provider who doesn't want to hold account in findasportclass.ie

### 2.2.10.2 Use case <Delete Account>

Description/Scope	
User permanently deletes his account from the system	
Actor	Consumer, Provider
Pre-Condition	User is signed in to the system.
Activation	Use case starts when user presses “Delete Account” button
Flow Description	
Main Flow	<ol style="list-style-type: none"> <li>1. User press “Delete Account”</li> <li>2. System pop-up dialog box with confirm/cancel button</li> <li>3. User clicks confirm button</li> </ol>
Exceptional Flow	<ol style="list-style-type: none"> <li>1. User press “Delete Account”</li> <li>2. System pop-up dialog box with confirm/cancel button</li> <li>3. User clicks cancel button</li> </ol>
Termination	Success: Registered account deleted

Post Condition

Exception: User fails to delete account

User is redirected to home page

## **2.3 Usability Requirements**

### **2.3.1 Performance/Response time requirement**

The web application must be responsive and react instantaneously without any delays. It is important that web application hosted on the correct hardware and software to avoid significant lags in response. FindaSportClass.ie will be hosted on AWS.

### **2.3.2 Availability requirement**

It is important for the web application that it is highly available for the users in multiple location. The application shall be available 24/7 basis.

### **2.3.3 Security requirement**

As the application provides access to the user personal information such as names, addresses, emails etc. there is a requirement that this information is protected from external threats.

### **2.3.4 Portability requirement**

The web application an access across different web browsers and devices.

### **2.3.5 Target Site Users**

Trainers





If you run sport classes, finding customers is time consuming process. Findasportclass provides platform for people who wants to advertise their sport classes.

### Customers

FindaSportclass provides platform for people who wants to participate in a sport class. User can perform quick search depending on their needs (location, price age).

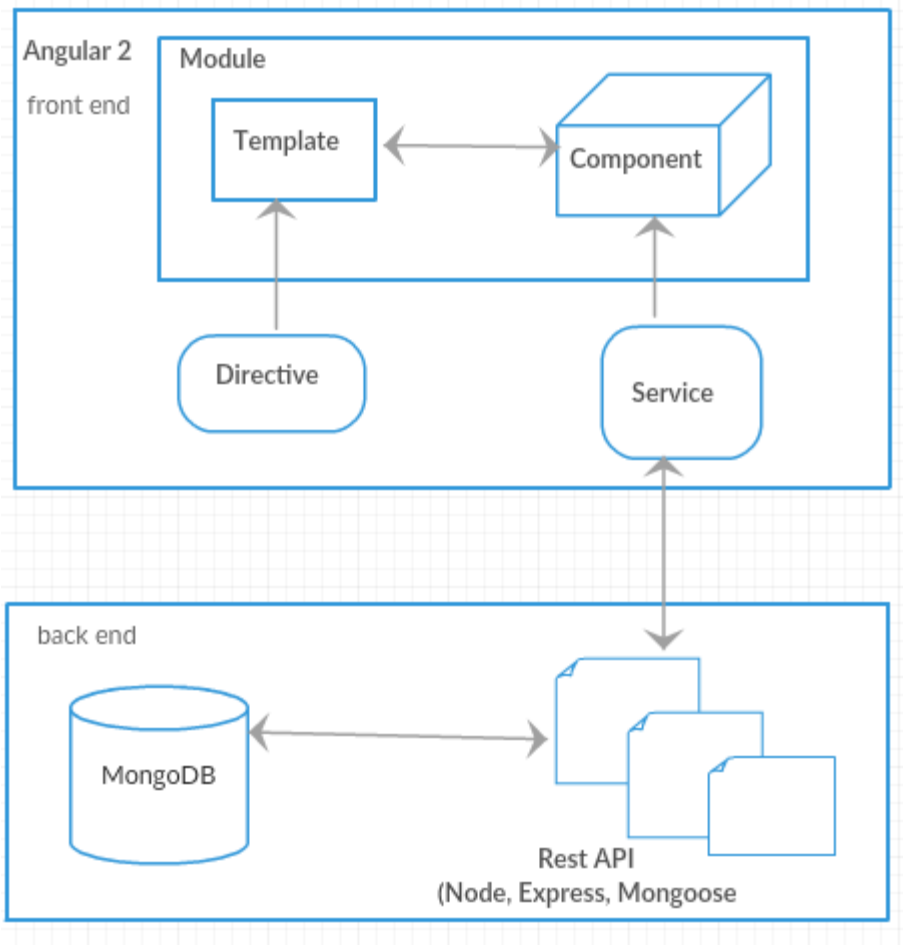
### 2.3.6 Scenario

Project Manager											
	<table><tr><td>Name</td><td>Kevin</td></tr><tr><td>Age</td><td>35</td></tr><tr><td>Location</td><td>Blackrock, Co Dublin</td></tr><tr><td>Status</td><td>Married</td></tr><tr><td>About</td><td>Kevin is working as project manager. He is living in Ireland, Blackrock, co Dublin. He has wife and 10 years old son.</td></tr></table>	Name	Kevin	Age	35	Location	Blackrock, Co Dublin	Status	Married	About	Kevin is working as project manager. He is living in Ireland, Blackrock, co Dublin. He has wife and 10 years old son.
Name	Kevin										
Age	35										
Location	Blackrock, Co Dublin										
Status	Married										
About	Kevin is working as project manager. He is living in Ireland, Blackrock, co Dublin. He has wife and 10 years old son.										
Scenario	Kevin got a new job in Swords County Dublin. He moved to the Swords, close to the job. His son was participating taekwondo classes in Blackrock. Now, Kevin wants to find similar Taekwondo classes in Swords.										
Task	Find taekwondo classes for 10 years old boy in Swords, County Dublin.										
Solution	Findasportclass.ie can provide list of sportclasses available in Swords by using search menu and providing location name and age.										

Football Trainer		
	Name	John Whale
	Age	46
	Status	Divorced
	About	John is working as real estate agent, but his part-time job is football trainer. He is running football classes on Saturday and Sunday.
Scenario	John wants to advertise his football classes to attract more kids. He already done it local community center, but he feels that not so much people are aware. John doesn't have resources to develop website and then maintain it as this is more hobby for him.	
Task	Advertise football class online	
Solution	John head to findasportclass site. He register on the site and list his football class with the description, age, price and times available. Customer can now see his advertisement online.	

## ***2.4 System Architecture and Implementation.***

This section gives a brief overview of how the different application components operate. The system is built using range of different technologies. The back-end of the system is developed using NodeJS Express with RESTful web services. The front-end is developed using Angular 2 platform with typescript and bootstrap libraries.

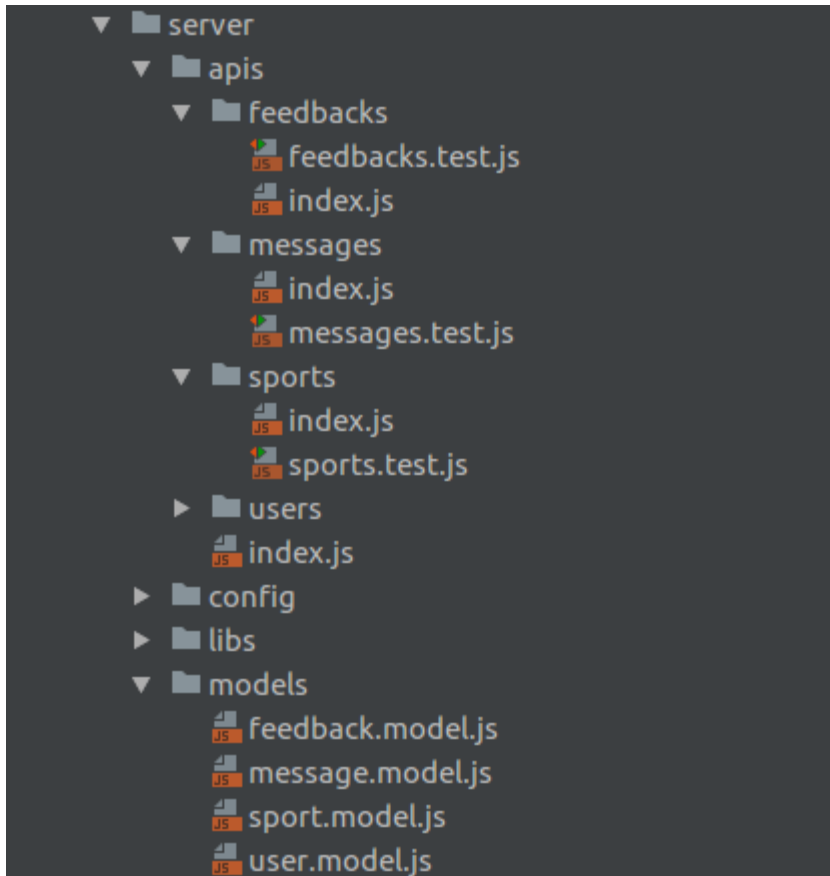


*System architecture*

### **2.4.1 Back end**

Backend is developed in NodeJS using express framework. It is fully restful.

In project, all backend is in directory called "server". For each model – feedback, messages, sports and users – is separate directory.



“Config” directory includes configuration for server, express framework, mongoDB, testing and development environment.

Each model has index.js file, where HTTP GET, PUT, POST, DELETE and error handling are implemented.

Example of GET:

```
feedbackDetail(req, res){

    let feedbackId = req.params.feedbackId;

    let promise = global.MongoORM.Feedback.findById(feedbackId)
        .populate('userId',[ 'name','email','address','location','gender','profilePic'])
        .populate('trainerId',[ 'name','email','address','location','gender','profilePic']);
    promise
        .then(function(feedback){
            res.sendResponse(feedback);
        })
        .catch(function(error){
            let errors = [];
            if(error.name == 'ValidationError'){
                Object.keys(error.errors).forEach(function(field){
                    let eObj = error.errors[field].properties;
```

```

        if(eObj.hasOwnProperty("message"))
            errors.push(eObj["message"]);
    });
} else if(error_name == 'MongoError'){
    errors.push(error);
} else
    errors.push('Internal server error. ');
res.sendError(errors);
});
}

```

## 2.4.2 API Documentation

### 1. Feedback

#### POST Add Feedback

*Sample request:*

```

curl --request POST \
  --url http://localhost:8080/api/feedback \
  --header 'content-type: application/json' \
  --data '{
    "content": "Nice trainer",
    "rating": 5,
    "trainerId": "58caa3238664432062aea426",
    "userId": "58ca54b437a8ac1437d1cd5f"
  }

```

#### GET List Feedbacks

*Sample request:*

```

curl --request GET \
  --url http://localhost:8080/api/feedback \
  --header 'content-type: application/json' \
  --header 'token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUz11NiJ9.eyJ1c2VybmFtZSI6ImltYXhNobmFslwiczGFzc

```

#### GET Feedback Detail

*Sample request:*

```
curl --request GET \  
  --url http://localhost:8080/api/feedback/58cacb26daabd62824e2f89c \  
  --header 'content-type: application/json' \  
  --header 'token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyTmFtZSI6ImtyaXNobmFslwiicGFzc3dvcmlhZG1pbilslmIhdCI6MTQ4OTUxNTQwNiwiZmVudC5ndG5NjAxODA2fQ'
```

## PUT Update Feedback

*Sample request:*

```
curl --request PUT \  
  --url http://localhost:8080/api/feedback/58cacb26daabd62824e2f89c \  
  --header 'content-type: application/json' \  
  --header 'token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyTmFtZSI6ImtyaXNobmFslwiicGFzc3dvcmlhZG1pbilslmIhdCI6MTQ4OTUxNTQwNiwiZmVudC5ndG5NjAxODA2fQ.e4bL9hX9MSmBS6hutLKejuRIY25dxWtnJ5zAoDvYWTs' \  
  --data '{  
    "content": "Very nice trainer"  
  }'
```

## DELETE Feedback

*Sample request:*

```
curl --request DELETE \  
  --url http://localhost:8080/api/feedback/58cacb26daabd62824e2f89c \  
  --header 'content-type: application/json' \  
  --header 'x-access-token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyTmFtZSI6ImtyaXNobmFslwiicGFzc3dvcmlhZG1pbilslmIhdCI6MTQ4OTUxNTQwNiwiZmVudC5ndG5NjAxODA2fQ'
```

## 2. Messages

### POST Send Message

*Sample request:*

```
curl --request POST \  
  --url http://localhost:8080/api/messages/58cacb26daabd62824e2f89c \  
  --header 'content-type: application/json' \  
  --header 'x-access-token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyTmFtZSI6ImtyaXNobmFslwiicGFzc3dvcmlhZG1pbilslmIhdCI6MTQ4OTUxNTQwNiwiZmVudC5ndG5NjAxODA2fQ'
```

```

--url http://localhost:8080/api/messages \
--header 'content-type: application/json' \
--header 'token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUz11NiJ9.eyJ1c2VyTmFtZSI6ImtyaXNobmFslwiGFzc3dvcmQiOiJhZG1pbilslmlhdCI6MTQ4OTUxNTQwNiwiZ XhwljoxNDg5NjAxODA2fQ.e4bL9hX9MSmBS6hutLKejuRIY25dxWtnJ5zAoDvY WTs' \
--data '{
    "title":"Test Message",
    "description":"how are you?",
    "senderId":"58caa3238664432062aea426",
    "recipientId":"58ca54b437a8ac1437d1cd5f"
}'

```

## GET List Messages

*Sample request:*

```

curl --request GET \
--url http://localhost:8080/api/messages \
--header 'content-type: application/json' \
--header 'token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUz11NiJ9.eyJ1c2VyTmFtZSI6ImtyaXNobmFslwiGFzc3dvcmQiOiJhZG1pbilslmlhdCI6MTQ4OTUxNTQwNiwiZ XhwljoxNDg5NjAxODA2fQ.e4bL9hX9MSmBS6hutLKejuRIY25dxWtnJ5zAoDvY WTs'

```

## GET Message Detail

*Sample request:*

```

curl --request GET \
--url http://localhost:8080/api/messages/58caa51431613f2133ce06bb \
--header 'content-type: application/json' \
--header 'token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUz11NiJ9.eyJ1c2VyTmFtZSI6ImtyaXNobmFslwiGFzc3dvcmQiOiJhZG1pbilslmlhdCI6MTQ4OTUxNTQwNiwiZ XhwljoxNDg5NjAxODA2fQ.e4bL9hX9MSmBS6hutLKejuRIY25dxWtnJ5zAoDvY WTs'

```

## DELETE Message Detail

*Sample request:*

```
curl --request DELETE \  
  --url http://localhost:8080/api/messages/58caa51431613f2133ce06bb \  
  --header 'content-type: application/json' \  
  --header 'token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyTmFtZSI6ImtyaXNobmFslwiicGFzc3dvcnQiOiJhZG1pbilslmlhdCI6MTQ4OTUxNTQwNiwiZm9udG5NjAxODA2fQ.e4bL9hX9MSmBS6hutLKejuRIY25dxWtnJ5zAoDvYWTs'
```

## 3. Sports

### POST Add Sports

*Sample request:*

```
curl --request POST \  
  --url http://localhost:8080/api/sports \  
  --header 'content-type: application/json' \  
  --header 'token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyTmFtZSI6ImtyaXNobmFslwiicGFzc3dvcnQiOiJhZG1pbilslmlhdCI6MTQ4OTUxNTQwNiwiZm9udG5NjAxODA2fQ.e4bL9hX9MSmBS6hutLKejuRIY25dxWtnJ5zAoDvYWTs' \  
  --data '{  
    "gender": "male",  
    "location": [150.644, -34.397],  
    "name": "Baseball",  
    "description": "baseball description",  
    "ownerId": "58caa3238664432062aea426",  
    "age": 18,  
    "price": 100,  
    "tags": ["baseball"],  
    "promptPicture": "mytest.jpg",  
    "address": {
```



```
"address1": "address street 1",
"address2": "address street 2",
"city": "Wales",
"state": "New South Wales",
"zipcode": "1234",
"country": "Australia",
"phone": "+412543322222"
}
}'
```

## GET List Sports

*Sample request:*

```
curl --request GET \
--url http://localhost:8080/api/sports \
--header 'content-type: application/json' \
--header 'token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyTmFtZSI6IjYyOTUxNTQwNiwiZmxhZjoxNDg5NjAxODA2fQ.e4bL9hX9MSmBS6hutLKejuRIY25dxWtnJ5zAoDvYWTs'
```

## GET Sports Detail

*Sample request:*

```
curl --request GET \
--url http://localhost:8080/api/sports/58caaa7889ac37226fb9e4d7 \
--header 'content-type: application/json' \
--header 'token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyTmFtZSI6IjYyOTUxNTQwNiwiZmxhZjoxNDg5NjAxODA2fQ.e4bL9hX9MSmBS6hutLKejuRIY25dxWtnJ5zAoDvYWTs'
```

## PUT Update Sport's Address

*Sample request:*

```

curl --request PUT \
  --url http://localhost:8080/api/sports/58caa47f6add8120dcc852f8/address \
  --header 'content-type: application/json' \
  --header 'token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyTmFtZSI6ImtyaXNobmFslwiwGFzc3dvcmlhZG1pbilslmlhdCI6MTQ4OTUxNTQwNiwiZmlybXN0dG5NjAxODA2fQ.e4bL9hX9MSmBS6hutLKejuRIY25dxWtnJ5zAoDvYWTs' \
  --data '{
    "address1": "street 1",
    "zipcode": "23456"
  }'

```

## PUT Update Sport

*Sample request:*

```

curl --request PUT \
  --url http://localhost:8080/api/sports/58caaa7889ac37226fb9e4d7 \
  --header 'content-type: application/json' \
  --header 'token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyTmFtZSI6ImtyaXNobmFslwiwGFzc3dvcmlhZG1pbilslmlhdCI6MTQ4OTUxNTQwNiwiZmlybXN0dG5NjAxODA2fQ.e4bL9hX9MSmBS6hutLKejuRIY25dxWtnJ5zAoDvYWTs' \
  --data '{
    "name": "Base Ball"
  }'

```

## DELETE Sports Detail

*Sample request:*

```

curl --request DELETE \
  --url http://localhost:8080/api/sports/58caaa7889ac37226fb9e4d7 \
  --header 'content-type: application/json' \
  --header 'token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyTmFtZSI6ImtyaXNobmFslwiwGFzc3dvcmlhZG1pbilslmlhdCI6MTQ4OTUxNTQwNiwiZmlybXN0dG5NjAxODA2fQ.e4bL9hX9MSmBS6hutLKejuRIY25dxWtnJ5zAoDvYWTs'

```

## 4. Users

### POST Signup User

*Sample request:*

```
curl --request POST \  
  --url http://localhost:8080/api/user \  
  --header 'content-type: application/json' \  
  --data '{  
    "email":"Andrei@gmail.com",  
    "name":"Andrei Ivy",  
    "password":"admin",  
    "gender":"male",  
    "location":[150.644,-34.397],  
    "type":"customer",  
    "profilePic":"mytest.png",  
    "address": {  
      "address1":"address street 1",  
      "address2":"address street 2",  
      "city":"Wales",  
      "state":"New South Wales",  
      "zipcode":"1234",  
      "country":"Australia",  
      "phone":"+412543322222"  
    }  
  }'
```

### GET List Users

*Sample request:*

```
curl --request GET \  
  --url http://localhost:8080/api/user \  
  --header 'content-type: application/json' \  
  --data ''
```

```
--header 'x-access-token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6ImtyaXNobmFslwiicGFzc3dvcmlhZG1pbilzImhhdCI6MTQ4OTUxNTQwNiwiZXhwIjoxNDg5NjAxODA2fQ.e4bL9hX9MSmBS6hutLKejuRIY25dxWtnJ5zAoDvYWTs'
```

## PUT Update User Profile

*Sample request:*

```
curl --request PUT \  
  --url http://localhost:8080/api/user/58c97e18497a771059e999fb \  
  --header 'content-type: application/json' \  
  --header 'user-agent: Mozilla/5.0 (Linux; U; Android 4.0.3; en-in; SonyEricssonMT11i' \  
  --data '{  
    "name": "Andrei",  
    "gender": "Male"  
  }'
```

## GET User Detail

*Sample request:*

```
curl --request GET \  
  --url http://localhost:8080/api/user \  
  --header 'content-type: application/json' \  
  --header 'x-access-token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6ImtyaXNobmFslwiicGFzc3dvcmlhZG1pbilzImhhdCI6MTQ4OTUxNTQwNiwiZXhwIjoxNDg5NjAxODA2fQ.e4bL9hX9MSmBS6hutLKejuRIY25dxWtnJ5zAoDvYWTs'
```

## PUT Update User Address

*Sample request:*

```
curl --request PUT \  
  --url http://localhost:8080/api/user/58ca54b437a8ac1437d1cd5f/address \  
  --header 'content-type: application/json' \  
  --header 'user-agent: Mozilla/5.0 (Linux; U; Android 4.0.3; en-in; SonyEricssonMT11i' \  
  --data '{
```

```
"address1":"street 1",  
"zipcode":"23456"  
}'
```

## POST User Login

*Sample request:*

```
curl --request POST \  
  --url http://localhost:8080/api/login \  
  --header 'content-type: application/json' \  
  --header 'user-agent: Mozilla/5.0 (Linux; U; Android 4.0.3; en-in; SonyEricssonM  
T11i' \  
  --data '{  
    "email":"andrei@gmail.com",  
    "password":"admin"  
  }'
```

## DELETE User Detail

*Sample request:*

```
curl --request DELETE \  
  --url http://localhost:8080/api/user \  
  --header 'content-type: application/json' \  
  --header 'x-access-token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2V  
yTmFtZSI6ImtyaXNobmFslwiicGFzc3dvcmQiOiJhZG1pbilzImIhdCI6MTQ4OTUx  
NTQwNiwiZXhwIjoxNDg5NjAxODA2fQ.e4bL9hX9MSmBS6hutLKejuRIY25dxWtn  
J5zAoDvYWTs'
```

## POST Check Login

*Sample request:*

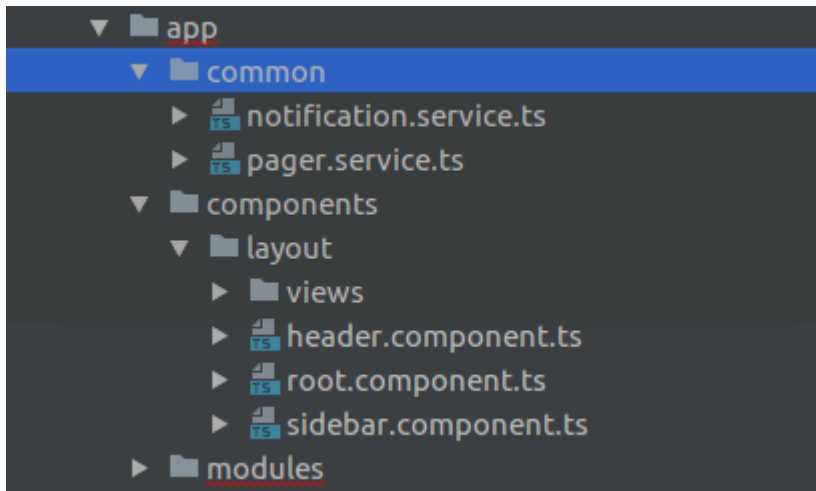
```
curl --request POST \  
  --url http://localhost:8080/api/checkLogin \  
  --header 'content-type: application/json' \  
  --data '{
```

```
"token": "e1yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI1OGNhYTMyMzg2NjQ0MzlwNjJhZWE0MjYiLCJ1cGRhdGVkQXQiOiIyMDE3LTAzLTE2VDE0OjM3OjZlY4OVoiLCJjb250b3R5ZXRzbnZlbn7ImFkZHI3MxljoiYWRkcmVzcyBzdHJIZXQgMSIsImFkZHI3MxljoiYWRkcmVzcyBzdHJIZXQgMmIsImNpdHkiOiJXYWxlcylsIn0iYXRlljoiTmV3IFNvdXRoIFdhbGVzliwiemlwY29kZSI6IjEjEYmzQiLCJjb3VudHJ5IjoiQXVzdHJhbGhliwicGhvbmUiOiIiNDEyNTQzMzlyMjlyliwiX2lkIjoiNThjYWZMjM4NDMyMDYyYyYwVhNDI3In0sInJlZ2lzdHJhdGlvdGkiOiIyMDE3LTAzLTE2VDE0OjM3OjZlY4MFoiLCJwcm9maWxlUGljIjoibXl0ZXN0LnBuZylsInR5cGUoiOiJ0cmFpbmVylwiZ2VuZGVyIjoibWFsZSI6ImVtYWVzIjoia3Jpc2huYWwuanFkYXZAZ21haWwuanY29tliwibmFtZSI6IktyaXNobmFslEphZGF2liwiX192IjowLCJpc0FjdGl2ZSI6dHJ1ZSwibG9jYXRpb24iOiI0IiwiaWF0IjoiMj01OTQ0MTc2NTEsImV4cCI6MTQ5MDYwNDA1MX0.HOGMAqmvrq-DWjbgImaPQI8nV9X14CmNcyQGifHI9s"
```

```
}
```

## 2.5 Front end Angular 2

In project, all the front end is in directory called "app".

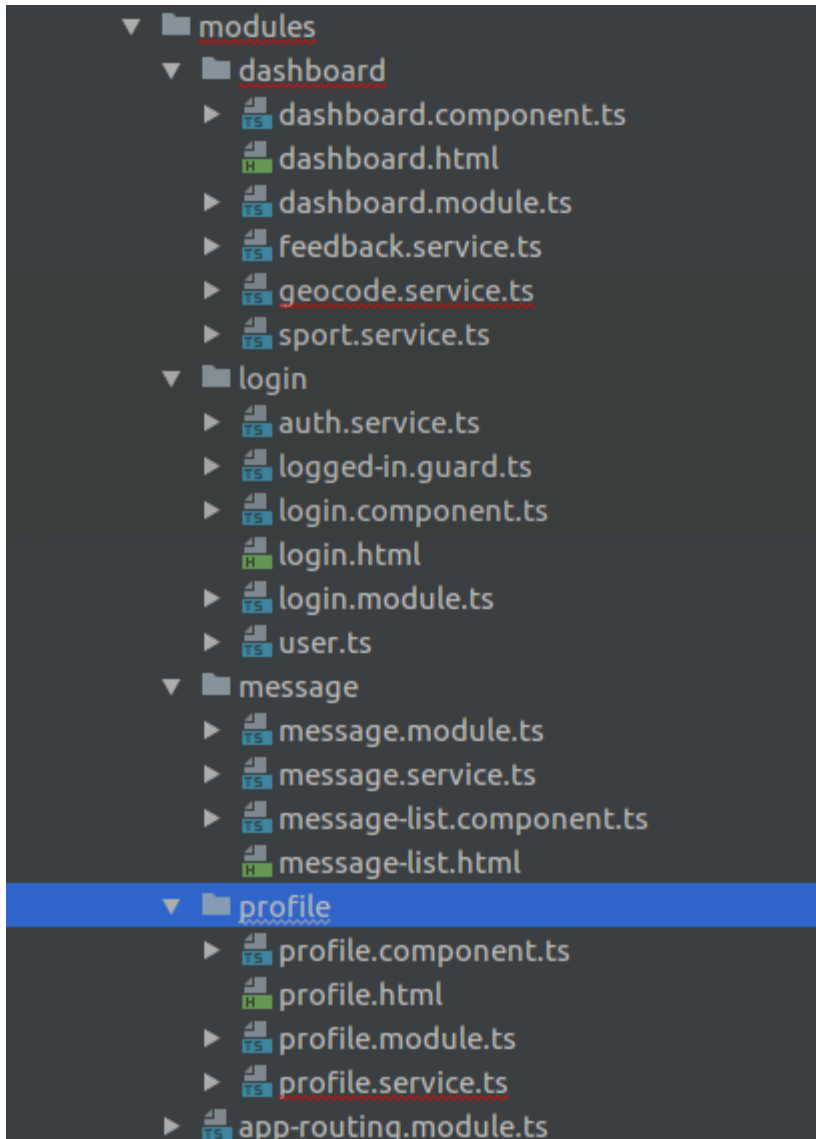


Common folder includes notification service and pager (pagination) services. These are used in different modules, such as feedback, sport and message modules, to avoid code duplication there are placed into separated directory from where they are injected into other modules.

Component folder consist of main component – root component – and header component (navbar). View folder consist html pages, where also sits main view page – app.html – that renders all the .html pages.

## **Modules**

All Angular 2 applications have modular structure, findasportclass is also broken down into modules. It has dashboard, login, message, profile and root module - app module. Modules contains components, providers (services) and pipes, html and CSS. Angular 2 provides built in services (like libraries) that can be importer and injected into the components. This makes developing application faster.



Example of module class (profile module):

```
@NgModule({
  declarations: [ProfileComponent],
  imports: [BrowserModule,
    CommonModule,
    ReactiveFormsModule,
    RouterModule,
    ModalModule.forRoot(),
    FlexLayoutModule,
    DropzoneModule.forRoot(DROPZONE_CONFIG)
  ],
  exports: [ProfileComponent],
  providers: [ProfileService]
})
export class ProfileModule {
  static ROUTES: any = routes;
}
```



As we can see “profile module” include component (profile.component.ts) html page (profile.html) and module(profile.module.ts) file. Modules are like separate units – small applications - that can be imported, not only in the same application, but across different Angular 2 applications.

## Component

Component contains data members and functions. It's like controller in MVC architecture that has application logic and controls a patch of the page called a template (.html). It interacts with the template through functions, bindings and API.

## Service

To reuse code over and over in angular2 services are implemented and then injected into components. Services allows us to create specific functionality and then reuse them in components. We can use Angular 2 native http module that exposes HTTP service to access web services over http.

Example of profile service with the update profile functionality:

```
@Injectable()
export class ProfileService {
  private headers = new Headers();
  private profileUrl = "api/user";

  constructor(private http: Http) {
    this.headers.append('Content-Type', 'application/json');
    this.headers.append('token', localStorage.token);
  }

  updateProfile(postArray, profileId): Promise<profile[]> {
    let apiRequestUrl = this.profileUrl + "/" + profileId;
    return this.http
      .put(apiRequestUrl, JSON.stringify(postArray), {headers: this.headers})
      .toPromise()
      .then(res => res as profile[])
      .catch(this.handleError);
  }
}
```

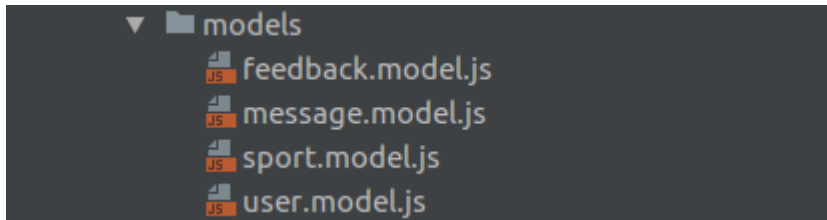
In profile service “updateProfile” function updates (http PUT) profile data by accessing update profile URL (API) and update profile accordingly in server side.

## Authorization and authentication

JWT (Jason web token) library is used to securely communicate JSON payload (object) across HTTP connection. When user sign-in and the password match, he is assigned with the valid token that is used to send request over http. If user does not have a token he can only access to log-in, register and main page of the sport classes.

### 2.5.1 Database

Database is developed in MongoDB using mongoose library. Mongoose is high level and uses MongoDB driver. System have 4 models:



1. User collection stores user details and it has address schema and user schema as following:

```
AddressSchema = new mongoose.Schema({
  address1: {type: String},
  address2: {type: String},
  city: {type: String},
  county: {type: String},
  zipcode: {type: String},
  country: {type: String},
  phone: {type: String}
});

UserSchema = new mongoose.Schema({
  name: { type: String },
  email: { type: String, required: [true,'Email is required'] },
  password: { type: String, required: [true,'Password is required'] },
  birthdate: { type: Date },
  gender: { type: String },
  registrationDate: { type: Date},
  address: AddressSchema,
  location: { type: Array},
  isActive: { type: Boolean, default:true },
  profilePic: { type: String},
```

```
about: { type: String},
type: { type: String}
}
```

2. Sport collection is used to store sport class details. It has address schema and user schema as following:

```
AddressSchema = new mongoose.Schema({
  address1: {type: String},
  address2: {type: String},
  city: {type: String},
  state: {type: String},
  zipcode: {type: String},
  country: {type: String},
  phone: {type: String}
});

SportSchema = new mongoose.Schema({
  name: { type: String, required: [true,'Sport name is required'] },
  description: { type: String, required: [true,'Description is required'] },
  ownerId : { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
  startDate: { type: Date},
  startTime: { type: String},
  address: AddressSchema,
  location: { type: Array},
  prompPicture: { type: String},
  minAge: { type: Number},
  maxAge: { type: Number},
  price: { type: Number, min: 1},
  tags: { type: Array},
  isActive: { type: Boolean, default:true },
  createdDt: { type: Date, default: Date.now},
}
```

3. Feedback collection is used to store feedback details and following schema is used.

```
FeedbackSchema = new mongoose.Schema({
  content: { type: String, required: [true,'Content is required'] },
  trainerId: { type: mongoose.Schema.Types.ObjectId,ref: 'User' },
  rating: {type: Number},
  userId: { type: mongoose.Schema.Types.ObjectId,ref: 'User' },
  createdDt: { type: Date, default: Date.now},
}
```

4. Message collection is used to store PM details.

```
MessageSchema = new mongoose.Schema({
  title: { type: String, required: [true,'Message title is required'] },
  description: { type: String, required: [true,'Description is required'] },
  senderId: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
```

```
recipientId: { type: mongoose.Schema.Types.ObjectId, ref: 'User'  
  createdDt: { type: Date, default: Date.now},  
  },
```

## **2.6 Deployment**

### **Deployment Process**

I used webpack for build process of angular 2 application and Nginx for setting routing for APIs to node.js application and serve angular app statistically.

#### **Webpack:**

Webpack is a JavaScript bundler. It takes in a bunch of assets (i.e. source, images, markup, CSS, JS) and turns that into minified and browser compatible format(For example typescript to JavaScript transpiled source with html and CSS).

#### **Nginx:**

nginx is an open source, lightweight, high-performance web server or proxy server. Nginx used as reverse proxy server for HTTP, HTTPS, SMTP, IMAP, POP3 protocols, on the other hand, it is also used for server's load balancing and HTTP Cache.

#### **How deployment works:**

Used webpack to generate application build. It packs all third-party module in vendor.bundle.js file, CSS in webapp.css file and all angular source in

app.bundle.js file. Index.html will have only these 4 files injected. It will copy images and assets in build folders using webpack copy plugins.

Nginx will serve index.html from build folder which is generated using webpack.

/api upstream block will catch all api calls and proxy it to node.js application running on different port.

### **How to generate build:**

"npm run build" script will execute "webpack --config webpack.config.js " command to generate build as per the config in webpack.config.js file. It will create build folder in project root directory which will be used by nginx to server static content.

## 3 Testing

### 3.1 *Functional Testing*

Mocha, xunit, jenkins-mocha, istanbul, should, and supertest nperms of node.js have been used to **unit test** our api code.

#### **Mocha:**

Mocha is an open source JavaScript test framework running on Node.js and in the browser, making asynchronous testing simple. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases.

#### **xunit:**

xunit is a plugin to integrate test with automation and CI using jenkins. xunit collects test results and prepares xml reports for viewers in jenkins, hudson etc.

#### **jenkins-mocha:**

Jenkins mocha is the wrapper npm to jenkins and mocha which acts like a task runner to execute unit tests and prepare reports according to reporter, In my case I provided xunit.

#### **istanbul:**

Istanbul is a coverage tool that calculates statement, line, function, and branch coverage with module loader hooks to transparently add coverage when running tests. It helps to maintain code coverage and reports non-tested dead codes.

#### **Supertest:**

Supertest is used to test http requests by asserting HTTP to the app. It makes requests and maintains dependency requirements.

## Should :

Should is an expressive and readable assertion library. It keeps clean code and error message helpful. For example to check attribute name with andrei value in the user object, we can check by 'user.should.have.property('name', 'andrei');' it's readable and easy to use.

## Test automation flow:

Npm script, 'npm test' are added in the package.json file. It executes Make command('make test').

Step 1:

Make script set's reporter to xunit so it can produce xml report which can be used prepare unit test document for code coverage and further integrate with jenkins.

Step2:

Set node environment as a test (NODE\_ENV=test) so that API will read the test environment config and it will use test database to create testing data.

Step 3:

Initiate test from index file of test directory which includes all test files. It will pass, artifact directory to store test results in html format.

All test passed with the following input and output (command line interface):

```
POST /api/user
Mongoose: users.insert({ updatedAt: new Date("Sun, 07 May 2017 22:11:16 GMT"),
createdAt: new Date("Sun, 07 May 2017 22:11:16 GMT"), address: { address1: 'address
street 1', address2: 'address street 2', city: 'Wales', state: 'New South Wales', zipcode:
'1234', country: 'Australia', phone: '+412543322222', _id:
ObjectId("590f9b8411626e1046f797df") }, registrationDate: new Date("Sun, 07 May
2017 22:11:16 GMT"), profilePic: 'mytest.png', type: 'trainer', gender: 'male', password
'21232f297a57a5a743894a0e4a801fc3', email: 'andrei@gmail.com', name: 'Andrei', _id:
ObjectId("590f9b8411626e1046f797de"), isActive: true, location: [ 150.644, -34.397 ],
__v: 0 })
POST /api/user 200 271.230 ms - 596
```

✓ should create trainer user and respond with json and 200 status (341ms)

```
Mongoose: users.insert({ updatedAt: new Date("Sun, 07 May 2017 22:11:16 GMT"),
createdAt: new Date("Sun, 07 May 2017 22:11:16 GMT"), registrationDate: new
Date("Sun, 07 May 2017 22:11:16 GMT"), type: 'trainer', password:
'21232f297a57a5a743894a0e4a801fc3', email: 'andrei.customer@gmail.com', name:
'Andrei', _id: ObjectId("590f9b8411626e1046f797e0"), isActive: true, location: [], __v: 0 })
```

POST /api/user 200 10.584 ms - 339

userId 590f9b8411626e1046f797e0

✓ should create customer user and respond with json and 200 status

POST /api/user 400 5.732 ms - 81

✓ should not create user without password

POST /api/login

```
Mongoose: users.findOne({ password: '21232f297a57a5a743894a0e4a801fc3', email:
'andrei@gmail.com' }, { fields: {} })
```

POST /api/login 200 29.001 ms - 1393

✓ should login user and respond with 200 status

```
Mongoose: users.findOne({ password: 'e00cf25ad42683b3df678c61f42c6bda', email:
'andrei@gmail.com' }, { fields: {} })
```

POST /api/login 400 6.661 ms - 55

✓ should not login user and respond with 400 status

GET /api/user

```
Mongoose: users.find({}, { limit: 10, skip: 0, sort: { createdAt: -1 }, fields: {} })
```

```
Mongoose: users.count({}, {})
```

GET /api/user 200 19.348 ms - 926

✓ should respond with json



```
Mongoose: users.findOne({ _id: ObjectId("590f9b8411626e1046f797e0") }, { fields: {} })
GET /api/user/590f9b8411626e1046f797e0 200 6.972 ms - 311
  ✓ should respond with 200 status

GET /api/user/15 400 3.003 ms - 55
  ✓ should respond with 204, record not found

PUT /api/user

Mongoose: users.findOne({ _id: ObjectId("590f9b8411626e1046f797e0") }, { fields: {} })
Mongoose: users.update({ _id: ObjectId("590f9b8411626e1046f797e0"), __v: 0 }, {
  '$set': { gender: 'male', location: [ 150.644, -34.397 ], updatedAt: new Date("Sun, 07 May
2017 22:11:17 GMT") }, '$inc': { __v: 1 } })
PUT /api/user/590f9b8411626e1046f797e0 200 28.706 ms - 342
  ✓ should update user and respond with json and 201 status

Mongoose: users.findOne({ _id: ObjectId("590f9b8411626e1046f797e0") }, { fields: {} })
PUT /api/user/590f9b8411626e1046f797e0 200 15.029 ms - 342
  ✓ should update address of user and respond with json and 200 status

Mongoose: users.findOne({ _id: ObjectId("590f9b8411626e1046f797e0") }, { fields: {} })
Mongoose: users.update({ _id: ObjectId("590f9b8411626e1046f797e0") }, { '$set': {
address: { address1: 'address line 1', address2: 'address street 2', city: 'Wales', state:
'New South Wales', country: 'Australia', zipcode: '1234', phone: '+412543322222', _id:
ObjectId("590f9b8511626e1046f797e4") }, updatedAt: new Date("Sun, 07 May 2017
22:11:17 GMT") } })
PUT /api/user/590f9b8411626e1046f797e0/address 200 7.775 ms - 577
  ✓ should update address of user and respond with json and 200 status

PUT /api/user/100 400 2.831 ms - 55
  ✓ should return no user found
```

DELETE /api/user

Mongoose: users.findAndModify({ \_id: ObjectId("590f9b8411626e1046f797e0") }, [], , { remove: true, fields: {} })

DELETE /api/user/590f9b8411626e1046f797e0 200 9.790 ms - 39

✓ should delete user and respond with json and 200 status

DELETE /api/user/100 400 1.988 ms - 207

✓ should return no user found

POST /api/feedback

Mongoose: users.findOne({ password: '21232f297a57a5a743894a0e4a801fc3', email: 'andrei@gmail.com' }, { fields: {} })

POST /api/login 200 8.584 ms - 1393

Mongoose: feedbacks.insert({ updatedAt: new Date("Sun, 07 May 2017 22:11:17 GMT"), createdAt: new Date("Sun, 07 May 2017 22:11:17 GMT"), userId: ObjectId("590f9b8411626e1046f797de"), trainerId: ObjectId("58caa3238664432062aea426"), content: 'Nice trainer', \_id: ObjectId("590f9b8511626e1046f797e7"), createdAt: new Date("Sun, 07 May 2017 22:11:17 GMT"), \_\_v: 0 })

POST /api/feedback 200 11.081 ms - 287

✓ should create feedback and respond 200 status

GET /api/feedback

Mongoose: users.findOne({ password: '21232f297a57a5a743894a0e4a801fc3', email: 'andrei@gmail.com' }, { fields: {} })

POST /api/login 200 7.344 ms - 1393

Mongoose: feedbacks.find({}, { limit: 10, skip: 0, sort: { createdAt: -1 }, fields: {} })

Mongoose: feedbacks.count({}, {})

GET /api/feedback 200 10.529 ms - 353

✓ should respond with json

```
Mongoose: feedbacks.findOne({ _id: ObjectId("590f9b8511626e1046f797e7") }, { fields: {} })
```

```
Mongoose: users.find({ _id: { '$in': [ ObjectId("590f9b8411626e1046f797de") ] } }, { fields: { name: 1, email: 1, address: 1, location: 1, gender: 1, profilePic: 1 } })
```

```
Mongoose: users.find({ _id: { '$in': [ ObjectId("58caa3238664432062aea426") ] } }, { fields: { name: 1, email: 1, address: 1, location: 1, gender: 1, profilePic: 1 } })
```

GET /api/feedback/590f9b8511626e1046f797e7 200 24.765 ms - 596

✓ should respond with 200 status

GET /api/feedback/111 400 2.680 ms - 55

✓ should respond with 400, record not found

PUT /api/feedback

```
Mongoose: users.findOne({ password: '21232f297a57a5a743894a0e4a801fc3', email: 'andrei@gmail.com' }, { fields: {} })
```

POST /api/login 200 7.461 ms - 1393

```
Mongoose: feedbacks.findOne({ _id: ObjectId("590f9b8511626e1046f797e7") }, { fields: {} })
```

```
Mongoose: feedbacks.update({ _id: ObjectId("590f9b8511626e1046f797e7") }, { '$set': { trainerId: ObjectId("590f9b8411626e1046f797de"), updatedAt: new Date("Sun, 07 May 2017 22:11:17 GMT") } })
```

PUT /api/feedback/590f9b8511626e1046f797e7 200 8.658 ms - 287

✓ should update feedback and respond with 200 status

PUT /api/feedback/100 400 3.063 ms - 55

✓ should return no user found

DELETE /api/feedback

```
Mongoose: users.findOne({ password: '21232f297a57a5a743894a0e4a801fc3', email: 'andrei@gmail.com' }, { fields: {} })
```

```
POST /api/login 200 6.195 ms - 1393
```

```
✓ login
```

```
Mongoose: feedbacks.findAndModify({ _id: ObjectId("590f9b8511626e1046f797e7") }, [], { remove: true, fields: {} })
```

```
DELETE /api/feedback/590f9b8511626e1046f797e7 200 7.163 ms - 43
```

```
✓ should delete feedback and respond with 200 status
```

```
DELETE /api/feedback/100 400 3.666 ms - 211
```

```
✓ should return no feedback found
```

```
POST /api/messages
```

```
Mongoose: users.findOne({ password: '21232f297a57a5a743894a0e4a801fc3', email: 'andrei@gmail.com' }, { fields: {} })
```

```
POST /api/login 200 11.200 ms - 1393
```

```
Mongoose: messages.insert({ updatedAt: new Date("Sun, 07 May 2017 22:11:17 GMT"), createdAt: new Date("Sun, 07 May 2017 22:11:17 GMT"), recipientId: ObjectId("58ca54b437a8ac1437d1cd5f"), senderId: ObjectId("58caa3238664432062aea426"), description: 'how are you?', title: 'Test Message', _id: ObjectId("590f9b8511626e1046f797ec"), createdDt: new Date("Sun, 07 May 2017 22:11:17 GMT"), __v: 0 })
```

```
POST /api/messages 200 11.789 ms - 318
```

```
✓ should create message and respond 200 status
```

```
GET /api/messages
```

```
Mongoose: users.findOne({ password: '21232f297a57a5a743894a0e4a801fc3', email: 'andrei@gmail.com' }, { fields: {} })
```

```
POST /api/login 200 5.681 ms - 1393
```

```
Mongoose: messages.find({}, { limit: 10, skip: 0, sort: { createdAt: -1 }, fields: {} })
Mongoose: users.find({ _id: { '$in': [ ObjectId("58caa3238664432062aea426") ] } }, {
fields: { name: 1, email: 1, address: 1, location: 1, gender: 1, profilePic: 1 } })
Mongoose: messages.count({}, {})
GET /api/messages 200 16.082 ms - 317
  ✓ should respond with json
Mongoose: messages.findOne({ _id: ObjectId("590f9b8511626e1046f797ec") }, { fields:
{} })
GET /api/messages/590f9b8511626e1046f797ec 200 4.501 ms - 318
  ✓ should respond with 200 status
GET /api/messages/111 400 2.407 ms - 55
  ✓ should respond with 400, record not found

PUT /api/messages
Mongoose: users.findOne({ password: '21232f297a57a5a743894a0e4a801fc3', email:
'andrei@gmail.com' }, { fields: {} })
POST /api/login 200 6.394 ms - 1393
Mongoose: messages.findOne({ _id: ObjectId("590f9b8511626e1046f797ec") }, { fields:
{} })
PUT /api/messages/590f9b8511626e1046f797ec 200 8.459 ms - 318
  ✓ should update message and respond with 200 status
PUT /api/messages/100 400 4.395 ms - 55
  ✓ should return no user found

DELETE /api/messages
Mongoose: users.findOne({ password: '21232f297a57a5a743894a0e4a801fc3', email:
'andrei@gmail.com' }, { fields: {} })
```

POST /api/login 200 5.874 ms - 1393

✓ login

Mongoose: messages.findAndModify({ \_id: ObjectId("590f9b8511626e1046f797ec") }, [], { remove: true, fields: {} })

DELETE /api/messages/590f9b8511626e1046f797ec 200 5.143 ms - 42

✓ should delete message and respond with 200 status

DELETE /api/messages/100 400 1.890 ms - 210

✓ should return no message found

POST /api/sports

Mongoose: users.findOne({ password: '21232f297a57a5a743894a0e4a801fc3', email: 'andrei@gmail.com' }, { fields: {} })

POST /api/login 200 4.273 ms - 1393

Mongoose: users.findOne({ \_id: ObjectId("590f9b8411626e1046f797de") }, { fields: {} })

Mongoose: sports.insert({ updatedAt: new Date("Sun, 07 May 2017 22:11:17 GMT"), createdAt: new Date("Sun, 07 May 2017 22:11:17 GMT"), address: { address1: 'address street 1', address2: 'address street 2', city: 'Wales', state: 'New South Wales', zipcode: '1234', country: 'Australia', phone: '+412543322222', \_id:

ObjectId("590f9b8511626e1046f797f2") }, promptPicture: 'mytest.jpg', price: 100, ownerId: ObjectId("590f9b8411626e1046f797de"), description: 'baseball description', name: 'Baseball', \_id: ObjectId("590f9b8511626e1046f797f1"), createdDt: new Date("Sun, 07 May 2017 22:11:17 GMT"), isActive: true, tags: [ 'baseball' ], location: [ 150.644, -34.397 ], \_\_v: 0 })

POST /api/sports 200 19.945 ms - 593

✓ should create sport and respond 200 status

GET /api/sports

```
Mongoose: users.findOne({ password: '21232f297a57a5a743894a0e4a801fc3', email: 'andrei@gmail.com' }, { fields: {} })
```

```
POST /api/login 200 6.561 ms - 1393
```

```
Mongoose: sports.find({}, { limit: 10, skip: 0, sort: { createdAt: -1 }, fields: {} })
```

```
Mongoose: users.find({ _id: { '$in': [ ObjectId("590f9b8411626e1046f797de") ] } }, { fields: { name: 1, email: 1, address: 1, location: 1, gender: 1, profilePic: 1 } })
```

```
Mongoose: sports.count({}, {})
```

```
GET /api/sports 200 14.379 ms - 943
```

```
✓ should respond with json
```

```
Mongoose: sports.findOne({ _id: ObjectId("590f9b8511626e1046f797f1") }, { fields: {} })
```

```
Mongoose: users.find({ _id: { '$in': [ ObjectId("590f9b8411626e1046f797de") ] } }, { fields: { name: 1, email: 1, address: 1, location: 1, gender: 1, profilePic: 1 } })
```

```
GET /api/sports/590f9b8511626e1046f797f1 200 11.082 ms - 896
```

```
✓ should respond with 200 status
```

```
GET /api/sports/111 400 6.741 ms - 55
```

```
✓ should respond with 400, record not found
```

```
PUT /api/sports
```

```
Mongoose: users.findOne({ password: '21232f297a57a5a743894a0e4a801fc3', email: 'andrei@gmail.com' }, { fields: {} })
```

```
POST /api/login 200 5.724 ms - 1393
```

```
Mongoose: sports.findOne({ _id: ObjectId("590f9b8511626e1046f797f1") }, { fields: {} })
```

```
Mongoose: sports.update({ _id: ObjectId("590f9b8511626e1046f797f1") }, { '$set': { ownerId: ObjectId("58caa3238664432062aea426"), updatedAt: new Date("Sun, 07 May 2017 22:11:17 GMT") } })
```

```
PUT /api/sports/590f9b8511626e1046f797f1 200 26.286 ms - 565
```

```
✓ should update sport and respond with 200 status
```

```
Mongoose: sports.findOne({ _id: ObjectId("590f9b8511626e1046f797f1") }, { fields: {} })
```

```
Mongoose: sports.update({ _id: ObjectId("590f9b8511626e1046f797f1") }, { '$set': {  
'address.address1': 'address line 1', updatedAt: new Date("Sun, 07 May 2017 22:11:17  
GMT") } })
```

```
PUT /api/sports/590f9b8511626e1046f797f1/address 200 20.788 ms - 591
```

✓ should update address of sport and respond with 200 status

```
PUT /api/sports/100 400 3.353 ms - 55
```

✓ should return no user found

```
DELETE /api/sports
```

```
Mongoose: users.findOne({ password: '21232f297a57a5a743894a0e4a801fc3', email:  
'andrei@gmail.com' }, { fields: {} })
```

```
POST /api/login 200 6.799 ms - 1393
```

✓ login

```
Mongoose: sports.findAndModify({ _id: ObjectId("590f9b8511626e1046f797f1") }, [], , {  
remove: true, fields: {} })
```

```
DELETE /api/sports/590f9b8511626e1046f797f1 200 5.699 ms - 40
```

✓ should delete sport and respond with 200 status

```
DELETE /api/sports/100 400 1.610 ms - 208
```

✓ should return no sport found

42 passing (1s)



## Users Module Test Coverage Report:

all files server/apis/users/

80.18% Statements 182/227 46.28% Branches 56/121 91.3% Functions 42/46 80.18% Lines 182/227

File	Statements	Branches	Functions	Lines
index.js	75.14%	133/177	46.22%	55/119
users.test.js	98%	49/50	50%	1/2

## Sports Module Test Coverage Report:

all files server/apis/sports/

78.99% Statements 203/257 44.87% Branches 78/156 93.18% Functions 41/44 78.99% Lines 203/257

File	Statements	Branches	Functions	Lines
index.js	73%	146/200	44.87%	70/156
sports.test.js	100%	57/57	100%	0/0

## Personal Message Module Test Coverage Report :

all files server/apis/messages/

89.92% Statements 116/129 47.62% Branches 20/42 92.31% Functions 36/39 89.92% Lines 116/129

File	Statements	Branches	Functions	Lines
index.js	82.67%	62/75	47.62%	20/42
messages.test.js	100%	54/54	100%	0/0

## Feedback Module Test Coverage Report :

all files server/apis/feedbacks/

88.15% Statements 119/135 47.73% Branches 21/44 92.31% Functions 36/39 88.06% Lines 118/134

File	Statements	Branches	Functions	Lines
feedbacks.test.js	100%	54/54	100%	0/0
index.js	80.25%	65/81	47.73%	21/44

## Metric explained:

**Statement:** how many statements/actions in code are executed. For example, loops or if statements.

**Branches:** some conditional statements may not be executed (for example if/else, if executes then else, will not execute). Then these statements create branches and metric tells how many of these branches have been executed.

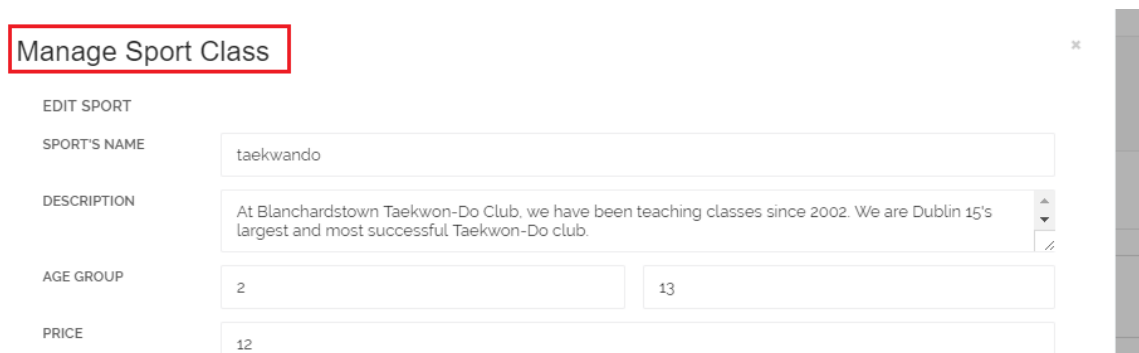
**Functions:** the percentage (proportion) of the functions is defined and which have been called.

**Lines:** the percentage (proportion) of the lines of code which have been executed.

## 3.2 User Testing (UX)

### 3.2.1 Design & Navigation

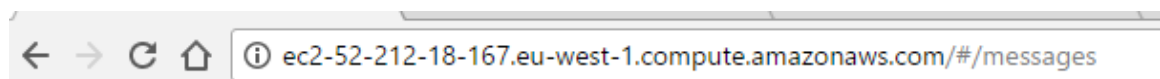
Main color scheme is white and gray (background, navbar) gives neutral, simple and calm look for user. However, some images uploaded by trainers are bright and colorful that stand out and attract user attention. Most of the pages were found to use defined headings and page titles. Page titles appears in search results however at the top of the browser's heading is missed. Defined heading and titles allow users to summarize and get overall view the content of the pages quickly.



Manage Sport Class	
EDIT SPORT	
SPORT'S NAME	taekwondo
DESCRIPTION	At Blanchardstown Taekwon-Do Club, we have been teaching classes since 2002. We are Dublin 15's largest and most successful Taekwon-Do club.
AGE GROUP	2 13
PRICE	12

### 3.2.2 URLs

Website uses appropriate URLs throughout. User can make an accurate guess about the contents of the by looking at the URL.



### 3.2.3 404 & HTTP 301

Findasportclass website does not have custom 404 pages that handles missing pages correctly and allow user to navigate correct page.

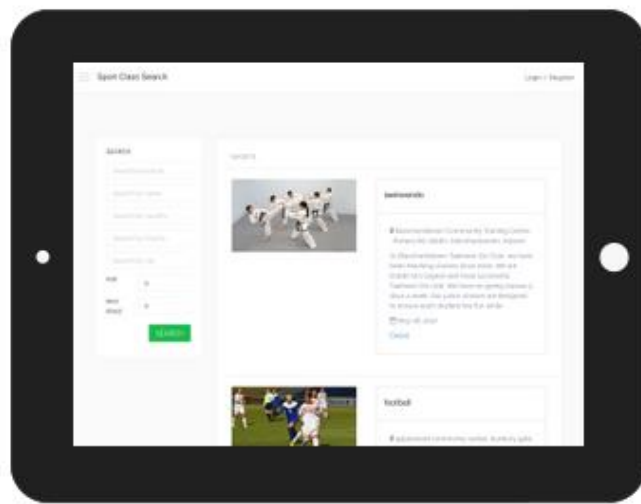
However, it follows the best practise of using permanent (HTTP 301) redirect between URLs with or without trailing slashes. For example, when user type in <http://ec2-52-212-18-167.eu-west-1.compute.amazonaws.com/#/messages/> it will redirect to <http://ec2-52-212-18-167.eu-west-1.compute.amazonaws.com/#/messages>.

### 3.2.4 Sitemap

Findasportclass doesn't have a sitemap. As it small site and all pages are easy to discover, it will probably doesn't need to have a sitemap.

### 3.2.5 Mobile & Tablet

It is important that content is optimized for a wide range of devices as a growing percentage of web browsing is done on phones and tablets. Findasportclass site is fully optimized for viewing content on a mobile or tablet.



### **3.2.6 Heuristic Evaluation**

#### **Visibility of system status**

Site has some visibility features implanted such as

- Changing mouse cursor (hand over) if user navigates to the navbar
- Highlights selected navbar
- Change text color on selected tab.

#### **Match between system and the real world**

Overall, there is a strong match between the system and the real world. However, as most of the content (class advertisements) on a website is uploaded by users there is not much control over vocabulary used on the website. So, the content might include slang or words that non-native English speakers may not understand.

#### **User control and freedom**

As site is SPA (single page application) and user are always on the same page or on the modal pop-up window that provides “emergency exit”, findasportclass follows good practice of “user control and freedom”.

#### **Error prevention**

Only one field fails on validation. During the registration user can provide and string on email field. This could be improved. Overall, fields has validation and users are not able to finish forms without proper input.

#### **Recognition rather than recall**

Website follows good practice of “recognition rather than recall”. For example, when user perform the search he gets back the list of sport classes. Each sport class has basic information such as exact address and brief description of the sport class. If user wants to get more information he can click on the “detail: link and modal pop-up window brings him to the full ad.

### **Flexibility and efficiency of use**

Site provide users with the search criteria like location, price age etc to narrow down their search result. This is very efficient; however, site could provide filtering by date added and price in descending or ascending order.

### **Aesthetic and minimalist design**

Site has minimalist design. Interface is not overloaded with features.

### **Conclusion & Recommendations**

The site meet its objectives, but require some further development in terms of UX. All functionality is working properly, however there some additional functionality needed to increase usability from user perspective.

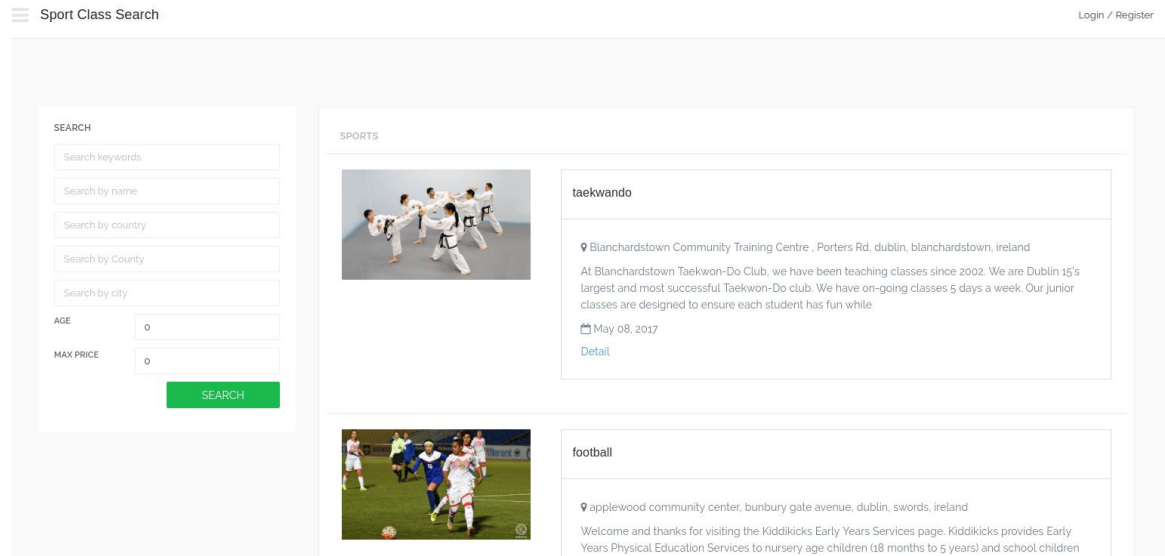
All links on a page operate correctly, validation is missed only on register form in "email" field.

List of recommendations:

1. Add filtering so that user can filter list of sport class by date and price in ascending and descending order.
2. Add validation to "email" field in "register" form.
3. Allow user to choose how many results he wants to be displayed by adding appropriate dropdown menu above and the bottom of the search result.
4. Display average price for list of sport classes in search result
5. Add custom 404 pages to handle missing pages.

## 4 GUI

The main page will contain a main screen with “login” and “register” buttons. Also, main page provides searching functionality and results section. User can search and view classes without registration.

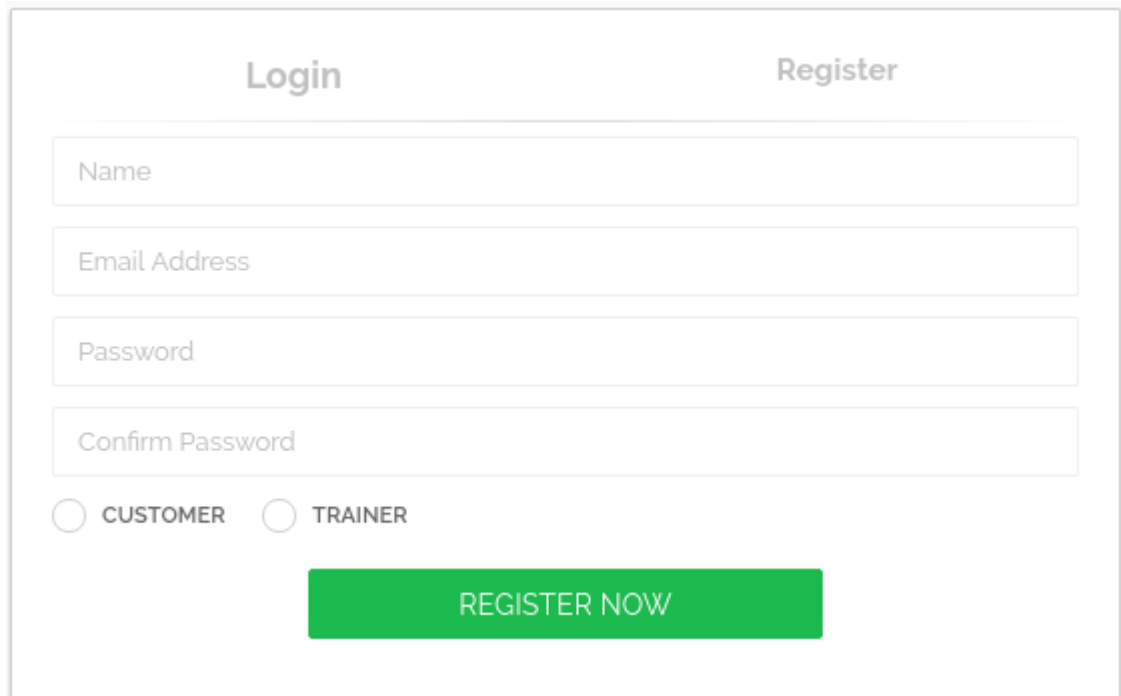


Main page

To **login** user has to provide his email and password used during registration.

The screenshot shows the login form. At the top, there are two tabs: 'Login' (active) and 'Register'. Below the tabs are two input fields: the first contains the email address 'andreivanov86@gmail.com' and the second contains a masked password '.....'. Below the password field is a checkbox labeled 'REMEMBER ME'. At the bottom of the form is a large blue button labeled 'LOG IN'.

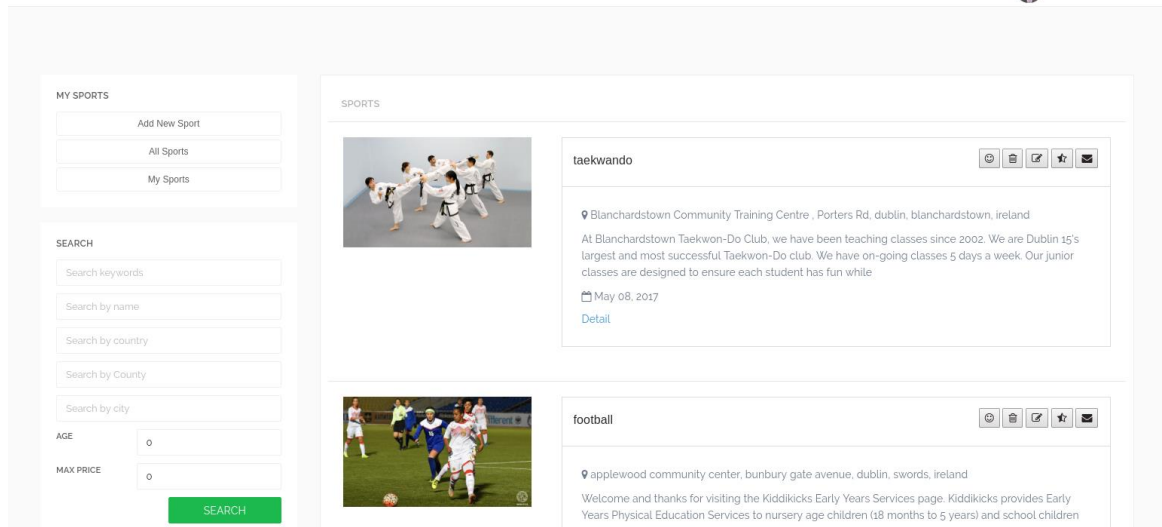
**Register page** will allow users to register their accounts. Some information is required to create valid account. If user is trainer that wants to advertise a service, then he/she must tick the appropriate checkbox to enable advertising functionality on the account.



The image shows a registration form with two tabs: "Login" and "Register". The "Register" tab is active. The form contains four input fields: "Name", "Email Address", "Password", and "Confirm Password". Below the fields are two radio buttons: "CUSTOMER" and "TRAINER". A green button labeled "REGISTER NOW" is positioned at the bottom center of the form.

*Register page*

After registration trainer can login and see extra functionality on his **trainer's authorized main page**.



*Trainer main page*

On the left side trainer has buttons view rating, delete class, edit class, write feedback, pm” (from left to right). Delete and edit class buttons are visible only if the trainer is class owner.



*Buttons*

On the left side trainer has “add new sport”, “all sport”, “my sports” buttons.

Clicking to “add new sport” will trigger modal pop-up window, where trainer can add new sport class details.



## Manage Sport Class

ADD SPORT

SPORT'S NAME

DESCRIPTION

AGE GROUP

PRICE

ADDRESS LINE 1

ADDRESS LINE 2

CITY

STATE

COUNTRY

ZIP CODE

*Add sport class modal pop-up.*

“My Sport” – allows trainer to view only his classes.

Registered consumer can contact the trainer via private message service, leave feedback, rate sport class from 1 to 5 scale and view rating and feedback. User can see average trainer rating based of the previous rating score.

## Feedback ✕

1 2 3 4 **5**

Your feedback

SAVE

*Add rating and feedback*

### Average rating 3.7 / 5 from 3 feedbacks ✕

FEEDBACK

good trainer. (5 / 5)  
👤

dssdsd (4 / 5)  
👤

ssss (2 / 5)  
👤

*View rating and feedbacks*

On the left user can open side menu where he can enter to the private message service (inbox)

MESSAGES

All  
Today  
Tomorrow

SENDER	TITTLE	DESCRIPTION	DATE	REPLY
andy	interested in taekwando classes	Hi Andrei I am interest in taekwandoo classes for my 7 years son. What age group he can fit? thank you Andy	May 08, 2017	📧

1

## 5 Evaluation

The website was evaluated in two ways, as a resource for trainers and costumers, who are looking for sport class for themselves or their kids. Trainers were agreed that most of them running classes is more hobby and they do it on their free time. So, they don't have extra financial resources to create their own website or spend time on advertising their services on a different social media sites (Facebook etc.) Findasportclass website would bring all enthusiastic trainers into one place. Then consumers will be happy to use quick and simple sport class search and interact with the trainer.

## 6 Conclusion

The concept of my final project was difficulty to find suitable sport class for my 5 years old son. Living in Swords, county Dublin I noticed that most of the adds are on the board in the local community centres. To get any information about the sport class you must physically go there.

The reason I chose angular 2 is, because I started my internship, where the company use angular 2 for the front end. I found that combining my final project and work experience into one will give me quicker and better understanding of cutting-edge technology – angular 2. At the start it was very hard to learn and understand as I did not have any experience, but now I am very confident with angular 2 and express framework.

## 7 References

### 7.1 Tutorials used

1. RESTful Web Services with Node.js and Express  
<https://app.pluralsight.com/library/courses/node-js-express-rest-web-services/table-of-contents>
2. Building Web Applications with Node.js and Express 4.0  
<https://app.pluralsight.com/library/courses/nodejs-express-web-applications/table-of-contents>
3. Angular 2 Fundamentals  
<https://app.pluralsight.com/library/courses/angular-fundamentals/table-of-contents>
4. Introduction to Mongoose for Node.js and MongoDB  
<https://app.pluralsight.com/library/courses/mongoose-for-nodejs-mongodb/table-of-contents>
5. Angular 2: Reactive Forms  
<https://app.pluralsight.com/library/courses/angular-2-reactive-forms/table-of-contents>
6. Angular 2 with TypeScript for Beginners: The Pragmatic Guide  
<https://www.udemy.com/angular-2-tutorial-for-beginners/learn/v4/overview>

### 7.2 References

1. <https://angular.io/>
2. <https://expressjs.com/>
3. <https://github.com/dwyl/learn-istanbul>
4. <http://unitjs.com/guide/mocha.html>
5. <https://wiki.jenkins-ci.org/display/JENKINS/TestComplete+JUnit+Plugin>
6. <https://www.npmjs.com/package/jenkins-mocha>
7. <http://mongoosejs.com/>

## **8 Appendix**

### **8.1 Project Proposal**

#### **8.1.1 Objectives**

I am looking to develop a web application that will act as an online portal. It will allow people find sport classes matching their criteria and interact directly with the trainers who runs the classes. The web application will allow trainers/instructors to create their own profile with their details and classes offered on a regular basis.

Once registered, end users will be able to search for sport classes by criteria they filter (sport category, location, price, time etc.), send private messages to the trainers, and leave feedback.

#### **Goals**

- To create web application that allow trainers/instructors advertise their services
- To allow consumers to search for sport classes matching their criteria
- To allow consumers and trainers interact with each other directly.
- To ensure that the site is easy to navigate and sign-up process is as straightforward as possible.
- To ensure that project is completed within specific timeline

#### **8.1.2 Background**

Using search engines such as google and yahoo users can search for sport classes by type and location they require. For example, if I am in Swords, County Dublin and are looking for football class for my 5 years old son I can use google or yahoo search engines and they will find a dozen of sport classes available. Google

provide links for different websites such as local community centres' webpages, trainer's outdated websites and so on. Also, I discover that lot of people advertise their classes on the board in the local shops and community centres. Finding sport class with all suitable criteria or even getting a list of available classes in your area is very time consuming. People have to ring or go personally to local community centres, visit all links that google search result provides and then manually filter available and suitable classes for them. "FindaSportClass.ie" web application will bring all trainers and their services into one place and make it easy to search for the end user and get more overall information about sport classes available in their area.

## **Audience**

The web application will be targeted at two groups of people.

### **Trainers/instructors**

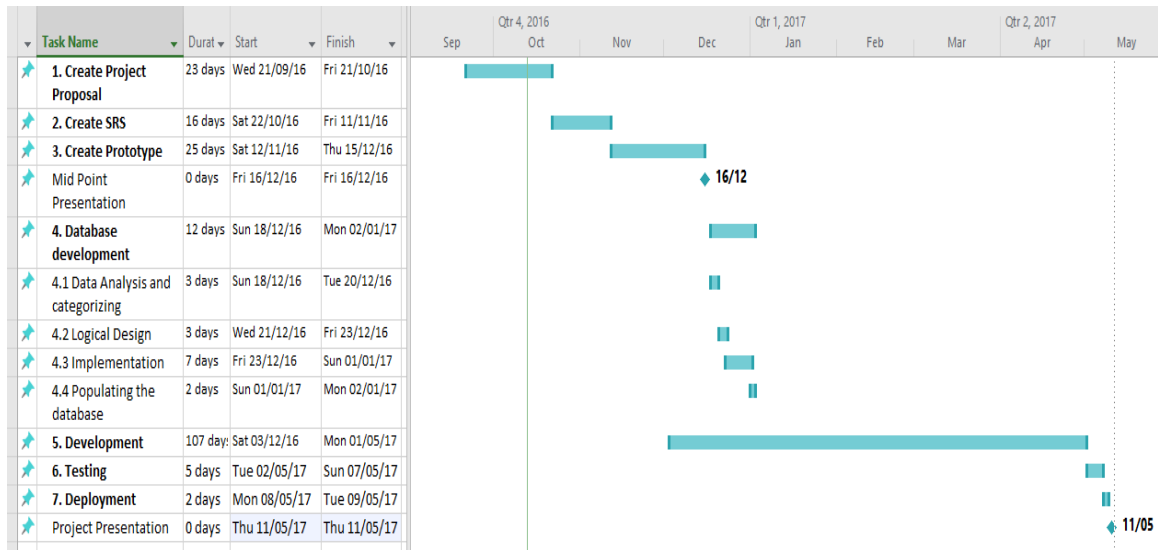
This will be the individuals who are running sport classes and want to advertise their services. It could be football, taekwondo, boxing or gymnastic etc

### **End users**

This will be anyone who are interested to find some suitable sport class or to enrol local once-off activity. "FindaSportClass.ie" will be very useful for parents who wants to find list of local sport classes available for their kids and keep them busy during specific time.

## **8.2 Project Plan**

Key project milestones have been detailed in the Gantt Chart. Screen shot of the Gantt chart is attached below.



### 8.3 Technical Details

The web application will be developed using different web technologies. It will be simple, responsive, fluid single-page application (SPA) where the single page dynamically updates as the user interact with the application. The front end application will be developed using HTML, CSS, Bootstrap, JavaScript, Angular 2, JQuery & AJAX where applicable. MongoDB database will be used to store and access the key records and information. For the back end NodeJS and Express framework will be used. Breaking code into modules allows it to be easily understand, develop and maintain. Web application will be hosted in Amazon Web Services (AWS).

### 8.4 Challenges and Considerations

In terms of key challenges, first consideration is the time to fully complete the project and meet all the project milestones as the workload on year 4 is huge. In terms of web development my knowledge in HTML, CSS and Bootstrap is strong, but I have no experience in NodeJS and Angular 2. My technical skills



in terms of NodeJS and Angular 2 are very limited. I will need to research and learn it as the project progresses and consider how much time learning Angular 2 will take.

## **8.5 Research and Interviews**

5753 people living in Ireland participate in the research. Questionnaire has been created using google research tool and link shared across the Facebook groups and friends.

### **Questions:**

**Do you run any sport classes?**

**if yes:**

**Where do you advertise your sport classes?**

- a) Web site
- b)community centres
- c)local newspaper

**if no:**

**What is your age group?**

- 1)3-7
- 2)7-11
- 3)11 - 15
- 4)15-18
- 5)18-24
- 6)24-40
- 7)40-

**Have you or your kids ever sign up or willing to sign up for the classes?**

1)Yes

2)No

**Are you thinking to sign up for the classes?**

1)Yes

2)No

**How you found the suitable sport class for you or for your kids?**

1)Online Search engine

2)Friends

3)Local community centres

**Do you found search engine effective?**

Yes, works for me fine

No

**What is the import criteria when you search for the sport classes? Please start from MOST IMPORTANT.**

1)Location

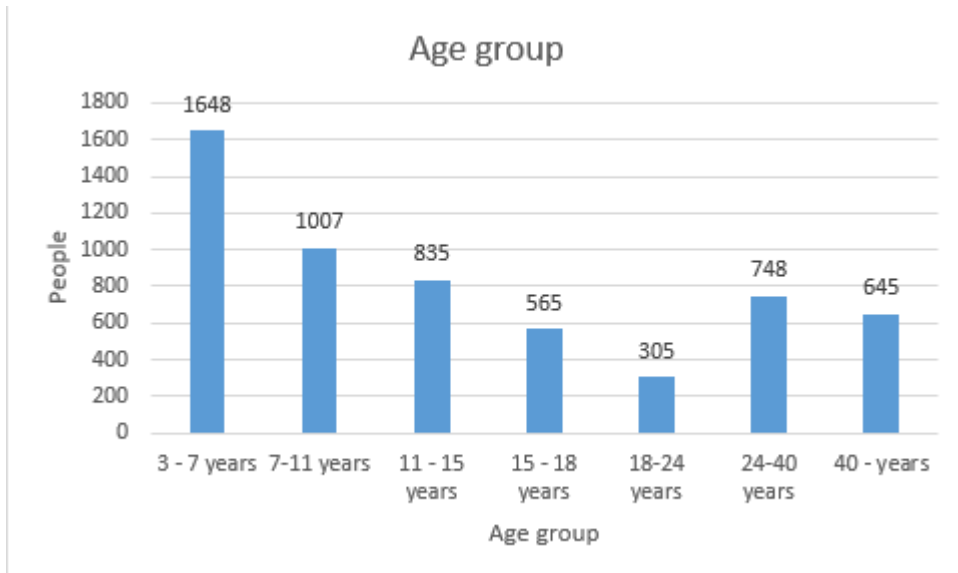
2)Time

3)Age

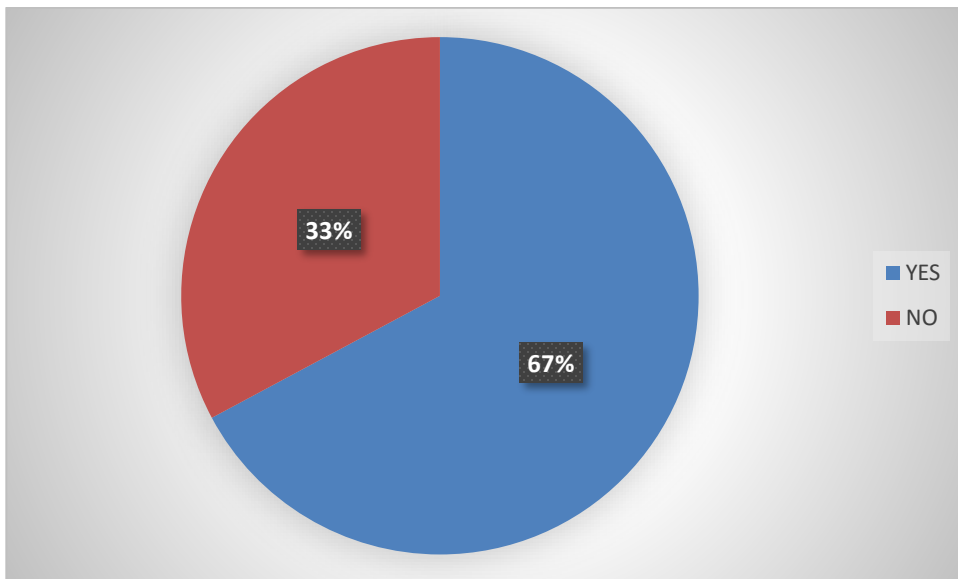
3)Sport category

4)price

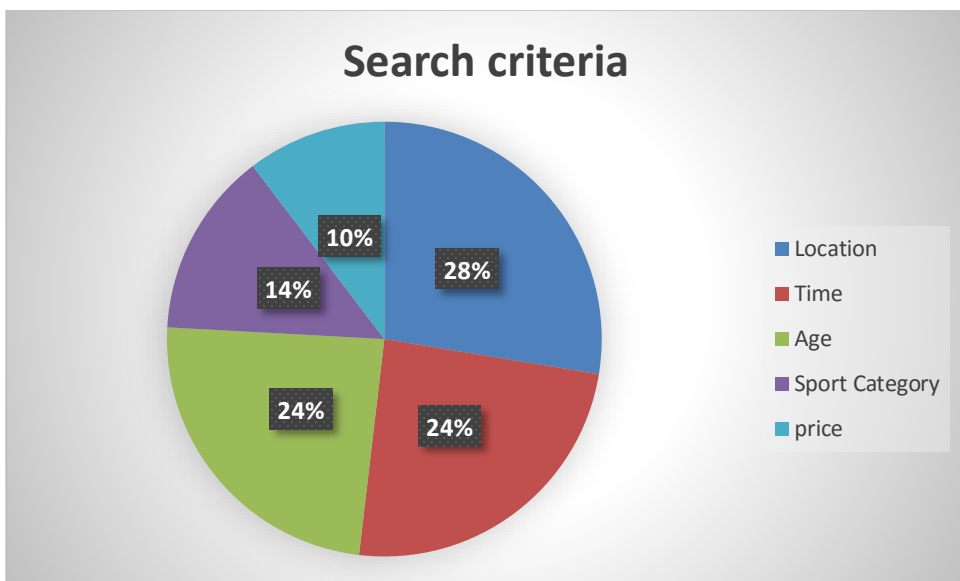
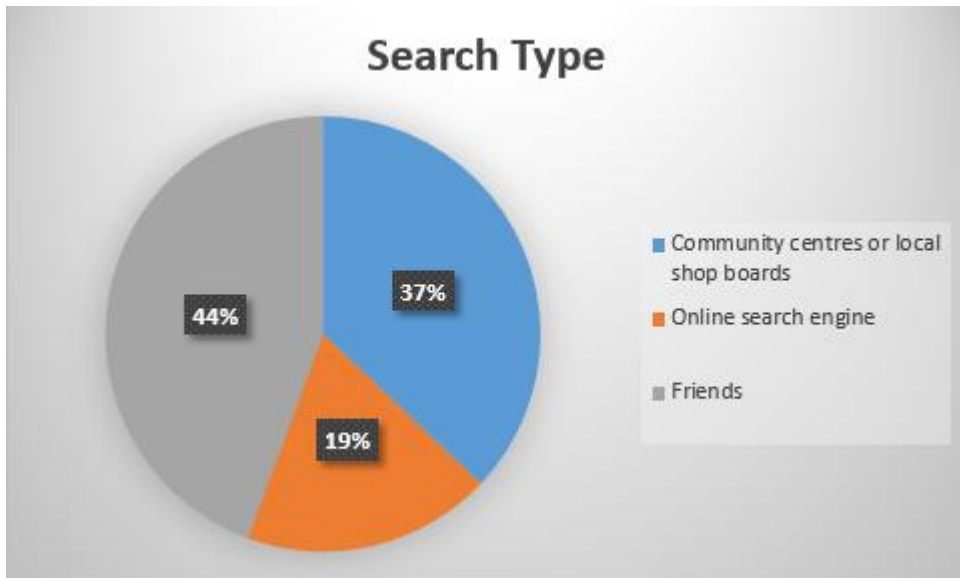
## RESULTS



Have you or your kids ever sign up or willing to sign up for the classes?



How you found the suitable sport class for you or for your kids?



## 8.5.1 Interviews

### Interview 1

**What is your name?**

My name is Ardy Mehist

**What is your occupation?**

Software Tester

**Do you participate in sport classes? What class?**

I am doing kickboxing 3 times a week.

**How did you find this sport class? Was it difficult to find?**

I was googling and got 4 websites with the kickboxing classes in my area. Then got more information over the phone.

**What you think about the web application that brings all trainers and instructors into one place. Users can search by sport category, location etc.**

This is good idea as the google give a lot of results. Some of the classes are not available any more at specific location, timetable is changed etc. Trainers order to create static "business card type" website, but never maintain or update information. This make end user difficult to find suitable class for him.

**Interview 2**

**What is your name?**

Emer

**What is your occupation?**

Third level support

**Do you participate in sport classes? What class?**

I do not go to any classes but my 4 years old son attend football classes 2 times a week.

**How did you find the sport class? Was it difficult to find?**

I went to the community centre and found advertisements of several classes around my area. Then I contacted trainers over the phone. It wasn't difficult, but I spend some time to go to community centre.

**What you think about the web application that brings all trainers and instructors into one place. Users can search by sport category, location etc.**

That would be great. The most important criteria for me would be the age. As most of classes are for 5+ years old kids I got the problem to find classes for my son who is 4 years old.

There is someone who provide classes for 4 years old kids, but it's very difficult to filter them out from rest of the classes.

### **Interview 3**

**What is your name?**

Conor

**What is your occupation?**

I am manager in Tesco, but I work part-time as a taekwondo trainer. I am training mostly kids who are 12 years old or under.

**What you think about the web application that brings all trainers and instructors into one place. Users can search by sport category, location etc.**

Well, that's what we don't have at the moment. Most of trainers work as trainers part-time only. We don't have funds to hire software developer to build proper website and then pay for maintaining and updating information.

We advertise classes on donedeia.ie, adverts.ie, but people use this website mostly for other purposes. If you are going to build web application, it would be good if it would have self-intuitive user interface. Trainers could easily add, delete, update information about the class.

## **8.6 Monthly Journals**

**Month: September**

### **Introduction**

This is the first month of the final project reflective journal. So I would like to introduce myself.

My name is Andrei Ivanov and I am 30 years old. I have my final year in Bachelor Honor in computing. Also I have two years full-time course (Software development stream) in progress run by FIT ICTP. I have two sons – Andrey 5 years old, and Steven 3 months old. For the last 4 years I live in Swords, County Dublin.

So my final year is very tough, considering that I have another full time course and new born baby. First weeks of September I was thinking to get deferral and come back next year, when the full time college is over. Last minute I changed my mind, and enrolled for the final year.

### **Project Idea**

21/09 2016, I spend all day thinking about project ideas. I wanted to develop something original, but had no luck to come up with something.

24/09/2016 – My wife asked me to find football class in Swords for older son. I started to google and got lot of links where the location or age were not suitable. I got few links for local community centers, classes were advertised all over the place with no detailed information provided. I did approximately 5 different phone calls to get picture about local football activities available for my son. I visited few local community centers and noticed ads on the board (“in 21<sup>st</sup> century ads on the board? They are probably kidding” I thought). I found out that there is lot of sport activities for kids and adults in Swords and also how much time I waste to find this out. So I started to think that it would be a great idea to develop web application and put all activities together by category, location, time, price and age. Trainers would advertise their classes and be able to manage their ads by adding

information, pictures and maybe videos. Parents could use this web application to find different activities (sport classes, dancing, drawing, drama, and so on).

Next few days I was thinking about additional functionalities such as possibility for trainers/organizers to create different events (such as charity marathons), kids events/parties and bootcamps for adults. So users can find them all by location, activity category and sub-category with all details.

### **Month: October**

This month, I finished my project proposal document and submitted it to the Moodle. The project proposal template we got was very detailed and some topics were not related to the “project proposal”. I followed the template, however, I felt that I am writing something mixed between proposal and requirements specs document.

### **Supervisor meetings**

I got Ralf Bierig for supervisor. Unfortunately, I haven't have any meetings with him yet. He wants everyone in on Fridays which does not suit me. I am going to arrange meeting and talk to him soon.

Anyways, Project proposal document is submitted and now I am going to focus on requirements specification document.

### **Month: November**

This month I started to write SRS. It was long and boring document. I accidentally met my supervisor Ralf and we had a quick chat about the project. Ralf wasn't happy that I wanted to develop back-end in PHP as it's old technology. I tried to defend myself that I need it in my further job I am starting in January.

### **Month: December**

In December, we will do mid-term presentation, where we should present prototype. I am facing ow with the problem: I don't have enough knowledge in angular 2 to develop even prototype.



I accidentally met Ralf again in Artificial Intelligence class and had a meeting. I tried to explain him that I can't develop prototype as I need to learn angular first. Ralf advised me to develop interactive bootstrap app that will fake my main app and present it. It was good advice, I spent just two evenings developing prototype.

Presentation went fine. Both, Padraig and Eammon, were happy even though I am very bad at presentation.

### **Month: January**

This month I haven't worked on the project as it was exam period. We had 3 exams and two of them was theory based that I really hate.

I started my internship where I got to do project to learn angular 2.

### **Month: February**

This month I just learned angular 2. I followed lot of tutorials on Udemy and pluralsight, but faced to the problem – All tutorials were in angular beta version. After angular 2 final version released (December) there were not much tutorials out there. So, my only mentor was official documentation and one tutorial on pluralsight that Ralf recommend me. In few weeks, I finished my job project and started to work on actual company SAAS project. In few days, I understood that I am not going to develop anything in back-end, so I abandon idea to do my final project back end in PHP. To be honest I wasn't happy to learn PHP, as I totally agreed with the Ralf that this is old technology. The only reason I chose PHP was because of work.

### **Month: March**

Every day I had tremendous progress in angular 2 at work. I was surprised how quickly you can learn at work. I learned how to create services and consume API's. I started to think about back-end for my actual college project and decided to go with nodejs express. I was thinking that will be cool that my app will be mean stack. Specially I found lot of tutorials in nodejs express RESTful. But this all requires me extra work – I needed to change my database into mongodb as all meaningful

tutorials in nodejs express were with the mongoDB. It wasn't hard. I followed lot of tutorials and adapted them to my project.

This month I created project skeleton and started to add models and create HTTP methods in express framework.

Suddenly my younger son got very sick. So, we decided that my wife and younger son go to Russia for treatment. They left 24<sup>th</sup> of March without return ticket as we did not know how long treatment will take. I left with the 5-year-old son in Ireland. I agreed at work 3 days a week as I needed to look after my son and couldn't afford childminder.