Sunil Maharjan

Markeplace web application with Foglab
(Foglab with Marketplace)

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

11th August, 2017

Helsinki Metropolia University of Applied Sciences

Author(s) Title Number of Pages Date	Sunil Maharjan Foglab with Marketplace 30 pages 11 August 2017
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Janne Salonen, Title (Thesis Instructor)

The purpose of the project was to create a web based application to make it easier for students to look for jobs and internships and employees to post their vacancy announcement. It is a job portal web application that is a part of the Foglab website of Metropolia. As part of the project also includes the redesigning the pages of the Foglab website with a new and innovative design as well.

The project was carried out by using two of the most popular javascript framework available combined with the powerful database to manage data which is popularly known as the MEAN stack. AngularJS is one of the most popular and powerful javascript framework for designing a web application. NodeJS is a powerful backend server framework that is entirely based on javascript which makes programmers easy to use. Also add on the new concept of saving data on a database in the form of objects which is perfectly done by the MongoDB database. Now, combine all these three ingredients and you get a super powerful recipe or tool known as the MEAN stack which was used to create this dynamic web application.There are various other tools and packages used alongside these frameworks to make the application better in terms of size, performance and management.

Keywords	Web application, IoT, MEAN stack, portal, Angular, NodeJS, HTMl, Javascript, frontend, backend, open-source, bootstrap, Jquery

Contents

Abbreviations

MEAN	MongoDB ExpressJS AngularJS NodeJS
HTML	HyperText Markup language
CSS	Cascading Style Sheet
SQL	Standardized Query Language
IoT	Internet of Things
ES6	ECMA script 6
JSON	JavaScript Object Notation
API	Application Program Interface
NPM	Node Package Manager
CLI	Command-Line Interface
MVC	Model View Controller
SMTP	Simple Mail Transfer Protocol

1 Introduction

The advancement in technology is one of the greatest achievements in human history. It has made lives of people easier, faster and brought the world closer. One of the most popular product of technology is the internet and the devices or other elements that can access data through the internet are known as the Internet of Things (IoT). Different products and services that we use today are all due to the advancements made in the IT sector or specifically in the field of web services and technologies.

Web technologies are growing in such an exponential scale that it has become hard to catch up to the new technologies in the market. There is a huge supermarket of technologies that one can use to create different types of application or services. This has all been available just a few years back. Before, that web technologies were simple, with some static content to show to the user and making boring old websites. Now, the technology has been very advanced and developers has been able to integrate a full-fledged working application that works in a web page.

The technology industry is growing so rapidly that it has become hard for web developers to keep up to the web world with new languages and frameworks surfacing every day. This has been even harder to the new comers who have just entered the web development field with technologies that they studied a year back has been changed with newer technologies.

The project consists of creating a web application, a job portal which would be integrated in to the foglab website of Helsinki Metropolia University of Applied Science. Apart from the web application, the project included the redesigning of the foglab website itself with changes done as instructed by the supervisor. The project was completed using one of the most popular combination of technologies that is used to create simple yet powerful dynamic web app, the technologies and methods of which are discussed further in the report.

2 Theoretical Background

The web application is created by using the MEAN stack framework combination. Every letter has a technology involved in it, the “M” stands for Mongo which is a database service. “A” stands for AngularJS, which is a popular and powerful javascript framework that is used to create the user interface or the front-end part of the application. The “N” stands for NodeJS, which is a powerful javascript framework that is used in the server side for creating API’s or also known as the back-end part of a web application. Finally, “E” stands for ExpressJS which is a web application framework created on top of NodeJS which makes coding in the backend a breeze.

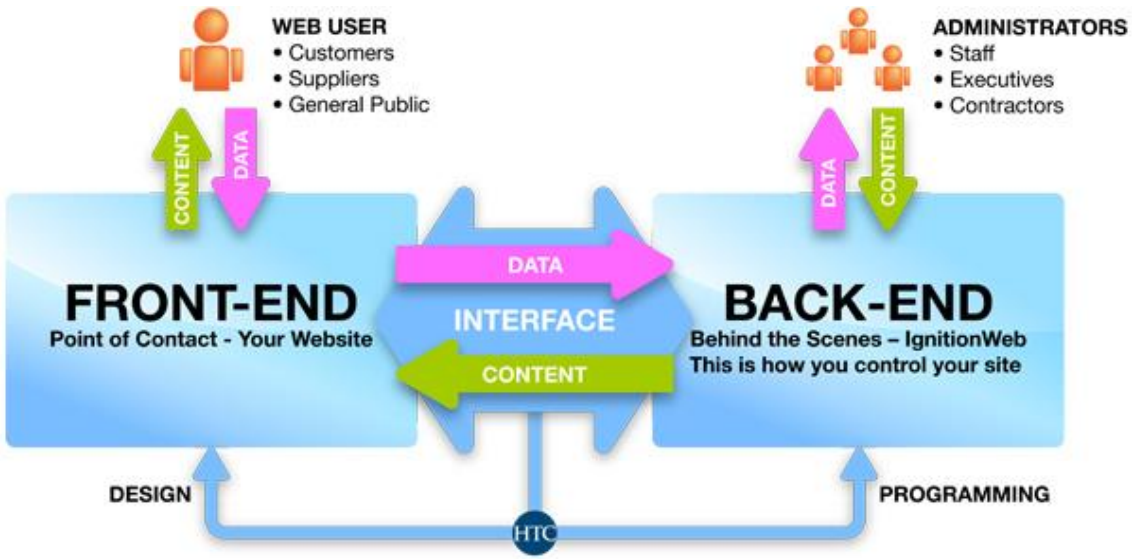

Figure 1 – Web Application Architecture

In a web application, there are two layers in the software architecture, the one that the user sees is known as the presentation layer i.e. front-end and the layer where all the functionality and services occur in the back or the server is known as the access layer i.e. back-end. There are different technologies involved in both the front-end and backend, such as HTML, CSS and Javascript are used for the former whereas technologies such as PHP, python, SQL are used for the latter.

2.1 Frontend Technologies

The frontend technologies or referred to as the presentation layer of a web application is the view layer of the application. In other words, the layer which the user can view and

interact with. The most basic and most important technologies used in the frontend are namely HTML, CSS and Javascript in which Javascript is the one with a wide range of variety in terms of features and uses. Javascript, with new plugins and framework surfacing everyday, has changed the way people view content in the internet.

Figure 2 – The frontend spectrum

As shown in Figure 2, the most popular Javascript framworks are Angular, React, Ember, vue, Jquery, etc. The frontend technologies that have been used in the project are

2.1.1 Angular 2

Angular 2, one of the most popular javascript framework and loved by frontend developers, was first designed by Misko Hevery and Adam Abrons in 2009 and is officially maintained by Google. AngularJS version 1.0, also known as AngularJS was released in 2012 as a add on feature like jQuery to create dynamic web applications. It was later transformed dramatically after 2 years of research and created a full working framework. The AngularJS version 2.0, also known as Angular 2 was released in September, 2016.

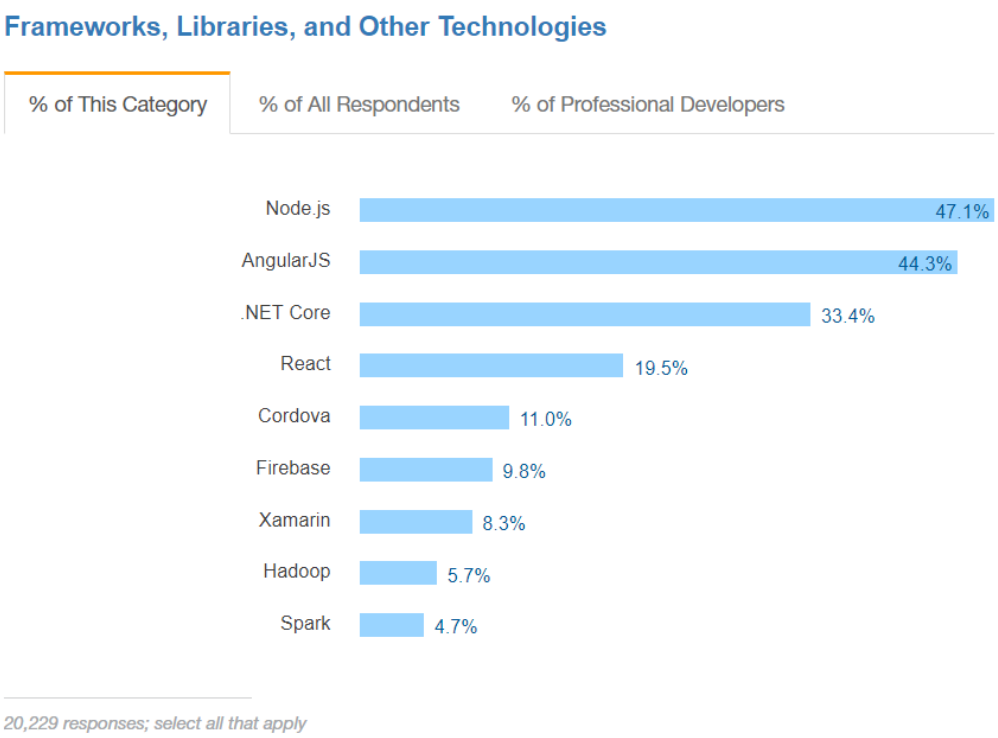

Figure 3 – Most used frameworks and technologies

According to the survey done by stackoverflow in 2017, NodeJS and AngularJS are the most commonly used technology in web development. The primary focus of AngularJS is to provide the best functionalities for server-side rendering for a perfect user interface. Angular2 was created over angularJS to overcome many issues that had been prominent with the angularJS. The benefits of using Angular2 over AngularJS are:

- Angular2 provides much better performance over AngularJS by using less memory and using superior template functionalities.
- Angular2 has advanced and powerful templating features giving better runtime and faster error detection.
- Angular2 focused on smooth server side rendering to the user interface and is dedicated to data capturing with the web works.
- Angular2 supports multiple platforms helping to make a fully functional native mobile UI on both IOS and android platforms.[4]

The basic building blocks of angular 2 are:

2.1.1.1 Typescript

Typescript is a open source language developed by microsoft on top of javascript to make it similar to a object oriented programming language like java. Typescript code can be easily written by a programmer and it should be compiled to javascript code on runtime by the browser. Angular 2 uses ES6 standard javascript language and typescript supports ES6 and is also recommended by angular 2 developers to use type script as opposed to others.

2.1.1.2 Modules

Module is a group of code that combines all the components, directives, pipes and services that are connected to each other. This combination helps to create an application. Modules can be public or private making the former visible while the latter's implementation details are hidden. The modules can export elements which then can be used by other modules. The elements of a module that can be exported should have a public attribute.

2.1.1.3 Components

Components are the building blocks of an angular2 application. Everything written in an angular app is a component. Like we build a house laying brick on top of another brick, components are stacked together to create a angular app. Component is a set of code that does a specific simple task and combining all these components together an angular application is created. Component in angular2 is the combination of Directive, controller and scope in AngularJS.

2.1.1.4 Services

Service in angular2 is just a simple class with a decorator injectable, meaning that it can be injected to other components as a dependency injection. Services are created with a specific purpose to perform a specific task. It can injected in a component all inside the application. The code in a service is normally to send or retrieve data to and from the server. That is why it is needed to be kept separate from the application for easier testing and debugging and for dependency injection.

2.1.1.5 Templates

Templates are basically the HTML code that is displayed in the web page. Templates are enclosed in the ‘ or “ mark which is also considered as a string. Also in ES6, templates have been made even easier by making it possible to write multiline code using the back ticks. Almost all the the HTML code is a valid syntax in a angular2 template. The template represents the view part of the view and controller model that was used in angularJS. Templates also help separating the view and the logic in a component.

2.1.1.6 Metadata

Metadata are the decorators for a class that is used to configure the behavior of the class. It allows developers to configure a class by setting different metadata on them which gives them a specific category. Metadata are built in in angular2 framework and can detect its own particular element by the help of metadata. The most commonly used metadata are @component, @pipe, @service, etc. So, if we want to make a class a component, we simply put the decorator @component in front of the class.

2.1.1.7 Data binding

The most prominent and powerful feature of angular2 is data binding. Data binding is a feature that helps to bind values of variables between the view and controller layer. There are four types of data binding in angular2. They are interpolation, event binding, property binding and two way binding. This new data binding feature helps a lot during coding in javascript and reduces a lot of code needed to be written to just simple few lines of code.

2.1.1.8 Dependency Injection

Dependency injection is also a new feature which allows to inject a dependency in a web application in the form of a service. By doing this, the service layer of the application can be created and debugged separately and used by dependency injection. Furthermore, it helps to reduce code duplicity because same service can be injected multiple times anywhere in the application. This approach helps to make the code flexible and make unit testing easier.

2.2 Backend Technologies

The backend technology used in a web project is known as the service layer of the application. In a web application, the frontend is the beauty, whereas the backend is the

brains. About 90% of the code that is required for an application to work successfully is done in the backend. Backend code runs on the server so the backend developers need to have a good understanding of the programming language and the databases as well as the server architecture. If an application constantly crashes, gives error or is really slow, it is always problem with the backend that is why backend development is the most important part in a software application.

There are many different backend programming languages. The most popular ones are ruby, Python, Sql, NodeJS, etc. NodeJS is growing to become one of the most popular Javascript backend development framework and also is one of its kind.

2.2.1 NodeJS

NodeJS is an open-source, cross platform Javascript runtime environment for executing Javascript code server side[5]. NodeJS is increasingly becoming one of the most popular and most used backend language. It enables javascript to be used for server side scripting and has a event driven architecture thus making it able to perform tasks asynchronously which is also one of the powerful feature that only NodeJS provides. NodeJS also uses different plugins or add-ons known as packages to be directly embedded to its application and be used for a specific purpose. It has its own package manager called NPM(Node Package Manager) which helps to install and manage all the packages installed in a web application.

2.2.2 ExpressJS

ExpressJS is a NodeJS web application framework or a package that is built on top of nodejs that provides different new features with minimum code for mobile and web applications. Even though NodeJS is very powerful and useful, coding in it is very long and cumbersome and doing simple tasks can require a lot of code. This is where ExpressJS comes in with its minimal code structure that does exactly what needs to be done without a big hassle. It is the most popular framework of NodeJS which is used 90% of the time by Nodejs developers.

ExpressJS is very popular because it provides a layer of fundamental features in a web application without obscuring Nodejs Features. There are many Frameworks based on

ExpressJS as well. ExpressJS also lets developers to have full customizability over its application unlike other languages like Ruby and Python.

2.3 Database

The main source of information in an application is the database. It is the core of a dynamic web application. An application doesn't have any value if it has no information to provide. The database gives life to an application. The database of an application should be well structure otherwise it becomes a huge mess to handle. There are different types of databases available now a days. There are traditional forms of databases such as MySQL, Postgre, etc. this application uses a new form of data storing called NoSQL database. MongoDB is one of them.

2.3.1 MongoDB

MongoDB is a cross-platform open source document base database which is totally different from the traditional realational database that has been used as the main form of data storage since a long time. It is a kind of NoSql database, meaning that it uses JSON like document structure rather than the table based structure used in relational databases such as MySQL. These type of databases are useful for certain type of application making them faster and easier to use. It is high performance, high availability and easy to scale. Also, deployment to large and complex multi-site from a single server is easy [6].

3 System Design and Implementation

The main goal of the project was to design a web application that could be used as portal for the employers and students to post and apply for jobs. So basically a job portal. Since this is a web application, not just a simple web site, dynamic web technologies were used to create a full-fledged web application with a backend API support. To complete the project, one of the popular web technologies were used namely Angular2, NodeJS, Expressjs and MongoDB. Different other technologies could be used but these technologies go hand in hand and work together and are compatible with each other so these technologies were chosen.

Also, there was some extra work included with the project. It was to update the already created foglab website with the above application integrated and some minor touch ups to make it better. For updating the foglab website, the already used wordpress framework was used and changes were made with the support for both Finnish and English language. The first section is about the design and implementation of the marketplace web application and the second section is the description of the foglab update.

3.1 Marketplace web application

The marketplace web application was created using the popular MEAN stack framework. The application is used as a mediator between the students and employers. Both user groups, the employer and the student, have a different page where the employer can post and delete jobs whereas the student can only search and apply for jobs. At first you have to register with you name and other information in the form and choose if you are an employer or a student and submit. After registering you have to validate your email and then you can login to the application using your username and your password.

3.2 Installation

To develop the application, all the required programming software has to be installed. The installation process and the versions of the software are given below:
Since all the languages and framework used are mostly based on packages, first a package manager should be installed. NPM provided by node is easy and faster to use and comes built in with NodeJS installation.

- NodeJS - Version 6.9.3

  To install NodeJS, we have to first go to the official node website which is www.nodejs.org and download the software related to their respective Operating System and follow the instruction. After installation, go to terminal if it is in linux or command prompt if it is in windows and check the version by typing the command

```
node -v
```

- Angular-cli – Version 1.0.0-beta.31

  Angular-cli is the command line interface of angular2 which helps to create an angular application from scratch using a single command. It is currently in beta version when developing. Angular-cli can be installed in the computer by typing the following command in the terminal

  ```
  npm install –g @angular/cli
  ```

  You can also check the version of the angular-cli by using the following command in the terminal

  ```
  ng --version
  ```

- MongoDB – Version 2.6.10

  The features and use of MongoDB can be directly used by adding a package in the NodeJS package.json file. However, the shell version, in which you can view all the data and make queries, can be installed by downloading and installing the .msi file for windows and .deb file for linux.

- ExpressJS – Version 4.15.2

  Since ExpressJS is a package used in the NodeJS, it is fairly easy to install and use in the application. There are two ways to install expressJS in the application. The first process is by editing the package.json file of the node application and adding express and its version in the dependencies section. Another way is to install Express through the terminal by using the command

  ```
  npm install –save express
  ```

  This command however installs the latest version of express rather than the version that you wanted which may later run to compatibility issues or issues related to deprecated code.

3.3 The backend API

The API of the application is the core of the application. The API contains all the relevant data and all the processes that run with it. It is very important to structure the code perfectly so that it runs as expected. The API of this application is divided in three sections. The first section deals with all the necessary stuffs that need to be carried out like initialization and dependency injection, creating paths and creating servers. The other two are the sub sections of the first one which are dedicated codes to work to create the model of the data to be stored and the other one for the queries of the database API.

3.3.1 The app.js

The app.js file is the main entry point to the application itself. This file controls all the route handling of the application. Here, all the basic things are coded such as initialization, route handling, dependency injection, Database connection and server creation.

The whole API is created as a NodeJS application. The app.js file needs to be initialized by using the “npm init” command and complete all the required instructions. Dependencies to the Node application can be installed using the command “npm install”. The package.json file that contains all the required information about the application and its dependencies is listed below:

```
{
  "name": "marketplace",
  "version": "1.0.0",
  "description": "app to post and apply for jobs",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit
1",
    "start": "node app.js"
  },
  "author": "Sunil Maharjan",
  "license": "ISC",
  "dependencies": {
```

```
    "bcryptjs": "^2.4.3",
    "body-parser": "^1.17.1",
    "cors": "^2.8.1",
    "express": "^4.15.2",
    "jsonwebtoken": "^7.3.0",
    "mongoose": "^4.9.1",
    "nodemailer": "^4.0.1",
    "passport": "^0.3.2",
    "passport-jwt": "^2.2.1"
  }
}
```

Listing 1. Package.json file of NodeJS application

The dependencies section on Listing 1 shows all the dependencies of the backend API. The name of the dependency and its required version is also displayed. The file shown on listing 1 is very important since it has all the information about the dependencies which can later be easily installed by using the command “npm install”.

“bcryptjs” is used to encrypt the password field of the database that is given by the user which would be encrypted when loading in the database. “body-parser” helps to easily parse the form data provided by the frontend. “jsonwebtoken” is used to transfer tokens between the front and back. “mongoose” is a useful dependency required to work with the database. “nodemailer” is used to send emails and “passport” and “passport-jwt” are used for secure authentication.

Apart from this, the app.js is used to install all the middleware. It is also used to create all the routes or links that can be used to access the API. It also creates an initial route that redirects to the homepage and it also creates a server to host the application and make it available as well.

3.3.2 Models

Models are the section of the API that determines what type of data can be stored in the system and how they can be stored. The model section helps to filter the data collected

from user forms and store in the database. The models also contain functions that can be used to store data and retrieve data based on certain specific queries and also can be used for verification as well.

```
var UserSchema = mongoose.Schema({
  name: {
    type: String,
    required: true
  },
  email: {
    type: String,
    required: true
  },
  contact: {
    type: String,
    required: true
  },
  username: {
    type: String,
    required: true
  },
  password: {
    type: String,
    required: true
  },
  status: {
    type: String,
    required: true
  },
  verified: {
    type: Boolean,
    required: true,
    default: false
  }
});
```

Listing 2. UserSchema from models folder

Listing 2 shows code for a typical schema for a user entry. The schema describes what type of data should go to which field and is it a required attribute or not. The schema also offers an additional option to give a default selection if there is no data provided whatsoever.

3.3.3 Routes

The routes section of the backend application is used to communicate with the frontend with the user request and responses, taking the request from user and respond with a specific route associated with it. The routes section is the API of the system, it gives the response coded in the application when the user tried to access the specific route. For instance, there is a route in the routes section called '/register'. When the user triggers this route from the frontend view when he/she is trying to register for the web application, since it is a post route the data provided from the form template is first checked if correct type of information is provided or not. Then, the route is configured to save the data in the database. The routes section is used to access the database of the system which is predetermined to what and which aspects can be accessed by who, which in addition provides some security as well.

3.4 The frontend

The frontend, also known as the design layer or display layer of the application, was created using one of the most popular Javascript framework known as AngularJS. The newest version of Angular known as Angular2 was used. Angular2 is build using a modular structure, stacking bricks on top of bricks to create a house. These bricks are known as components. The subsections of this topic discusses about all the components and other features of angular that are used and how they are used.

Angular2 is a Javascript framework with a large scale of applications and it has different packages for different features and purposes. Thus, different features are bundled to different packages and only the packages that are necessary are to be included and thus

used. This makes angular very powerful because despite being so vast it can retain its size depending on the type of application and the features that are needed. Since, it also uses packages NPM helps on this section as well and there is also a package.json file that keeps in check all the dependencies and the information that are needed.

```
"dependencies": {
    "@angular/animations": "^4.0.1",
    "@angular/common": "^4.0.1",
    "@angular/compiler": "^4.0.1",
    "@angular/compiler-cli": "^4.0.1",
    "@angular/core": "^4.0.1",
    "@angular/forms": "^4.0.1",
    "@angular/http": "^4.0.1",
    "@angular/platform-browser": "^4.0.1",
    "@angular/platform-browser-dynamic": "^4.0.1",
    "@angular/platform-server": "^4.0.1",
    "@angular/router": "^4.0.1",
    "angular2-flash-messages": "^1.0.5",
    "angular2-jwt": "^0.2.0",
    "angular2-moment": "^1.3.3",
    "core-js": "^2.4.1",
    "ng2-datepicker": "^1.8.3",
    "ng2-order-pipe": "^0.1.3",
    "ng2-pagination": "^2.0.1",
    "ng2-quill-editor": "^2.0.0",
    "rxjs": "^5.0.1",
    "ts-helpers": "^1.1.1",
    "typescript": "^2.2.2",
    "zone.js": "^0.7.2"
  },
  "devDependencies": {
    "@angular/cli": "1.0.0-beta.31",
    "@angular/compiler-cli": "^2.4.0",
    "@types/jasmine": "2.5.38",
    "@types/node": "^6.0.42",
    "codelyzer": "~2.0.0-beta.1",
```

```
    "jasmine-core": "2.5.2",
    "jasmine-spec-reporter": "2.5.0",
    "karma": "1.2.0",
    "karma-chrome-launcher": "^2.0.0",
    "karma-cli": "^1.0.1",
    "karma-jasmine": "^1.0.2",
    "karma-coverage-istanbul-reporter": "^0.2.0",
    "protractor": "~5.1.0",
    "ts-node": "1.2.1",
    "tslint": "^4.3.0",
    "typescript": "~2.0.0"
  }
```

Listing 3. Dependencies for the angular2 app

Listing 3 show the different dependencies or the packages that are required for the application to perform smoothly. Most of the dependencies are built in when creating the app through the angular-cli since those are the core elements of the application. But other dependencies are added as we need more feature and functionality in our application.

For instance, the “ng2-pagination” dependency is required to auto create a pagination for the listing of the jobs that are displayed in the application. “ng2-quill-editor” dependency allows to add a powerful text editor with heavy features in the application with just a few lines of code. In Listing 3, there are two different set of dependencies namely “dependencies” and “devDependencies”. While both of them are necessary, the latter is only used during the development phase of the application.

Angular2 used uses a superset of Javascript known as typescript. In Typescript, coding is easier and powerful features can be implemented using simple code. This makes coding in typescript much easier and more powerful. But, the browser cannot understand typescript. So, typescript code needs to be compiled into plain Javascript code during rendering.

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

Listing 4. Code written in Typescript

```
/* tslint:disable:no-unused-variable */
import { async, ComponentFixture, TestBed } from '@an-
gular/core/testing';
import { By } from '@angular/platform-browser';
import { DebugElement } from '@angular/core';

import { HomeComponent } from './home.component';

describe('HomeComponent', () => {
  let component: HomeComponent;
  let fixture: ComponentFixture<HomeComponent>;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ HomeComponent ]
    })
    .compileComponents();
```

```
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(HomeComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

Listing 5. Code written in Javascript

Listing 4 and Listing 5 illustrate the difference in using typescript over plain javascript. Listing 5 has 20 lines of code whereas to perform the same task the code from Listing 4 was used which is a mere 7 lines of code. This demonstrates how write less do more strategy works great with typescript.

3.4.1 Module

The module of the application is the app.module.ts file that is used to bootstrap the application. In other words, make the application run. In this file, all the typescript files such as the components, services, guards needs to be added to the file with its location. All the files need to be addressed because this gives access to the files in the application and any file at any time could access other files through this main module. Module is a structure that holds all the angular pieces together.

Angular is mainly used to create single page applications. This means that the application has multiple pages but acts as a single page since all the other pages are dynamically added to a single page. This helps in many things. One is that it becomes a light application since it doesn’t have to load every time a new page is visited. Other is that the links become easier to manage since all the link redirect back to the same location.

To create this single page links angular uses something called routes giving each page a specific route to trigger and associating a specific route with a specific component. This route function can also be added in the module file.

3.4.2 Components

Since components are the building block of the application, each piece of the design has its own component. There is one component for each one of the pages of the application and also two or more components combined for a single page of the application. There are components such as Home component, register component, login component, reset component.

Components can be easily created through the angular-cli using the command

```
ng generate component [name]
```

Using the generate command automatically generates the components and its required files and also adds it to the module as well. The generate command generates a type-script file, a HTML file, a CSS file and a javascript file which is automatically compiled and run. The logic of the component is written in the typescript file and the HTML and CSS is written in their respective file.

Angular uses a MV* architecture as opposed to the normally used MVC architecture. The model, logic layer and View, display layer are both connected via the controller but in the MV* architecture the model and view is automatically combined which saves a lot of code and the code becomes much more less and simple. In angular2, the HTML file is dis-played to the user and content are dynamically added or removed in the view using the model or the typescript file which is compiled to javascript in runtime.

For instance, in the register component there are 4 files namely register.component.ts, register.component.spec.ts, register.component.html, register.component.css. The names follow the angular naming convention since it is created using the angular-cli. The register.component.html file contains the new user register form where user can register by inputting the required information. Then the register.component.ts file handles the data that comes from the form of the html file and sends it to the backend route for further processing and saving to the database.

3.4.3 Services

Services in this angular2 application are the files that are the bridge between the frontend and the API. Writing huge lines of code is cumbersome task and if some error occurs it is even harder to debug. Moreover, the code that needs to be used many times should not be written over and over again. Code repeat is a huge red flag in a web application since it takes a lot of time to write and manage.

Thus, services are used. Services can be imported in the module and can be used by any component. Services in this application are mostly used to communicate with the server or to check data.

```
//user authentication
  authenticateUser(user){
    let headers = new Headers();
    headers.append('content-type',            'applica-
tion/json');
    return   this.http.post(this.server+'/users/authen-
ticate', user, {headers: headers})
      .map(res => res.json());
  }
```

Listing 6. Code from authentication.service.ts

Listing 6 shows how a typical service is coded. Services have different function written mainly to communicate with the server. As shown on Listing 6, the code is used to authenticate a user who has just tried to login by sending the information with the required types and headers to the API '/users/authenticate'.

3.4.4 Guards

Guards in angular2 is used to prohibit access to certain content of the application without authentication. Guards are very useful for preventing unauthorized access. This is done by adding guards to the components that needs to be protected. For instance, a user

who has not been registered or does not have the login credentials that match with the database cannot access the dashboard of the application.

```
import {Injectable} from '@angular/core';
import {Router, CanActivate} from '@angular/router';

import {AuthenticationService} from '../services/au-
thentication.service';

@Injectable()
export class AuthGuard implements CanActivate{
    constructor(private authService: Authentication-
Service, private router: Router) {}

    canActivate(){
        if(this.authService.loggedIn()){
            return true;
        } else {
            this.router.navigate(['']);
            return false;
        }
    }
}
```

Listing 7. Code from auth.guard.ts

Listing 7 illustrates the code of a typical guard component. These files have a injectable decorator since they need to be injected to another file to have access or work. The code in listing 7 creates a AuthGuard class that implements the CanActivate feature of angular which lets the incoming request to navigate to that section if the user is authenticated, otherwise the access is denied.

4 Result

After starting the project from scratch, the application was finally developed. After development, the application needs to be tested for different errors and bugs in the system. This section discusses how to application works, what errors occurred and how the errors were solved. During the development phase, the node application and angular application both run in their own server so a package called cors was used to connect between these two servers. In the testing phase, the angular code is embedded in the node application so the need for a mediator disappeared. The application was uploaded to heroku after testing.

4.1 Testing

After typing the web address, the homepage of the application is displayed which has the option to either login or register. First, one must register before they can access their account. After clicking the register button, it directs to a new page that displays a form. The user has to enter all the required information in the form and an option in the form gave the user two user options, first option to create account as a student and second option to create account as an employer. Two different user groups have two different dashboards. After submitting the form, the application checks for already used usernames and emails and if both are unique then an account is created for the user. Still, the account is not activated. To confirm the user hasn’t entered a fake email, an email is sent to the users email address with a verification link to verify his account and after that the user can login.

Having two user groups, two use cases occur and the application is created to handle both the use cases.

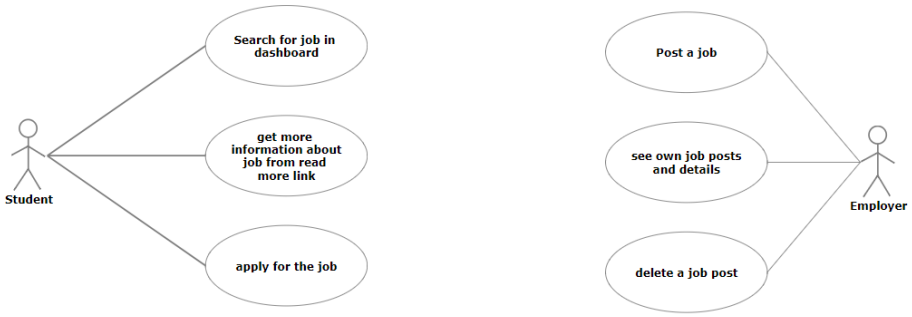

Figure 4 – use case diagram for marketplace application

The use case of the student group includes searching for a job in the portal, getting more information and applying for the job whereas the employer group can post a job in the system that the students can view, look at all of their posts and even delete them if wanted.

Since different user groups need to have different pages displayed, this section is handled by the dashboard component which is further divided into the student component and the employer component.

```
<div *ngIf="user">
  <div *ngIf="user.status == 'student'">
    <app-student></app-student>
  </div>
  <div *ngIf="user.status == 'employer'">
    <app-employer [employer_id]="user.id"></app-em-
ployer>
  </div>
</div>
```

Listing 8. Code to differentiate user groups

Listing 8 illustrates how to dashboard is created to display pages targeted to specific user groups. “ngIf” which is a if statement in angular2 determines the status field of the

user coming from the server and if the status of the user is student, it directs it to the student page and if the status is employer, it is directed to the employer page.

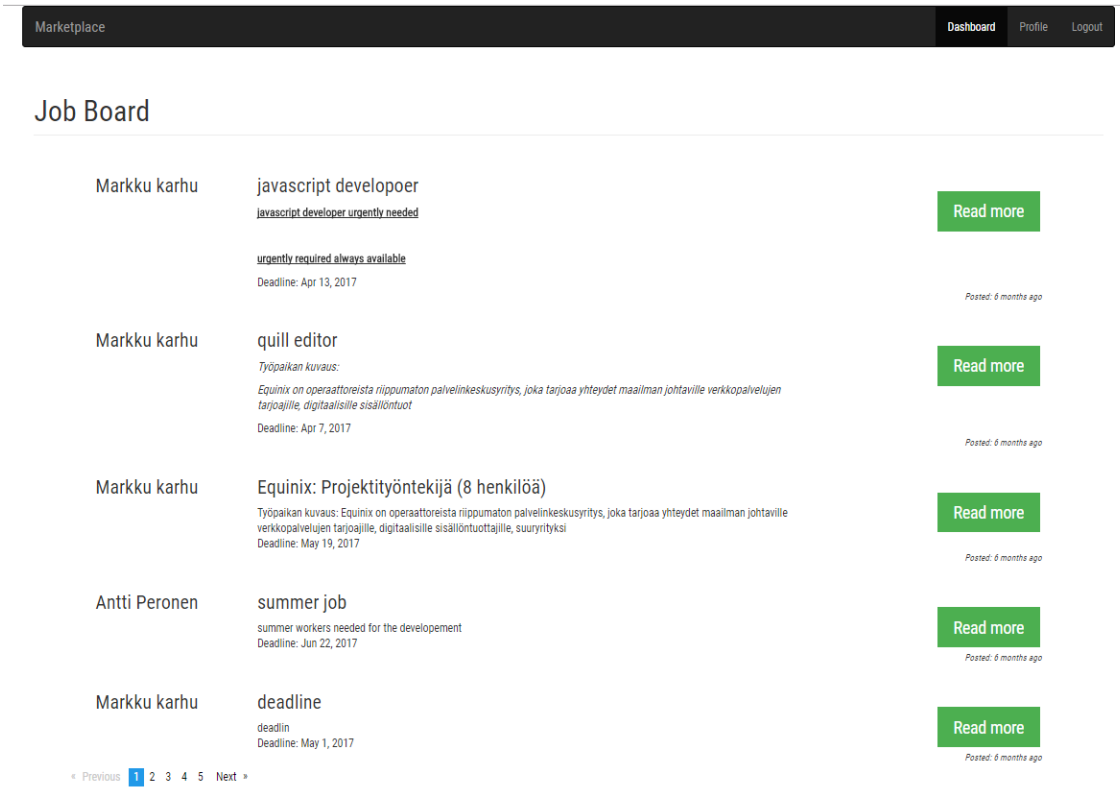

Figure 5 – Job listing from the marketplace application

Figure 5 displays the page as seen by a student who has logged in. the student can see all the jobs that has been posted, click read more to get more information about the job such as the requirement, the deadline and more. And apply for the job if the user is interested.

The page for the employer user group is totally different. The user can add a new job post to the system and view all of the user's own post.

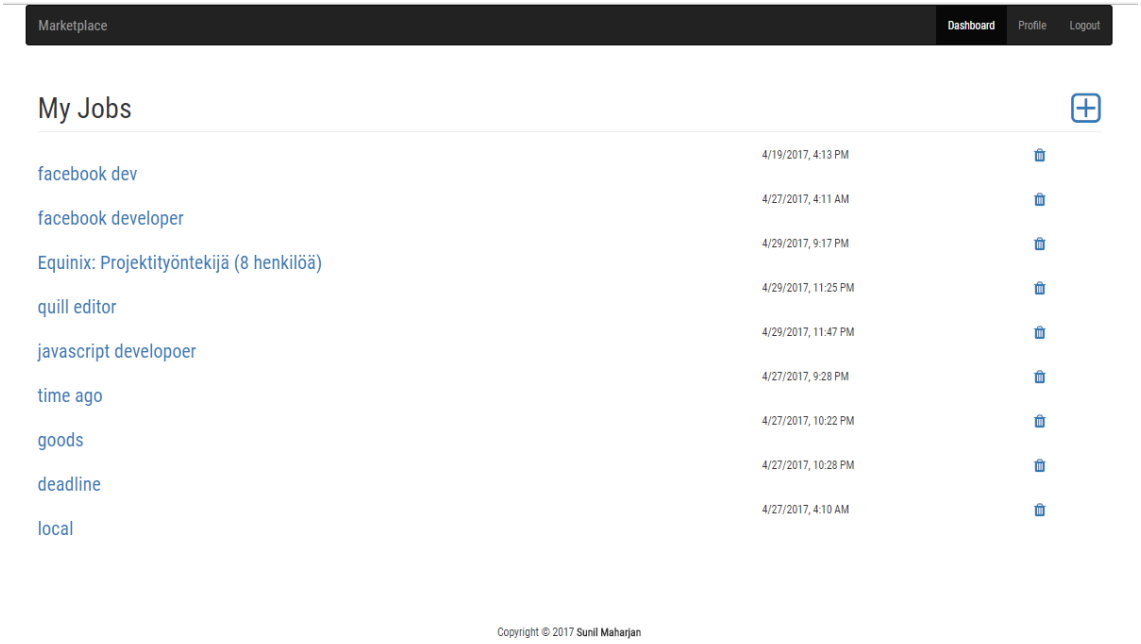

Figure 6 – List of posted jobs in the marketplace application

Figure 6 shows the page displayed in the employer group. The page has listed all the jobs that the user has posted with an option to delete each one of them. Also on the top right corner aligned with the my jobs text, there is a plus sign that directs to the add job link when clicked.

Also, the guards were tested for unauthorized access. Due to use of token based authentication and using passportJS for authentication, the security of the application has been really good. Without logging in, any user would not be able to access either the student or the employer page of the application.

Apart from the user verification system, there was also an added feature in the application which is also very important as well. The feature that helps user reset their password in case they forget it back to their email. The user goes to the application and uses the forgot password link and asks for the email that was used during registering. An email is sent to that address giving them access to change their password through a link. The link expires in a few hours to ensure security. This feature gives the user the convenience to access own account in case the user forgets the password to his account.

There were also few problems due to the email sending service for verification as well as password reset. There were many SMTP service providers but each one had its own problems. Some were paid providers, some were not been able to set up and some were pretty much useless. The one used on this application is known as Nodemailer which is a package in NodeJS as well. It was tried and not working before during development. Later during testing phase, the school's email address was used and the host and port were changed and it worked correctly.

```
let transporter = nodemailer.createTransport({
        host: 'smtp.metropolia.fi', //smtp service
        port: 465,
        secure: true,
        auth: {
            user: 'sunilma@metropolia.fi',  //user
            pass: '**********'             // password
        }
    });
```

Listing 9. Code for email service prodiver initialization

Listing 9 shows the initialization of the email service provider using the nodemailer package in NodeJS. The host smtp server and the valid port is provided to make it work correctly. The auth field used to give a authentic user valid on that server whose account is used to send emails.

During the testing phase, there were few errors that were encountered. The details of the errors and its solution are discussed in the next section.

4.2 Errors and Solution

There were a few errors that were encountered during testing. One of the first errors was with browser compatibility issues. The application was solely developed and tested in Google Chrome web browser. Google chrome web browser is the most popular choice as a browser for most users as well as testing platform for most developers as well. There are different features added to google chrome such as the inspection mode, developer options and most features already added that makes development easier.

Google chrome is also constantly updating itself and more so often than other browsers in the market.

The good features were overestimated while developing the application. While posting the job as an employer, there is a input field named deadline which is of date type. This date type was easily processed and there was a date type input field during testing in the chrome browser. But when it was tested in Mozilla firefox browser, this resulted in a blank input field. Many different options were tried. The solution was to use a third party plugin that was tried and worked across all browsers.

Another hindrance encountered was when there was issues with the user verification and the password reset feature. Both of these features were relied on the same data field of the user information named id. Since id field was already being used by the verification section, there was need for a different form of identification for the password reset feature.

The problem was solved by using a different schema named token which would create a unique token for a specific user when a password reset request is made and associate that token to that particular account and use it as identification. Thus having two different forms of identification for two different features, it worked like a charm.

5 Foglab

Foglab is a different section of Metropolia's official website. The link to the website is http://fog.metropolia.fi/ and it contains information about IoT. To describe the website, it is a single page website meaning all its content are in the same page and the links are just anchors to different sections. The first section is the landing page, after it comes the information about IoT page. Then there is a news section and after that contact information are given. The foglab webpage had been made using the wordpress plugin and all its content were written in wordpress.

For the project, the foglab website had to be updated and the marketplace application had to be linked to the website. The update in the website included changing the Fog laboratorio page, change the contacts page, link the marketplace web application to the

website and add two language features in the website namely English and Finnish, which previously was only available in Finnish.

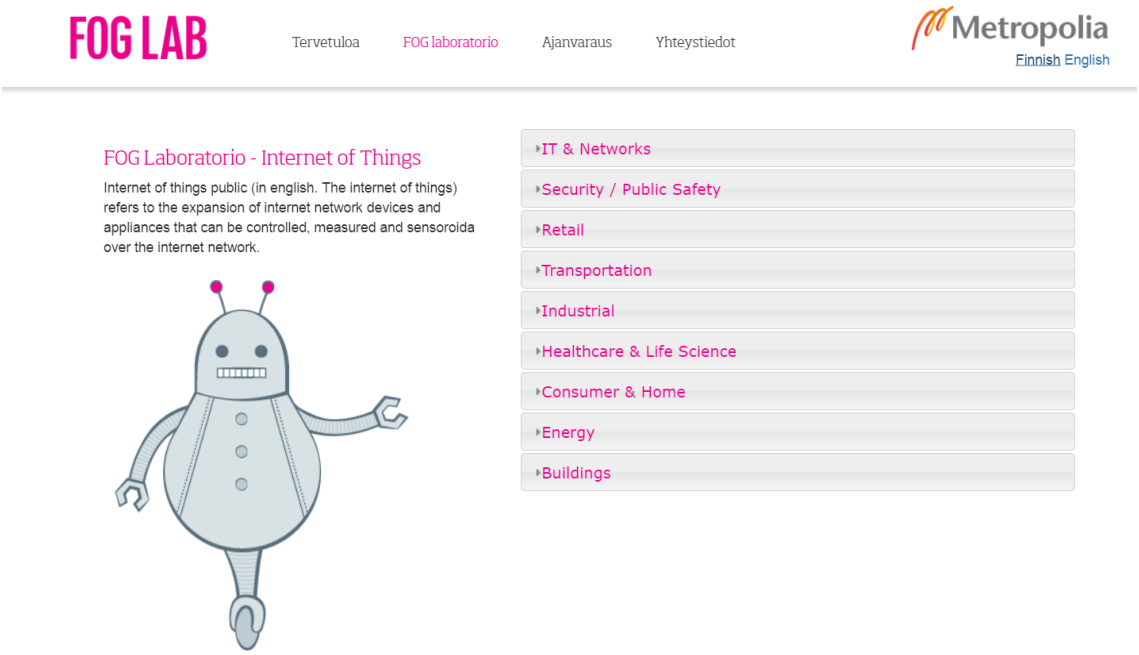

Figure 7 – IoT page of foglab metropolia

Figure 7 illustrates the design and layout of the foglab page after it has been updated. Figure 7 gives the description of the IoT page that was update. The links on the right side are clickable and after clicking it slides down the information on that particular section and clicking again reverses the action. This feature was achieved by using the JqueryUI plugin. JqueryUI is a added feature that has to be downloaded separately that can be easily used to add functionality to a particular section. This particular feature used is known as the accordion in jqueryUI.

Since the webpage was created in wordpress and the new sections was an integral part of it. It was recommended to update the website using wordpress to keep the new section intact. The contact page was updated with the new maps feature provided by google that pin points the exact location of the Metropolia campus located in Leppävaara, Espoo.

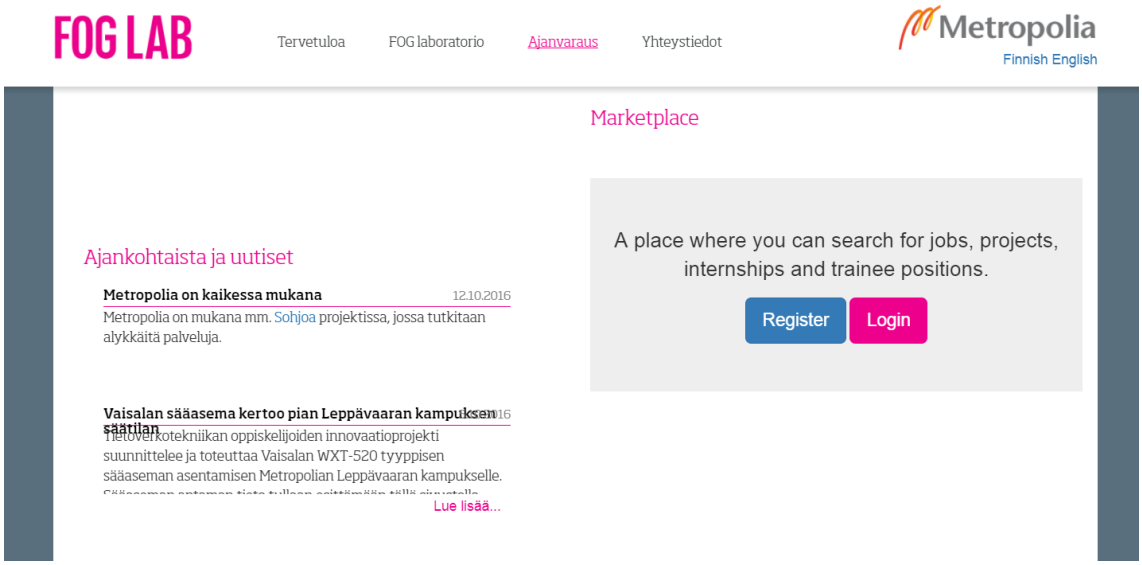

Figure 8 – Marketplace linked to the foglab website

Figure 8 shows how the marketplace web application was added to the foglab website. The news section was divided into two sections. One which would displays the current news and the other one had links related to the marketplace application. Clicking on either login or register link would redirect the user to the marketplace web application where the user could use all the features of the application.

The feature to translate the webpage was quite challenging to do in the wordpress. It was achieved by using two different classes for different languages and the classes would be displayed only when their specific link was active. For instance, if the finnish link active, the content in the finnish class would be displayed and the other would be hidden and vice versa. This process made the translation process easier and even very fast because it would not load another page to translate the content, rather it would directly translate using the same content which is very much faster.

6 Conclusion

There were two different goals of the project. The first one was to create a job portal web application that could be used to be a medium between the employers and students to exchange information about new jobs and its requirements. The other one was to redesign the foglab website to embed the previous application to this website. Both of the goals were achieved successfully. The web applications API was developed and the application was fully tested. The same API could be used again in different areas where the same information can be used. The application was also linked to the FogLab website of Metropolia.

With the increase in interest and the huge growth in the Internet of Things and being a student of Information Technology, it is necessary to give information about the internet and its applications to new students. The web is rapidly increasing and developing creating vast application in every field available. Everything now can and is done in the web. Banks, hospitals, companies all have their own website and web applications for its customers. Web applications are getting more popular and more useful and its complexing is increasing every day. Thus creating a huge demand for web developers that can fulfil the growing need.

Thus creating these types of simple applications helps to get knowledge and experience in the web application development world which benefit for future endeavours where one show his work and impress the employers. The demand for Angular and node developers are high so increasing your skills in this particular field is very likely to land a good job and also create powerful applications and make the life to many people a bit easier.

References

[1] Korolova, Luliia. (2016). Frontend vs Backend web development. https://www.aog.jobs/blog/frontend-vs-backend-web-development-whom-do-you-need-for-your-project

[2] Ribeiro, Joao. (2017). The Front-end Web Developer Spectrum. http://joaoperibeiro.com/the-front-end-developer-spectrum/

[3] Developer survey results. (2017). https://insights.stackoverflow.com/survey/2017?utm_source=so-owned&utm_medium=social&utm_campaign=dev-survey-2017&utm_content=social-share

[4] AngularJS. Angular 1 and angular 2 integration. http://Angularjs.blogspot.fi/2015/08/Angular-1-and-Angular-2- coexistence.html

.[5] Wikipedia, The free encyclopedia. NodeJS. https://en.wikipedia.org/wiki/Node.js

[6] Technopedia. MongoDB. https://www.techopedia.com/definition/30340/mongodb

[7] Bhardwaj, Rachit. (2017). Basic building blocks of Angular 2 Architecture. https://dzone.com/articles/components-of-angular2-architecture

[8] what is angular?. Angular 2 Documentation. https://angular.io/docs

[9] Nwamba, Chris. (2015). Seamless ways to upgrade Angular 1 to Angular 2. https://scotch.io/tutorials/seamless-ways-to-upgrade-angular-1-x-to-angular-2

[10] Simons, Eric. (2017). A better way to learn Angular2. https://thinkster.io/tutorials/learn-angular-2

[11] Nemeth, Gergely. (2017). What's new in NodeJS 8.5?. https://blog.risingstack.com/whats-new-in-node-js-8-5/

[12] Weiss, Manuel. (2017). The absolute beginners guide to NodeJS. https://blog.codeship.com/node-js-tutorial/

Title of the Appendix

Content of the appendix is placed here.