Summer Internship Report

# FULL STACK WEB DEVELOPMENT INTERNSHIP
# FINAL REPORT

Completed By:Amala Peter
Supervised By:Asst.Prof.Albert Sunny,IIT Palakkad

TABLE OF CONTENTS:



## Introduction

Motivation

This project aimed at implementing a full stack web development using Technologies like
HTML,CSS,JavaScript,Mongodb,Node.js.Full stack development refers to the development of
both front end(client side) and back end(server side) portions of web application.

Front End Development : involves the actual presentation of your website, how the information
in your website is laid out in browsers and on mobile devices as well.It is done by languages like
HTML,CSS as well as scripting Language,Javascript.The main goal of a front end is to provide
the platform for visitors to interact with, a platform which provides and receives information. It
deals with everything that we actually see on a website, the layout, the positioning of text and
images, colors, fonts, buttons, and so on.

Back End Development :Creation, edit/update and recollection of data are some of the
processes that are most often associated with back-end development. Some examples of
common scripting languages used are PHP, Ruby, and Python.We need to write code to
receive the information input from the user and also save it in a database. There are two main
types of databases: relational (like PostgreSQL and MySQL) and non-relational management
systems (like Mongo). The language used for database management is SQL, which helps the
developer interact with the database.Another component of back-end development is server
management, which are applications that host the database and serve up the website.

Full Stack Development :In this phase we need to get focused on front-end, back-end,
frameworks, as well as server, network and hosting environments. So the full stack of
Development.

Using front-end JavaScript libraries, and a JSONbased database puts JavaScript on all parts of the stack, increasing efficiency, scalability, speed, and resource re-usability. This stack is dubbed full-stack JavaScript.

Throughout the development, I explored various solutions to build my full-stack JavaScript application, and I modified it to solve the problems I faced. The sole motivation was to learn and test different open source APIs and to improve my personal skills in web development. Because of this shift in the focus of the project. The report will focus more on the process of development than the final product.

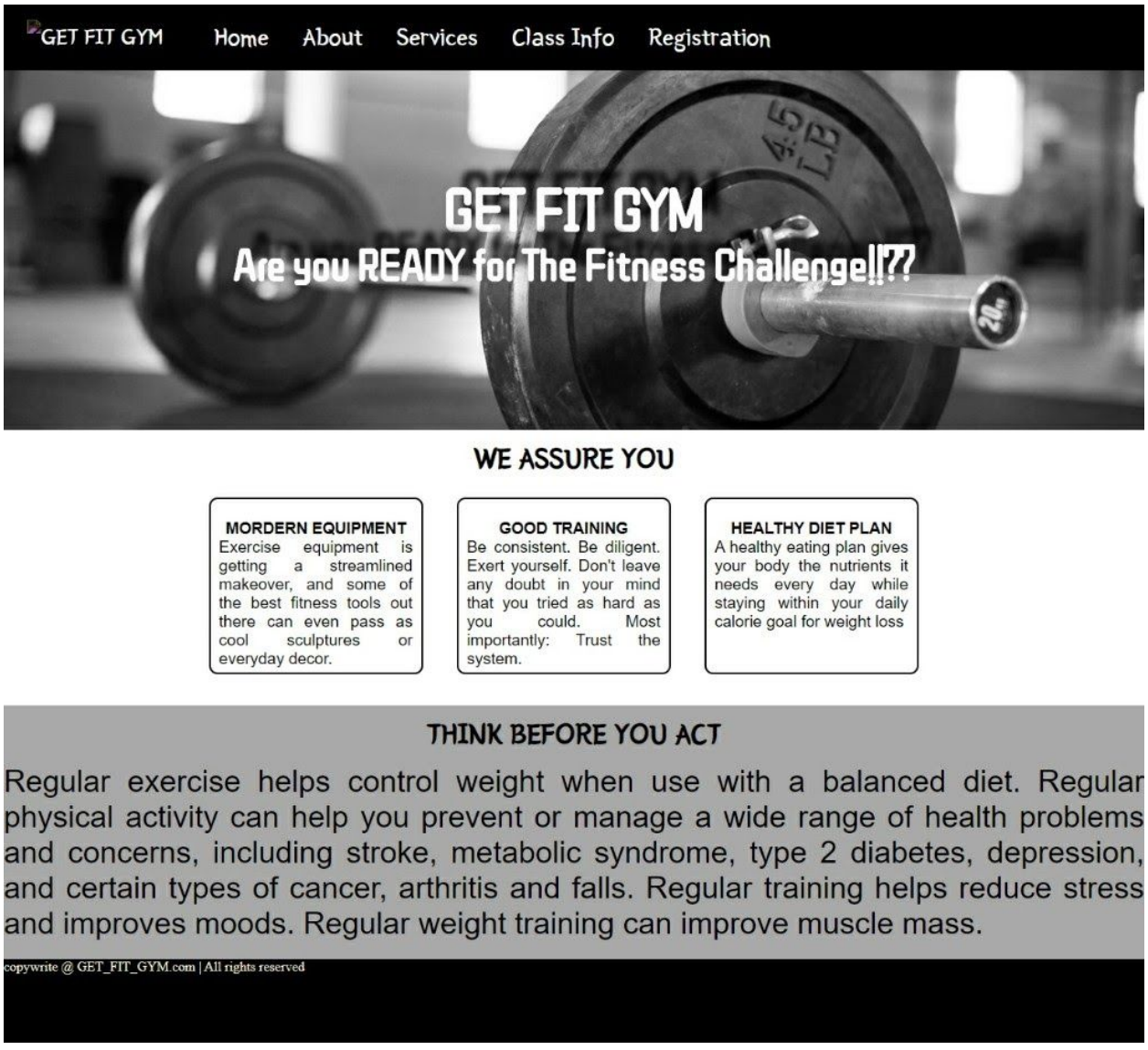
Deliverables

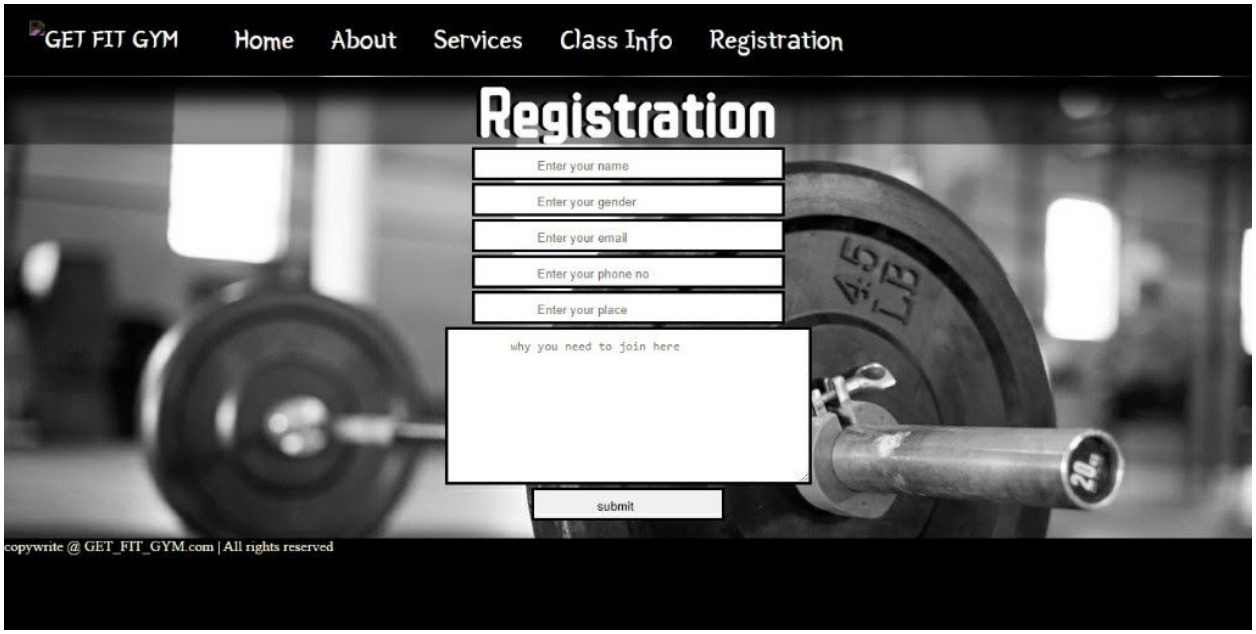
A web application that is fast and scalable and built in a multi-page application format. which provides the users with the ability to go for  a gym website.
A full-stack JavaScript application with a server built with Node.js using the Express.js web framework, and a secure Database that holds all user data built with MongoDB, and a responsive, single-page front-end developed using React. The server is hosted on DigitalOcean and the database on Mongodb thus the users can access the application through the ip address.

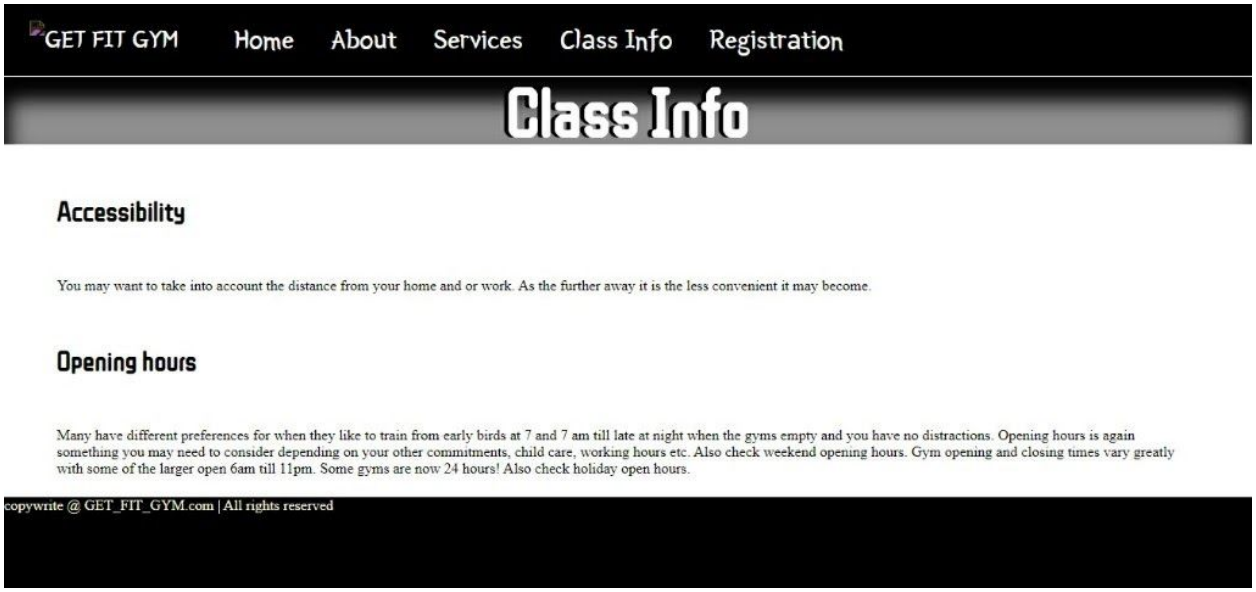
Features

This website allows the users to interact with a gym site and undergo registration for the new users.

- Multi Page Application
  As seen in the picture below, the application follows the Multi-page pattern, under which further reloading occurs.




GET FIT GYM
Home
About
Services
Class Info
Registration
Registration
Enter your name
Enter your gender
Enter your email
Enter your phone no
Enter your place
why you need to join here
submit
copywrite @ GET_FIT_GYM.com | All rights reserved


GET FIT GYM
Home
About
Services
Class Info
Registration
Class Info
Accessibility
You may want to take into account the distance from your home and or work. As the further away it is the less convenient it may become.
Opening hours
Many have different preferences for when they like to train from early birds at 7 and 7 am till late at night when the gyms empty and you have no distractions. Opening hours is again something you may need to consider depending on your other commitments, child care, working hours etc. Also check weekend opening hours. Gym opening and closing times vary greatly with some of the larger open 6am till 11pm. Some gyms are now 24 hours! Also check holiday open hours.
copywrite @ GET_FIT_GYM.com | All rights reserved

# Services

## What can I expect?

Free weights

Gyms have a main work out area which tends to be divided into a free weights section including dumbbells, bar bells. They may also have exercise machines with free weights or more commonly exercise machines will have their own section.

## Cardiovascular

The Cardiovascular area with have machines including treadmills (running machines), rowing machines, exercise bikes (stationary).

## Classes

Classes are group sessions and separate room from the cardio and free weights. Examples of group class exercise include aerobics, Spin (cycling on a stationary bike), boxercise, high intensity training (HIT) and yoga.

Staff members tend to be qualified personal trainers or the club will tend to be able to put you in contact with a personal trainer who will devise a training plan and dietary advice to achieve your goals.

## Facilities

Facilities can vary greatly from gym to gym. Not just on a broad range of cardiovascular equipment including:

Treadmills, Rowers, stationary bikes, cross trainers, versa climbers (step machines), steppers, and rowing machines.

## Resistance training equipment including:

Fixed weight machines, dumbbells, barbells and weight plates.

Also clean toilets, changing and shower rooms and exercise mats, stability balls and space to exercise.

When choosing your gym you should check there is a number of the equipment you feel you will use more. If you are going to focus on free weight training (dumbbells etc) check they have multiple sets. On weight machines check they have setting from minimal weight to maximum weight to suit your capabilities. Also check have they been maintained and in good condition. If you are going to mainly use of cardio machines again check they have been well maintained, check they have adequate number of machines compared to gym members.

Larger gyms also may have studio classes, sports and injury therapy. Sports shop, juice bars, cafe and restaurant. Swimming pool, sauna, Jacuzzi, steam rooms. Tennis, squash and badminton courts. Some gyms also offer child care facilities.

# About

## What is a gym?

A gym - physical exercises and activities performed inside, often using equipment, especially when done as a subject at school. Gymnasium is a large room with equipment for exercising the body and increasing strength or a club where you can go to exercise and keep fit.

A gym is a gymnasium, also known as health club and fitness centre. Gymnasiums have moved away just being a location for gymnastics. Where they had gymnastics apparatus such as bar bells, parallel bars, jumping boards and running path etc.

## Choosing the right gym for you

Most gyms have a wide range of ages and levels of fitness. Don't buy into the preconception that it will be full of supreme athletes!

## Personal training

Staff members tend to be qualified personal trainers or the club will tend to be able to put you in contact with a personal trainer who will devise a training plan and dietary advice to achieve your goals.

## Staff

Members of staff do tend to be fit and healthy, smartly dressed in corporate clothing. They should be happy to explain and demonstrate whenever necessary. They should have knowledge across a wide range of exercise disciplines and many are also personal trainers. When you are on you initial visit look out for if the staff are helping and advising all members or only willing to advise clients who have paid for additional personal training session. You should check on staff qualifications and also view feedback from previous clients. Some memberships come with a specific training schedule designed to achieve your goals.

Hypothesis

JavaScript

JavaScript originally named Mocha is a “multi-paradigm language, supporting object oriented, imperative, and functional programming styles.”JavaScript started on web browsers as a language to dynamically interact with the user and control the components of the page. JavaScript’s language architecture is quite unique from other languages which is especially handy when developing web applications that rely on non-blocking operations. JavaScript architecture combines object-oriented, functional and scripting languages paradigms. Syntax wise, JavaScript is based on C’s structured layout, with subroutines, block structures and loops. While it is object-oriented, JavaScript does not have classes, but instead rely on object prototypes for inheritance. This means that the methods and fields of the objects’ prototypes may be dynamically changed for future construction of these objects. Functions in JavaScript are also objects, and can be sent as arguments to other functions as is the case in functional languages; functions can also have methods and properties of their own.
JavaScript also provides dynamic typing as well as static typing. Variables could be defined with a specific type such as Number of String, or casted dynamically by initializing it as a var. Moreover, JavaScript, like scripting languages, allows the use of variadic functions (number of arguments is not defined), and supports Regular expressions such as in Perl. Finally, JavaScript’s support for associative arrays is the basis for the construction of JSON data formats. Using a single language throughout the stack enables the reuse of resources such as JSON objects which can be manipulated the same way by any part of the stack.
There were several attempts aiming to place JavaScript on the server side ever since the mid-1990s. But, none of these solutions were as successful as Node.js which is pioneering a new way of server programming focused mainly around asynchronous operations. There are three main reasons that made JavaScript the language of choice of Node.js.
1. Google’s V8 is an open-sourced high-performance JavaScript execution engine built for Chrome. It compiles JavaScript to native machine code instead of interpreting it in real time. V8 and other JavaScript runtime engines also provide a concurrency model using a message queue and an event-loop which allows JavaScript to stack operations and their callbacks and execute the callback when the operation is done.

2. JavaScript is built around an event-driven interaction model because it depends on user actions, which makes asynchronous, non-blocking and callbacks natural, as they are event driven as well.

3. JavaScript is an interpreted language which makes it platform independent.
4. JSON (JavaScript Object Notation). JavaScript has been the language of choice to control the web for a long time; and since the old days, data had to be marshalled into JSON objects when sent to the web. Because of the high dependence on JSON objects, a new kind of JSON-like database was created, enabling the easy exchange of data between back-end and front-end.

Node.JS

Node JS is a JavaScript platform for building fast, scalable, network applications built on Google's V8 Engine. Node is single threaded and built around the paradigm of non-blocking IO. With Node.js each incoming request by the user is handled by one single thread in opposition to the multi-threaded techniques used by PHP to scale the operations. Each request handled by this thread is coupled with a callback function that is called upon completion of the task. This is possible due to the fundamental support of JavaScript for events, Asynchronous operations, and callbacks; and Node.js puts JavaScript on the server side.
Node has several advantages, some of which are:

- RESTful API. As Node can build an HTTP server out of the box, it can communicate with all other components through HTTP methods for CRUD operations based on the RESTful paradigm.
- Single Threaded. Since it is not blocking I/O operations, Node can handle all user requests using a single thread, instead of allocation of new thread for each request, which has a large memory footprint. Nonetheless, Node does use a thread pool at the kernel level to guarantee that the operations are being executed asynchronously without blocking the event-loop. This is necessary because the kernel does not support all operations asynchronously.
- NPM: The Node Package Manager is based on JavaScript's npm. A built-in module that supports package management, it can be used to easily download and install modules for a Node application. Moreover, Node already has many packages and libraries developed to work on top of it; all of which confine to the asynchronous nature of Node.js.
- Non-blocking I/O. Because Node uses an Event-loop to allow asynchronous operations and callback functions, it is capable to setting the requesting function on the side till the task is completed and meanwhile handling other tasks.
- There is also an advantage for using JavaScript on the server because data structures and logic may be shared by both front-end and back-end which

increases the synergy of the developers' team as resources may be shared. Furthermore, using a unified data structure across the stack makes transfer easier and removes the overhead of data conversion and casting.

- Finally, Node is supported by a great community that is very vibrant and dedicated to constantly improve Node.js itself or develop new modules for it.

Nevertheless, Node.js is not problem free and has flaws that sometimes hinder production level websites. The most confusing part with Node.js for a newcomer, is to get adapted to the asynchronous programming. To assure that the single thread Node.js provides does not get delayed, it is necessary to guarantee that no blocking operations occur in the stack so that the thread does not get changed which would deter the whole performance of the server. Getting used to the Async paradigm is not easy though, and callbacks tend to get nested and hard to trace back.

There is also the issue of scaling on different CPUs which Node is incapable of performing due to its single thread; it can only run on a single CPU and cannot be scaled on big server pools in contrast to distributed programming possible with other languages. At the same time this limits Node’s capabilities. Nodes tend to perform very well with basic I/O operations which are blazingly fast with callbacks. But, since a single thread is used, all operations.

[callbacks], will be executed in the stack, on that single thread, which would harm the performance significantly if the task blocks for a relatively long time. This limit’s Node capabilities as a full server, and tends to do well only in simple websites like chat rooms and blogs. Node.js has faced many problems due to its immaturity at this stage, especially when using modules which are not tested in big production means that there is a big margin of error. This may cause in some cases memory leaks that would increase latency and CPU usage, and if it occurs while there are thousands of simultaneous requests, the server tends to crash and must be restarted. There are hundreds of different programs, frameworks, plugins, and other software developed for Node.js. The main programs my stack will need is a database, a web server framework for Node, and a front-end engine.

Express.js

Although Node is capable of independently acting as a web server, there are frameworks designed to make it more powerful and efficient -the most popular being

Express.js. It is fast, minimal, and has several applications built for it. Express was chosen for several reasons among them are:

- Minimalistic. Express makes it possible to build an HTTP server very easily by wrapping the backend code of Node.js, and it makes the building of a RESTful API very simple.
- Express supports middlewares which are functions called in a sequential order on a request or response. For example, a middleware I am using is body parser, which parses the body of incoming HTTP requests with a form submission. This allows me to access the parameters of the request directly. Middlewares can do anything, and they end with next(), which calls the next middleware.
- Routes are Express's way to handle incoming requests on a certain URL. Express after detecting the incoming requests directs it to a specific routing JavaScript file that handles all the logic associated with that URL. This is precisely helpful to keep the code lean and organized.
- Supports a wide range of available templates engines out of the box. Provides a range of extra helping features such as being able to redirect the user to another URL from within the server.
- A good community. Express is one of the original Node web frameworks, the community has developed very comprehensive tutorials, online examples, extensive documentation, and powerful tools for Express.

Express though, does not provide a proper MVC structure to the application, and models would have to be created using other libraries such as Mongoose which is being used in this project.

MongoDB

Additionally on the backend, there should be a database that stores user data. It would be preferable to use a JSON –NoSQL- based database to leverage the benefits of using JavaScript across the stack when the same objects stored in the DB can be processed by the server and the front-end without any additional conversion. I started my testing with a pioneer of NoSQL databases, and a member of the MEAN stack, MongoDB. Mongo is just the database software itself, and a driver is needed to connect nodes with the database instance and provide a layer for I/O with the database. Preferable also, there should be defined schemas for the database documents.

There are several advantages for using MongoDB such as:

- BSON (Binary JSON) storage types. JSON types used by JavaScript are very similar to xml in terms of its construction; instead of using a table, the whole

collection (MongoDB's equivalent of an SQL table) can be written as a nested list of fields and values.

- On its own, MongoDB is schema less, which provides the users with an easier way to append objects into the database. Certain documents within the collection may have different values from others. For example, a user can have as many phone numbers' fields in their document as they own without any need to change the other documents in the same collection.
- Indexing. This is a feature of MongoDB in which any field that is 'required' in a collection, if indexed, would be added to a sorted array with the values of this field in all documents. This way searching the collection can be made very quickly.
- Auto-sharding. In a production system with more than one server, MongoDB can split each collection across the different servers, thus enabling an easy way to scale horizontally.
- Location based data. MongoDB natively supports geo-spatial coordinates and data indexing accordingly.
- A strong, active community with many tutorials and examples available online.
- MongoLab. MongoLab is a cloud database hosting service for MongoDB providing up to 500MB of free storage and an online UI for monitoring and managing the database.

Mongoose enabled me to create schemas and models for my documents in the database, which I could use to verify data being inserted into the DB. The schema modeling includes "built-in type casting, validation, query building, business logic hooks and more, out of the box." It also provides abstraction to top level Mongo operations such as creation of documents and collections, connecting to a database, and defining indexes, requires fields in a document, and more. Another important feature of Mongoose is population, which is similar to joining in relational databases.

## Conclusion

Node.js is a JavaScript platform for building network applications. It is built around the concept of using asynchronous operations and non-blocking IO. It is fundamentally different from competitors such as PHP, which uses a multi-thread paradigm to scale its operations, since it uses a single thread to handle all incoming requests to the server, and relies on its asynchronous model to manage concurrency.

Another great opportunity Node.js provides is the ability to use JavaScript at both ends of the application; making resource usability between developers more applicable. Overall the experience was very fulfilling and I managed to grasp the important parts of Node.js and full-stack JavaScript applications.

I gained extensive knowledge in JavaScript . Node.js becomes very simple and the benefits of JavaScript over the full stack definitely improved and simplified the experience. Most importantly, I found myself capable of solving many of the big challenges using the different libraries available for Node.js and with the help of the community.

Moreover, I appreciated the opportunity to try the different new libraries built for Node.js that tackle many aspects of the development. Nonetheless, I also understand now the reasons behind the slow adaptation of Node.js in its current premature state for intensive applications. While Node.js does IO very efficiently, it lacks the ability of heavy processing because of its single thread functionality which limits Node.js possible applications.

Full-stack JavaScript development tool set is a great way for developing basic http application that relies mostly on message sending and basic IO. It is also a great tool for teaching because it relies on a single language, has great resources, and many great tools, all of which are completely free and open source.

Furthermore, JavaScript is a great language that is considered less difficult than other alternatives such as Java or Scala. In conclusion, I greatly recommend Node.js as a gateway to learning web development and getting engaged in the open-source community, and as a tool for rapid development of http applications.

The amount of tools and passion the community has put into the node ecosystem and web applications in general makes development straightforward and abstracts the complex deployment difficulties.

Bibliography

1. Crockford, Douglas. JavaScript: The World's Most Misunderstood Programming Language. http://www.crockford.com/javascript/javascript.html. Retrieved May 29th, 2015.

2. Init.js: A Guide to the Why and How of Full-Stack JavaScript. http://www.toptal.com/javascript/guide-to-full-stack-javascript-initjs. Retrieved May 29th, 2015.

3. An Introduction To Full-Stack JavaScript. http://www.smashingmagazine.com/2013/11/21/introduction-to-full-stack-javascript. Retrieved May 29th, 2015.

4. The MEAN Stack: MongoDB, ExpressJS, AngularJS and Node.js. http://blog.mongodb.org/post/49262866911/the-mean-stack-mongodb-expressjs-angularjs-and. Retrieved May 29th, 2015.

5. Learning React.js: Getting Started and Concepts .https://scotch.io/tutorials/learning-react-gettingstarted-and-concepts. Retrieved May 29th, 2015.

6. Understanding the Node.js event loop. https://nodesource.com/blog/understanding-the-nodejsevent-loop. Retrieved May 29th, 2015.

7. Comparing Node.js vs PHP performance. http://www.hostingadvice.com/blog/comparing-nodejs-vs-php-performance/. Retrieved May 29th, 2015.

8. Understanding Express. http://evanhahn.com/understanding-express/. Retrieved May 29th, 2015.

9. Object Modeling in Node.js with Mongoose .https://devcenter.heroku.com/articles/nodejsmongoose. Retrieved May 29th, 2015.

10. You're Doing Node.js Wrong! Avoid Synchronous Code. http://www.nodewiz.biz/your-doingnode-js-wrong-avoid-synchronous-code/. Retrieved May 29th, 2015.