Ranjan Yadav

Building a Blog Project using JavaScript, NodeJS and MongoDB

Metropolia University of Applied Sciences
Bachelor of Engineering
Information Technology
Bachelor’s Thesis
9 May 2021

Abstract

Author:	Ranjan Yadav
Title:	Building a blog project using JavaScript, NodeJS and MongoDB
Number of Pages:	27 pages
Date:	9 May 2021
Degree:	Bachelor of Engineering
Degree Programme:	Information Technology
Professional Major:	Software Engineering
Supervisors:	Janne Salonen, head of ICT department

Demand and popularity of full stack (frontend with backend) websites has been growing steadily with time. JavaScript language became the language for developing the front-end part of applications while to develop the backend part developers would need to use other languages like python, java, PHP etc. The invention of NodeJS changed this as it can execute the JavaScript code on the server. With JavaScript being popular for web development, the popularity of NodeJS also grew with it. To understand and learn Nodejs, a blog project will be developed using JavaScript, NodeJS, Express and MongoDB. Users will be able to write their blogs, store it with the help of a database and will be able to access and modify it as per their convenience.

Keywords: JavaScript, NodeJS, MongoDB

Contents

List of Abbreviations

HTML:	Hyper Text Markup Language
API:	Application Programming Interface
TC39:	Technical Committee 39
XML:	Extensive Markup language
HTTP:	Hyper Text Transfer Protocol
AJAX:	Asynchronous JavaScript and XML
JSON:	JavaScript Object Notation
I/O:	Input/Output
ECMA:	European Computer Manufacturer’s Association

1 Introduction

Developing an application which has both the front-end and back-end is known as full stack web development. The front-end refers to the presentation of the application user interface along with the interaction of the application with the user. The back end refers to the computation of user input as per the application demands while also interacting with databases. [1]First time the term “Full Stack Developer” was inquired on Google in 2010. [2] Full stack is a recent phenomenon which grew with the growth in Information Technology (IT) start-ups and Multinational Corporations (MNCs). These companies or start-ups required developers who could not only develop and manage either the front-end or back-end but also could interchange their roles as per the organization demands. [2] With the commercial success of online business and applications, the demand for full stack developers has also grown over the years. The blog application is going to have front-end build in JavaScript while the back end will be built in NodeJS with Express.js framework, and the database used will be MongoDB.

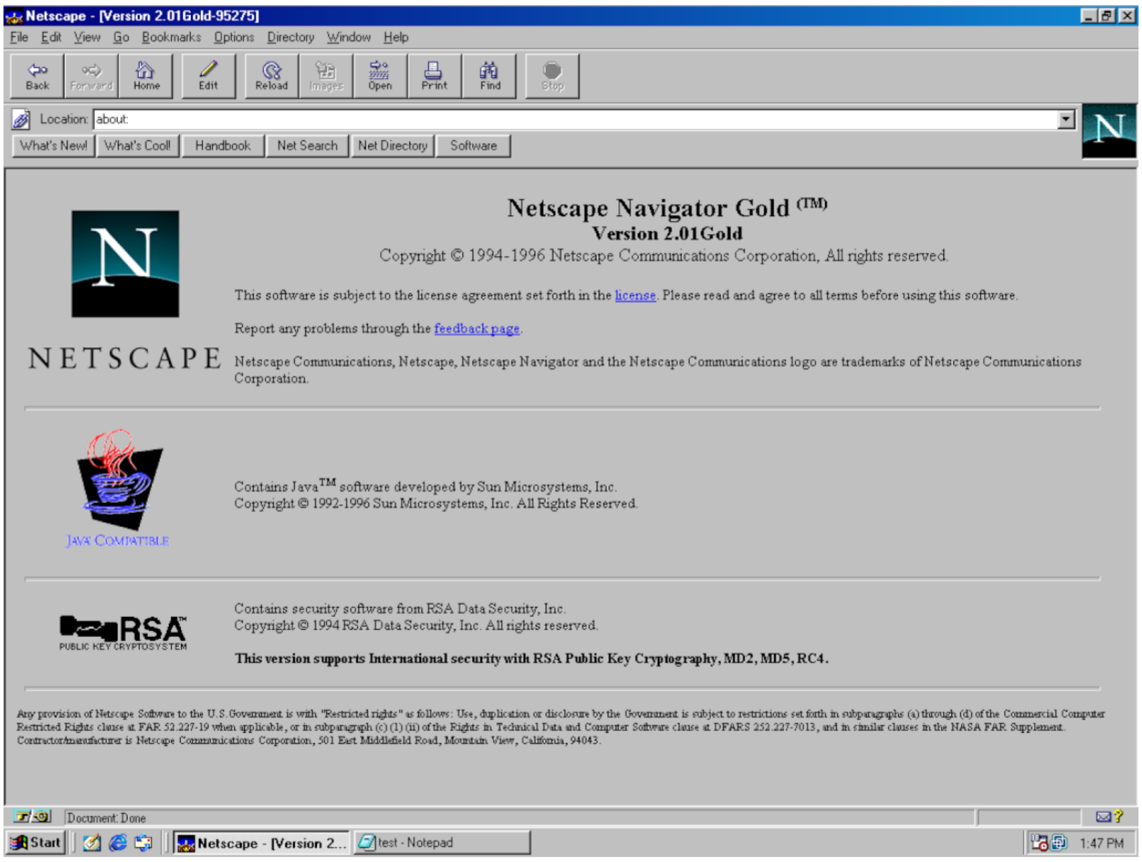
2 JavaScript

JavaScript is an interpreted programming language which is lightweight and has object-oriented abilities. [3] It is one of the most common programming languages used today and its demand has skyrocketed recently where more websites built today use JavaScript in some form. [4]

JavaScript brought changes in web pages where the web pages no longer had to be written in static Hyper Text Markup Language (HTML) instead web pages can interact with users and control its content dynamically. [3]

2.1 History of JavaScript

A Netscape programmer, Brendan Eich, developed a new scripting language in September of 1995 which was then called LiveScript. Later LiveScript went on to be called JavaScript. The invention of JavaScript was the outcome of browser competition between Netscape and Microsoft. [5]. JavaScript first was released for public in 1995 where it was integrated with Netscape Navigator 2.0 as shown in Figure 1.


Netscape - [Version 2.01Gold-95275]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop
Location: about:
What's New! What's Cool! Handbook Net Search Net Directory Software
NETSCAPE
JAVA COMPATIBLE
RSA PUBLIC KEY CRYPTOSYSTEM
Document: Done
Start Netscape - [Version 2... test - Notepad 1:47 PM

Netscape Navigator Gold (TM)
Version 2.01Gold
Copyright © 1994-1996 Netscape Communications Corporation, All rights reserved.

This software is subject to the license agreement set forth in the license. Please read and agree to all terms before using this software.

Report any problems through the feedback page.

Netscape Communications, Netscape, Netscape Navigator and the Netscape Communications logo are trademarks of Netscape Communications Corporation.

Contains Java™ software developed by Sun Microsystems, Inc.
Copyright © 1992-1996 Sun Microsystems, Inc. All Rights Reserved.

Contains security software from RSA Data Security, Inc.
Copyright © 1994 RSA Data Security, Inc. All rights reserved.

This version supports International security with RSA Public Key Cryptography, MD2, MD5, RC4.

Any provision of Netscape Software to the U.S.Government is with "Restricted rights" as follows: Use, duplication or disclosure by the Government is subject to restrictions set forth in subparagraphs (a) through (d) of the Commercial Computer Restricted Rights clause at FAR 52.227-19 when applicable, or in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, and in similar clauses in the NASA FAR Supplement. Contractor/manufacturer is Netscape Communications Corporation, 501 East Middlefield Road, Mountain View, California, 94043.

Figure 1:Netscape Navigator 2.0 (Copied from https://auth0.com/blog/a-brief-history-of-javascript/)

2.1.1 ECMAScript

The rapid growth of JavaScript created the need for standardizing and maintaining it. Netscape gave this task to the European Computer Manufacturers Association (ECMA) [6]. JavaScript went through changes and revisions over the years leading to oscillating popularity during its initial years.

2.1.2 ECMAScript 1

A committee was formed at ECMA which was named Technical Committee 39(TC39). TC39 was tasked to come up with standardization of JavaScript. The first standard version of JavaScript was called ECMA-262 and was released in 1997. ECMA-262 still lacked major features for web development.

2.1.3 ECMAScript 2

ECMA-262 was handed over to the International Standards Organization (ISO) as the final step for JavaScript standardisation process [7]. The second version of JavaScript was released in 1998 in which the differences between ECMA-262 and ISO were addressed and corrected. No additional language features were added in the second edition. The final accepted version was named ECMA-262, Second Edition [7]

2.1.4 ECMAScript 3

During the development of the first version of JavaScript, the TC39 committee discussed adding extra features to the language. Later, they agreed to postpone these features until the JavaScript language was standardised. [7] Once the ECMAScript 1.0 and ECMAScript 2.0 achieved standardisation, work on adding more features to the language commenced. The third edition brought major changes in the language which included added features such as regular expressions, do-while block, exceptions and try/catch blocks, more built-in functions for strings and arrays, formatting for numeric output, the ‘in’ and ‘instanceof’ operators and much better error handling [8]. The third version was specified as ECMA-262 third edition. This version of JavaScript became popular and was also supported by web browsers.

During the development of ECMAScript 3.0, Microsoft implemented a new browser API (Application Programming Interface) called XMLHTTP in its Outlook Web Access (OWA). XMLHTTP API allowed JavaScript to perform asynchronous Hyper Text Transfer protocol (HTTP) requests to the server. This enabled JavaScript web pages to send and transfer data from the server without reloading the same page frequently [7]. In 2006, Jesse James Garrett coined the term “AJAX (Asynchronous JavaScript and XML)” [7]. AJAX concept was used a lot by developers to develop web pages over the following years which led to its standalone standardisation as part of the Web Hypertext Application Technology Working Group (HATWG) and World Wide Web Consortium (W3C). [8]

2.1.5 ECMAScript 4

Work on the fourth edition commenced right after the release of ECMAScript 3.0. There was a proposal to include class definitions in JavaScript from the start in order to manage large programs [7]. During the development of the fourth edition, disagreements on the future of JavaScript began to appear more frequently within the TC39 committee members. While some wanted JavaScript language to have features for developing big applications, some were not convinced by it. The lack of agreement and difficulty of few properties prolonged the development of the fourth edition [8].

2.1.6 ECMAScript 3.1 or 5

Crockford put forth the idea of developing a manageable set of properties which could be agreed upon. Syntax of the language remained the same while enhancement of user experience was added. This was later known as ECMAScript 3.1. ECMAScript 4.0 though was put aside; it was already accepted as an ECMAScript even without any release. Thus, to avoid uncertainty, TC39 decided to name ECMAScript 3.1 to ECMAScript 5 in 2009. [8]

ECMAScript 5 did not have any syntax changes but included features that were already being used by developers and recognised by browsers. It included features which were aimed at improving programming and handling errors. JavaScript Object Notation (JSON) was included. This version of JavaScript was supported by different web browsers. [8]

In 2011, ECMAScript 5.1 was released which was aimed at correcting misconceptions regarding standardisation. [8]

2.1.7 ECMAScript 6 and Beyond

ECMAScript 6 came with consequential improvements along with correcting the imperfections of ECMAScript 5 [9]. It was released in 2015 and known as

ES2015. There were significant syntactic changes brought by ECMAScript 6 with the intention to take JavaScript to a larger public and improve programming in JavaScript significantly [8]. The support for ECMAScript 6 has been on the rise in browsers over time as it will pave the way ahead for application development in JavaScript. [9]

ECMAScript went through minor changes the following years in 2016 and 2017, respectively. ECMAScript released in 2016 was the seventh edition of JavaScript also called ES2016. The eight editions of JavaScript released in 2017 is also called ES2017 [10]. For next edition of JavaScript, a term ES.Next was given. [10]

2.2 Pros and cons for JavaScript

Here are advantages of JavaScript:

- Wherever JavaScript is hosted, it often runs on the client environment to save bandwidth and accelerate the execution process.
- The primary benefit of JavaScript is its ability to support and deliver similar results in all modern browsers.
- Global corporations make a significant contribution by initiating significant initiatives. Google (which invented the Angular framework) or Facebook (created the React.js framework) are two examples
- JavaScript integrates well with other programming languages and can be used in a wide variety of applications.
- Numerous open-source projects assist developers in using JavaScript.
- There are many methods for using JavaScript through Node.js servers. It is possible to create a complete JavaScript application utilizing just JavaScript. [11]

Some Limitations of JavaScript are listed below:

- No matter how powerful the JavaScript interpreter is, the JavaScript DOM (Document Object Model) is slow and will never render as quickly as HTML.
- The primary issue or downside of JavaScript is that the code is still visible to everyone.
- With complex front-end projects, configuration is always a repetitive process due to the number of resources that must be integrated to create an environment suitable for a project.
- If an error occurs in the JavaScript, the entire website will fail to render. [11]

3 Node JS

Node.js, written by Ryan dahl in 2009, is an amalgamation of the V8 JavaScript chrome engine, a I/O (Input/Output) API and an event loop [12]. Node.js utilizes event driven, non-blocking I/O model making it lightweight, coherent and ideal for data-intensive real-time web applications that run across distributed devices [13]. V8 provides Node.js with a boost in performance by performing straight compilation into native machine code instead of using an interpreter. [13] Node.js provides JavaScript developers with a unique opportunity as the backend program can be written using JavaScript. It avoids the inconvenience for JavaScript developers to learn other programming languages for the back end [14].

3.1 Node.js Features

Features of Node.js are as follows:

3.1.1 Asynchronous and Event Driven

APIs of Node.js library are asynchronous which allows servers built on Node.js to move from an API to another without waiting for response. When there is a response to the previous API call, the events of Node.js notifies it [15]. As asynchronous I/O operations do not block the script execution, this allows applications to execute without being slowed down while performing I/O operations. This allows browsers to handle multiple requests from users while also responding to them. [13]

3.1.2 Modules

Node.js was developed as a server runtime for JavaScript having access to the filesystem. Access to filesystem was lacking JavaScript which allowed Node.js to implement a distinct way to maintain modules. Node.js implemented CommonJS which allowed JavaScript to have a module system in servers or domains outside of browsers. CommonJS gained popularity in the browser environments which was also helped by module bundlers like Browserify and Webpack. [16]

Module system assists in conveying the basic requirement of software engineering. It assists in maintaining the code in an ordered way while also helping to expand and examine several functionalities individually. This independence provided by a module system allows for repeated use of the same piece of code over several projects. Module system lets the desired part of code implementation be hidden from public interfaces that are consumed by users of a particular module. It helps developers to write over existing modules which provides consumers to use the modules over time. [16]

3.1.3 Single threaded and Scalable

The basic concept to learn for scaling applications is load distribution. Load distribution is the process of distributing load of an application over multiple processes and machines. Node.js application runs on a single thread which seems like a limitation but the non-blocking I/O model of Node.js allows applications to optimize the utilization of resources to handle simultaneous requests. Thus, Node.js applications running on a single thread is an advantage for scaling applications over time. [16]

3.1.4 Fast

Node.js runs on the V8 engine which powers Google Chrome [13]. Competition between multiple browsers to develop support and increase performance for JavaScript over time has boosted performance of web applications. Node.js is huge on streams and streaming. Streams are data distributed over time. Since data keeps coming in blocks, it allows developers to work with each block of data as they come instead of waiting for the whole data. This helps in saving time and developing applications faster. [13]

4 Express JS

Express was designed by TJ Holowaychuk who outlined Express as a web framework influenced by Sinatra, a web framework built on Ruby [17]. “Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications” as stated by Express website. [18]It is built upon Node.js core http module and connect components. Express provides libraries for core Node.js modules that are robustly tested and maintained. This helps developers to avoid the inconvenience of writing similar code for the same Node.js module while building applications. Express also allows for sophisticated reuse of code along with a Model-View-Controller (MVC)-like design for web applications. [19]

4.1 Philosophy of Express

4.1.1 Minimal

Express limits the layer between the developer and the server. This allows developers to have freedom with implementation of their ideas. Express allows developers to use Express functionalities that are required for their applications while excluding functionalities that are not used. Thus, it helps to avoid complexity and bloating of applications. [17]

4.1.2 Flexible

Express provides flexibility to projects as developers can decide on the third-party libraries and functionalities required for their applications. [19]

4.1.3 Web Application Framework

Express is the back-end part of the web framework which can be merged with other services through web APIs. Also, Entire application can be built using backend rendering with Express making Express complete web application framework. [17]

4.2 Core of Express

4.2.1 Routing

Routing is the process of determining the application response on finding a particular endpoint requested by the client. The endpoint is a URL with a HTTP request method. [20]

Routing can be defined using Express methods that respond to certain HTTP requests for example App.get() handle get request and similarly App.post() handle POST requests. [21]

4.2.2 Middleware

Middleware is a term that refers to something that sits within two layers of software. Express middleware is a collection of functions that run during the lifecycle of an Express transaction. [22]

Express uses the middleware in a wide array of fields. These may include serving files, error handling, third parties middleware apps make web development easier by working with session, cookies, user authentication etc. [23]

4.2.3 Templating

A template engine simply adds the dynamic data from the program to the HTML page and merges it. The template file replaces this dynamic data with actual values and display the HTML page [24]

Template engines such as EJS, Pug and Mustache are built on Express that simplify the web development process. [24]

5 MongoDB

MongoDB is a flexible, scalable and document-oriented database. It is not a relational database. [25] It is based on a document data model where data is stored in key-value pairs. [26]

The size of data stored by applications are expanding which has brought the need for scaling databases of applications. This leaves developers with two choices, either scaling up or scaling out. Scaling up requires expensive and big machines and in due course even that machine will not be enough. The other option is scaling out for which MongoDB was developed. The document data model of MongoDB helps to slice data across several servers. The maintenance of data and load are taken care of by MongoDB along with rearranging data and routing operations to the expected server. [25]

MongoDB not only allows users to perform database operations but also has features that make it unique from other databases. These features are following

- MongoDB has common sub-indexes and gives distinct, composite, geospatial and complete text indexing abilities. Sub-indexes on hierarchical design are supported, enabling developers to have the capability to model in procedures that complement their application.
- MongoDB offers a platform for data integration built on the principle of data production channels. Aggregation channels allow you to create complex analytics engines by manipulating data on the server side in a sequence of steps, maximizing the benefits of database optimizations.
- MongoDB allows time-to-live (TTL) collections for data ought to expire after a specified period, such as meetings, and fixed-size (capped) collections for data that should be retained indefinitely, such as logs. MongoDB also supports fragmented indexes that are restricted to documents that meet a defined requirements filter, which improves performance and reduces the amount of storage needed.
- MongoDB provides an intuitive protocol for storing massive files and their metadata. [25]

6 Technology Used

6.1 Visual Studio Code

Virtual Studio Code is an efficient code editor with built-in support for JavaScript and Node.js and supported across the major operating system. [27]

It has extensions built-in which enables to edit application code in a manageable and efficient way. It also comes with extensions that are helpful in catching syntax errors during the programming phase along with debugging features that helps to avoid crashing of application. [27]

6.2 NPM

Npm is the largest software registry used by open-source developers to distribute and use packages. Organizations can manage packages on their own for private development of their applications. It consists of three unique components which are the website, the Command Line Interface (CLI) and the registry. Developers use the website to find packages as per their need, the CLI to communicate with npm while the registry is a common database of JavaScript software along with meta-information surrounding it. Npm can also be used to limit access to certain developers, maintain versions of code and its dependencies and finding developers pursuing same projects. [28]

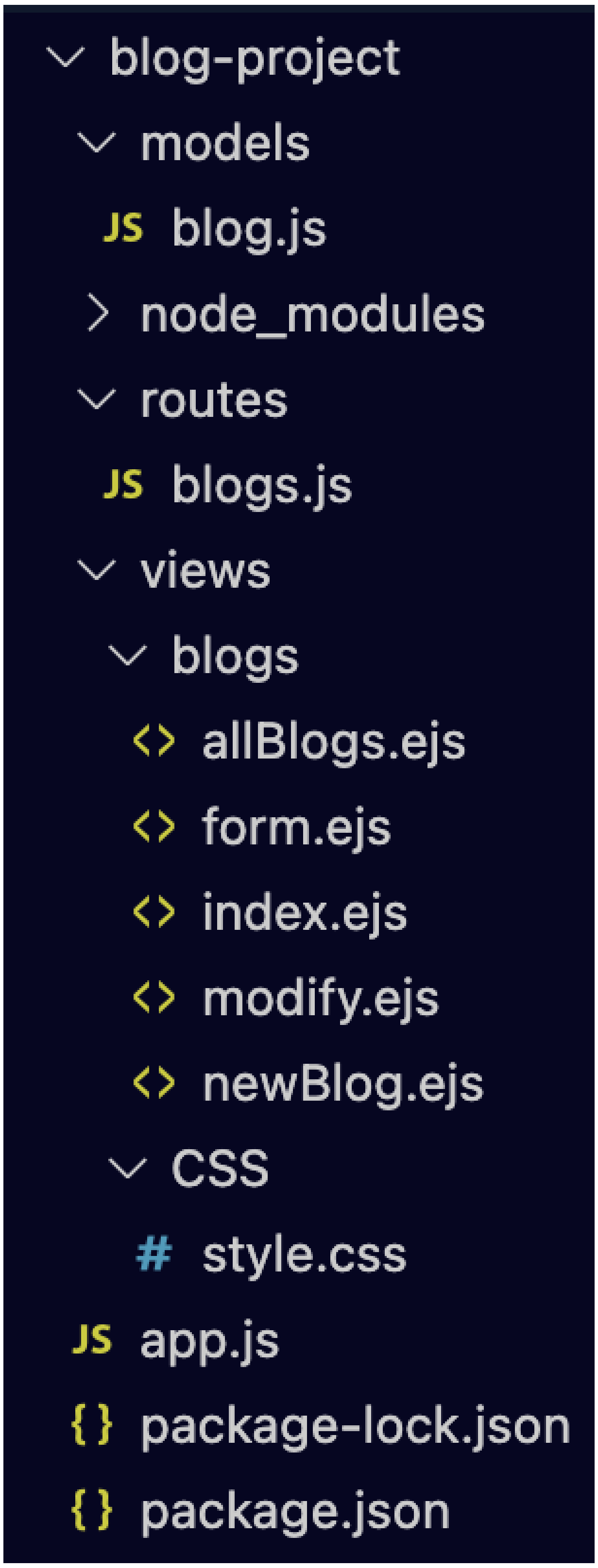
6.3 Bootstrap

Bootstrap is a popular HTML, Cascading Style Sheet (CSS) and JavaScript framework used for creating websites that are responsive and mobile friendly. Bootstrap was developed in 2010 by two developers Mark Otto, Jacob Thornton. Formerly known as Twitter Bootstrap, bootstrap has gone through many updates and finally today we have Bootstrap 5. Bootstrap is used by around more than one fourth of the world's websites roughly. [29]

7 Implementation

Implementation of the blog project was done in two phases. The first phase was for designing the User Interface of the application and the second phase was for taking in user input, saving it in the database and displaying it back to the user with functionality using which the user can view, modify and delete their blog.

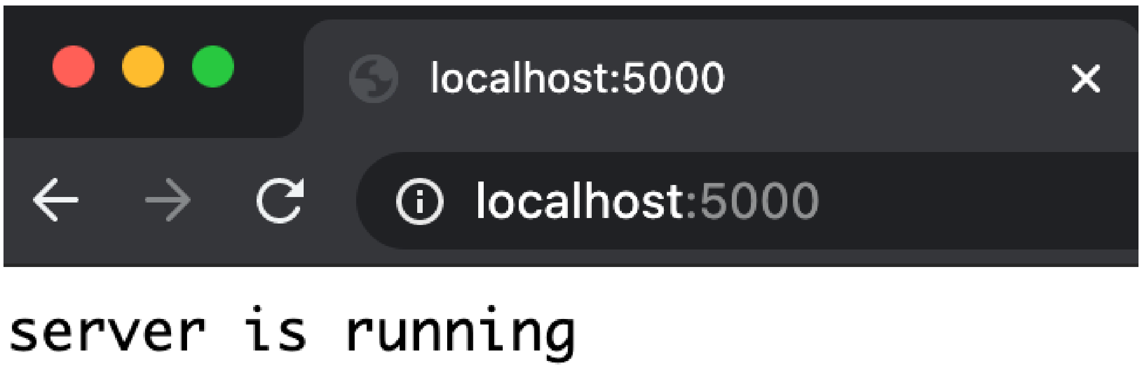
The blog project was initiated with npm and Express, ejs and mongoose were installed. The folder structure of the project is shown in figure 2.



```
blog-project
  models
    blog.js
  node_modules
  routes
    blogs.js
  views
    blogs
      allBlogs.ejs
      form.ejs
      index.ejs
      modify.ejs
      newBlog.ejs
    CSS
      style.css
  app.js
  package-lock.json
  package.json
```

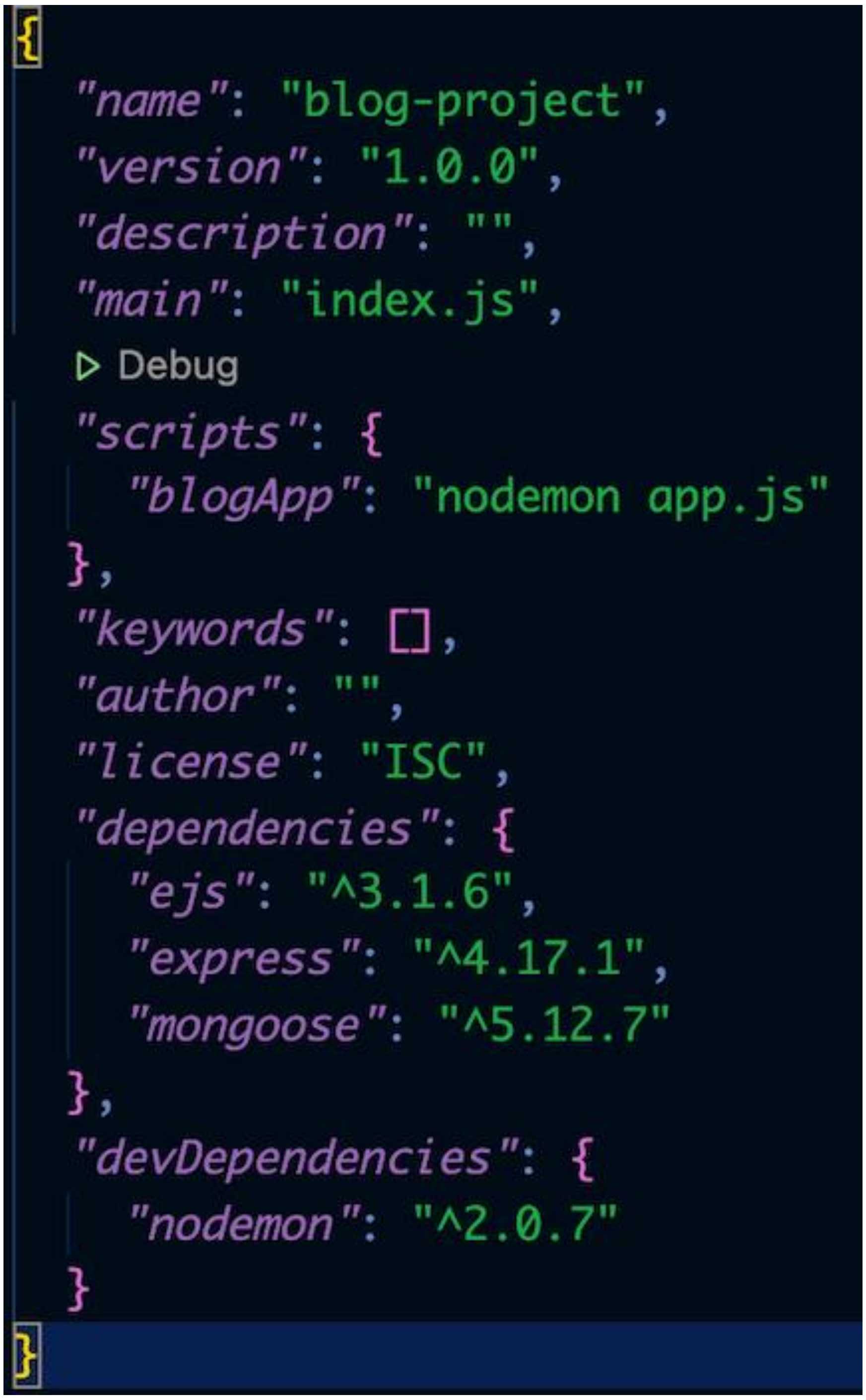
Figure 2: Folder structure of blog project

The model folder is for writing models for database, the routes folder is for writing the routing file, views folder for writing the views. Express.js was used to write the app.js file and it was assigned port 5000 to listen in. The server was up and running as shown in figure 3 and the initial package.json file is shown in figure 4 respectively.



```
server is running
```

Figure 3: shows server is running



```
{
  "name": "blog-project",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  ▷ Debug
  "scripts": {
    "blogApp": "nodemon app.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "ejs": "^3.1.6",
    "express": "^4.17.1",
    "mongoose": "^5.12.7"
  },
  "devDependencies": {
    "nodemon": "^2.0.7"
  }
}
```

Figure 4: initial package.json file

Implementation of the user interface was commenced. The index.ejs page was formed using the view engine ejs and rendered through Express. The index.ejs file was styled using bootstrap in order to make it responsive and simple CSS was used to style the header and footer of the page. In the initial phase, hard coded data was supplied from the app.js file to the index.ejs file to check whether the data was displayed with proper styling. When the data was displayed correctly, the construction of newBlog.js file commenced to take in the user input.

A button-styled link was displayed in the index.ejs file as shown in figure 5 which when clicked would take the user to the newBlog.ejs page. To make this happen, a routes folder was created where blogs.js file was created to handle the routing of the application.

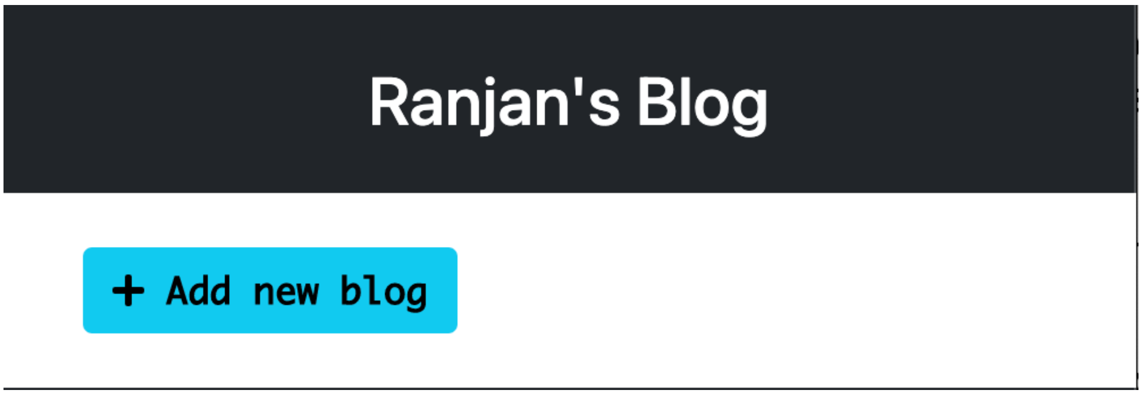


Figure 5. The button-styled link.

A form.ejs file was created. form.ejs file was used to make the form and controls for the application where the user can enter information. The form fields in form.ejs file had required attributes so that the user did not leave them empty. The motivation behind putting the form in a separate file was to increase reusability of the file and avoid writing duplicate codes for other functionalities which may use similar design. Then, the form.ejs file was passed to the newBlog.ejs file. The rendered version of newBlog.ejs file is shown in the figure 6.

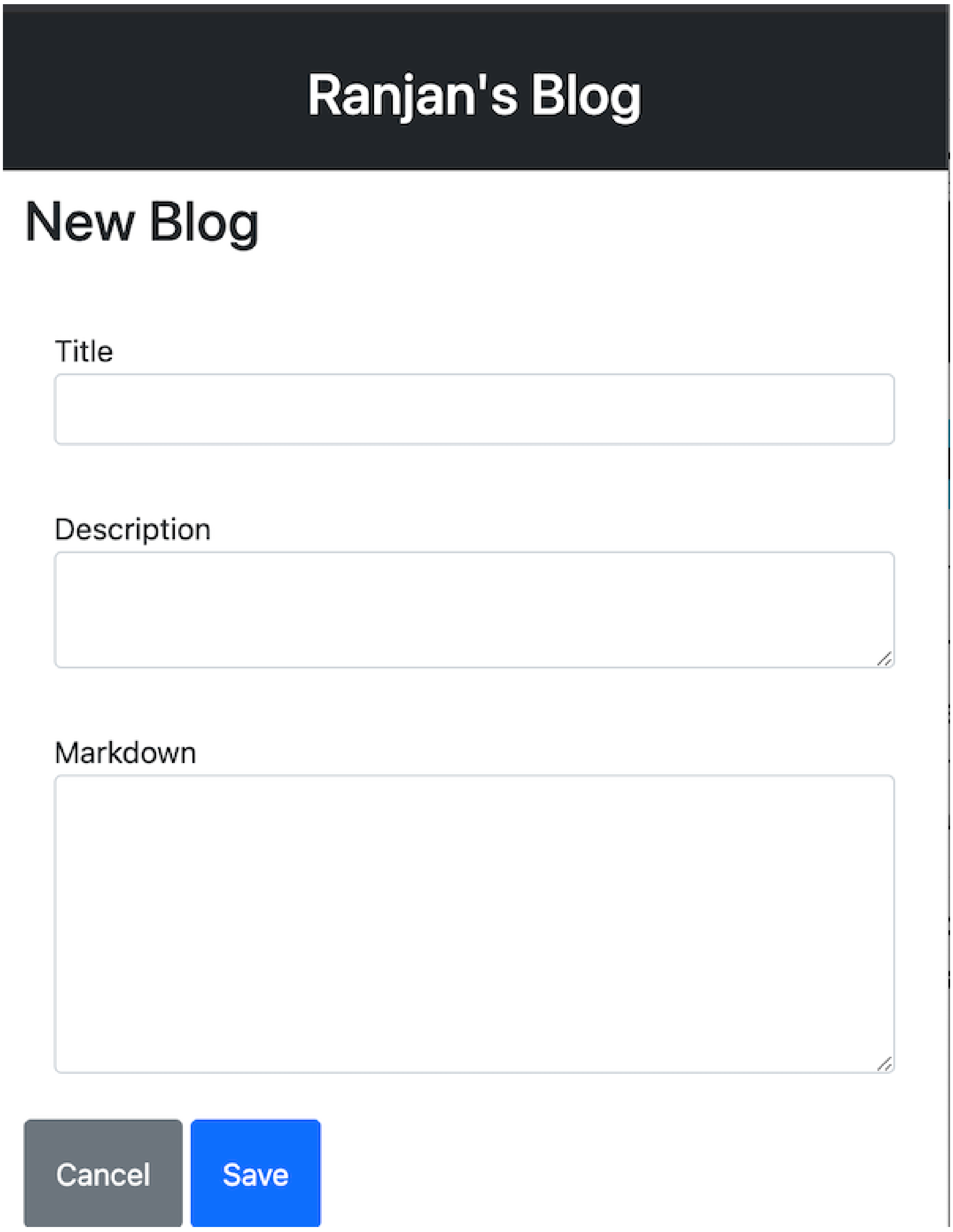
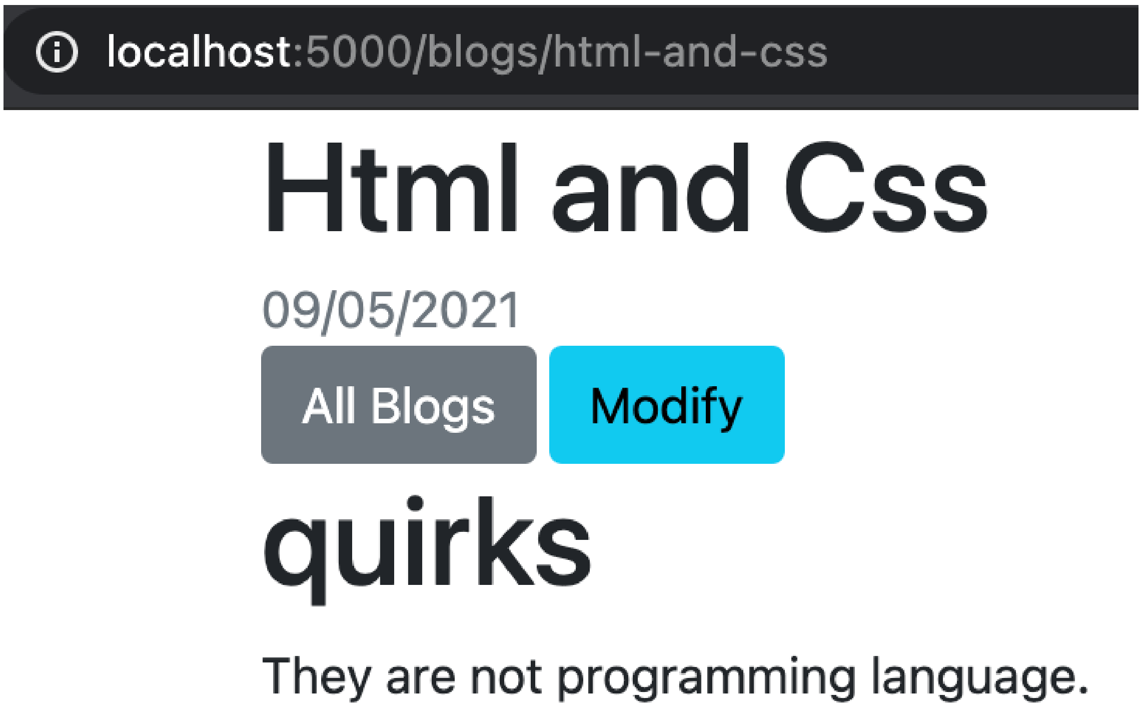


Figure 6. The rendered file newBlog.ejs.

As can be seen from figure 6, the user has been provided with two functionalities on the rendered version of the newBlog.ejs page. On clicking the cancel button, the user will be taken back to the index.ejs file while on clicking the save button the user input will be saved, and the user will be routed to the allBlogs.ejs file.

When the user clicks the save button on the newBlog.ejs file, the input from the user is sent to the blog.js file in the model folder by the help of the routing file blogs.js. The database saves the user input.



Html and Css

09/05/2021

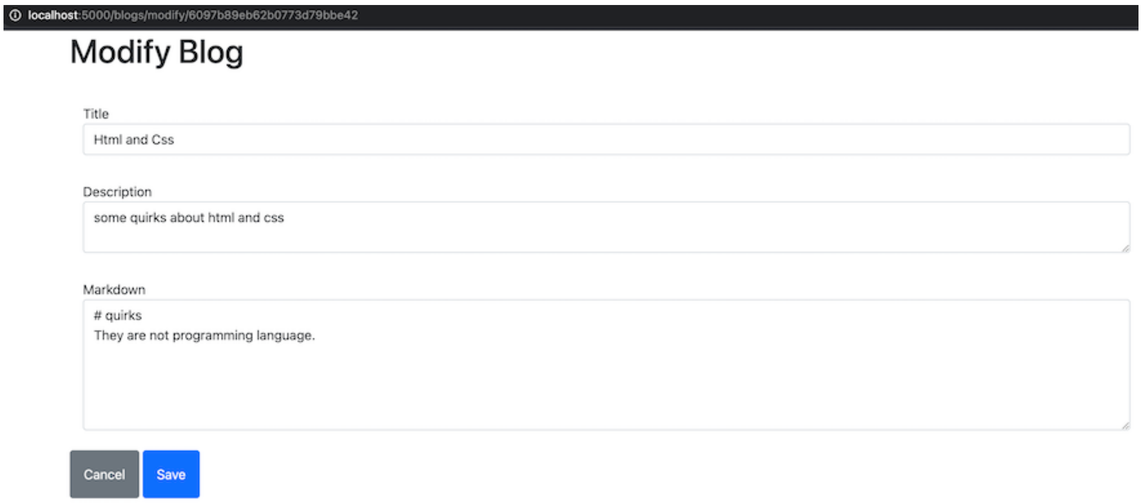
quirks

They are not programming language.

Figure 7:shows rendered version of allBlogs.js file

The allBlogs.js file displays the saved user input as shown in figure 7. It also provides the user with two functions, either the user can go to index.ejs file by clicking the All Blogs button or can be routed to modify.ejs by clicking the modify button where the user can change the earlier inputs.

The modify.ejs file has the same design as the newBlog.ejs file. The form.ejs file is passed to the modify.ejs file which helps to avoid duplicating code and saves time. The user can modify earlier input and save the changed file. The earlier inputs are displayed while modifying as shown in figure 8. The changes done by the user is then passed to the blogs.js file in the routes folder. The blogs.js sends the changed information to the blog.js file in the model folder where the changes from the user are updated and saved in the database.



localhost:5000/blogs/modify/6097b89eb62b0773d79bbe42

Modify Blog

Title

Html and Css

Description

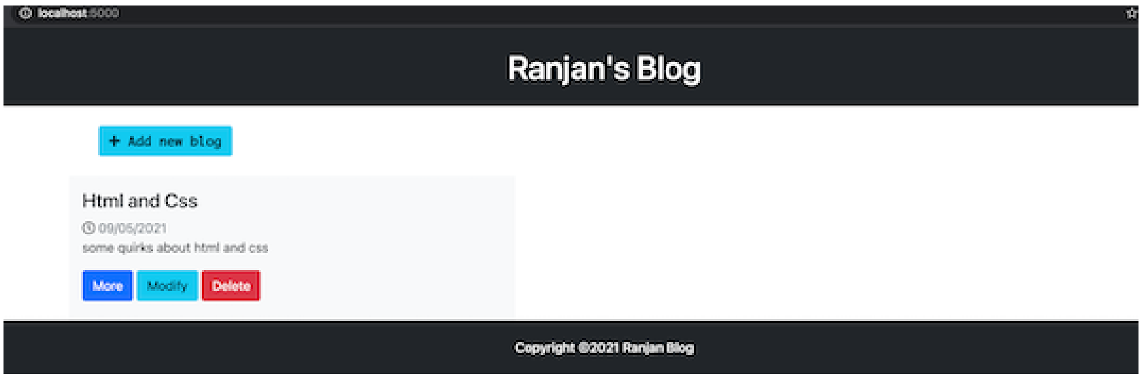
some quirks about html and css

Markdown

quirks
They are not programming language.

Figure 8: shows modify.js file with data from initial inputs

The next functionality to add was the delete button which the users could click to delete the blogs from the database. To make a delete button, methodOverride library was installed using npm. This library allowed us to create a link with delete method. The function for deleting blogs was created in the blogs.js file in the routes folder. So, when the user clicks the delete button, the delete route in the blogs.js file is called, and it deletes the particular blog form the database and reloads the index.ejs file displaying the updated information. Along with delete, users were provided with more and modify buttons when displaying each blog using which full text of a blog could be read and modified respectively as shown in figure 9.



localhost:5000

Ranjan's Blog

Html and Css

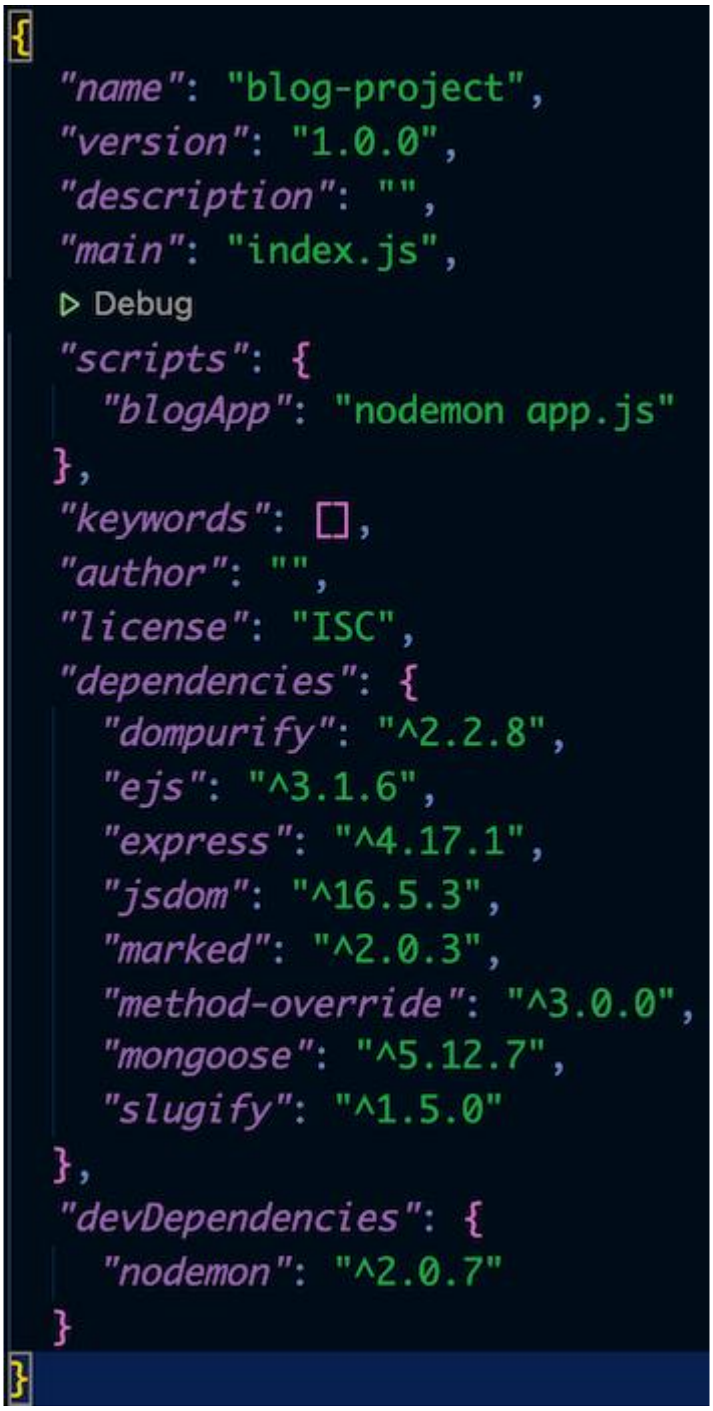
09/05/2021

some quirks about html and css

Copyright ©2021 Ranjan Blog

Figure 9: shows all the buttons within the single blog display.

Once the main functionalities of the application were finished, some extra libraries were installed, and their features were added. Slugify was used to manage the links to each blog. Instead of using id of the links from the database, it was replaced by s slug corresponding to the value of title of each blog. Dompurify and jsdom libraries were used to provide basic security to the application. Marked library was used to provide markdown when displaying the text entered in the markdown field. The package.json file after the completion of project is shown in figure 10.



```
{
  "name": "blog-project",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  ▷ Debug
  "scripts": {
    "blogApp": "nodemon app.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "dompurify": "^2.2.8",
    "ejs": "^3.1.6",
    "express": "^4.17.1",
    "jsdom": "^16.5.3",
    "marked": "^2.0.3",
    "method-override": "^3.0.0",
    "mongoose": "^5.12.7",
    "slugify": "^1.5.0"
  },
  "devDependencies": {
    "nodemon": "^2.0.7"
  }
}
```

Figure 9:Package.json file after the completion of project

8 Results

Multiple inputs of blog were inserted in the applications to populate the database and the main page. Figure 11 shows the application index page.

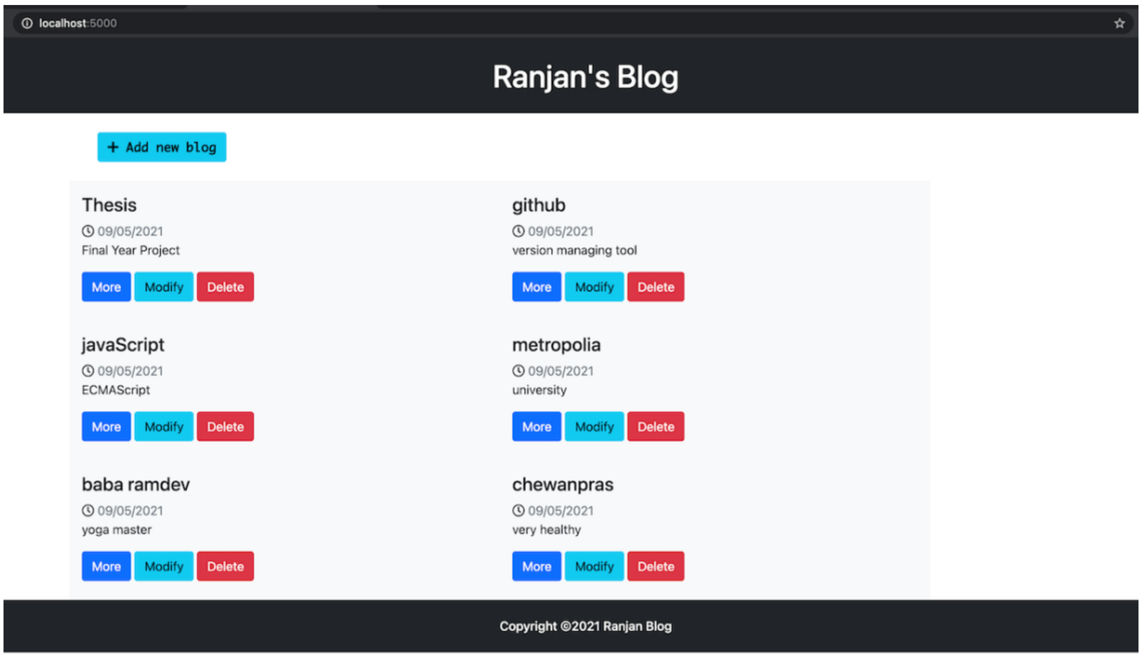
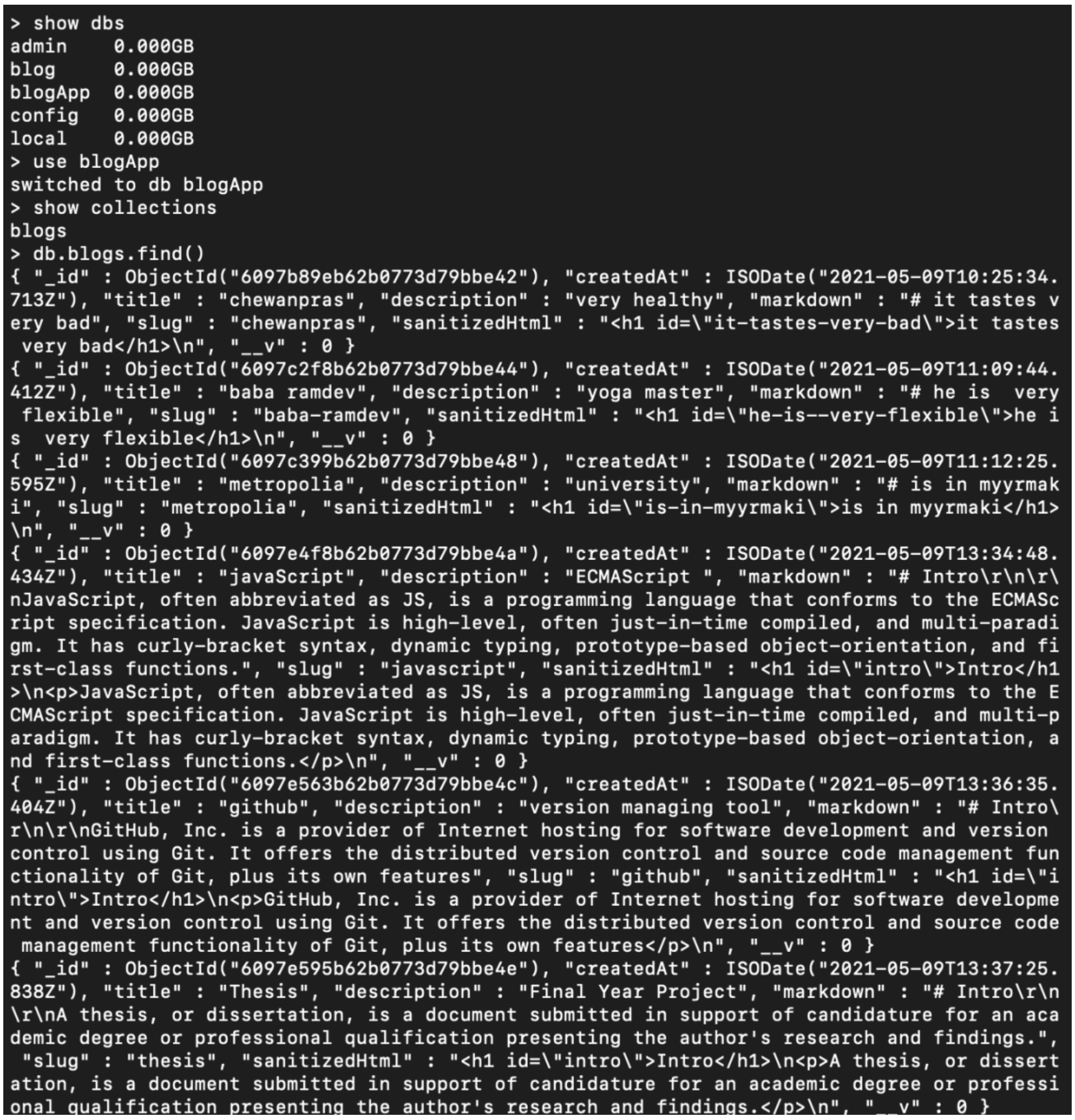


Figure 10: index.ejs file after populating the database

The mongoDB database was also checked to verify the data displayed on the index.ejs file and inside the database matched. The mongoDB database was accessed through the terminal and is shown in figure 12.



```
> show dbs
admin    0.000GB
blog     0.000GB
blogApp  0.000GB
config   0.000GB
local    0.000GB
> use blogApp
switched to db blogApp
> show collections
blogs
> db.blogs.find()
{ "_id" : ObjectId("6097b89eb62b0773d79bbe42"), "createdAt" : ISODate("2021-05-09T10:25:34.
713Z"), "title" : "chewanpras", "description" : "very healthy", "markdown" : "# it tastes v
ery bad", "slug" : "chewanpras", "sanitizedHtml" : "<h1 id=\"it-tastes-very-bad\">it tastes
 very bad</h1>\n", "__v" : 0 }
{ "_id" : ObjectId("6097c2f8b62b0773d79bbe44"), "createdAt" : ISODate("2021-05-09T11:09:44.
412Z"), "title" : "baba ramdev", "description" : "yoga master", "markdown" : "# he is  very
 flexible", "slug" : "baba-ramdev", "sanitizedHtml" : "<h1 id=\"he-is--very-flexible\">he i
s  very flexible</h1>\n", "__v" : 0 }
{ "_id" : ObjectId("6097c399b62b0773d79bbe48"), "createdAt" : ISODate("2021-05-09T11:12:25.
595Z"), "title" : "metropolia", "description" : "university", "markdown" : "# is in myyrmak
i", "slug" : "metropolia", "sanitizedHtml" : "<h1 id=\"is-in-myyrmaki\">is in myyrmaki</h1>
\n", "__v" : 0 }
{ "_id" : ObjectId("6097e4f8b62b0773d79bbe4a"), "createdAt" : ISODate("2021-05-09T13:34:48.
434Z"), "title" : "javaScript", "description" : "ECMAScript ", "markdown" : "# Intro\r\n\r\
nJavaScript, often abbreviated as JS, is a programming language that conforms to the ECMASc
ript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradi
gm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and fi
rst-class functions.", "slug" : "javascript", "sanitizedHtml" : "<h1 id=\"intro\">Intro</h1
>\n<p>JavaScript, often abbreviated as JS, is a programming language that conforms to the E
CMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-p
aradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, a
nd first-class functions.</p>\n", "__v" : 0 }
{ "_id" : ObjectId("6097e563b62b0773d79bbe4c"), "createdAt" : ISODate("2021-05-09T13:36:35.
404Z"), "title" : "github", "description" : "version managing tool", "markdown" : "# Intro\
r\n\r\nGitHub, Inc. is a provider of Internet hosting for software development and version 
control using Git. It offers the distributed version control and source code management fun
ctionality of Git, plus its own features", "slug" : "github", "sanitizedHtml" : "<h1 id=\"i
ntro\">Intro</h1>\n<p>GitHub, Inc. is a provider of Internet hosting for software developme
nt and version control using Git. It offers the distributed version control and source code
 management functionality of Git, plus its own features</p>\n", "__v" : 0 }
{ "_id" : ObjectId("6097e595b62b0773d79bbe4e"), "createdAt" : ISODate("2021-05-09T13:37:25.
838Z"), "title" : "Thesis", "description" : "Final Year Project", "markdown" : "# Intro\r\n
\r\nA thesis, or dissertation, is a document submitted in support of candidature for an aca
demic degree or professional qualification presenting the author's research and findings.",
 "slug" : "thesis", "sanitizedHtml" : "<h1 id=\"intro\">Intro</h1>\n<p>A thesis, or dissert
ation, is a document submitted in support of candidature for an academic degree or professi
onal qualification presenting the author's research and findings.</p>\n", "__v" : 0 }
```

Figure 11: Database of blog application

9 Conclusion

The purpose of this study was to carry out a blog project using JavaScript, NodeJS and MongoDB, which worked as expected. Using Node.js to develop applications was made simple by using npm. Npm, being a rich repository of libraries and packages in JavaScript, helped with extra packages which were needed to develop certain functionalities in the project. MongoDB was effortless to maintain as the model can be written in JavaScript. Accessing the MongoDB database was effortless because of easy-to-understand documentation available on its official website.

Using Node.js for server-side programming has provided JavaScript developers with comfort. The communities built around Node.js have many libraries or packages which help to build full stack applications in JavaScript that might not have been possible earlier.

References

[1] “https://www.leewayhertz.com,” Leewayhertz, [Online]. Available: https://www.leewayhertz.com/full-stack-development/. [Accessed 25 04 2021].

[2] A. Anuradhac, “https://devopedia.org/,” Devopedia, 28 10 2020. [Online]. Available: https://devopedia.org/full-stack-developer. [Accessed 27 04 2021].

[3] D. Flanagan, “Introduction to JavaScript,” in *JavaScript: The Definitive Guide, Fourth Edition*, California, O'Reilly Media, Inc., 2001.

[4] S. Powers, in *Learning JavaScript*, Sebastopol, O'Reilly Media, Inc., 2006.

[5] S. Powers, Learning JavaScript, Sebastopol: O'Reilly Media, Inc, 2006.

[6] T. DeGroat, “https://www.springboard.com/,” SpringBoard, 19 08 2019. [Online]. Available: https://www.springboard.com/blog/data-science/history-of-javascript/#:~:text=JavaScript%20Origins&text=In%20September%201995%2C%20a%20Netscape,LiveScript%20and%2C%20later%2C%20JavaScript.t. [Accessed 21 04 2021].

[7] B. E. Allen Wirfs-Brock, “JavaScript: The First 20 Years,” 2020.

[8] S. Peyrott, “https://auth0.com,” auth0, 16 01 2017. [Online]. Available: https://auth0.com/blog/a-brief-history-of-javascript/. [Accessed 02 05 2021].

[9] E. Brown, Learning JavaScript, 3rd Edition, Sebastopol: O'Reilly Media, 2016.

[10] L. Groner, Learning JavaScript DataStructures and Algorithms, Birmingham: Packt Publishing, 2018.

[11] “https://www.geeksforgeeks.org/,” GeeksforGeeks, 25 11 2020. [Online]. Available: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-javascript/. [Accessed 01 05 2021].

[12] “https://www.section.io,” Section, 25 08 2020. [Online]. Available: https://www.section.io/engineering-education/history-of-nodejs/. [Accessed 05 05 2021].

[13] M. Cantelon, N. Rajlich, M. Harter and T. Holowaychuk, Node.js in Action, New York: Manning Publications, 2013.

[14] “https://nodejs.dev/,” OpenJS Foundation, [Online]. Available: https://nodejs.dev/learn. [Accessed 02 05 2021].

[15] “https://www.tutorialspoint.com,” Tutorialspoint, [Online]. Available: https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm. [Accessed 02 05 2021].

[16] M. Casciaro and L. Mammino, Node.js Design Patterns - Third Edition, Birmingham: Packt Publishing,, 2020.

[17] E. Brown, Web Development with Node and Express, 2nd Edition, Sebastopol: O'Reilly Media, Inc, 2019.

[18] “https://expressjs.com/,” [Online]. Available: https://expressjs.com/. [Accessed 02 05 2021].

[19] E. Hahn, Express in Action, Manning Publications, 2016.

[20] “https://expressjs.com,” OpenJS Foundation, [Online]. Available: https://expressjs.com/en/starter/basic-routing.html. [Accessed 05 05 2021].

[21] “https://expressjs.com/,” OpenJS Foundations, [Online]. Available: https://expressjs.com/en/guide/routing.html. [Accessed 03 05 2021].

[22] L. Brandt, “https://developer.okta.com,” Okta, 13 09 2018. [Online]. Available: https://developer.okta.com/blog/2018/09/13/build-and-understand-express-middleware-through-examples. [Accessed 30 04 2021].

[23] “https://developer.mozilla.org/,” Mozilla and individual contributors, [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction. [Accessed 21 04 2021].

[24] “https://www.pabbly.com,” Pabbly, [Online]. Available: https://www.pabbly.com/tutorials/template-engine-in-expressjs/. [Accessed 01 05 2021].

[25] S. Bradshaw and K. Chodorow, MongoDB: The Definitive Guide, Sebastopol: O'Reilly Media, Inc., 2019.

[26] W. d. R. França, MongoDB Data Modeling, Birmingham: Packt Publishing, , 2015.

[27] “https://code.visualstudio.com/,” Microsoft, [Online]. Available: https://code.visualstudio.com/docs. [Accessed 09 05 2021].

[28] “https://docs.npmjs.com/,” npm Enterprise,, [Online]. Available: https://docs.npmjs.com/about-npm. [Accessed 09 05 2021].

[29] “https://getbootstrap.com/,” Bootstrap, [Online]. Available: https://getbootstrap.com/docs/4.1/about/overview/#:~:text=Bootstrap%20was%20created%20at%20Twitter,in%20without%20any%20external%20guidance. [Accessed 09 05 2021].