Cuong Cao Nguyen

A social platform using modern web stack (MERN)

Bachelor’s thesis

Information Technology

Bachelor of Engineering

2021

XAMK

South-Eastern Finland
University of Applied Sciences

Author (authors)	Degree title	Time
Cuong Cao Nguyen	Bachelor of Engineering InformationTechnology	February 2021

Thesis title	
A social platform using modern web stack (MERN)	43 pages 0 pages of appendices

Commissioned by

Supervisor

Timo Hynninen

Abstract

The objective of this thesis was to use modern web stack (MERN) to develop several features of a social-platform-like application. The work could be a reference for anyone who is getting familiar with MERN stack. It showed the process of building a front-to-back application from the beginning.

Before doing the implementation part, the background knowledge about each core element of the stack was outlined. A database system to store data was introduce, frameworks and libraries to support interacting with the back end and the front end were mentioned also. In the implementation part, needed tools were listed, the process of setting up environments was shown, the approaches to build the features of the application were explained.

After developing several features, conclusions and comments were made about the process and the technologies.

Keywords

ReactJS, NodeJS, ExpressJS, MongoDB, a social network

CONTENTS

LIST OF ABBREVIATIONS 4
1 INTRODUCTION 5
2 BACKEND TECHNOLOGIES 6
2.1 JavaScript 7
2.2 Node JS 9
2.3 Express 12
2.4 JSON and Postman 14
2.4.1 JSON 14
2.4.2 Postman 16
2.5 UUID 17
2.6 JWT 18
2.7 Asynchronous programming 19
3 DATABASE 20
3.1 MongoDB 20
3.2 Mongoose 22
4 REACT JS 23
5 PRACTICAL IMPLEMENTATION 26
5.1 Tools and technologies 26
5.2 Back end 27
5.2.1 Preparation 27
5.2.2 Developing features 30
5.3 Front end 37
6 CONCLUSION 40
REFERENCE 41

LIST OF ABBREVIATIONS

MERN	MongoDB, Express, React, Node
URL	Uniform Resource Locator
HTML	HyperText Markup Language
JS	JavaScript
CSS	Cascading Style Sheets
JRE	Java Runtime Environment
IO	Input Output
PHP	PHP: Hypertext Preprocessor
HTTP	Hypertext Transfer Protocol
JWT	JSON Web Token
SSR	Server-side rendering
JSON	JavaScript Object Notation
AJAX	Asynchronous JavaScript And XML
XML	Extensible Markup Language

1 INTRODUCTION

In the digital era that we are currently living in, there are more and more approaches and supportive applications for a person to be connected to the rest of the world. The Internet was invented on January 1, 1983 by two computer scientists Bob Kahn and Vinton Cerf. Since then, it has been easier for people to be connected to the society as well as finding information from all around the world. The World Wide Web was a revolutionary invention for the humanity: everything is now available on the web for everyone to learn, share and grow as a person.

A social network, also known as a social platform, is one of the most popular approaches for people to connect to each other. Sharing things in life is one of the necessities. A person needs to be heard, to seek help or to share thoughts, emotions, outlooks with his or her acquaintances, friends and family. Human beings exist in herds, and we cooperate with each other to improve social quality as a whole. Cooperation is the orthodox way for everyone to work. With the existing social networks, information is spread rapidly to every single region.

For instance, referring to the ongoing COVID-19 pandemic which has caused us severe damages including the economy and casualties, the Internet is on our side. With the help of social platforms, people understand more quickly the intense degree of the pandemic, the dangerous level of the virus, therefore, they raise awareness for themselves, encourage other people to use respiratory protection and repel the disease.

The aim of this thesis is to make a social platform sample using the modern tech stack MERN. I chose this topic because it is a practice for me to grow as a web developer. The thesis covers related background knowledge and technologies, including interaction with the backend, database, and front-end framework. There is a demonstration of the knowledge written. It is useful for students who are learning MERN. They will know the noticeable points of making an application with MERN, therefore, understand the stack better.

2 BACKEND TECHNOLOGIES

When the word “backend” is mentioned, it can be written as "back end" or "back-end" depending on the context, it refers to the process behind the computer screen which cannot be displayed visually to the users. And since we have back end, we also have front end, everything which is shown to users. By oversimplified definitions mentioned, there is a stark difference between these two terms. When it comes to application architecture, there are three distinct layers (from top to bottom): presentation layer, business layer and data access layer. The front end is located in the first layer while the back end is at the third one. When you surf the web and browse through web pages, there is an URL to each specific web page. With that URL, you are accessing the content on the web server. Each URL triggers a different set of scripts, generating the page. Everything happening behind the scene, so that you can see the page visually, is back-end related.

Some typical back-end activities can be listed as follow:

- Triggering a script to show HTML content
- Retrieving an image from a database
- Downloading, uploading data
- Updating a database
- Saving a password in a hashed version instead of plain one for better security
- Response to users’ requests

There is a programming language that allows developers to write code for both back end and front end. That language is called JavaScript. It is common for a programmer to write an application front to back (a person who is capable of doing this is called full-stack developer). For instance, ReactJS is a framework that can handle the front parts, meanwhile, NodeJS is responsible for the back parts.

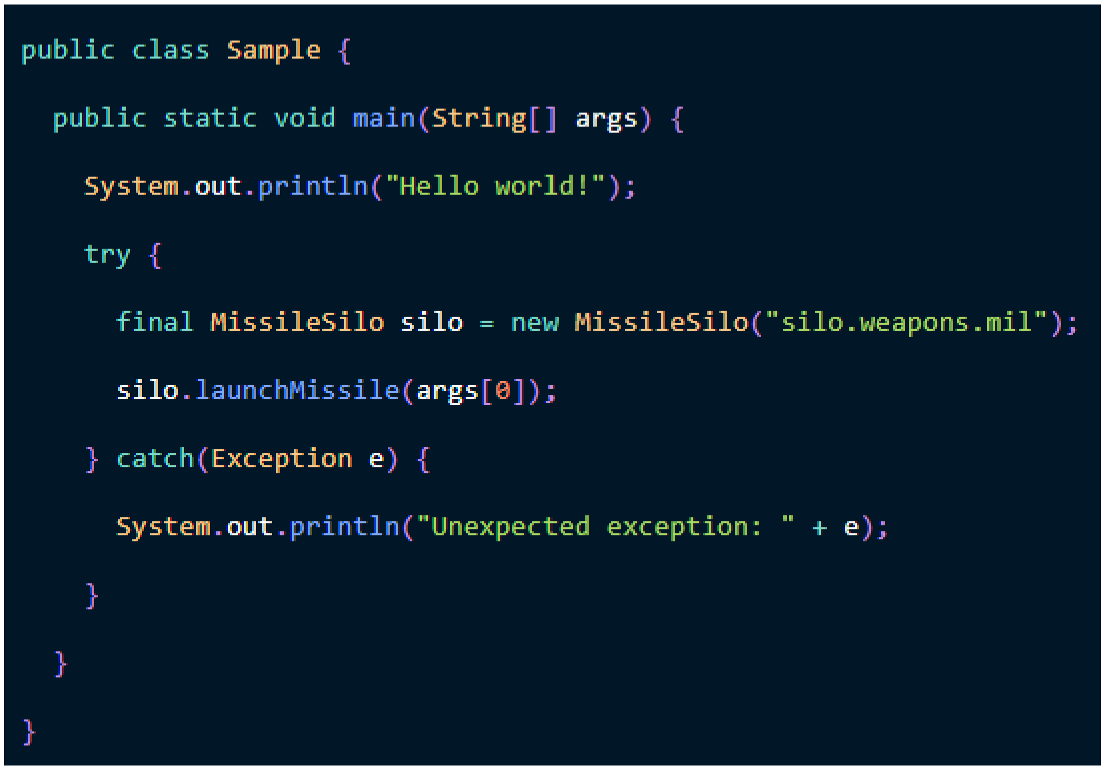
2.1 JavaScript

There is an undeniable fact that JavaScript is one of the most popular programming languages, since its learning curve is not steep and its flexibility allows web developers to develop a full-stack application, front to back, using only one programming language with the help of support frameworks. JS, short for JavaScript, is voted to be the most used programming language in 2019 by a survey conducted within the Stack Overflow community. JS has been on the first place for seven consecutive years. Statistics show that it was employed by 67.8% of web programmers in 2019. More than 95% of websites nowadays, with an equivalent of 1.52 billion websites, have utilized the language. The most used social network Facebook, the largest video database YouTube, the world digital map Google Maps, JS has played an essential role in the process of building those applications.

A glance at the first release of the programming language, in September 1995, a scripting language was invented by a developer named Brandan Eich within a relatively short amount of time. In the beginning, it was called Mocha but was shortly renamed to LiveScript and eventually, JavaScript.

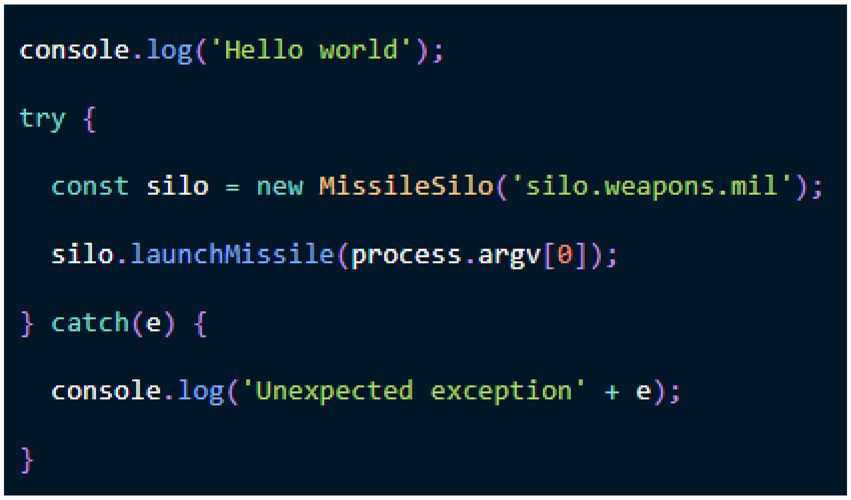
There are three core elements that create a typical website: HTML, CSS and JavaScript. HTML is responsible for the structure, CSS is responsible for the presentation of the page. Meanwhile, JS brings functionalities and behaviors to the website, makes it lively and interactive with the user. Viewing the web technologies as a human being, HTML is the skeleton of the body, CSS acts as the skin tone, the clothes which we put on ourselves each day. JS is how we interact, communicate with other people, society. JavaScript is a client-side language, active mainly in the browser of the computer. Since the advent of NodeJS, JS can run on servers as well, handling the back-end parts. JS is used more commonly compared to Java, Flash and the remaining programming language mainly because it helps web programmers to build applications rapidly.

Many people have mistaken JavaScript as Java or vice versa since the names are somewhat similar to each other. But there is a stark difference between those two and people should be able to distinguish between them. There was a myth saying that the naming convention of JavaScript was actually an approach to shift attention away from Java because Java was the trending language at the time. Figure 1 and figure 2 below show the same purpose but different syntax.



```java
public class Sample {

  public static void main(String[] args) {

    System.out.println("Hello world!");

    try {

      final MissileSilo silo = new MissileSilo("silo.weapons.mil");

      silo.launchMissile(args[0]);

    } catch(Exception e) {

      System.out.println("Unexpected exception: " + e);

    }

  }

}
```

Figure 1. Java example



```javascript
console.log('Hello world');

try {

  const silo = new MissileSilo('silo.weapons.mil');

  silo.launchMissile(process.argv[0]);

} catch(e) {

  console.log('Unexpected exception' + e);

}
```

Figure 2. JavaScript example

JS has simplified code for basic functionalities but the syntax was still familiar for people at the time. This obviously helps JS to quickly become known to programmers. And it keeps developing and changing every day. No matter how different it becomes, JS will still be around for a long time.

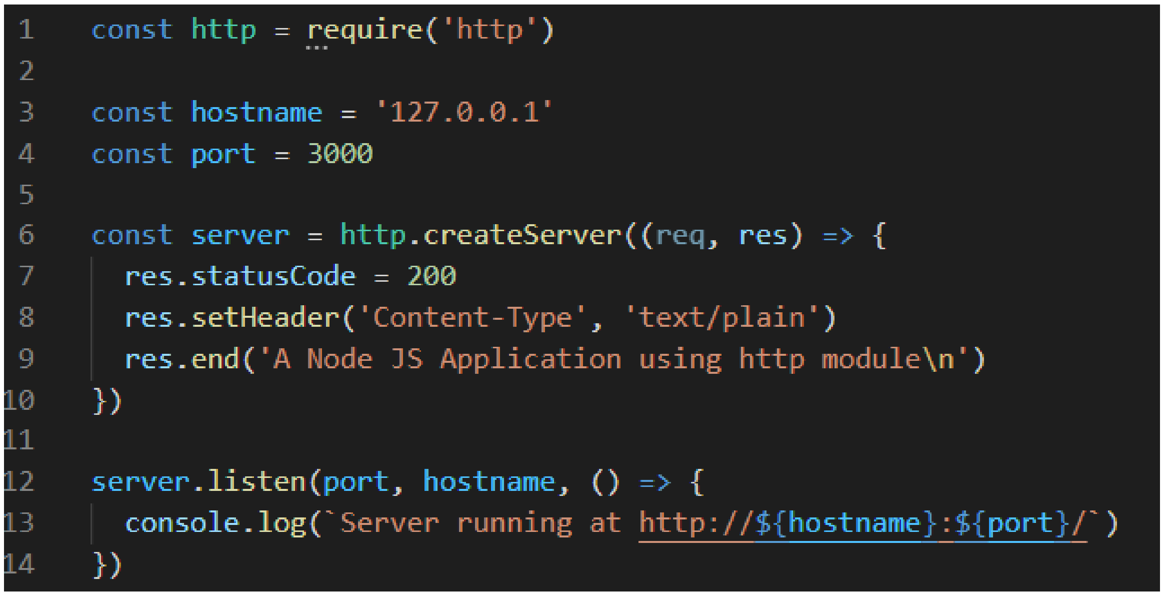
2.2 Node JS

Node JS allows web developers to do server-side programming, interact with the server using JS. Node JS is not considered as either a programming language or a framework, it is an open source, cross platform Java Runtime Environment (JRE). “A typical JRE includes three core parts: Java Classloader, Java Class Library, and Java Virtual Machine” (Effectus Software blog). The Classloader job is to prepare all the components needed to run a program. The Java Class Library is a library of code snippets to create programs, while the Java Virtual Machine enables computers to run Java applications.

Node JS has all three parts mentioned above because it utilizes V8, an open-source JavaScript engine invented by Lars Bak. One of the most popular web browsers currently, Google Chrome, also utilizes V8. Normally, JS needs to be run in a browser, but Node JS was built based on V8 from Chrome, it enables web programmers to use JS outside of browsers. If you want to build an application using JS in its back end, then Node JS is a mighty approach.

Applications using Node JS are single-threaded, non-blocking IO (input output) paradigms. That makes a stark difference between Node JS and remaining programming languages such as PHP, Ruby. Those languages are multi-threaded, blocking IO model. There are many existing tasks while a program is running. Let us assume that the first job took 1 millisecond to complete and in that 1ms, if there is no other job, the program is paused and waits for the job to be finished. It is a waste of time so PHP, Ruby deal with this issue by opening new threads to do other jobs. Node JS app does not open new threads for jobs. It uses a single thread to handle every job. Each job has its own call back function if needed, and these functions are layered on top of one another. This is called asynchronous programming. In Node JS, we can also implement synchronous programming, blocking IO paradigm, but the use cases are limited and it is considered an exception. Asynchronous programming helps Node JS to deal with many tasks with a single server, avoiding handling thread concurrency, which produces remarkable issues.

Let us take a look at a simple Node JS application using the HTTP module, which is in the huge standard library. Figure 3 shows the code for this application.



```
const http = require('http')

const hostname = '127.0.0.1'
const port = 3000

const server = http.createServer((req, res) => {
  res.statusCode = 200
  res.setHeader('Content-Type', 'text/plain')
  res.end('A Node JS Application using http module\n')
})

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`)
})
```

Figure 3. A typical NodeJS app

It is intuitive to understand the result from the code snippet. First of all, an http module is imported from the NodeJS library and is assigned to a variable called “http”. Hostname and port are declared so we can log the data in the terminal for presentation purposes. The method createServer() within the HTTP module is used to make a server. This server analyzes the request (req) and returns responses (res). We set the statusCode, setHeader, and the response with the data “A Node JS Application using http module”. Finally, the server listens on the specific port as well as hostname, and log data to the terminal. We end up with a server running in localhost (127.0.0.1) in port 3000. Messages are showing in the application and the terminal. It takes a little time to create the server so the server listens to the specific host and port first. Then when the creation is completed, the application calls the callback function, showing the message in the terminal. The benefits of this asynchronous programming were mentioned earlier, but there are also challenges. Contemporary CPU chips usually utilize many cores, 16 or even 64 cores, and contemporary server racks include numerous CPR chips. An intrinsic limitation of a single-threaded event loop lies in the absence of vertical scalability. A single-threaded program will be unable to efficiently use the 24+ cores available in a sturdy server rack.

Node JS covers a wide range of utilities, thanks to npm, the largest registry in the world. Npm is a package manager of thousands of open-source Node JS projects. It is considered a powerful library and can help Node JS application in installing different packages, managing software versions, managing dependencies. Let us assume that you desire to build this specific functionality within your personal project, and in common sense, you build it from scratch. But in the big world out there, someone has built ideally the same functionality. You would want to import that feature to your application and add several improvements to it. Npm acts as an open library for you to search for your need and install it to your application. Npm registry is currently holding approximately 1.2 million packages of free, open-source Node JS code snippets. New projects are uploaded to npm all the time, it is constantly evolving.

I list below a few stand-out packages that are stable and used by programmers all around the globe:

- **Express**: Enabling the creation of web servers with the simplest but most powerful methods. It becomes popular due to its unpretentious approach, which is focused only on the essential qualities of a server. I will have a dedicated section to discuss this web framework.
- **Gatsby**: A framework built on React and powered by GraphQL, generates static sites with an extensive ecosystem. It has all the advantages of static websites, helps you build blogs, dashboards, etc. It can retrieve data from any source and has good performance. You can develop many scalable Gatsby sites hosted only on Gatsby Cloud and related services.
- **Loopback**: Facilitates the rapid development of contemporary apps that need sophisticated interconnections.

- **Next.js**: A React framework that has great developer experience with a rich amount of features, such as supporting SSG and SSR, TypeScript support, route pre-fetching.
- **Socket**: A real-time bidirectional event-based communication that enables the development of network applications.

In summary, we discussed mostly about the advantages of Node JS, therefore, it should be applicable in:

- Applications that run in real-time
- Websites that utilize non-HTTP connection protocols: Those protocols can be run on top of the TCP protocol.
- Stateful sites: Caching is simplified with Node, it can be assigned to a global variable then all requests may use the cache. We do not lose external memory to store a client's state, we can share the state with other clients through the global variable.

Node JS should not be applicable in these cases:

- Resource-intensive applications: With the single-threaded working mechanism, Node JS encounters bottlenecks when handling huge files.
- When you have an inadequate understanding of Node JS: With non-blocking/async APIs, if you do not comprehend the issue, then you will face errors that you have no idea where it originated from.

2.3 Express

Express is the most commonly used Node web framework. It serves as the foundation for numerous widely used Node web frameworks. It enables the following:

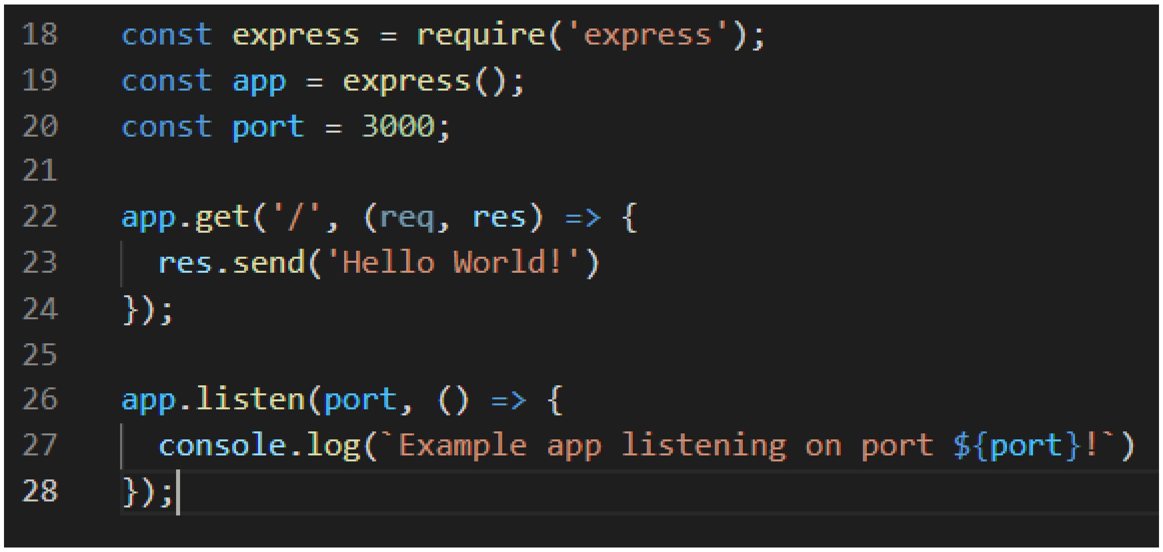
- Building handlers to handle requests that have a variety of HTTP verbs at different URLs.

- Integrating with “view” rendering engines to produce replies by populating templates with data.
- Configuring typical web application settings such as using this specific port and the location of the templates used to display the answer.
- Adding several middlewares in the process of analyzing the requests

While Express is a very simple framework just by itself, thanks to the middleware packages contributed by many developers, Express can utilize those to tackle many web development issues. Existing middlewares may be able to handle cookies, user login, sessions. But there are two sides to every coin, although practically all issues can be solved by different middlewares, choosing the most suitable one for your application is tougher since each has benefits and drawbacks. That is why there is not a single optimal solution for an application. Sometimes, the app does what it needs to be done but wastes resources Picking the right packages requires experience and experiments from developers. A deep understanding of the specific middleware is needed when building large, scalable projects.

Express was first launched in November 2010, approximately one year after the advent of Node, in 2009 for Unix users only. Since then, it has been used in many applications with 14 million weekly downloads. Express version at the moment is 4.17.1.

Figure 4 shows a simple code snippet with Express. Let us analyze it.



```
18  const express = require('express');
19  const app = express();
20  const port = 3000;
21
22  app.get('/', (req, res) => {
23    res.send('Hello World!')
24  });
25
26  app.listen(port, () => {
27    console.log(`Example app listening on port ${port}!`)
28  });
```

Figure 4. Simple code with Express

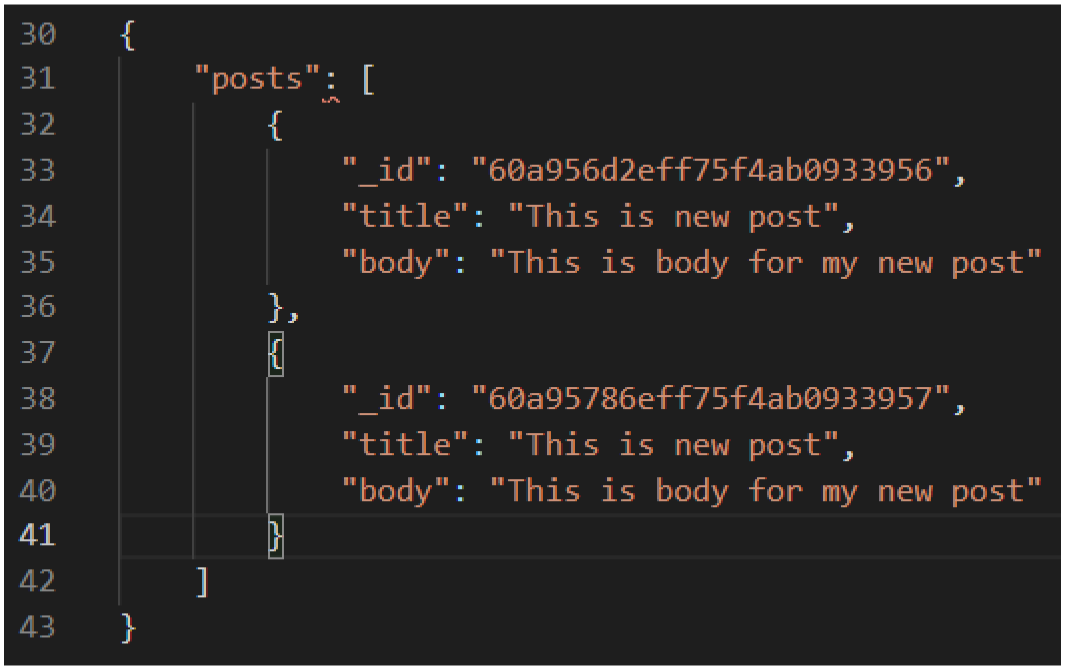
These lines of code result in a live server using Express and logging the port used to the terminal. Compared with the previous HTTP server created by Node JS, this has the same purpose. The notable point here is in line 22. This is a callback function, triggered when a HTTP GET request to the root of the site is received. It accepts the request and response by sending “Hello World!” with the method send(). This shows the get() method from express, moreover, there are post(), put(), delete() corresponding to create, update and delete. GET() method refers to read operation. Those operations are common in back-end interaction.

2.4 JSON and Postman

2.4.1 JSON

JSON which stands for JavaScript Object Notation is a data format that computers can understand and process. It may contain hierarchical data structures. It was released in 2001 by Douglas Crockford and gained popularity quickly at the time because of its advantages over other formats. JSON has JavaScript-like structures and notations. It is composed of name-value pairs that occur within the context of a parent-child relationship. JSON is widely used in millions of websites around the world at the time being. One popular way to communicate between webpages and servers, called AJAX, also uses JSON as the format for communication.

Figure 5 below denotes a typical chunk of data in JSON.



```
30  {
31      "posts": [
32          {
33              "_id": "60a956d2eff75f4ab0933956",
34              "title": "This is new post",
35              "body": "This is body for my new post"
36          },
37          {
38              "_id": "60a95786eff75f4ab0933957",
39              "title": "This is new post",
40              "body": "This is body for my new post"
41          }
42      ]
43  }
```

Figure 5. JSON file

As you can see, the data is concise, it describes some attributes of typical posts in a forum. This is a list of posts created by users, each post consists of a unique id to differentiate from other posts, a title summarizing the whole post in short, and the main body part of the post. In line 35, “body” is a name and “This is body for my new post” is a value, they exist together as a pair. The relationship between “post” and “body” is called parent-child relationship. This file is exactly the data transferred from server to end devices or vice versa. After being delivered to the end devices, some actions can be done on this data to showing it nicely to end-users. JSON only contains necessary value types, which is why it is lightweight and convenient. Even though the format was made based on JavaScript, the included value types are used in almost every single programming language. This is the reason why the format is language-independent and can be utilized to communicate between services.

Before JSON, the format being used at the time was XML. XML is more verbose compared to JSON, brings much more complication when using in a JavaScript-friendly data structure. The conversion to JavaScript object from XML takes hard work depending on the object. Meanwhile, with JSON, one line of code can do the trick.

Many advantages of JSON have been mentioned previously, let us discuss the disadvantages:

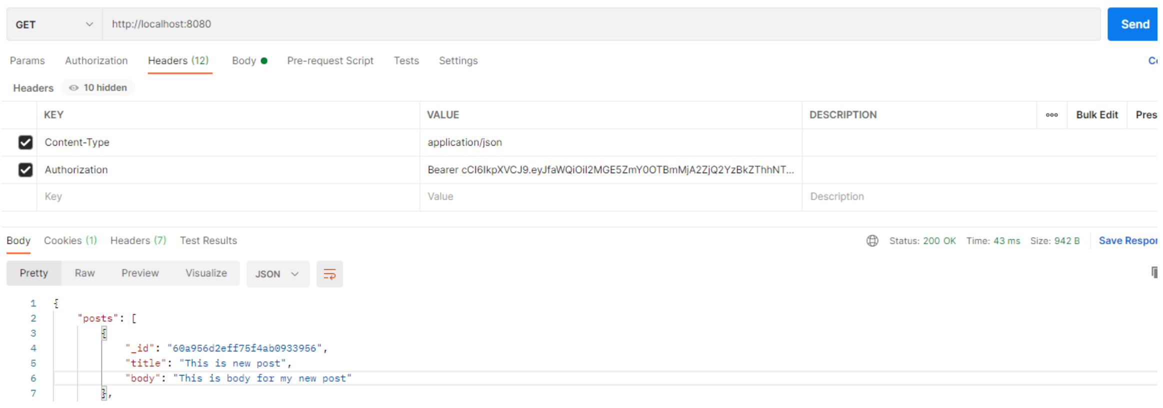
- There is no schema. It increases the chance of you creating malformed data.
- Concision: Though being more concise compared to XML, when used in a high scale project, issues happen. In this case, we have to rely on even more concise format.
- There is no comment. We have a higher probability of misunderstanding and may need extra support documentation.

2.4.2 Postman

Let us talk about what is Postman and what can we do with the application. Postman is a testing tool to check if API’s functionality behaves exactly as you expected. When you want to analyze the data sent to end-users by yourself or other people, that is when Postman comes in handily. Without Postman, you have to write many lines of code and format the data nicely enough for you to comprehend what is happening. It is a tedious chore because it is only for the purpose of testing, no one wants to spend time on that while we have an application like Postman.

For instance, when building an application, suppose I wanted to go to a specific URL to check if all the data I expect is there nicely. I need to perform an HTTP GET request to that URL. If I do not utilize Postman, I will need to create a new route and function to do the request, format the response how I want, display it to the console or where I can see and analyze. The process seems like developing the whole viable application around only that API while I only need to test it.

All the hard work can be performed simply through Postman. You choose the suitable type of request (in this example, it is GET), add the correct URL to the address bar. After that, probably you may put the needed keys and values in the header, indicate the response format to be “pretty” JSON. As a result, you can view the response in easily-readable JSON format along with a status code of 200, meaning successful, everything is working properly. Figure 6 shows the illustration.



	KEY	VALUE	DESCRIPTION
✓	Content-Type	application/json	
✓	Authorization	Bearer cCI6IkpXVCJ9.eyJfaWQiOiI2MGE5ZmY0OTBmMjA2ZjQ2YzBkZThhNT...	
	Key	Value	Description

Status: 200 OK Time: 43 ms Size: 942 B

```
{
    "posts": [
        {
            "_id": "60a956d2eff75f4ab0933956",
            "title": "This is new post",
            "body": "This is body for my new post"
        },
```

Figure 6. Postman example

All the commonly-used HTTP requests (GET, POST, PUT, DELETE) can also be performed on Postman.

2.5 UUID

Npm has been mentioned earlier as the package manager of Node JS. It is the reason why Node JS is powerful because it contains many useful and necessary packages. UUID is one of those packages. This package is used to generate short, random, URL-friendly, unique identifiers. These ids can be utilized as a key to hash sensitive information such as passwords, PIN codes, etc. They are generated with cryptographically strong values so they are trusted to protect our information. UUID is platform-independent and has more than 42 million weekly downloads.

Installation of the package is simple as many other packages found in npm. You type in “npm install uuid” then you will be able to import the package to your project and use it normally. Some usages of UUID may be listed:

- Changing an array of bytes to a string
- Generating a timestamp with uuidv1
- Generating a namespace using the less secure hashing algorithm MD5
- Making a random string with uuidv4
- Generating a namespace using the more secure algorithm SHA-1
- Validating a string

I plan to develop a social platform, and the platform cannot be alive if there is no user. Every time a new user registers an account on my system, he or she has to input a password directly. The password saved in my database should not be the one that the user typed in but the hashed version of it for the purpose of security. UUID helps me to create this hashed version.

2.6 JWT

JWT has been described in RFC 7519. It specifies a concise and self-contained means for sending information as a JSON object between devices. The send data is secure because it is signed digitally with the HMAC technique or with the public-key cryptosystem RSA. The tokens in JWTs can either be signed or encrypted. There is a slight difference between these two. While the encrypted ones conceal any claims that they contain, the signed ones can check the validity of such claims. If the token is signed with the public key cryptosystem RSA, the method makes sure that only the devices having the private key can sign the token.

JWT has three main parts. They are header, payload, and signature. Each part is separated with a dot. In the header, we have the signing algorithm (HMAC/ RSA) and JWT, token type. The payload holds the claims, that specify information about an entity. Claims are classified into three categories: register, public, and private. The signature is produced by the encrypted header, the algorithm given in the header, the encrypted payload, and a secret. Combining these three together and we will have a JWT, three Base4-URL strings with dots in between. The token is considered less verbose compared to XML-based standards.

JWT can be applicable in these cases:

- Transmitting information: JWTs are an excellent approach to secure communicating between devices. With the signed JWTs, we can validate the integrity of the data, as well as the identity of the senders.

- Authorization: JWT may be more widely applied in this case since we can permit what users do when they are already authenticated. After logging in, each future request will have the JWT in it, enabling the user to do only within the token limit.

2.7 Asynchronous programming

The code in JavaScript is executed in a non-blocking approach, same as in Node JS, that I referred to. This concept is called asynchronous programming. It is one of the foundation knowledge, therefore needed to be fully understood. To put it shortly, the program does not necessarily wait for a task to be completed, continue handling the consecutive task while the previous task is still being processed. The program returns to the previous task when it has received signals stating that the task is done. The section is dedicated to explaining asynchronous programming, before going further.

Callbacks were the original mechanism for dealing with asynchronous programming. They are functions positioned at the end of each synchronous task, called after the task has been carried out. This mechanism produces a few issues such as callback hell, inversion of control.

Later, developers built a solution to the aforementioned issues. The solution is called Promises. JavaScript users may rely on Promises to ensure two things:

- The “resolve” and “reject” functions force users to adhere to a certain convention.
- Connecting and aligning the callback functions, with the flow from top to bottom

The flaws in Promises lie in data accessing. Each task can only access data from either the immediately preceding step or parent scope.

The most recent approach to dealing with asynchronous programming is async/await. The approach appears in the ES2017 language specification. It places all related tasks under the same scope, therefore, the data accessing issue with Promises no longer exists. Using async /await is convenient, even though, it is also a double-edged sword. There is a higher chance of your code having performance loopholes if you use “await” wrongly. There is a blocking nature of the keyword and you should be aware of it.

3 DATABASE

When we refer to the term “database”, we are mentioning a place in the computer system used to keep a structured collection of data. Moreover, we need a database management system and supportive technologies to efficiently write or query data. All the related software aforementioned can be called database in short. There are many factors to consider when choosing an appropriate database for your application such as structure, the quantity of data, throughput, speed and scale. In my application, I have decided to choose MongoDB as the database.

3.1 MongoDB

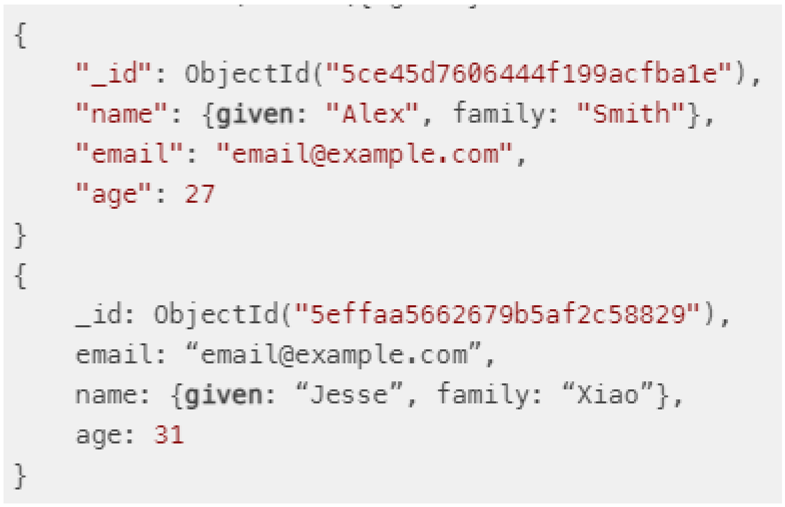
There are many categories when it comes to databases. They are mainly classified based on how they store information. The database can be relational, document-oriented, column-oriented, or use key/value, graph to keep the relationship between entities. MongoDB is a document-oriented database. It is considered the most widely used NoSQL database at the time being. The type said something about MongoDB, it uses collections of documents, each of which has key/value properties. In comparison with relational databases, one document in Mongo acts as a row in a table, with each key corresponding to a column name and key value corresponding to a particular cell in the table. What makes Mongo different is that one document is not restricted to a specific schema as the row in a table.

For example, my clothing store sells clothes and belts. There is a mutual property to every product is the price. But with belts particularly, I want them to have the entire different sets of property since they are different from clothes in terms of characteristics. MongoDB facilitates this feature. This is called dynamic schema change. Even though MongoDB has this advantageous point, it does not sacrifice any speed or resources. It remains high availability and adaptable to the amount of data. We do not need to worry about the scaling of our application in the future.

A few popular firms utilizing the database can be listed, Forbes, MetLife, CERN, Under Armour, etc. The primary use cases of MongoDB can be the following:

- **Administering and delivering data**: Handle a huge amount of data in a centralized data repository that enables fast modifications and quick responding.
- **Mobile and scalable projects**: These projects can make use of the advantages of MongoDB such as high availability, scalability.
- **Handling data from customers**: Dynamic schema changes allow MongoDB to handle sophisticated data models in large quantities.

Figure 7 shows an example of documents in MongoDB.



```
{
    "_id": ObjectId("5ce45d7606444f199acfba1e"),
    "name": {given: "Alex", family: "Smith"},
    "email": "email@example.com",
    "age": 27
}
{
    _id: ObjectId("5effaa5662679b5af2c58829"),
    email: “email@example.com”,
    name: {given: “Jesse”, family: “Xiao”},
    age: 31
}
```

Figure 7. documents in MongoDB

“_id” is used to differentiate documents in a big collection. “email” and “age” look good but pay attention to the “name” field. In relational databases, a new table is made up to hold these values (given, family). Meanwhile, with MongoDB, these values are embedded in the field “name” in the collection. This denotes one of the distinctions between MongoDB and other relational databases.

3.2 Mongoose

Both NodeJS and MongoDB use the same ODM. That Object Data Modeling is called Mongoose. Its responsibility consists of handling data associations, validating schemas and acting as a middle man so that there is a connection between objects in code and objects in MongoDB. Mongoose contains methods and functions to help NodeJS and MongoDB understand each other better. Figure 8 denotes the relationships between NodeJS, Mongoose, and MongoDB.

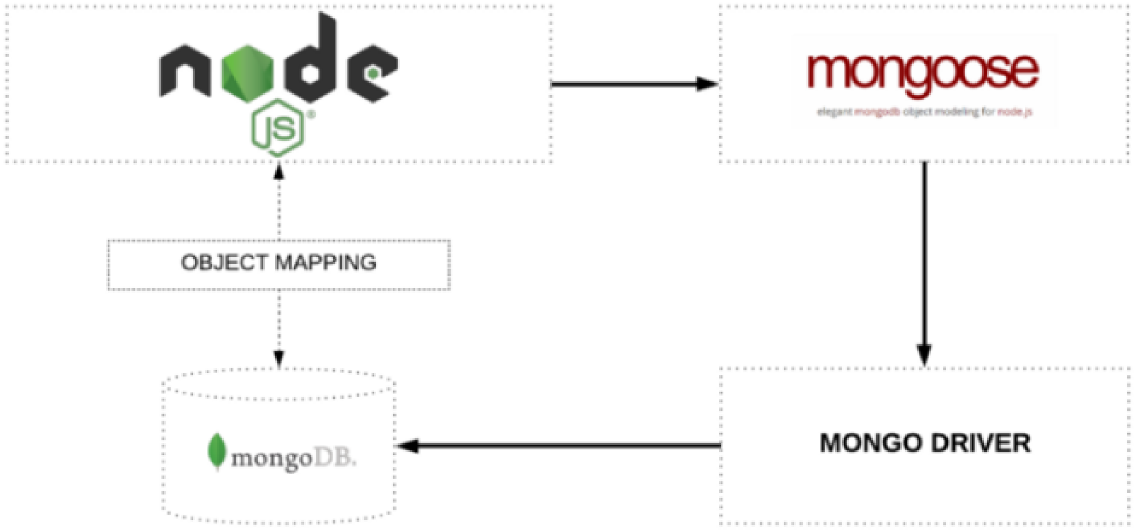


Figure 8. Relationships Node, Mongoose, Mongo

Basic operations of Mongoose include the following:

- Making record
- Retrieving data
- Updating data
- Deleting record
- Creating multiple helper functions and properties

Mongoose is a mighty library consisting of many helpful functions that assist us well while handling data models. An alternative to using Mongoose is to work on Mongo driver, but it is considered more complicated.

4 REACT JS

React is the most popular JavaScript framework used to develop user interfaces. The two alternatives are Angular and Vue. Each framework has its strengths and weaknesses. With React, you can create sophisticated user interfaces using tiny, independent code snippets referred to as “components”.

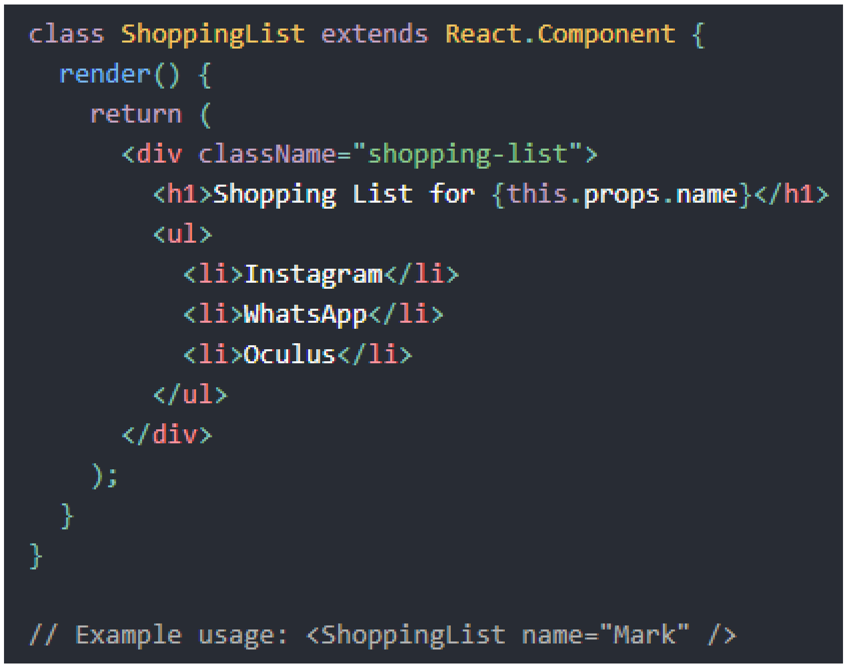
User interfaces mentioned are things that end users see on the device screen while using a webpage or an application, such as navigation bar, hamburger menu, buttons, etc. React was created to reduce the effort when working with these UIs so it is completely front-end related. React is used entirely for client-side programming. Prior to the advent of React JS, web programmers had to create user interfaces manually using plain JavaScript, also known as “vanilla JavaScript”. jQuery was the technology supporting developers at that time, but it is becoming deprecated because of more modern technologies. Old technologies tend to be more time-consuming and increase the likelihood of bugs. React JS was first released in 2011 by Jordan Walke.

Along with offering the reusable React code, there are two main aspects that make the React framework become more and more widely used:

JSX: HTML is considered the skeleton of any web page. The installed browser on your computer takes HTML files as input and produces websites on the screen as output. A Document Object Model (DOM) is constructed behind the scene, defining how the elements are combined in a website. Web programmers use JavaScript to change the DOM, making the application more interactive. JSX is created to facilitate this task. JSX can perform properly with cross-browser. Moreover, it makes the site having better performance and better efficiency.

Virtual DOM: Virtual DOM is generated when you use JSX to manage the DOM. As indicated, virtual DOM is a duplicate of the site DOM. The purpose of its existence is to keep track of the changes needed to apply to the site DOM when an event occurs. For example, on my Facebook page, one of my friends has recently commented on my profile picture. If React is not used, the whole web page needs to be refreshed to apply new changes. But since Facebook has been built on React, React locates the part of the DOM where changes were made and updates that part only instead of the whole page. This is the reason why React-based applications can have better performance.

Figure 9 shows an example of React.



```
class ShoppingList extends React.Component {
  render() {
    return (
      <div className="shopping-list">
        <h1>Shopping List for {this.props.name}</h1>
        <ul>
          <li>Instagram</li>
          <li>WhatsApp</li>
          <li>Oculus</li>
        </ul>
      </div>
    );
  }
}

// Example usage: <ShoppingList name="Mark" />
```

Figure 9. a React example

The class ShoppingList acts as a React component. One typical component accepts props as input and produces views for web pages as output. There is a render() method used to create the views. This method produces a more concise description of what needs to be rendered.

Let us compare React with another widely used front-end framework Angular. There is a distinction between the two. With React, you apply the needed methods in certain places. Meanwhile, with Angular, a frame has been made up for you so that you can insert framework code to designated spots. In a house, there are home appliances that help you not to do everything from scratch. These appliances can describe React. On the other hand, Angular is a model house frame which used as a template when creating the house.

With all the aforementioned features, React has benefits as well as drawbacks.

Benefits of ReactJS:

- Data changes can be applied conveniently.

- For someone who has solid JavaScript knowledge, with React, things become easier.

Drawbacks of ReactJS:

- The size of React is large, almost equals to Angular.
- The steep learning curve for people who are new to web development.
- Importing React components to MVC frameworks needs modifications.

5 PRACTICAL IMPLEMENTATION

I have decided to develop an application that can have basic functionalities of a typical social platform. There will be a sign-up system so that new users can join the platform and have their information kept in a database. After successfully signed in, users will be able to perform action in the application, for example, creating a new post. The goal of this implementation is to develop my own skill as a web developer, document my experience with new technologies, help others to have a general view on the chosen tech stacks.

5.1 Tools and technologies

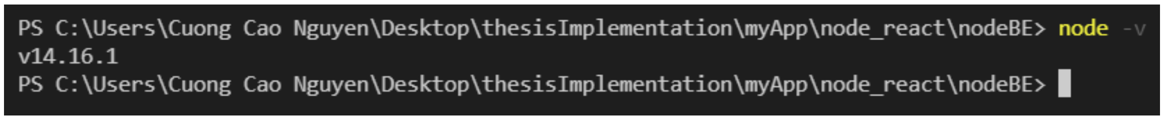
I have chosen MERN stack to build this application. It consists of MongoDB, ExpressJS, ReactJS, and NodeJS. NodeJS and ExpressJS are used to perform certain back-end operations and interactions. The data is stored in MongoDB, the most popular NoSQL database at the time being. ReactJS is responsible for the front-end operations, that are displayed to end users. The MERN stack is considered a modern stack and nowadays, more and more applications are developed using the stack.

About tools, we need a source code editor, a modern web browser, Postman and MongoDB Atlas. I use Visual Studio Code by Microsoft and Google Chrome for the first two tools. Postman and MongoDB Atlas are available on the Internet. Postman is used to perform API calls while MongoDB Atlas shows us what kind of data we are having in our database.

5.2 Back end

5.2.1 Preparation

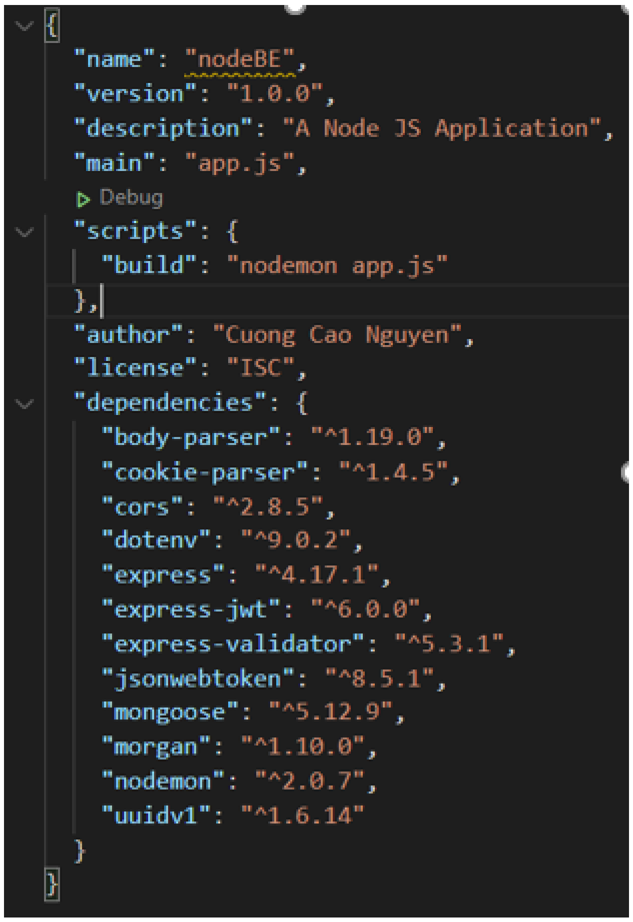
Let us work on the back-end related features first before handling the front end. First of all, we need to install NodeJS as it is one of the four core elements of the tech stack. It can be downloaded on its home page. You can verify if it is in your computer by this command (Figure 10). This shows the current version of Node you are having.



```
PS C:\Users\Cuong Cao Nguyen\Desktop\thesisImplementation\myApp\node_react\nodeBE> node -v
v14.16.1
PS C:\Users\Cuong Cao Nguyen\Desktop\thesisImplementation\myApp\node_react\nodeBE>
```

Figure 10. Checking nodeJS

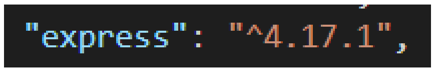
Whenever you want to build an app with NodeJS, you need to initialize the project with npm. Running npm init will do the work. It will add a few basic information of your project in package.json (Figure 11)



```
{
  "name": "nodeBE",
  "version": "1.0.0",
  "description": "A Node JS Application",
  "main": "app.js",
  ▷ Debug
  "scripts": {
    "build": "nodemon app.js"
  },
  "author": "Cuong Cao Nguyen",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "cookie-parser": "^1.4.5",
    "cors": "^2.8.5",
    "dotenv": "^9.0.2",
    "express": "^4.17.1",
    "express-jwt": "^6.0.0",
    "express-validator": "^5.3.1",
    "jsonwebtoken": "^8.5.1",
    "mongoose": "^5.12.9",
    "morgan": "^1.10.0",
    "nodemon": "^2.0.7",
    "uuidv1": "^1.6.14"
  }
}
```

Figure 11. package.json File

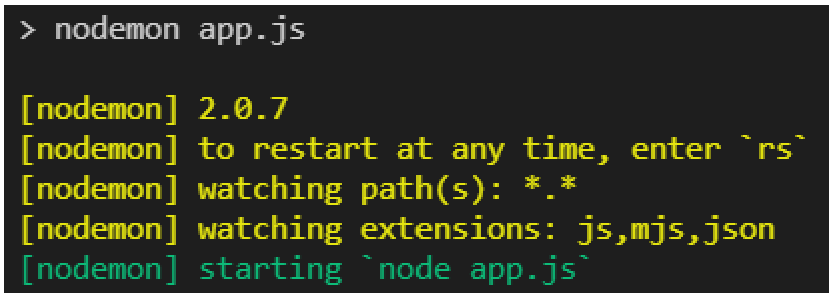
We need Express in our app since it is another foundation technology. Simply install it with one-line command, then, you can check if it is there in package.json, under ”dependencies” (Figure 12).



```
"express": "^4.17.1",
```

Figure 12. Express version

With Node and Express installed, you can test them by creating a simple server to see if it behaves properly. After running the project, the output may look like this (Figure 13).



```
> nodemon app.js

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
```

Figure 13. A running server

When working with a large project, it is necessary to group files into certain folder for better management as well as clear structure. The folders may be named as controllers, models, and routes. Models contain constructors and schemas. Routes consists of different URLs to certain sets of functions in Controllers. The relationship between these three can be describe as in figure 14.

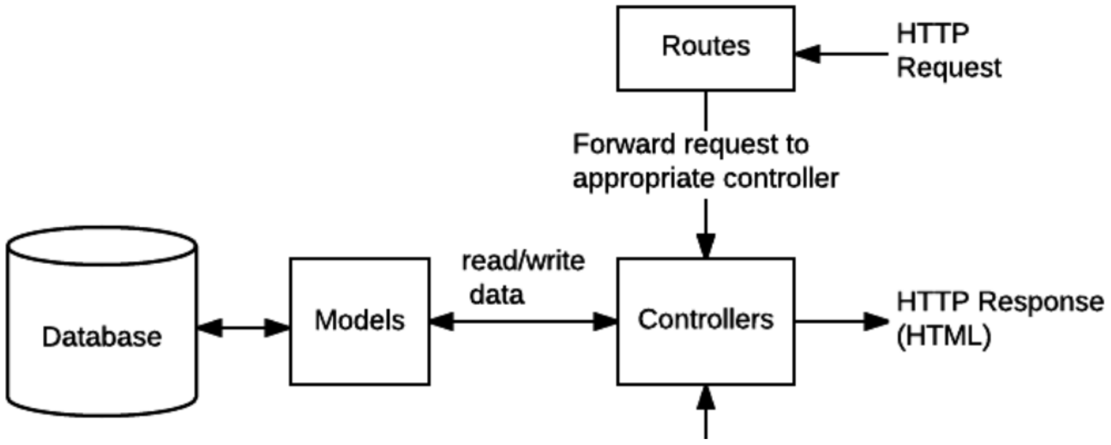
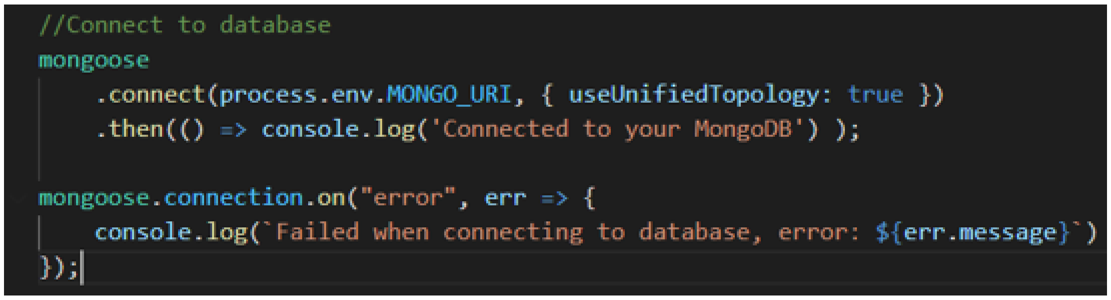


Figure 14. Relationships

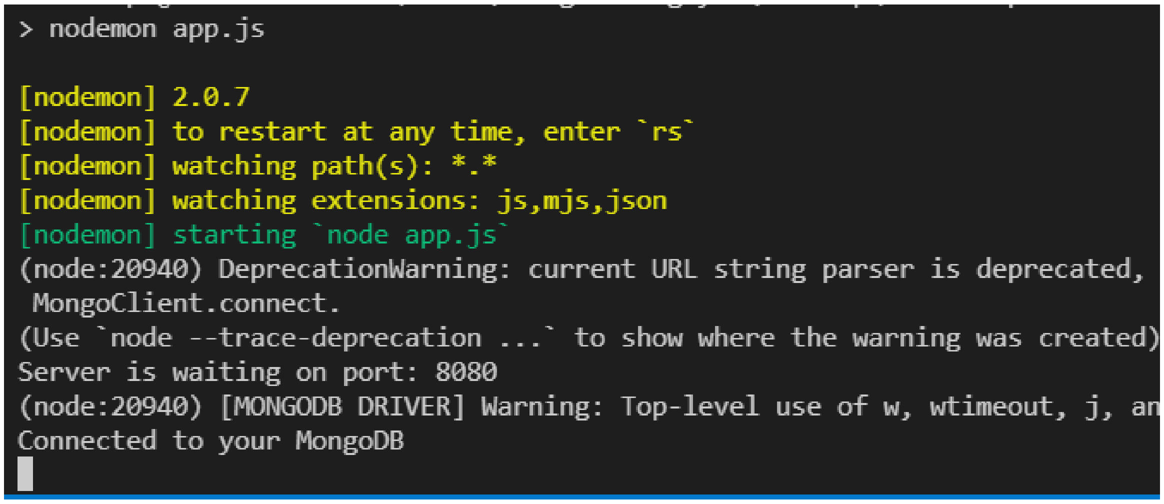
Your data is saved in MongoDB so let us establish a connection between the server and the database. Mongoose is utilized in this part because it contains necessary methods when working with MongoDB. You can use Mongo driver to interact with MongoDB but as mentioned previously, it is more convenient with Mongoose. Figure 15 shows the connection to database.



```
//Connect to database
mongoose
    .connect(process.env.MONGO_URI, { useUnifiedTopology: true })
    .then(() => console.log('Connected to your MongoDB') );

mongoose.connection.on("error", err => {
    console.log(`Failed when connecting to database, error: ${err.message}`)
});
```

Figure 15. Mongoose used to connect

The methods are self-explanatory based on their name. The server logs successful messages if connected to the database and displays error messages if failure happens. I connected to the database successfully (Figure 16).

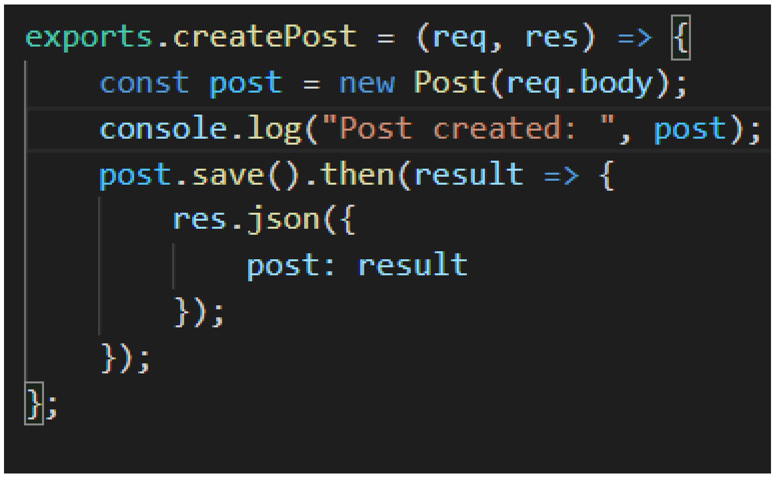


```
> nodemon app.js

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
(node:20940) DeprecationWarning: current URL string parser is deprecated,
 MongoClient.connect.
(Use `node --trace-deprecation ...` to show where the warning was created)
Server is waiting on port: 8080
(node:20940) [MONGODB DRIVER] Warning: Top-level use of w, wtimeout, j, an
Connected to your MongoDB
```

Figure 16. Connection with db

5.2.2 Developing features

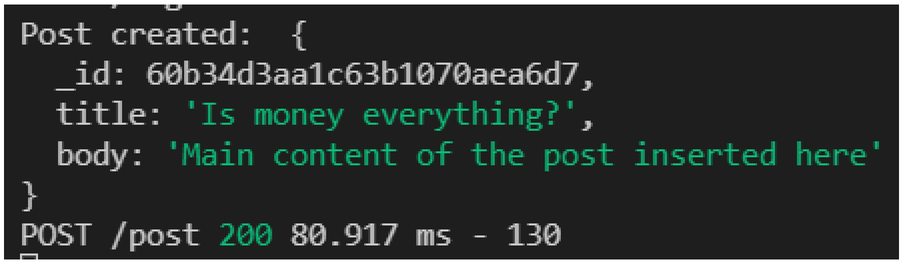
Now, let us think about what users can do when they are in our social platform. On Facebook, users can create their own post and upload to the social network to share with everybody. We may implement that feature in our application. To be able to create a post, one needs to build the structure for a typical post. It may consist of a concise tittle followed by the main content, same as the format of a letter. In “controllers” folder, we create a method called when user creates a post. This is my method to complete the task (Figure 17).



```
exports.createPost = (req, res) => {
    const post = new Post(req.body);
    console.log("Post created: ", post);
    post.save().then(result => {
        res.json({
            post: result
        });
    });
};
```

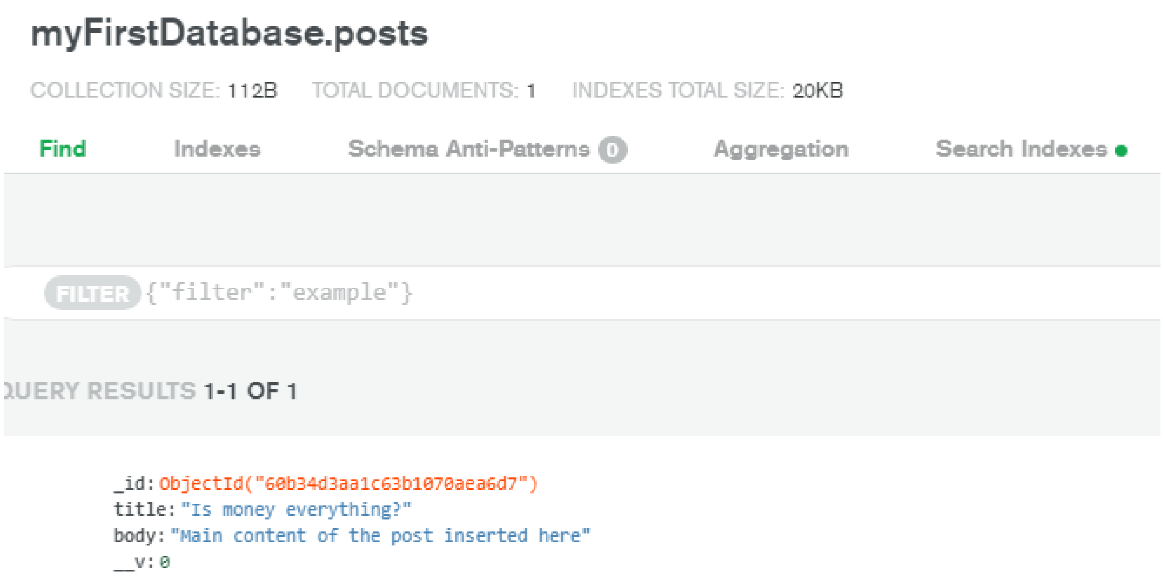
Figure 17. createPost method

We use the information comes to us from the user (req), create new instance of the Post model, log the created post to the terminal if success (Figure 18), and save the created post to the database. Figure 19 shows a new document added to the database from the user's input. Notice that they have the same ObjectId, title and body, stating the same unique post logged to the terminal and saved to the database.



```
Post created:  {
  _id: 60b34d3aa1c63b1070aea6d7,
  title: 'Is money everything?',
  body: 'Main content of the post inserted here'
}
POST /post 200 80.917 ms - 130
```

Figure 18. Post created successfully


myFirstDatabase.posts

COLLECTION SIZE: 112B TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 20KB

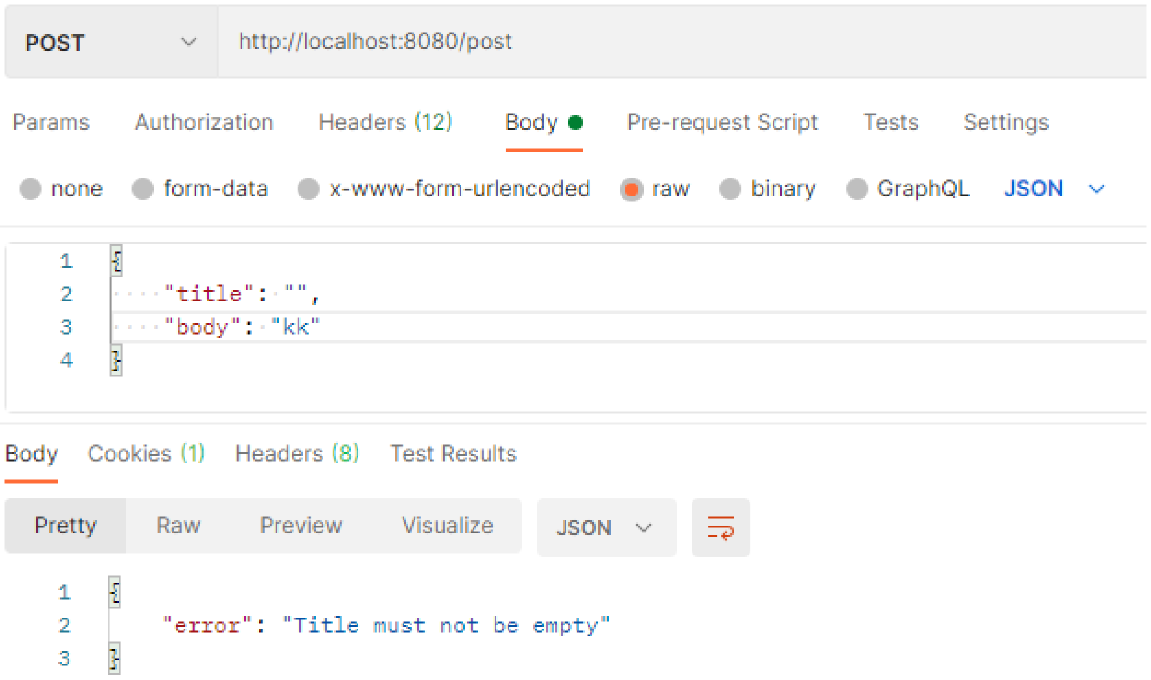
FILTER {"filter":"example"}

QUERY RESULTS 1-1 OF 1

```
_id: ObjectId("60b34d3aa1c63b1070aea6d7")
title: "Is money everything?"
body: "Main content of the post inserted here"
__v: 0
```

Figure 19. Post added to the db

I want to implement certain standards for my post. For instance, the title must always be inserted, not a blank space. The body should contain more than a number of character, otherwise it is not clear what the user is posting or to avoid spamming posts. The application can apply those standards (Figure 20).

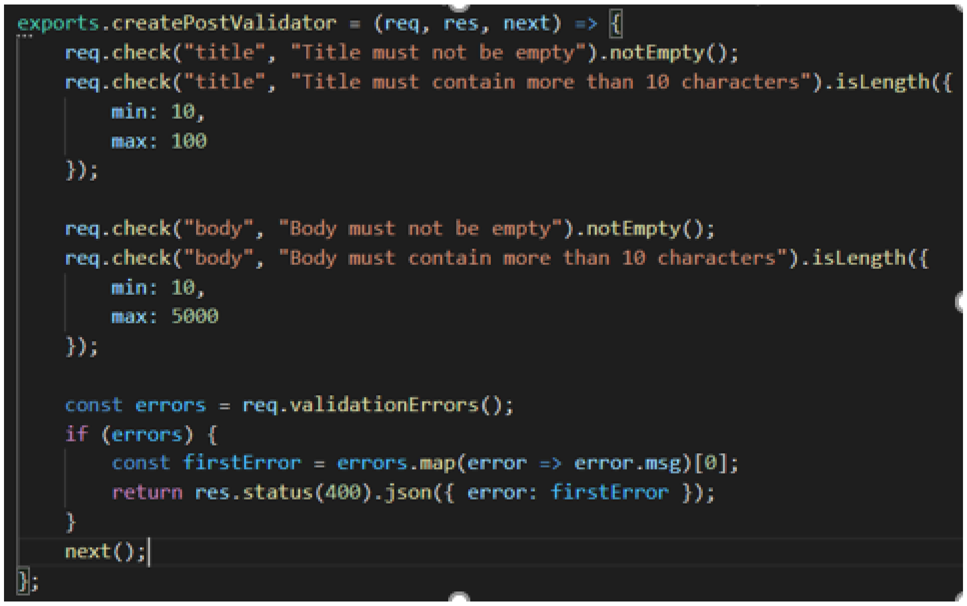

POST http://localhost:8080/post

```
{
    "title": "",
    "body": "kk"
}
```

```
{
    "error": "Title must not be empty"
}
```

Figure 20. Empty title not accepted

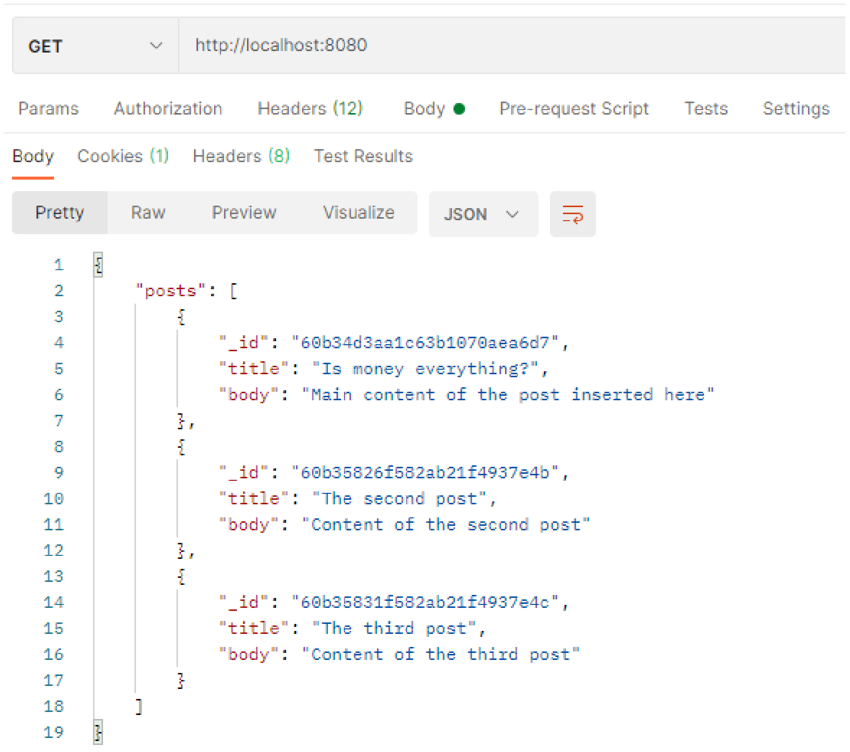
The method creating this behavious may look like this (Figure 21) and it is clear what each line of codes does.



```
exports.createPostValidator = (req, res, next) => {
    req.check("title", "Title must not be empty").notEmpty();
    req.check("title", "Title must contain more than 10 characters").isLength({
        min: 10,
        max: 100
    });

    req.check("body", "Body must not be empty").notEmpty();
    req.check("body", "Body must contain more than 10 characters").isLength({
        min: 10,
        max: 5000
    });

    const errors = req.validationErrors();
    if (errors) {
        const firstError = errors.map(error => error.msg)[0];
        return res.status(400).json({ error: firstError });
    }
    next();
};
```

Figure 21. createPostValidator method

All the created posts in the database should be visible to users when we access the social network (Figure 22).



GET http://localhost:8080

Params Authorization Headers (12) Body ● Pre-request Script Tests Settings

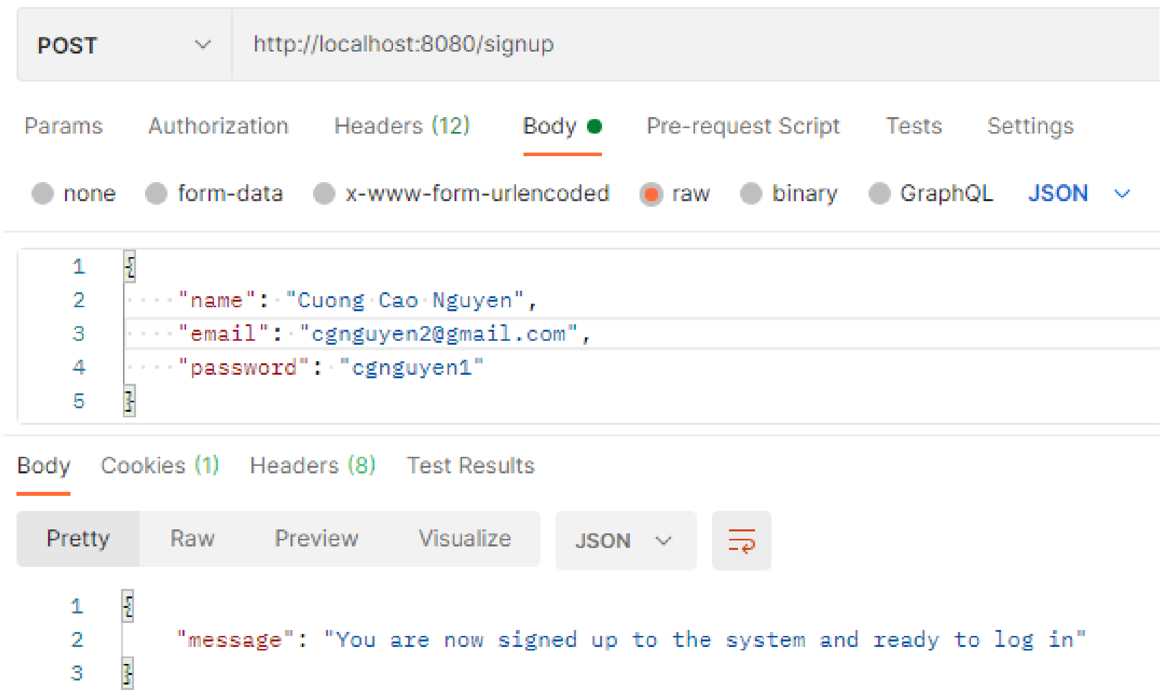
Body Cookies (1) Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "posts": [
        {
            "_id": "60b34d3aa1c63b1070aea6d7",
            "title": "Is money everything?",
            "body": "Main content of the post inserted here"
        },
        {
            "_id": "60b35826f582ab21f4937e4b",
            "title": "The second post",
            "body": "Content of the second post"
        },
        {
            "_id": "60b35831f582ab21f4937e4c",
            "title": "The third post",
            "body": "Content of the third post"
        }
    ]
}
```

Figure 22. Displaying post

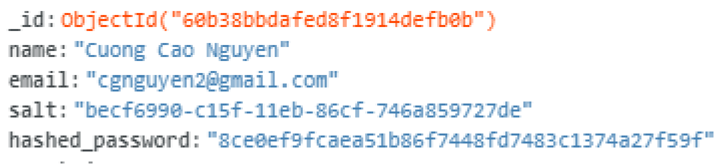
Every social platform has a signup system, so that new users can insert their information and have it saved in the database. Later on, we may provide certain privileges to the signed users. In widely used application, when signing up, you need to fill in name, email and password. The behavior looks as follow (Figure 23) and data should be saved in the database (Figure 24). The method making this feature is shown in Figure 25.



```
POST http://localhost:8080/signup
raw JSON
{
    "name": "Cuong Cao Nguyen",
    "email": "cgnguyen2@gmail.com",
    "password": "cgnguyen1"
}
```

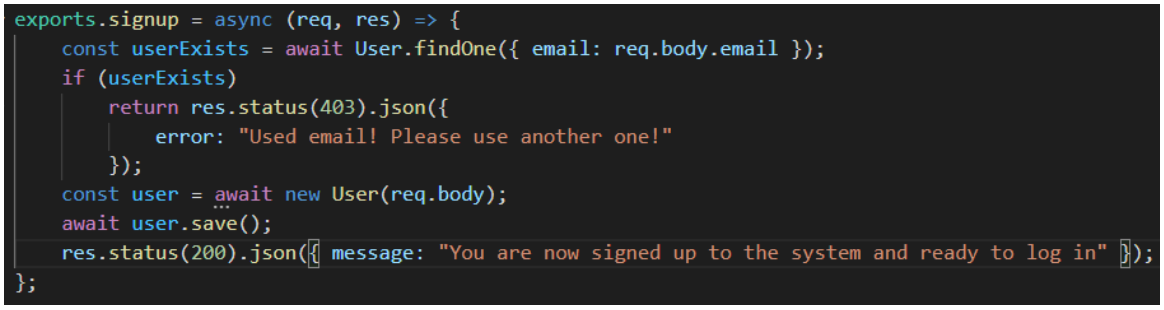
```
Body  Pretty  JSON
{
    "message": "You are now signed up to the system and ready to log in"
}
```

Figure 23. Sign up



```
_id: ObjectId("60b38bbdafed8f1914defb0b")
name: "Cuong Cao Nguyen"
email: "cgnguyen2@gmail.com"
salt: "becf6990-c15f-11eb-86cf-746a859727de"
hashed_password: "8ce0ef9fcaea51b86f7448fd7483c1374a27f59f"
```

Figure 24. New user in database

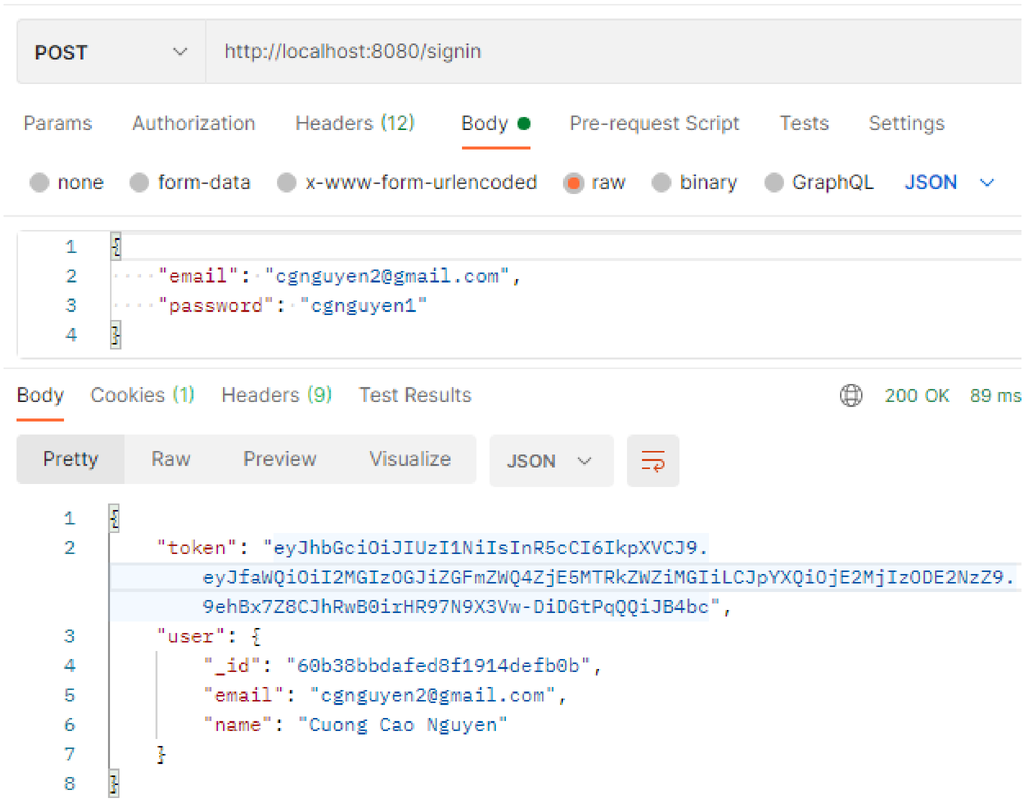


```
exports.signup = async (req, res) => {
    const userExists = await User.findOne({ email: req.body.email });
    if (userExists)
        return res.status(403).json({
            error: "Used email! Please use another one!"
        });
    const user = await new User(req.body);
    await user.save();
    res.status(200).json({ message: "You are now signed up to the system and ready to log in" });
};
```

Figure 25 signup method

The method takes input request and return a response. After user information is inserted in the front end, the app checks if the user is already existed by comparing the new email to the saved ones in database. If a match is detected, it prompts the user. If it is a new user, then it creates a new document in the database and saves it. The method is asynchronous because it takes some time to compare the emails and create new document in database.

After signing up and having information stored, users should be able to sign in the system (Figure 26).



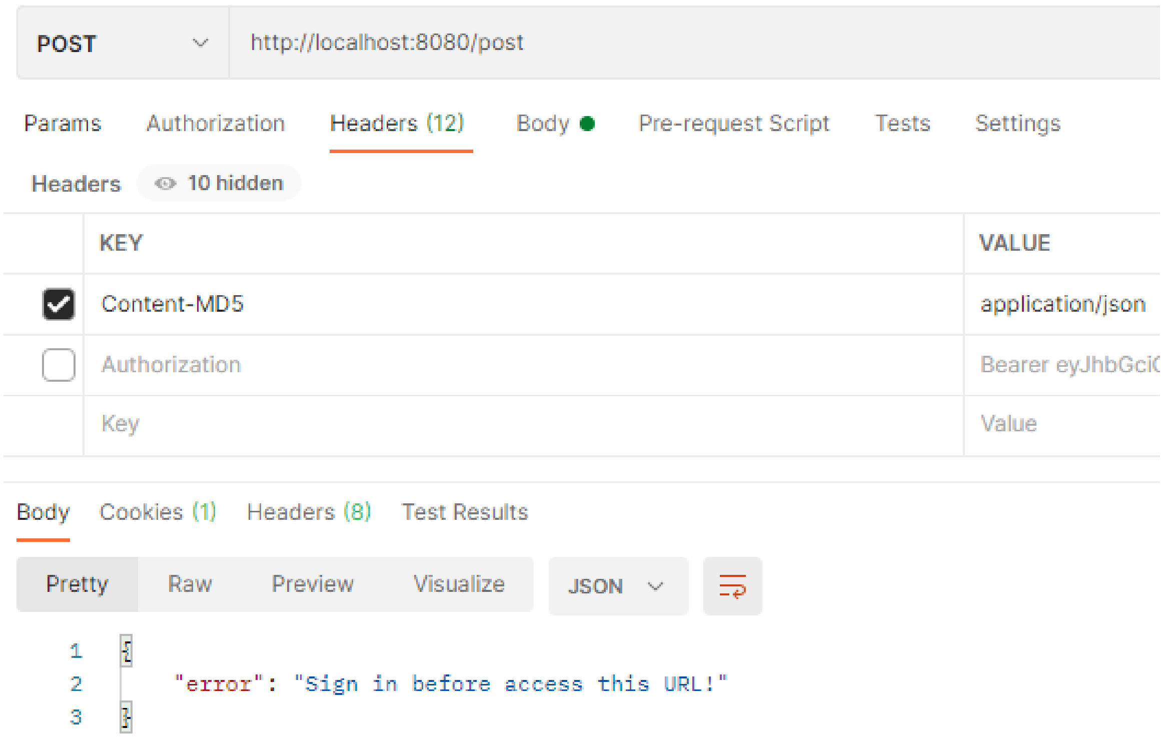
POST http://localhost:8080/signin

```
{
    "email": "cgnguyen2@gmail.com",
    "password": "cgnguyen1"
}
```

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJfaWQiOiI2MGIzOGJiZGFmZWQ4ZjE5MTRkZWZiMGIiLCJpYXQiOjE2MjIzODE2NzZ9.
        9ehBx7Z8CJhRwB0irHR97N9X3Vw-DiDGtPqQQiJB4bc",
    "user": {
        "_id": "60b38bbdafed8f1914defb0b",
        "email": "cgnguyen2@gmail.com",
        "name": "Cuong Cao Nguyen"
    }
}
```

Figure 26. Sign In

Inserting the sign up email and correct password will let the user sign in to the system. The client returns information about the signed up user and a token. This token is used to verify that the user is logged in and to provide permission for him or her to create a new post. That is the purpose of this sign in feature. We want only the signed users are able to create a new post. The users who are not signed, let us call them guests, receive this response when posting (Figure 27). People who are signed can create post normally (Figure 28).

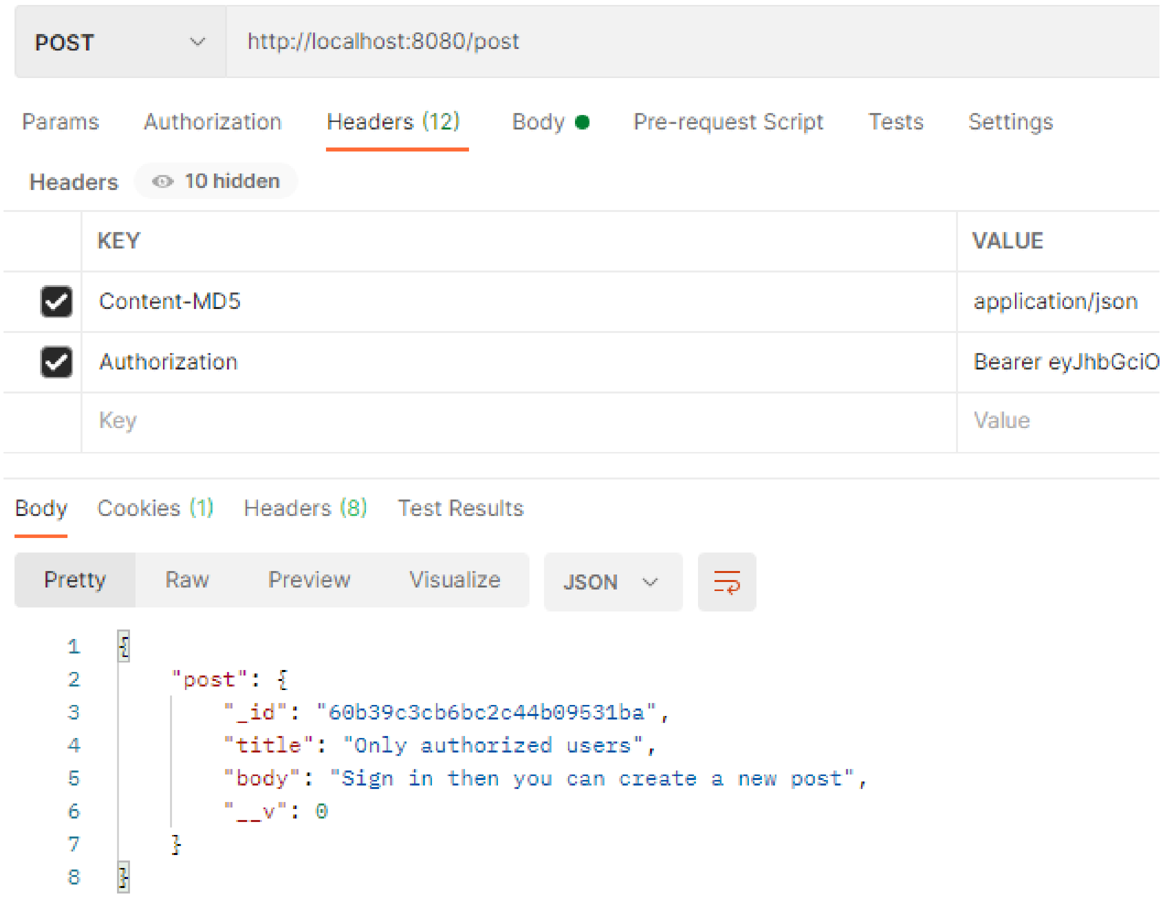


http://localhost:8080/post

	KEY	VALUE
✓	Content-MD5	application/json
☐	Authorization	Bearer eyJhbGciC
	Key	Value

```
{
    "error": "Sign in before access this URL!"
}
```

Figure 27. Prevent guests from posting



```
http://localhost:8080/post
```

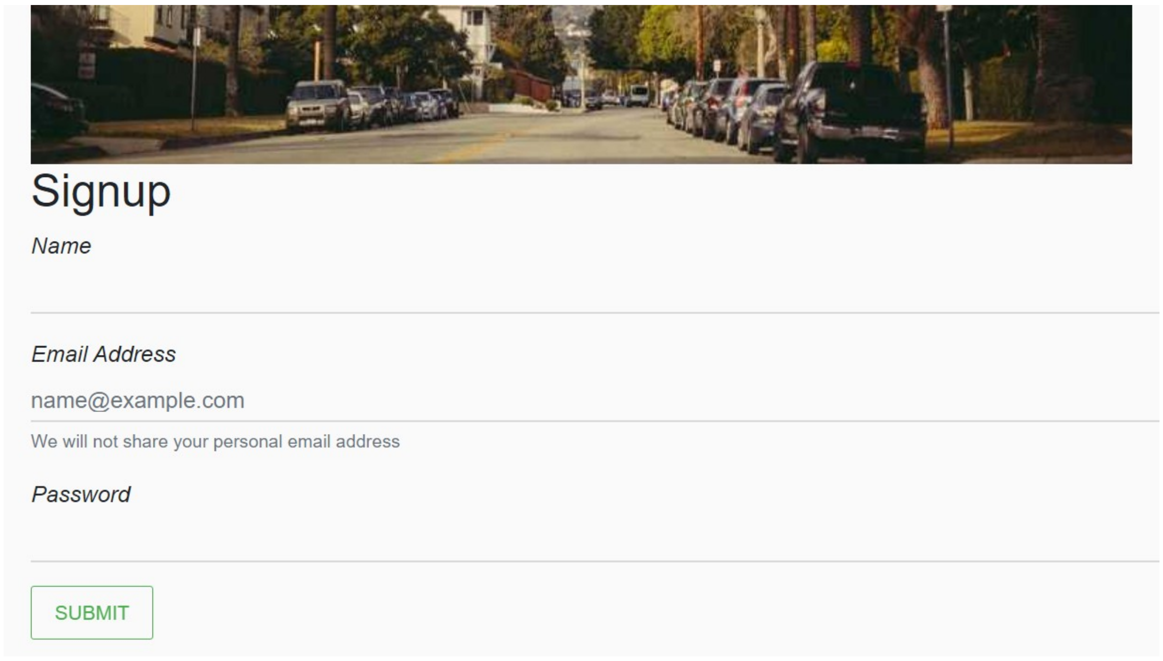
	KEY	VALUE
✓	Content-MD5	application/json
✓	Authorization	Bearer eyJhbGciO
	Key	Value

```
{
    "post": {
        "_id": "60b39c3cb6bc2c44b09531ba",
        "title": "Only authorized users",
        "body": "Sign in then you can create a new post",
        "__v": 0
    }
}
```

Figure 28 Users posting

5.3 Front end

Let us move to the front-end related stuff. We have touched three core elements in MERN stack. It is time to experience React. Previously, due to no front end, we had to rely on Postman to test all the back end functionalities. Normally, in a complete application, all the successful or error messages will be displayed directly to end users. We will display the sign up feature with React. First of all, we need to install React properly. Then, when signing up, we need name, email address and password from user so we use form to achieve this. Our form may looks as in Figure 29.


Signup

Name

Email Address

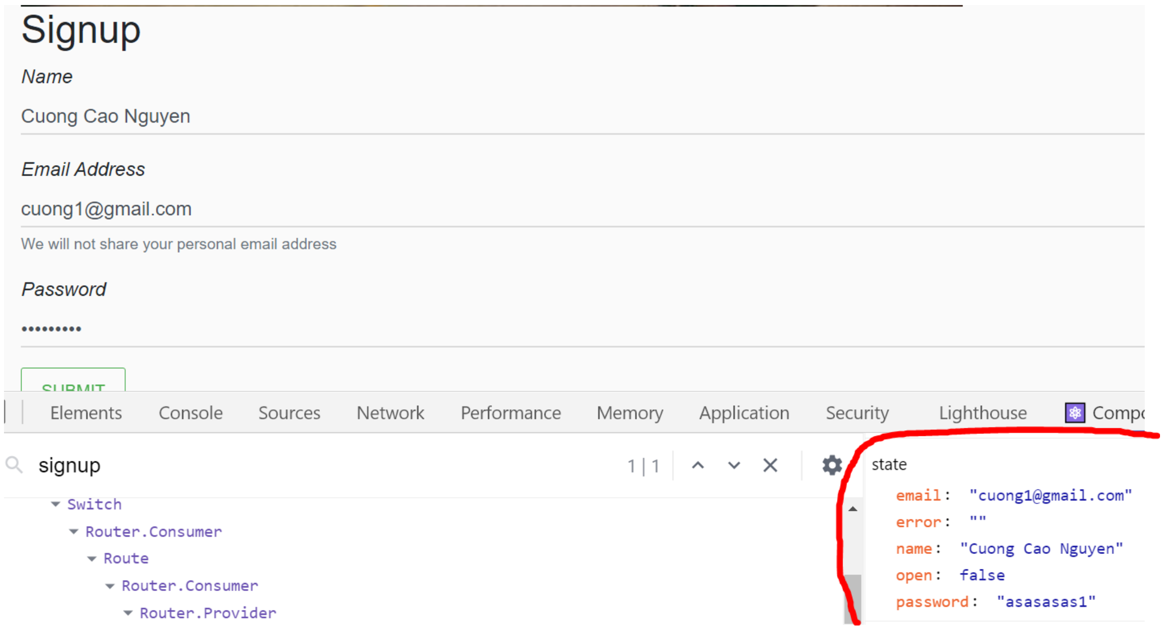
name@example.com

We will not share your personal email address

Password

Figure 29. signup form

Whenever a user type in the information, the app needs to save it in the current state. With React Developer Tools, we can see the data in the current state (Figure 30). This is done with a simple method (Figure 31).


Signup

Name

Cuong Cao Nguyen

Email Address

cuong1@gmail.com

We will not share your personal email address

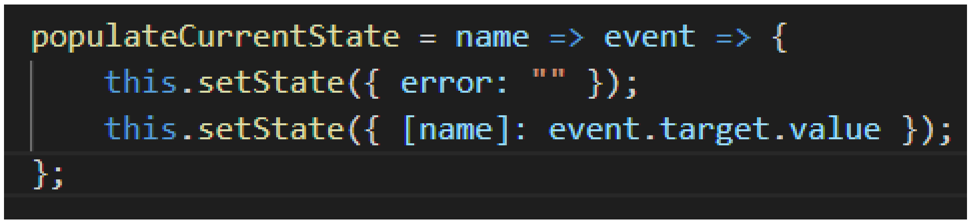
Password

•••••••••

```
▾ Switch
  ▾ Router.Consumer
    ▾ Route
      ▾ Router.Consumer
        ▾ Router.Provider
```

```
state
  email: "cuong1@gmail.com"
  error: ""
  name: "Cuong Cao Nguyen"
  open: false
  password: "asasasas1"
```

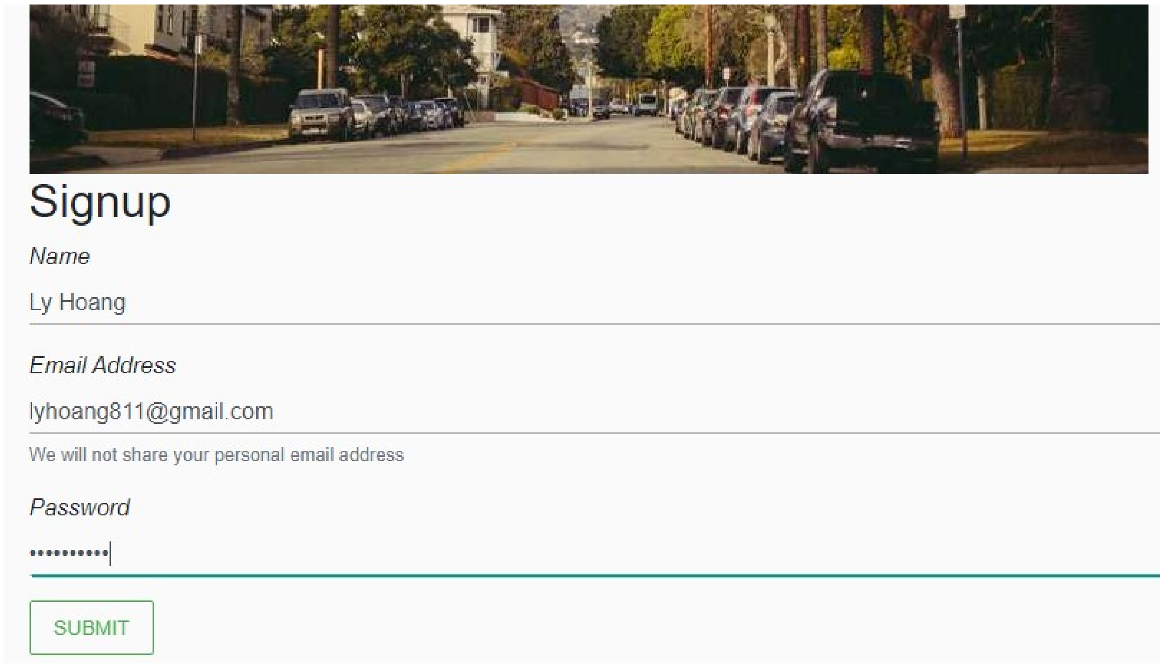
Figure 30 Populate the current state with data



```
populateCurrentState = name => event => {
    this.setState({ error: "" });
    this.setState({ [name]: event.target.value });
};
```

Figure 31. populateCurrentState method

Then, when clicking the Submit button, we want all inserted information transmitted to the database in the back end. The transmission is by a POST request to signup URL. After the request, a new user should exist in the database. Information is posted with the fetch() method that comes with the browser by default so no need to install any new package. There is a client HTTP API called Axios that does the similar task. When we submit all the necessary data in the form (Figure 32), there will be a new user created in the database (Figure 33).



Signup

Name

Ly Hoang

Email Address

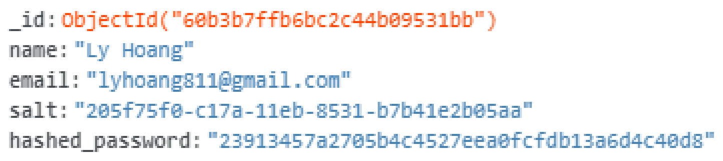
lyhoang811@gmail.com

We will not share your personal email address

Password

••••••••••

Figure 32. a new user signing up



```
_id: ObjectId("60b3b7ffb6bc2c44b09531bb")
name: "Ly Hoang"
email: "lyhoang811@gmail.com"
salt: "205f75f0-c17a-11eb-8531-b7b41e2b05aa"
hashed_password: "23913457a2705b4c4527eea0fcfdb13a6d4c40d8"
```

Figure 33 data in the database

6 CONCLUSION

The goal of thesis is to make several features of a social-platform-like application with the MERN technology stack including Node JS, Express JS, the database MongoDB and the front-end library React JS. The goal is considered achieved. Even though it is not a complete application, it has resulted in a document showing the process of developing a full-stack application. The paper showed that users could sign up, sign in to the application and have their information stored in the database. The signed users are provided certain permissions to create posts on the social platform.

The missing parts are rooms for further development. There are many features that can be added to the application in the future. For instance, basic operations such as creating, reading, updating and deleting entries in the database can be made. Users may have their own profile page that they are free to modify. Users may like, unlike posts and follow, unfollow other users. Moreover, users can be allowed to use their Google or Facebook account to log in to the application without signing up. The aforementioned features will provide better user experience.

REFERENCE

T.J. Degroat 2019. History of JavaScript. WWW document. Available at: 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.
[Accessed 1 April 2021].

Effectus Software Inc. 2019. Is Node JS a Framework? Learn about the popular platform. WWW document. Available at:
https://effectussoftware.com/blog/node-js-a-framework/
[Accessed 3 April 2021].

Node JS community contributors 2021. Introduction to Node JS. WWW document. Available at:
https://nodejs.dev/learn/introduction-to-nodejs
[Accessed 3 April 2021].

Martin Heller 2020. What is Node.js? WWW document. Available at:
https://www.infoworld.com/article/3210589/what-is-nodejs-javascript-runtime-explained.html#:~:text=is%20not%20easy.-,Node.,I%2FO%20requests%20to%20return.
[Accessed 3 April 2021].

MDN contributors 2021. Express/Node Introduction. WWW document. Available at:
https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction
[Accessed 8 April 2021].

Johnathan Freeman, 2019. What is JSON? WWW document. Available at:
https://www.infoworld.com/article/3222851/what-is-json-a-better-format-for-data-exchange.html
[Accessed 9 April 2021].

Students and Alumni of DigitalCrafts, 2019. What is Postman, and Why use it? WWW document. Available at:
https://www.digitalcrafts.com/blog/student-blog-what-postman-and-why-use-it
[Accessed 10 April 2021].

Npm community, 2021. Uuid. WWW document. Available at:
https://www.npmjs.com/package/uuid
[Accessed 12 April 2021].

Auth0 Inc. 2018. Introduction to JSON Web Tokens. WWW document. Available at:
https://jwt.io/introduction#:~:text=JSON%20Web%20Token%20(JWT)%20is,parties%20as%20a%20JSON%20object.&text=JWTs%20can%20be%20signed%20using,pair%20using%20RSA%20or%20ECDSA.
[Accessed 22 April 2021].

Sebastian Eschweiler, 2019. Async programming with JavaScript – Callbacks, Promises and Async/ Await. WWW document. Available at:
https://medium.com/codingthesmartway-com-blog/async-programming-with-javascript-callbacks-promises-and-async-await-980e3f144185
[Accessed 22 April 2021].

Michael Wilson, 2014. MongoDB Explained. WWW document. Available at:
https://www.credera.com/insights/mongodb-explained-5-minutes-less#
[Accessed 25 April 2021]

Nick Karnik, 2018. Introduction to Mongoose for MongoDB. WWW document. Available at:
https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/#:~:text=Mongoose%20is%20an%20Object%20Data,library%20for%20MongoDB%20and%20Node.&text=It%20manages%20relationships%20between%20data,of%20those%20objects%20in%20MongoDB.
[Accessed 31 April 2021]

Scott Morris, 2020. What is React JS? WWW document. Available at:
https://skillcrush.com/blog/what-is-react-js/
[Accessed 31 April 2021]

Facebook Inc. 2021. Intro to React. WWW document. Available at:
https://reactjs.org/tutorial/tutorial.html#overview
[Accessed 31 April 2021]