

Bachelor's thesis

Bachelor of Engineering, Information and Communications Technology

2020

Purna Baral

ROLE-BASED USER ACCESS CONTROL IN MERN STACK APPLICATIONS



BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Bachelor of Engineering, Information and Communications Technology

2020 | 55, 1

Purna Baral

ROLE-BASED USER ACCESS CONTROL IN MERN STACK APPLICATIONS

Web technologies have been rapidly evolving. They have evolved along with the development and advancement of web browsers and scripting languages. Development of modern browsers, cascading style sheets and open source concepts are the main reasons behind the latest revolution in web tools and technologies. Web technologies have come a long way from static HTML pages to complex, powerful, full-stack web applications.

Inside the application, there can be some sensitive data or content that should not be accessible to every user. It should be accessible to only authorized users. To tackle this problem, the concept of Role-Based User Access Control was developed.

The main objective of this thesis was to research the various aspects of Role-Based User Access Control and implement them in a MERN Stack application.

This thesis provides a brief induction to a current approach to web development. There have been several addition to the web development stack recently. The stack that has been used during the technical implementation of this thesis is the MERN stack. This thesis describes the general idea of Role-Based User Access Control in the MERN stack application. The Application that has been developed with this thesis elaborates how the Role-Based User Access Control can be implemented on the MERN Stack applications from scratch with the tools and libraries such as PassportJS, ExpressJS, JSON Web Tokens, and BcryptJS.

KEYWORDS:

Web Development, MERN Stack, Full-Stack Applications, Role Based User Access Control

CONTENT

LIST OF ABBREVIATIONS	6
1 INTRODUCTION	6
2 CLASSICAL WAY OF WEB APPLICATION DEVELOPMENT	8
3 MODERN APPROACH FOR WEB APPLICATION DEVELOPMENT	10
4 MERN STACK APPLICATION	13
4.1 Building Blocks of MERN Stack	14
4.2 MongoDB	15
4.3 ExpressJS	18
4.4 ReactJS	21
4.5 Features of ReactJS	21
4.5.1 Component Based	22
4.5.2 Declarative	22
4.5.3 Isomorphic	23
4.6 NodeJS	24
4.6.1 Node Package Manager	25
4.7 How MERN Stack Work?	25
4.7.1 Model	25
4.7.2 View	26
4.7.3 Controller	26
4.8 Strength and Weakness of MERN Stack Application	27
4.9 Strengths	27
4.9.1 Everything is in JavaScript	27
4.9.2 Data in JSON format	28
4.9.3 NodeJS runtime	28
4.9.4 Interactivity	29
4.10 Weaknesses	29
4.10.1 Consistency	29
4.10.2 Dependencies	30
4.10.3 Limited Computational Capacity	30
4.10.4 Difficulties to learn	31

5 ROLE-BASED USER ACCESS CONTROL	32
5.1 JWT	36
5.2 BcryptJs	36
5.3 PassportJS	37
6 APPLICATION ARCHITECTURE	39
7 SCALABILITY WITH MERN STACK ARCHITECTURE	41
8 TESTING	45
9 CLOUD HOSTING	48
10 CONCLUSION	49
10.1 Results	49
REFERENCES	50

APPENDICES

Appendix 1. Final View of Application

FIGURES

Figure 1. Web Development trends over the years.	9
Figure 2. Traditional Page vs Single Page Application Architecture.	10
Figure 3. Architecture of MERN Stack.	14
Figure 4. Components of MERN Stack.	15
Figure 5. MongoDB Object example	16
Figure 6. SQL Terms vs MongoDB Terms.	17
Figure 7. Database Scaling.	18
Figure 8. ExpressJS Architecture.	19
Figure 9. NPM command to install ExpressJS.	20
Figure 10. An example of basic ExpressJS Routing.	20
Figure 11. Web libraries and Frameworks trend.	21
Figure 12. React DOM update.	23
Figure 13. MERN application architecture	26
Figure 14. User Roles and privileges.	33
Figure 15. Mongoose Schema with user roles.	33
Figure 16. Express route for super admin and admin.	35
Figure 17. Hash and Salt for user passwords using bcryptJS.	37
Figure 18. Architecture for whole application.	40

Figure 19. Main routes used by ExpressJS.	42
Figure 20. Command to install memcahe.	43
Figure 21. NPM command to install memjs.	43
Figure 22. Memcache configuration used by main server.js file.	44
Figure 23. API testing with Postman.	45
Figure 24. Mocha and Chai testing results.	46
Figure 25. Mocha and Chai test code.	47

LIST OF ABBREVIATIONS

AJAX	A technique used by web applications to push and retrieve data asynchronously from server.
API	It is an Application Programming Interface. It is a set of functions to develop applications which can access data.
Base64URL	Encoding for files and URLs.
CLI	Command Line interface is a place where commands as texts can be processed in the computer.
CSS	Cascading Style Sheet is a tool, which styles HTML pages.
DOM	Document Object Model is the space in the browser where we can interact programmatically.
HMAC	Cryptographic message encryption code.
HTML	HyperText Markup Language is a skeleton of any web pages.
HTTP	HyperText Transfer Protocol which is used for transferring messages between browsers and servers.
I/O	It means input-output in NodeJS applications. It is a set of functionalities like giving command to application as input and application completing command as output.
JSON	It is a file format which means JavaScript Object Notation. It stores data as an object.
JWT	JSON Web Token is an encrypted token library which helps to perform token based authorization and authentication.
LAMP	Acronym for Linux, Apache, MySQL, PHP. A stack for building full-stack web applications.
MERN	Stack developed by the combination of MongoDB, ExpressJS, ReactJS and NodeJS.
MSSQL	It is a Microsoft SQL-based database.
MVC	Model, View, Controller is an application architecture.
NoSQL	It stands for Non relational or SQL based databases.
NPM	Node Package Manager is a package provider to the NodeJS applications.
OAuth	Open standard authorization framework.
OpenBSD	Security focused unix-like operating system.
OpenID	Open standard and decentralized authentication protocol.

ORM	Object Relational Mapper is a tool which helps to connect the data keys in relational databases.
PHP	HyperText Preprocessor is a web scripting language.
PWA	Progressive Web Application is a web application which can run in any platform and have offline browsing capabilities.
RBAC	Role-Based Access Control is a technique to control the user access according to the user roles.
REST	It is an acronym for Representational State Transfer which develops API to inbound and outbound data in JSON format.
RFC 7159	Standardization type for JSON format.
RSA or ECDSA	Strong Security focused digital signature algorithm.
SEO	Search Engine Optimization is a process of making web applications visible and accessible to the users via search engine.
SOC	Separation of Concern is a way of separating the individual entities within an application.
SPAs	Single Page Applications which are component-based and tied up in the root component within the single page.
SQL	Acronym for Structured Query Language which is a query language to communicate with the relational databases.
SQLite	It is a SQL-based database.
TDD	Test Driven Development is a technique where development task is guided by tests.
UI	The visible part of any software or application on browser or display, where user interacts. It means User Interface.
URL	Uniform Resource Locator is a web address.
WWW	World Wide Web is a combination of resources using HTTP.

1 INTRODUCTION

In recent times, if we look back and analyze the progress and revolution of web technologies, we can feel that web technologies have been developed well and provide cutting-edge performance. Due to advancements such as Responsive web design, Mobile-first web design in web tools and technologies, Mobile Web browsing has increased rapidly. According to the statistical report of Statista 2019, the Mobile web browsing holds the 52.2% of internet traffic (Topic: Internet usage worldwide, 2019).

Web technologies started with the evolution of the internet, especially the web. Then by that time, Static web pages were most commonly used on the web. Static web pages contain hard-coded content and information, e.g., text, images, videos, etc. on the web. Static means there is no database from where the web pages could interact with the real-time dynamic data. The content of the web page will not change programmatically. If it has to be changed somehow, then the programmer or the administrator of that page has to change it.

To make a web page dynamic, the concept of the database or the dynamic web pages evolved. In this process, there can be some sensitive data or the content that should not be accessible to every user. It should be accessible to only authorized users. To tackle this problem, the concept of Role-Based User Access Control was developed.

This thesis provides brief information on the modern approach to web development. There have been lots of addition to the web development stack recently. The stack that has been used during the technical implementation of this thesis is the MERN stack. This thesis describes the general idea of Role-Based User Access Control in the MERN stack application. The Application that has been developed with this thesis elaborates on how the Role-Based User Access Control can be implemented on the MERN Stack applications from scratch.

This thesis contains 9 chapters. These different chapters elaborate on the approach of research done during the thesis writing and how those findings have been implemented technically. From beginning to the end, the thesis follows the progressive pattern with

the introduction chapter at the start. The second chapter explains the classical approach of web development and the third chapter explains the modern approach of web development with the latest tools and technologies. The fourth chapter briefs the different components of MERN Stack along with its pros and cons in detail and the reason behind choosing it for the development. The fifth chapter dives deeper into the theoretical and practical implementation of Role-Based User Access Control. It describes the logical and algorithmic ideas behind the Role-Based User Access Control. The sixth chapter visualizes the whole application architecture and the logic flow. Seventh chapter describes the scalability of the MERN stack architecture. Eighth chapter contains the testing done for the application and its results. The process of cloud hosting or deployment of the application are explained in the ninth chapter of this thesis. The last chapter i.e tenth chapter contains the conclusion of this thesis.

2 CLASSICAL WAY OF WEB APPLICATION DEVELOPMENT

In 1989 Tim Berners-Lee, a software engineer invented the term World Wide Web, also known as WWW. This happened almost 20 years after the invention of the Internet. By that time, Berners-Lee recognized that the millions of computers all around the world could be connected through the internet. By October 1990, Tim specified HTML, URL and HTTP as the three fundamental technologies in which the web is developed.

Most of the web pages or websites developed by that time were static. They were simply static HTML pages interconnected with each other. They did not have a database connected to them. The web development mostly revolved around the templating with HTML. Client-side programming languages, such as JavaScript did not exist in that period, neither did server-side scripting languages such as PHP. In 1990 researchers from different parts of the world succeeded in inventing HyperText Markup Language (HTML). HTML had not various sets of features, it was mostly focused in the text. HTML did not have support for images, videos, positioning of the contents. Even though, then, HTML became the fundamental block of website development, developers and programmers around the globe used it effortlessly to build websites.

In 1993, the Mosaic browser was released. It changed the approach of website development. Mosaic introduced features like bookmarking, images along with text displayed on the website. It had support for different colors and it established the user-friendly browsing experience to its users. At that time the Mosaic browser became the future of the web development and the browser. After the Mosaic browser, the revolution began with the introduction of game-changers such as Flash video, video streaming, cookies, cascading style sheets (CSS), PHP, AJAX. All of these brought the richness and dynamic behavior in the websites. They provided lots of modern features to the users which brought the modernization of web development (Web Development History - Progress of Web in Detail - Intlum, 2018). Figure 1 visualizes the major web development achievement over the years (Intlum, 2018).

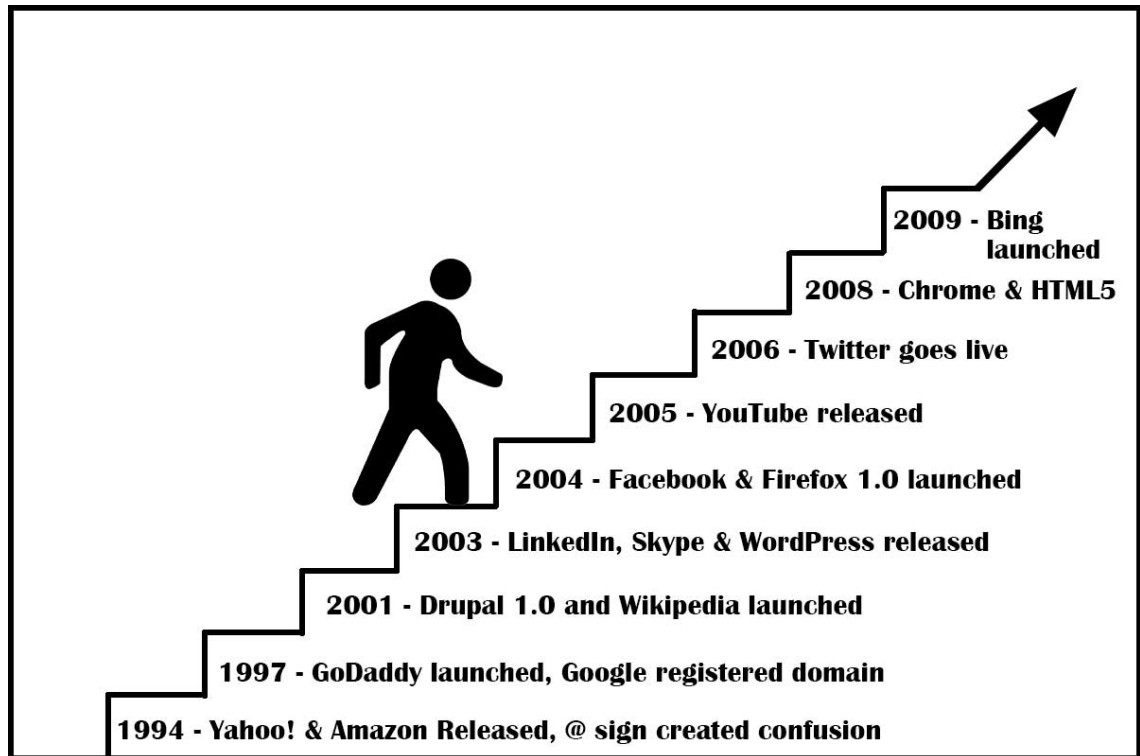


Figure 1. Web Development trends over the years.

3 MODERN APPROACH FOR WEB APPLICATION DEVELOPMENT

The modern approach of web development started with the initial release of the Mosaic browser. The mosaic browser was revolutionary then. It brought a different flavor to the contemporary web development practices. The introduction of web cookies in 1994 along with the Mosaic browser led to the best browsing experience for the users. Cookies are tiny memory storage in the browser, where users' preferences and choices can be stored. By utilizing cookies files, websites could provide an effective and optimized performance to its users. Even though cookies came under huge criticism as the service provider could track and monitor the user activities, it made a visible difference to modern web development.

In 1994, CSS (Cascading Style Sheets) was introduced by the Opera browser. It brought the colors and visualization to the HTML tags. CSS helped to separate the styling files from the website contents. CSS is a major milestone for modern web development (Intlum, 2018). Figure 2 differentiates the lifecycle of traditional page and single page (Diakogiannis, 2018).

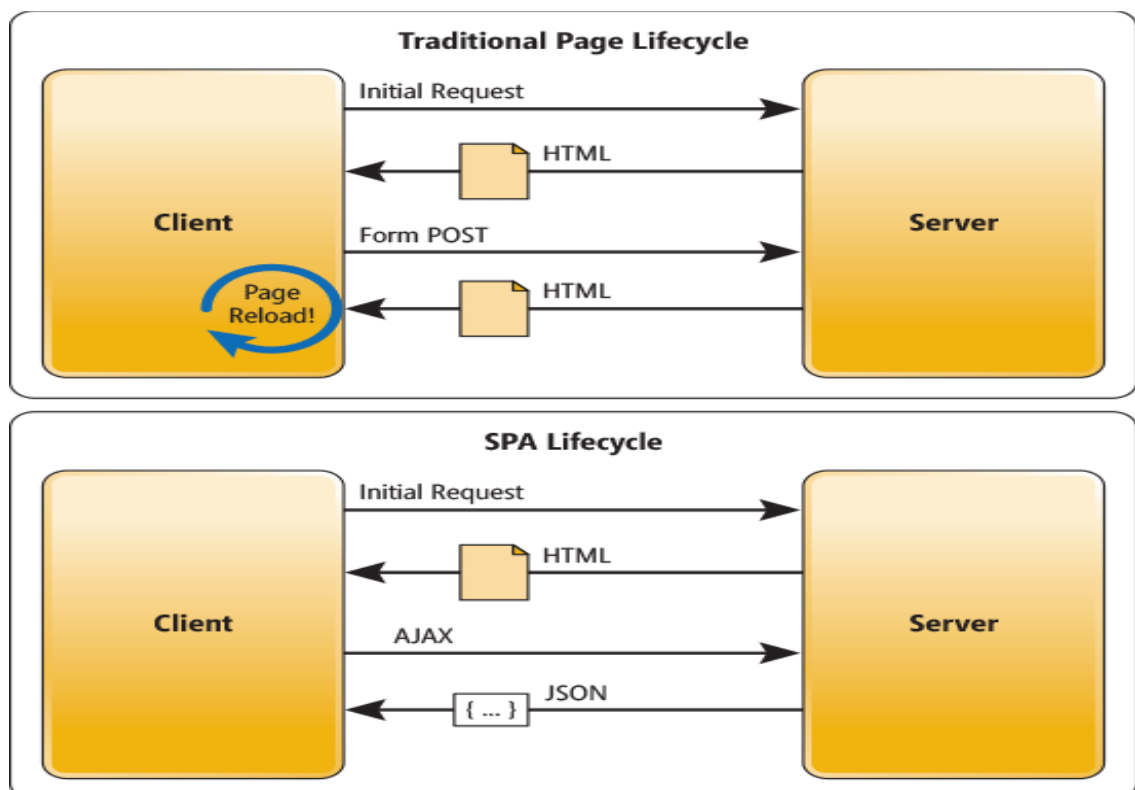


Figure 2. Traditional Page vs Single Page Application Architecture.

Researchers and developers wanted to add dynamic behavior to the websites. Before the release of PHP in 1996, all of the websites used to be static i.e. all of the contents are hardcoded. PHP added the dynamic behavior to the websites as it could be added to the server, connect to the databases along with SQL queries and it could be directly embedded with the HTML. At the moment, PHP runs in more than 80% of the websites globally. Along with its dynamic features, PHP became the major web development language.

JavaScript is Dynamic, high level and Object-Oriented programming language which operates mostly in the Client-side or Front-end. It was founded by the Brendan Eich in 1985 while working in the Netscape Corporation. The leading team in the Netscape organization decided that they want new programming or scripting language which should be lighter, faster, and should add the dynamic behavior to the web pages. Their target was to develop a scripting language which should be similar to the Java syntactically. Although they named and shipped it as LiveScript initially. After three months, they renamed it as JavaScript. Since its creation JavaScript runs in most of the modern, famous browsers like Google Chrome, Opera Mini, Internet Explorer, Mozilla Firefox, etc. Before the creation of JavaScript all of the web pages used to be static means, they did not have dynamic behavior. Dynamic behavior means a user can interact with the web page with the help of buttons, links, input fields, checkboxes, radio buttons, and the connection to the database. By then web pages used to consist of hardcoded contents like images, texts, and videos. After the introduction of JavaScript, it completely changed the way of developing and working on websites. JavaScript can add various behaviors, which can be experienced by Users to a static computer program by just adding new lines of code (Emery, 2016).

Along with the usage of JavaScript in web development, different organizations, tech companies developed various kinds of libraries and frameworks on top of core JavaScript make it more versatile, efficient, and easy to use. The development of libraries and frameworks made JavaScript more user friendly and famous among software engineers and web developers. The most used JavaScript libraries and frameworks are jQuery, Angular, React, Express, etc. Among the libraries and frameworks, jQuery is the most used JavaScript library. jQuery runs in more than 70% of websites worldwide.

In recent times, JavaScript has evolved a lot regarding its variety of libraries and frameworks. Most of the modern libraries and frameworks help to build and develop single-page applications (SPAs). A single page application means a single root page that contains various components embedded in it. Single-page applications are completely different from the PHP application or websites. PHP websites and applications contain page per view, but a single page application contains a single page and all the views are tied with the dedicated components and they are connected in one page. Single-page applications are faster because of less code and absence of multiple pages. The most famous Single page application libraries and frameworks are ReactJs, Angular, VueJs.

4 MERN STACK APPLICATION

In 2009, Ryan Dahl released the initial version of NodeJS. NodeJs is an open-source, cross-platform, modern JavaScript runtime environment that can compile and execute JavaScript code besides the web browser. NodeJs allows the developer to develop a Full-stack web application based on complete JavaScript codebase. Before the release of NodeJs, JavaScript was running only in frontend (client-side) i.e. the application view layer. By the introduction of NodeJs, JavaScript can run in the backend (server-side) as well. NodeJs has made JavaScript a complete solution to the development of full-stack applications. Stack means the combinations of different tools and technologies which can provide a complete solution for developing web applications. A web application can be developed by utilizing multiple libraries, frameworks, databases, query language. All these essential components make a stack.

A few years back, web development was not similar as it is now. The most famous stack to develop web applications as LAMP (Linux, Apache, MySQL, PHP) stack. LAMP stack played a very crucial role to make wonderful web applications. Along time, due to the necessity of the users and the evolution of technologies, the demand for powerful web applications has increased. Nowadays, we can do almost everything like playing games, online payment, shopping, creating animations, graphics, etc. with the web application. That's why the web application should leverage heavy user requests easily and smoothly. LAMP stack has also done most of the thing but the architecture and working principle are heavy. With the evolution of the Single page application, the LAMP stack has fallen behind. The single-page application only refreshes the change in the browser, not the whole page. This is the major strength of Single Page Applications. If there are any changes in the page, the LAMP stack refreshes the whole page and it makes the performance slower.

MERN stack is a combination of MongoDB, ExpressJs, ReactJs, NodeJs. All of the components of this stack are open-sourced. When the single page web application became famous, the MERN stack rose to popularity. It can develop high-quality web applications with rich user experience and smoother performance. Figure 3 visualizes the architecture of MERN stack with client-side, server-side and database.

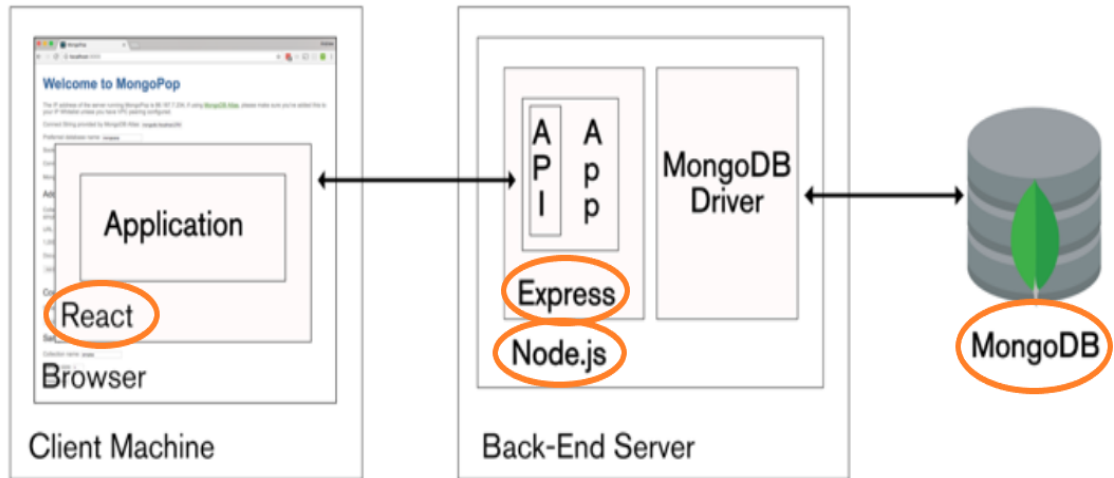


Figure 3. Architecture of MERN Stack.

4.1 Building Blocks of MERN Stack

The term “MERN” is an acronym for the combination of four stacks i.e. MongoDB, ExpressJs, ReactJs, NodeJs. All these building blocks are developed, maintained, and upgraded by the community of developers all around the globe. This is one of the main reasons that’s why it is famous among the developers. Another important aspect of using the MERN stack for development is that it uses JavaScript through and through. It makes development effortless. It is easier to learn as only one development language is used. Beginners can easily learn it and develop full-stack applications in no time. Figure 4 explains the connection between the components of MERN stack (How do MERN stack technologies work together?, 2020).

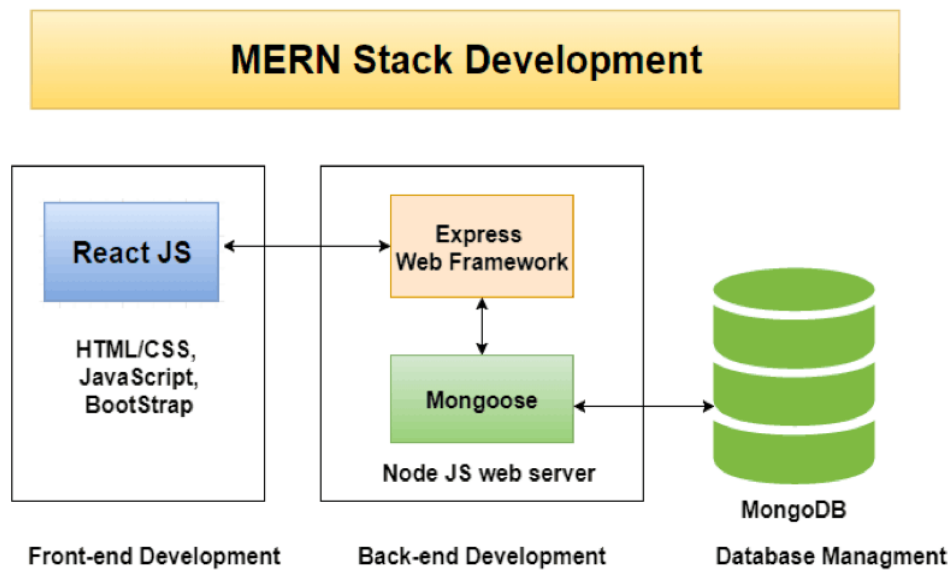


Figure 4. Components of MERN Stack.

4.2 MongoDB

MongoDB is one of the major components of the MERN stack. It is a cross-platform, open-source, NoSQL, asynchronous database management system. It is a document-based database as it saves data in collections and documents instead of rows and tables like in relational SQL databases. Technically, MongoDB saves data as JSON (JavaScript Object Notation) like objects. MongoDB stores data as dynamic schemas, which makes it more scalable and easier to use. Due to its asynchronous nature, it fits very well along with the NodeJs for development. Below is an example of a MongoDB document object.

The filed like address can vary from document to documents, as MongoDB provides the flexibility to change the data structure over the requirement of the users. MongoDB is a strongly document-based database. Let's assume the documents that we store in the drawer or table. We just keep the documents therein specific places. If we need to find one document, then we open the specific drawer or the place in the table. The same principle applies to MongoDB. It perceives and stores all the data as documents. That's why it is called a document-oriented database (Apress, 2017). Figure 5 is an example of MongoDB object.

```
{
  _id: "3dfd65jkdjf85ls2i439",
  firstName: "John",
  lastName: "Doe",
  phoneNumber: "+358123456789",
  address: {
    street: "Yö-kylä",
    city: "Turku",
    state: "South-west",
    zipCode: "20540"
  }
}
```

Figure 5. MongoDB Object example

The MongoDB document model maps to the objects in the application code as it makes easier to work with the data. MongoDB stores data as an array of objects. Each element of that array of objects is called a document. As we noted earlier, it has a flexible data structure, so that two documents can have different sets of properties. Due to this MongoDB is schema-less and inconsistent. The multiple documents are called “Collection” collectively in MongoDB, which is similar to the “table” in the relational database. Every document in the collection has a unique 12-byte identifier as “_id”, which is indexed automatically while creating the document. By using this “_id”, we can access the particular document from a certain collection. Figure 6 compares the relevant SQL and MongoDB/NoSQL terms (It’s high time to say goodbye to LAMP Stack and hi to MEAN Stack, 2019).

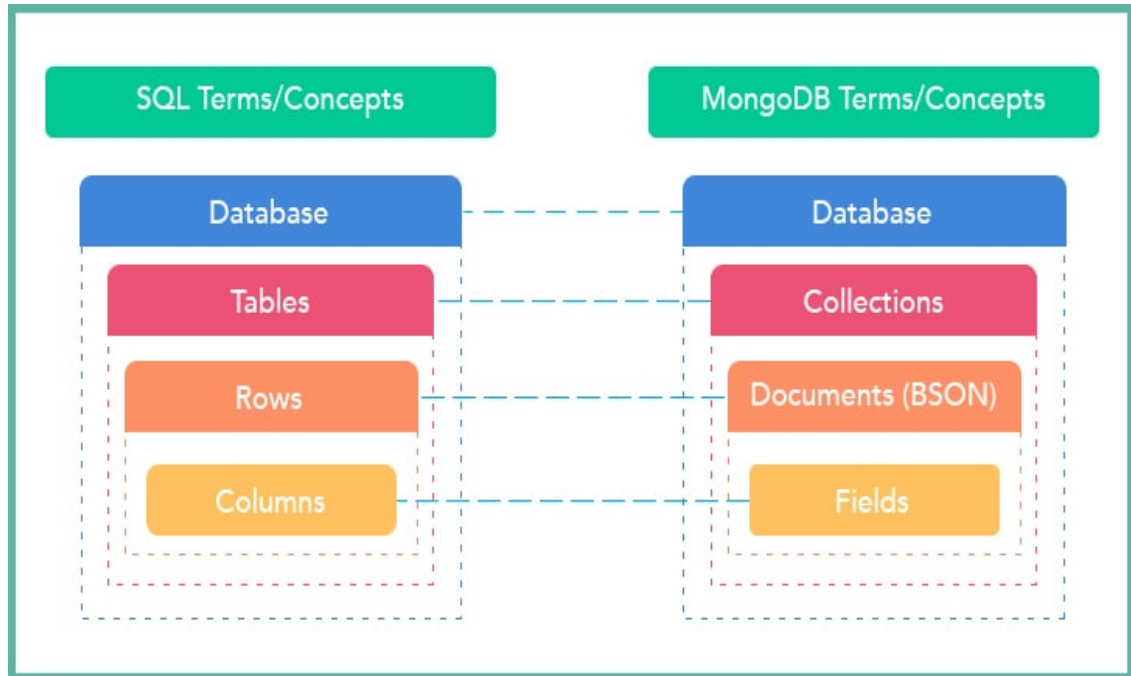


Figure 6. SQL Terms vs MongoDB Terms.

At the core, technically MongoDB is a distributed database system, which means it is highly available, geographically distributed, strongly supports horizontal scaling. Especially, horizontal scaling provides the strength for MongoDB to operate. Horizontal scaling distributes the loads that come to the database, over the multiple servers. But horizontal scaling has one bad aspect if we compare with the traditional relational database, i.e. consistency. MongoDB lacks consistency in this aspect but in reality, very few real-time applications need strong consistency. These attributes make MongoDB easy to use. Figure 7 explains the difference between vertical scaling and horizontal scaling (NoSQL Tutorial: Learn NoSQL Features, Types, What is, Advantages, 2018).

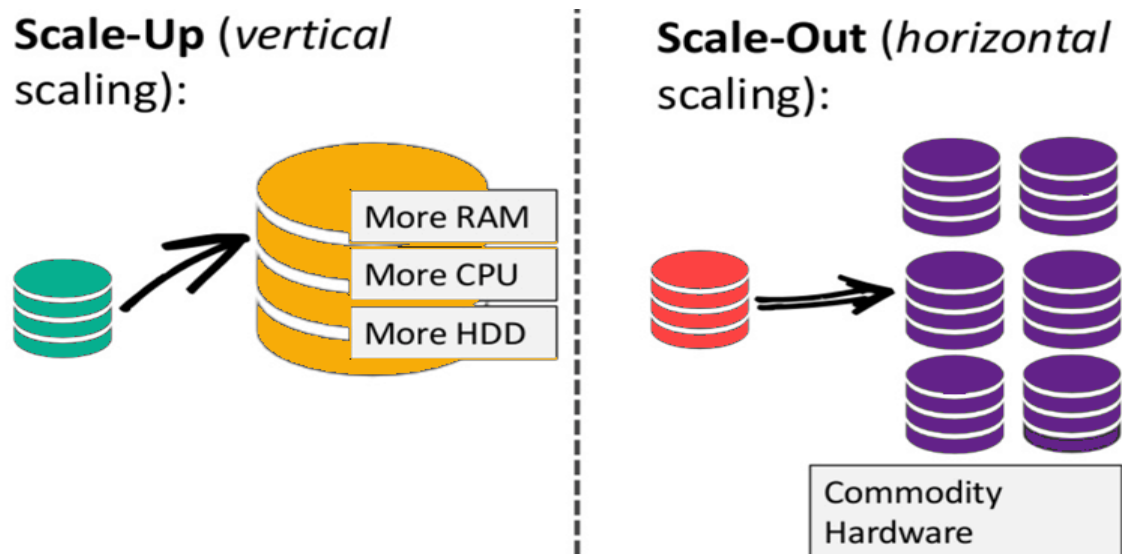


Figure 7. Database Scaling.

4.3 ExpressJS

ExpressJs is another important component of the MERN stack. It represents the “E” part in the MERN stack. ExpressJs is a flexible and minimal web application development framework with rich sets of features to build a robust and powerful web and mobile applications. Building a full-fledged web server with a complete JavaScript code is not an easy task. To solve and simplify this thing, ExpressJs becomes a handy tool to execute it. ExpressJs runs modules in NodeJS run time environment. It enables JavaScript code to run on the webserver. ExpressJs especially provides a solution for the backend part of the application. ExpressJs handles the multiple requests from users or the frontend part and allocates the respective requests with the response from the server accordingly. ExpressJs also helps to guide the request from the frontend part as routes. Routes provide the way form the incoming request to the server and outgoing response from the server. ExpressJs efficiently manages the routing (Apress, 2017).

There are many applications of ExpressJs depending upon the domain and context as it is a very flexible framework. One of the main applications is that ExpressJs executes the code that controls the application’s business logic and responds to the HTML file to be compiled by the web browser. Another important and most widely used and accepted feature of ExpressJs is building REST API. REST API provides the gateway for the frontend application to access and communicate with the backend server. REST means “Representational State Transfer”. It is an architectural style that provides styles and

rules to develop web services. Any services running in a computer or electronic devices that can be accessed and communicated by internet protocols like HTTP are web services. Web services that follow the REST API architectural rules and patterns are called RESTful web services. REST API provides stateless protocol as it aims for faster performance, reliability, simplicity, scalability. REST API communicates with the data in JSON format. REST API takes data requests in JSON format and responds to the data in JSON format. It makes easier to store data in the MongoDB database. Figure 8 explains the architecture and data flow of ExpressJS framework (It's high time to say goodbye to LAMP Stack and hi to MEAN Stack, 2019).

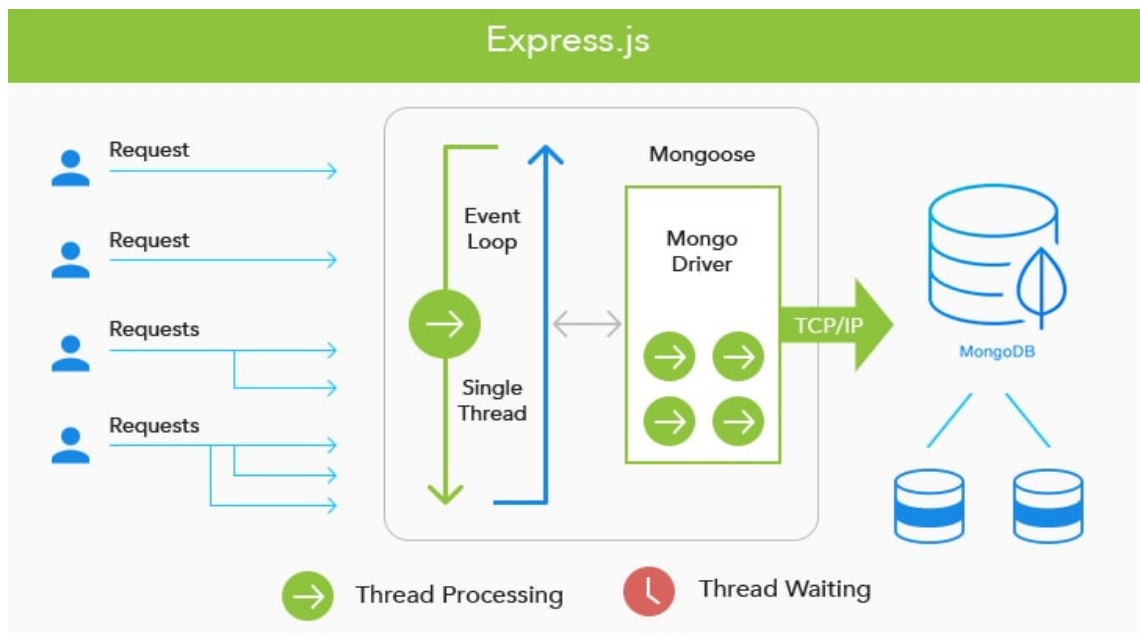



Figure 8. ExpressJS Architecture.

ExpressJs is very easy to use as it is well documented and published in node package manager i.e. npm. By using npm we can install and use lots of packages, which is required to develop NodeJS based applications. ExpressJs can be installed using the following command as shown in Figure 9.



```
npm install express --save
```

Figure 9. NPM command to install ExpressJS.

Even though ExpressJs does not have any built-in template engine, it exceptionally supports other JavaScript template engines like jade, ejs, handlebars, etc. For Single page applications, it is not necessary to use a server-side template engine as the dynamic content generation is handled by frontend libraries and frameworks. ExpressJs as a web server only generates the static files and serves the data via API calls. In MERN stack templating is handled by ReactJs as it provides the frontend solution. Figure 10 is an example of basic ExpressJS routing.



```
var express = require("express");
var app = express();

//a get request to respond "hello world" with ExpressJs get method
app.get("/", (req, res) => {
  res.send("hello world");
});

//a post request to send "hello world" with ExpressJs post method
app.post("/", (req, res) => {
  res.send("hello world");
});
```

Figure 10. An example of basic ExpressJS Routing.

4.4 ReactJS

ReactJs is another important component of the MERN stack. It holds the “R” part in the MERN stack. It is responsible for providing an interactive interface. ReactJs was initially released in 2013 by Facebook as a JavaScript-based frontend UI library to develop a reactive user interface for web and mobile applications. Since then, it is maintained by Facebook and a huge community of developers around the globe. Earlier ReactJs was developed and used internally by Facebook but later on, Facebook made it open source. Since its release, ReactJs has gained a lot of popularity among the developers and tech companies (Apress, 2017).

Let’s take a look at google trends about the trending libraries and frameworks in Figure 11 (GoogleTrends, 2020).

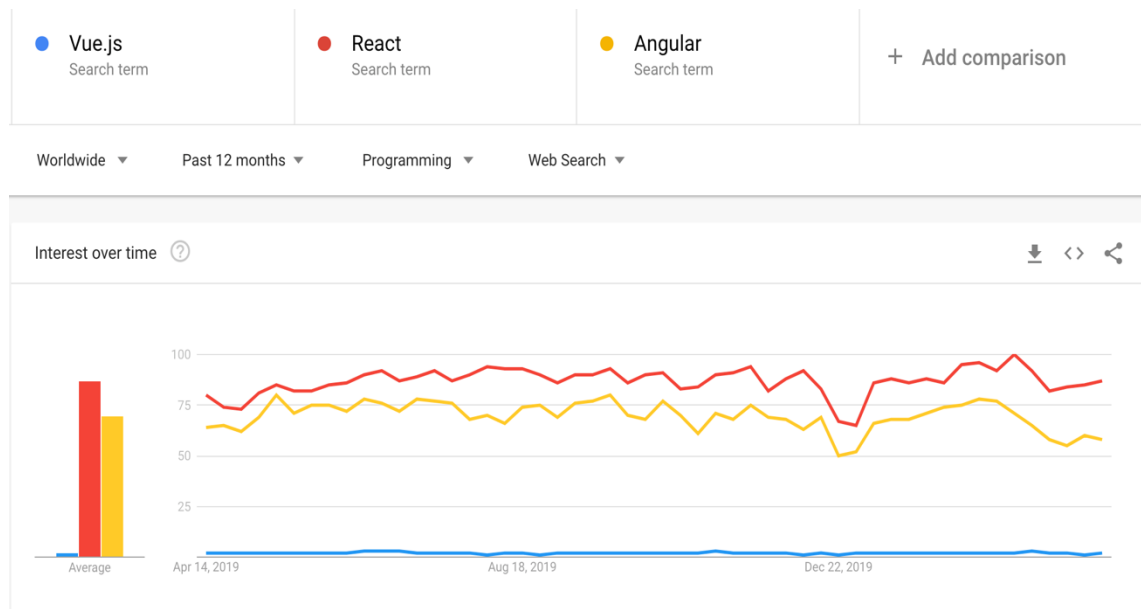


Figure 11. Web libraries and Frameworks trend.

4.5 Features of ReactJS

These are some important features that make ReactJS a handy tool for developing user interfaces.

4.5.1 Component Based

ReactJs is a component-based JavaScript library. It helps to breakdown the whole application to individual responsible components. Each component consists of its responsibility and the components are reusable. In ReactJs, each component holds its state. There can be two types of components in ReactJs, parent component, and child component. Parent component can pass its state down to the child component and child component can only pass back the changes to the parent component. This process is also known as data immutability. Due to this, ReactJs only re-renders the changes that have been made in the component, not the whole page. This is one of the major features of the Single page application. ReactJs strongly works on that basis (Patel, 2019).

4.5.2 Declarative

The main purpose behind the creation of ReactJs is to simplify the development of a powerful user interface. React enables designing of simple views along with its state which would update according to the data changes and renders the view of the respective component. Tracking and updating of the changes are all handled by ReactJs by itself. Developers do not need to worry about managing the effects of the changes in the state or the data. Every ReactJs component has its state and view. ReactJs stores the state and view and whenever there are any changes in the state it creates a new view on its own internally. Then it compares the old view with the new view and updates the actual view in the React DOM, where the state has been changed. The creation of a new view, maintaining all those changes, and updating the view accordingly seems tedious but practically ReactJs manages everything effortlessly with the help of Virtual DOM. Virtual DOM helps to store the view as a virtual representation in the memory data structure. ReactJs effectively manipulates the Virtual DOM and figures out the changes between those Virtual DOM. Then it applies the changes to the actual DOM in the browser. This makes the DOM manipulation very effective, faster as it is algorithmically applied in the ReactJs library. Let's check this example in Figure 12 to understand how ReactJs updates its element in the actual DOM (Rendering Elements – React, 2013).

Hello, world!

It is 12:26:46 PM.

```

Console Sources Network Timeline
▼ <div id="root">
  ▼ <div data-reactroot="
    <h1>Hello, world!</h1>
    ▼ <h2>
      <!-- react-text: 4 -->
      "It is "
      <!-- /react-text -->
      <!-- react-text: 5 -->
      "12:26:46 PM"
      <!-- /react-text -->
      <!-- react-text: 6 -->
      "."
      <!-- /react-text -->
    </h2>
  </div>
</div>

```

Figure 12. React DOM update.

Here we can see the browser inspect tools view for debugging. It has one root div and inside of that, it has another div. Inside data-reactroot div, there are two header elements h1 and h2. Element h2 is rendering the real-time in the browser. It changes its value every second. Another element h1 has not changed its value. Only h2 element has changed its value that's why in the inspect tools its highlighted with purple color. This explains the working principle and Virtual DOM of ReactJs in practically. ReactJs only re-renders those elements, where state and data have been changed. This is the declarative nature of ReactJs as it declares its view and with this nature, debugging and maintenance of code become very easy (Rendering Elements – React, 2013).

4.5.3 Isomorphic

ReactJs is isomorphic in nature. We can run ReactJs in the browser and can be used in a server with NodeJS to generate pages for SEO (Search Engine Optimization) optimizations as well. It follows the principle of learning once, write anywhere. ReactJs can be used to develop modern, native mobile applications using React Native. The same code can be run in the browser, server, and mobile apps.

These all features make ReactJs a powerful tool. As we know that how big is Facebook application. Facebook had faced a tough challenge to manage its state changes before the invention of ReactJs. Facebook has huge codebase and handling all those complex state changes and data in the view is a tough task. Facebook then thought to develop something more declarative tool to manage view rather than imperative (Patel, 2019).

4.6 NodeJS

NodeJS is the major component in the MERN stack. NodeJS represents the part “N” in the MERN stack. NodeJS was released in 2009 by Ryan Dahl. NodeJS is an open-source, cross-platform, JavaScript runtime built on top of Google chrome’s V8 JavaScript engine, which executes JavaScript code inside the browser and outside of the browser. Since the initial release of NodeJS, JavaScript has evolved more as a complete solution for powerful applications. With NodeJS runtime, JavaScript can be used to develop the backend and the RESTful microservices. Due to the NodeJS runtime, the other two major components of MERN stack, ExpressJs and ReactJs are developed. There are lots of other frameworks and libraries like VueJs, Angular, NestJs, etc. that have been developed according to the necessities.

NodeJS runs as an asynchronous, event-driven JavaScript runtime. The prime reason for its creation is to build scalable applications without using the threads. On connection, NodeJS fires a callback function, and if there is no connection NodeJS stops. In NodeJS there are not any locks like deadlocks, so it operates as non-blocking input/output (I/O) process. Due to its non-blocking feature, building scalable applications and systems is more feasible with NodeJS. NodeJS can perform multitasking easily. Even though it does not use multiple threads, multitasking is achieved by utilizing event loops. The event loop is a sequence or queue of the events that need to be processed by NodeJS. NodeJS performs an event loop with the help of callbacks. Callbacks are called on when an event is created. The event can be a command to read files, etc. Then callbacks run on that event and return results. This event-based working principle makes NodeJS systems and applications very fast (Patel, 2019).

4.6.1 Node Package Manager

NPM is an acronym for the Node package manager. It is a default package manager for installing third parties' packages and dependencies in NodeJS applications. It is publicly available at www.npmjs.com as it is the world's largest software registry. There are thousands of NodeJS packages developed by open source community and by NodeJS team to simplify the development process. Even in the MERN stack, we need lots of NPM packages to develop the application. As an example we need ExpressJs packages, MongoDB packages, React Packages to implement in MERN stack. We can install the packages and uninstall the packages via the NPM command-line interface. To use NPM packages we must install NodeJS and NPM in our local machine or system. NPM also provides effective updates from time to time for its packages. Most of the time it includes security and performance updates. NPM makes development tasks quite easier and it is an essential part of the NodeJS runtime environment (Node Package Manager, 2013).

4.7 How MERN Stack Work?

Before diving deep into the working principle of MERN stack, let's divide MERN stack into two parts i.e. frontend and backend. Frontend part is handled by the ReactJs and the backend part is handled by ExpressJs, NodeJS and MongoDB. Most of the modern web applications follow the various architectural patterns to make powerful web applications. One of the most used and famous architectural patterns is MVC. MVC is an acronym is Model, View, and Controller. These three blocks of MVC represent the interconnectivity and the logical data flow inside of the web applications. MERN stack strictly follows this architectural pattern.

4.7.1 Model

It manages the business logic of the application, what sort of data to be stored, how the data should be stored and manipulated. In MERN stack model layer is controlled by MongoDB.

4.7.2 View

It manages the user interface of the applications and it helps to visualize the data. In the MERN stack, the view layer is handled by ReactJs.

4.7.3 Controller

It controls the logical flow of data inside the application. It takes the commands from the view layer and connects to the model layer to get or update the data. In the MERN stack, the controller layer is managed by ExpressJs and NodeJS.

We can see that the MVC architectural pattern differentiates the layers into a clear picture for developers and business personnel. Another important aspect is the separation of concern (SOC) as the whole application can be separated into frontend and backend parts. It makes the development and management of the project and application easier and cleaner (Wilson, 2018). Figure 13 shows the logic flow in MERN stack application (Common Web Application Architecture Explained For Recruiters, 2019).

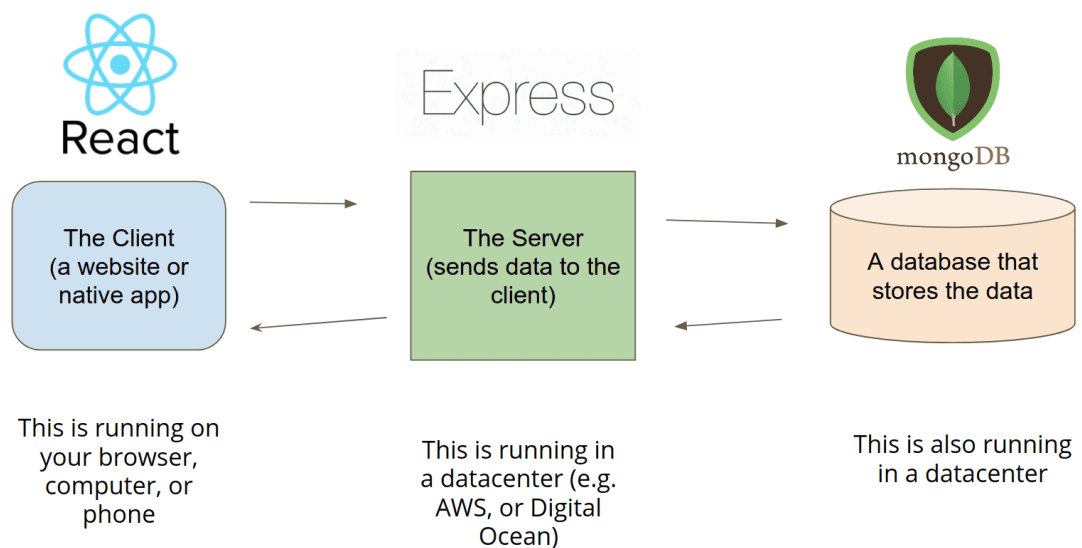


Figure 13. MERN application architecture

In Figure 13, when a user sends the request from the ReactJs user interface, the ExpressJs/NodeJS web server receives the request and then processes the request to

do according to the command queries in the request. The request communicates with the MongoDB database and queries the database. Once, querying the database is done, the database sends the data back to the ExpressJs/NodeJS web server. Then the webserver sends the data or update as a response to the ReactJs user interface. Then ReactJs updates the DOM and re-renders the particular piece of a component where the change has happened. This is an abstract workflow of the basic MERN stack application.

4.8 Strength and Weakness of MERN Stack Application

Everything has its advantages and disadvantages. When we use something practically for a while then we can put our experience in terms of good or bad about that thing. MERN stack also has its good and bad parts in the sense of usability, productivity, performance, etc. Developers from all around the world share their experience and problems regarding the stack they are using in the online community platform like stack overflow, GitHub, etc. MERN stack became popular around 2014 and developers started using it for development tasks.

4.9 Strengths

Lets discuss the strengths of the MERN stack applications.

4.9.1 Everything is in JavaScript

MERN is a JavaScript stack. The development language used in both frontend and backend is JavaScript. ReactJs, ExpressJs, and NodeJS all of them are built on top of plain Vanilla JavaScript. There is only one development language in the whole environment. Even in the MongoDB database, developers can write database scripts in JavaScript. This makes development and maintenance easier. There is no need to learn or switch to another programming language while developing the applications. It saves lots of time, effort, and money during development and posts development. The other important feature of the MERN stack is that no need to use template languages for generating pages. In MERN stack it is handled by ReactJs with HTML and DOM elements programmatically. JavaScript runs the whole stack thoroughly. Developers can

mix JavaScript with markup languages like HTML and CSS to make excellent applications (The Inside Scoop on MERN Stack | TECLA, 2019).

4.9.2 Data in JSON format

In MERN stack all of the data and object representation are done in JSON format. From frontend to backend everything is in JSON format. MERN stack uses RESTful API in the backend to communicate with the frontend. RESTful API lets the data be inbound and outbound in JSON format only. This feature saves lots of time and effort as a developer does not need to think about Object Relational Mapping (ORM), serializing and de-serializing the code. These attributes prevent the hassle of fitting the object model into rows and columns. Instead of Object Relational Mapper, an Object Document Mapper like Mongoose forces the object and data to follow the defined data schema, which is also a JSON structure. JSON format provides consistency in the data format all over the application.

4.9.3 NodeJS runtime

ReactJs, ExpressJs runs on top of NodeJS. MongoDB as a NoSQL database and absence of Object Relational Mapping, it outputs the higher performance with NodeJS. NodeJS itself is an asynchronous, event-driven, non-blocking input, output (I/O) webserver runtime environment. These features make the NodeJS a very fast webserver. When there is a huge amount of network traffic, then the event-driven architecture and non-blocking input, output feature play a vital role to handle it smoothly. This saves cost and time. Due to its architecture and better performance NodeJS runtime is a perfect choice for server-side development and it complements the MERN stack to be a handy and powerful stack.

4.9.4 Interactivity

As we know, the MERN stack is used for Single Page Application development. ReactJs helps to develop a single page application. The complex user interface development makes easier by ReactJs. ReactJs is component-based as it breakdowns the complete view and application into small components, it's easier to wire up the complex user interface structure. The states in the application hold the data and business values. ReactJs manages states in the best possible way algorithmically. It updates the part or component or state when there is any change in state or data. This makes the interactivity of the application user interface smarter and creates a great value for the users. MERN stack is a complete solution for complex applications due to its interactivity feature. Another important feature for developers is that the debugging tools like React developer tools make the development visually easier. It saves time and costs.

4.10 Weaknesses

These are the major weaknesses in MERN stack applications (The Inside Scoop on MERN Stack | TECLA, 2019).

4.10.1 Consistency

MongoDB database is a NoSQL database. It does not consist of rows and columns like in the traditional relational database. Instead of rows and columns, it stores data in a schema, which is a JSON object. MongoDB database also does not have tables like in the relational SQL database. It has collections instead of tables. MongoDB has its Object Document Mapper instead of Object Relational Mapper. MongoDB strongly supports horizontal scaling. This feature harms one of the main attributes of any database i.e. consistency. Consistency means the database transaction should be done in the way that it should apply the data changes in the allowed way only. MongoDB database lacks consistency.

4.10.2 Dependencies

MERN stack as a full-stack solution for the development of applications uses ReactJs for frontend and user interface development. ReactJs is a library for developing complex user interfaces by utilizing its Virtual DOM and rendering page according to the state changes. ReactJs does not have everything that needs to develop the complex user interface because it is a library. It has limited things and tools shipped upon as a library. It uses multiple third-party libraries and packages like React Router for routing, Redux for state management, Axios for API calls, etc. Due to its dependency upon third-party libraries, it affects the performance to leverage large scale applications. Another important thing is that the development of Enterprise application is a bit hectic with the MERN stack due to the architecture and dependencies. For example, LinkedIn chose MEAN stack over the MERN stack for its application development. The difference between MEAN and MERN stack is the frontend component. MEAN stack uses Angular and MERN stack uses ReactJs for frontend development. Angular manages the architecture of the application efficiently than as it is a framework. Though the MERN stack strongly supports MVC architecture pattern, MEAN stack is built on MVC architecture pattern. It ships with its own sets of tools to tackle Enterprise application development challenges.

4.10.3 Limited Computational Capacity

If there is any heavy mathematical computation or data processing on the server-side, then the NodeJS server is not the best option to choose for the development. NodeJS is relatively younger than other technologies like PHP, Java, Python, .Net. Other proven technologies can handle heavy mathematical calculations, Machine Learning, Complex Algorithms. For example, C++, Java, .Net, Python can handle these heavier computational necessities smoothly. NodeJS has a single thread so it processes a single request at a time. Even though it has non-blocking input-output (I/O), the processing can be easily blocked by one heavy computation. There are various ways to overcome this scenario but still, NodeJS lacks firepower for heavier backend computation. It is suitable for the development of an application with lighter backend computation.

4.10.4 Difficulties to learn

MERN stack is the combination of tools and technologies that are built upon the pure JavaScript. It does not use a traditional relational database like MySQL, MSSQL, SQLite, etc. Developers or the beginners who are not familiar with the NodeJS runtime and JavaScript development might find this stack overwhelming to learn and frustrating. The learning curve could be steep for the developers coming from old stack like LAMP (Linux, Apache, MySQL, PHP), Java Web, C++ web. Especially to master the backend development in MERN stack, developers need to approach completely different than the existing frameworks. Architecture is also different than the existing backend stacks. The asynchronous nature of NodeJS can surprise the beginners while understanding the MERN stack.

5 ROLE-BASED USER ACCESS CONTROL

Role-Based Access Control (RBAC) is an algorithmic approach to control access of the data and resources for users. It helps to allocate the data and the resources according to the role of the users. Most of the applications have some kind of role-based access control to safeguard the application resources. In simple words, role-based access control is similar to the access card for the door locks. The user authentication system authenticates the user's role and then provides access according to the role. Every role defined inside the application has its own sets of privileges. For example, a general grocery store application can have roles like manager, cashier, supervisor, and normal user. The manager can have the highest privilege, then descending privilege to the supervisor, cashier, and normal user. Roles with the higher privilege can perform read and write operations while roles with lower privilege are limited to read operation only. The roles are dynamic, and they are owned by the owner i.e. super admin and admin. Only the owner can determine the access rights inside the application.

The technical part of this thesis has implemented Role-based access control. The designated roles are super admin, admin, team Manager, and player. The super admin and admin has both read and write privilege. Superadmin can write over admin and others but the admin role is not allowed to write over super admin.

Admin can perform write operations to team managers and players. Then team manager can perform write operations to its own documents only. Team managers cannot perform write operations all over the database (Dunna, 2016).

Figure 14 below illustrates the privilege according to the user roles.

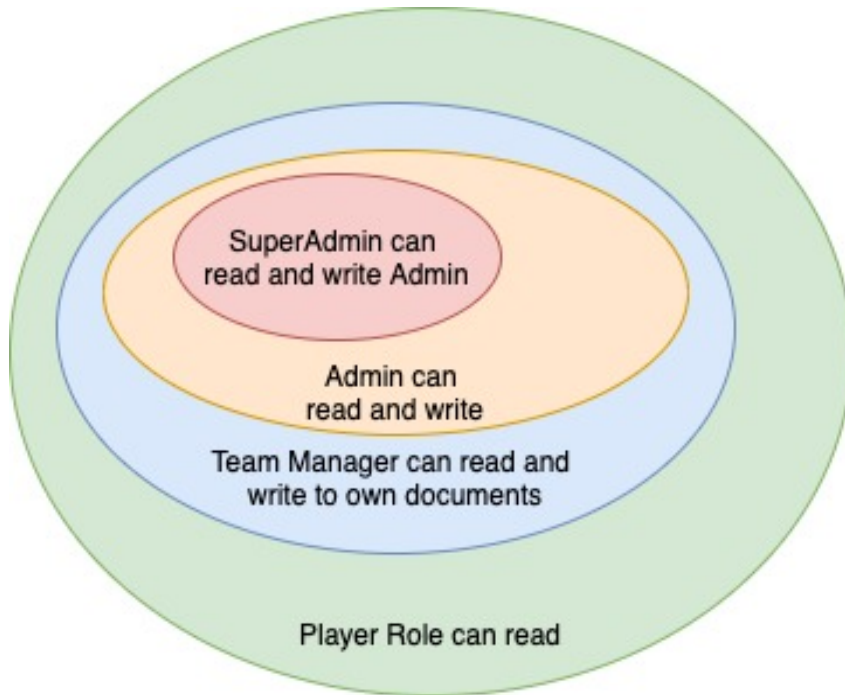


Figure 14. User Roles and privileges.

Technically user roles can be implemented as a mongoose schema object, which is a blueprint for user object creation. As already mentioned above, MongoDB has its Object Document Mapper. Mongoose is a library that performs the task of Object Document Mapper. Figure 15 is a code implementation for dynamic user roles.

```
const mongoose = require('mongoose');

const UserSchema = new mongoose.Schema({
  role: {
    type: String,
    enum: ['superAdmin', 'admin', 'teamManager', 'player'],
    default: 'player'
  }
});
```

Figure 15. Mongoose Schema with user roles.

In the Figure 15 above, user schema is defined by using mongoose. A role is an object under the mongoose user schema. Then the role object has three fields. The first field type defines the data type of that field, which is a string. The second field is an enum data type, which is an array with string values. The second field defines the user roles as super admin, admin, team manager, and player. The third field default sets the default value for the role. In the picture, it sets the player as the default role. While creating the new user, the role can be either of the role enum depending upon the user registration. The super admin and admin role is assigned in the database by a database administrator. The team manager role is assigned to the one who registers the team details and the players in the system after successful registration and login. In this way by using mongoose, we can define roles under mongoose schema.

The core workflow of role-based user access control starts with user registration and login. When a user is registered and logged in successfully, the default role of the user is player. When a user registers with name, username, password, and role then the password is encrypted to hash and salt value by bcryptJs library before saving to the database. When the user successfully logs in to the application, at the same time an encrypted token is generated by utilizing the JWT (JSON Web Token) generator and saved in the local storage of the browser. The encrypted token holds the information of logged in user except than the password and then the PassportJs authenticator authenticates the user by decrypting the encrypted JWT token. PassportJs is the user authenticator middleware library for NodeJS applications. Then the authenticator authenticates the user role and it redirects the user to the respective route according to the user role in the encrypted token. Before redirecting the user, each route is authenticated for the user role by PassportJs. The algorithmic combination of JWT token, bcryptJs, and PassportJs helps to make the secure role-based user access control system.

Below is an example of super admin and admin routes to access all of the team bookings and registration for the event. The router points to the allTeams route to get all bookings and then passport.authenticate method provided by passportJs authenticates the user role by checking in JWT token. PassportJS only allows the super admin and admin role to access that route, otherwise, it returns the error message. Figure 16 is a code implementation of role-based user authentication using PassportJS for super admin and admin.

```
//Admin All Bookings,Teams
router.get('/allTeams', (req, res, next) => {
  passport.authenticate('jwt', (err, user) => {
    if (user.role === 'superAdmin' || user.role === 'admin') {
      Booking.find((err, teams) => {
        // checks if there is any error
        if (err) {
          res.json({
            success: false,
            message: err
          }); // Returns error message
        } else {
          // Check if bookings were found in database
          if (!teams) {
            res.json({
              success: false,
              message: 'No Teams found.'
            }); // Returns error of no bookings found
          } else {
            res.json({
              success: true,
              teams: teams
            }); // Return success and bookings array
          }
        }
      }).sort({
        '_id': -1
      });
    } else {
      return res.send('not logged in');
    }
  })(req, res, next);
});
```

Figure 16. Express route for super admin and admin.

Security is an important aspect of any kind of application. Role-based user access control helps to implement the secure access of applicational resources and functionalities to the designated roles only. In the MERN stack application, there are several options to implement security inside the applications. The application developed for this thesis has utilized three important tools that are trusted by the developers around the world and have successfully set the industry standard i.e. JWT, BcryptJs, PassportJs.

5.1 JWT

JSON Web Token is a digitally signed, an open standard (RFC 7159) which helps to transmit information securely between the parties as a JSON object. HMAC algorithm or public/private key pair (RSA or ECDSA) signs the JWT token to be secure. Signed tokens verify the integrity of the information (JWT.IO, 2017).

The JWT token is used for user authorization as it is simple to use and secure. When a user is logged in then each request from the user contains JWT token along with it and it authorizes the JWT token to match. Once the token is matched then it sends the Base64Url encoded payload, which contains the user info. Then only the authorized user can access the resources, services, and routes within the application. The JWT token also applies the expiration time for its token, as it updates the token after a certain time.

5.2 BcryptJs

BcryptJs is another important component library that helps to protect user passwords by hashing it. It also implements salt in user passwords to protect from rainbow attacks. Its salt value can be increased and more the salt value, it is more resistant against the brute force attacks. Storing plain passwords without hash and salt in the database is not the recommended security practice. Plain passwords are easy to crack and they are vulnerable against brute force attacks and rainbow attacks. BcryptJs helps to make

implement password hashing by using the OpenBSD algorithm. OpenBSD algorithm is an open-source, security-oriented, UNIX-like operating system-based algorithm (Bcryptjs, 2017). It is used to hash and salt the user passwords before saving to the database in the application developed for this thesis. Figure 17 is an example of bcryptJs to hash and salt user passwords.

```
const bcrypt = require('bcryptjs');

//hash and salt for user passwords before storing to the database
module.exports.addUser = function(newUser, callback) {
  bcrypt.genSalt(10, (err, salt) => {
    if(err) throw err;
    bcrypt.hash(newUser.password, salt, (err, hash) => {
      if(err) {throw err;}
      newUser.password = hash;
      newUser.save(callback());
    });
  });
}
```

Figure 17. Hash and Salt for user passwords using bcryptJS.

5.3 PassportJS

PassportJS is an open-source authentication library for NodeJS applications. It is flexible and easy to perform user authentication. The implementation of user authentication for this thesis application has been done by using passportjs. It has made the task of developing role-based user access control easier and simpler inside the thesis application. PassportJs has only one responsibility to perform authentication on every request that comes inside the application to access the services or functionalities.

In modern applications, user authentication can be done in various ways. One common way is to username and password authentication. Besides this, there is a token-based authentication service provider like OAuth, Facebook, Google, Twitter, etc. The need for user authentication is compulsory as it is related to security and privacy. Different applications need different kinds of authentications depending upon the scenario. PassportJs works in this principle as it can implement authentication in several ways and strategies to provide flexibility and simpler implementation. It can provide authentication like token-based, session-based, OAuth based, OpenID based, etc. It is an industry-standard and trusted by developers and users around the globe. It is supported by the famous authentication service provider OAuth as well. The important features of PassportJs are customizable, lightweight code, secure, and 300+ authentication strategies (PassportJS library, 2017). These all features make it reliable and secure for user authentication.

6 APPLICATION ARCHITECTURE

The application architecture consists of four layers. First layer and second layer represent the frontend of the application. It contains all of the view and pages. Third layer consists of all API's used by the application. It is called everytime when user interacts with a certain page. Fourth layer provides the authentication, authorization and database services which is used by the third layer. Figure 18 below visualizes the different layers of the application.

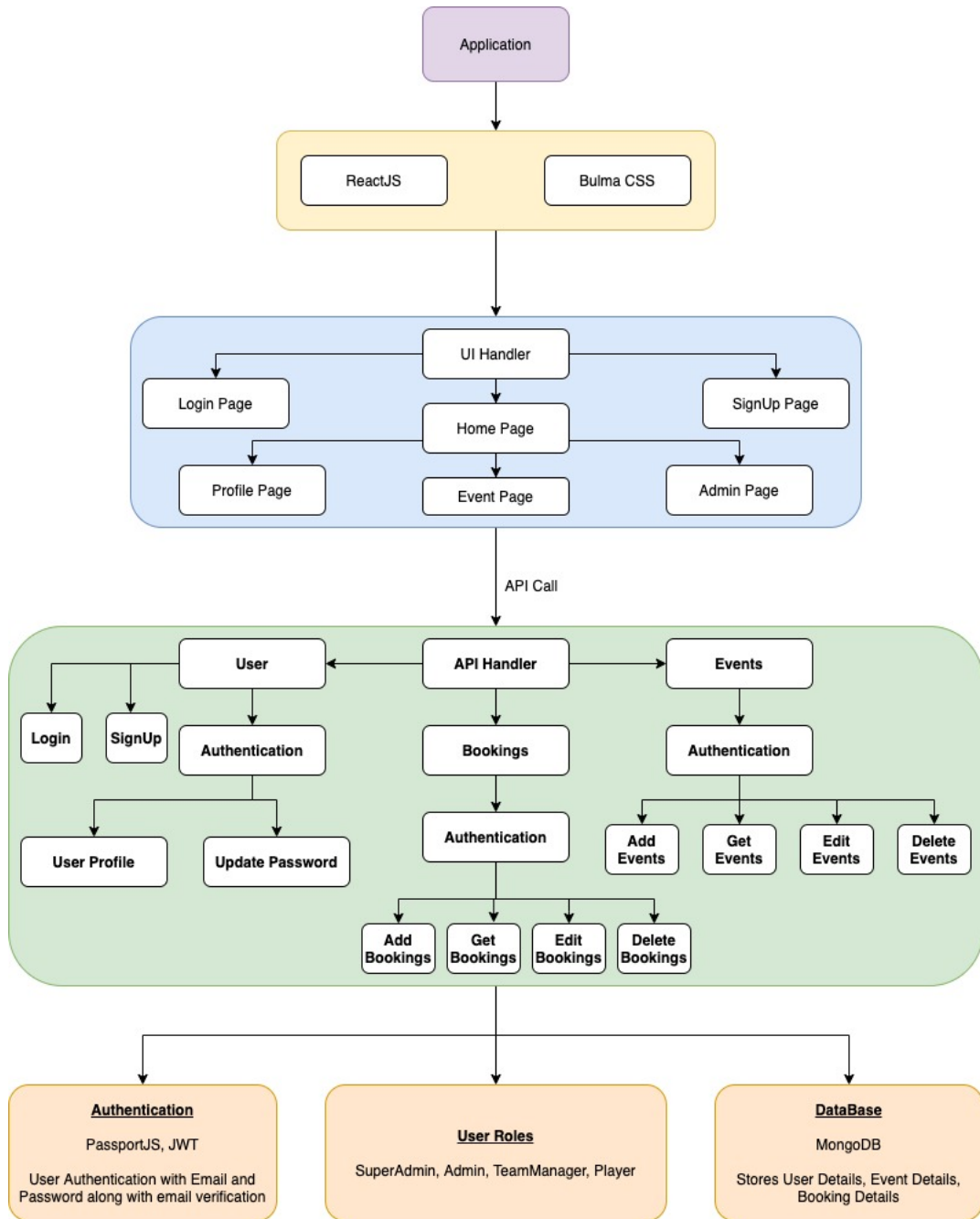


Figure 18. Architecture for whole application.

7 SCALABILITY WITH MERN STACK ARCHITECTURE

The Mern Stack is not the only stack available to develop web applications. There are other technologically similar stacks like Django, Java Spring boot, ASP .NET, MEAN, LAMP, etc. Every stack has its characteristics and architecture, in which it outputs the performance.

MERN stack has excellent frontend components as ReactJs which can develop complex user interfaces with efficient performance. As ReactJS is a component-based library, Components can be easily scaled up or reused according to the necessity. The data between the parent and child components can be easily passed via props. Complex data state rendering can be handled either by its built-in hooks, class-based state or by plugging the state management library such as Redux. Redux is very efficient for managing complex states as it stores all of the states in one built-in store to maintain a single source of truth. Redux assists ReactJs to make it highly scalable. One of the main reasons for opting to choose the MERN stack is ReactJs.

On the backend side, the MERN stack operates on top of NodeJS runtime. As explained under the sub-title, 4.10.3 Limited Computational Capacity, NodeJS lacks the power to perform heavy mathematical computation or Machine Learning Algorithms. Nodejs architecture is an event-based, single-threaded processing environment. It does not consist of enough computational firepower to perform highly intensive CPU tasks. NodeJS is lighter than PHP and it does not need another webserver to run, while PHP needs Apache webserver. Django is a Python-based web application framework. Python is extensively used to perform CPU-intensive tasks and Machine Learning. It is a powerful language as the same goes for Java spring boot and ASP .Net. MERN stack is suitable for developing e-commerce applications, enterprise applications, applications with less intensive CPU tasks. MERN is used to some extent along with other stacks by famous tech companies like Facebook, Netflix, Instagram, WhatsApp, Dropbox, etc. Even though Nodejs has limited computational power, it is highly scalable. Its performance can be easily upgraded as its architecture strongly supports the scalability. Scalability provides an option to enhance the computational power of the application by adding the infrastructure.

MERN uses ExpressJS as a backend framework to develop REST API and API routes. ExpressJS provides an excellent option to scale the API routes. New Routes can be easily added to the existing routes. Inside the main server.js file, all of the routes inside the “./routes/users” folder is assigned to a constant called users. This users constant is used by the ExpressJS method, app.use(“/users”, users). All of the new routes for the users can be defined inside users routes and the main server.js file can access it automatically. This process makes the API development easier and scalable. Figure 19 explains the scalability of ExpressJS routes.



```
const users = require("./routes/users");
app.use("/users", users);
```

The image shows a code snippet in a dark-themed editor. The code consists of two lines: the first line declares a constant 'users' which is assigned the result of 'require' for the path './routes/users'; the second line calls 'app.use' with the path '/users' and the 'users' constant as arguments.

Figure 19. Main routes used by ExpressJS.

MERN uses MongoDB as a database. There are lots of cloud-based MongoDB database solution provider and MLAB is one of them. MLAB helps to scale database easily and its cloud-based which makes it highly available.

There are various ways to scale NodeJS applications. When the incoming load increases in the application then the scaling of the infrastructure throughout the application requires. It enhances performance and balances the load across the application. NodeJS architecture supports scalability efficiently, whether it is horizontal or vertical. By utilizing the multiple processing and load balancer, all of the core's processing power can be used to increment the performance. NodeJS has built-in support for scalability e.g. cluster module and PM2 cluster module. These modules help to scale applications when multiple processes are running in the same machine.

The application developed for the thesis has been deployed in the Heroku cloud. Heroku cloud also provides different tools and ways to scale the application. Heroku provides a solution called Memcache, it is a technology that improves the scalability and poor performance of applications. When the application is loading slowly or having the scalability issues, then Memcache becomes helpful. Memcache enables smooth and faster response as it pulls responses from the cache. This also reduces the computational burden of the server. Figure 20 provides the command to install Memcache in Heroku.

```
heroku addons:create memcachier:dev
```

Figure 20. Command to install memcache.

Then memjs should be installed via npm. Figure 21 provides the command to install memjs.

```
npm install memjs
```

Figure 21. NPM command to install memjs.

In the main server.js file we can configure Memcache. The variable mc can be used in the each routes to enable Memcache. Figure 22 describes the Memcache configuration in ExpressJS routes.

```
var memjs = require('memjs');
var mc = memjs.Client.create(process.env.
MEMCACHIER_SERVERS, {
  failover: true, // default: false
  timeout: 1,     // default: 0.5 (seconds)
  keepAlive: true // default: false
});
```

Figure 22. Memcache configuration used by main server.js file.

8 TESTING

Testing is a very important part of any software development. Testing ensures the quality and assurance to the users. The application for this thesis has been developed according to the TDD (Test Driven Development) approach. This means development is strictly followed by the testing process. Test cases are defined before and the code is developed to pass those test cases. It maintains the quality and performance. The test results reflect the code quality and provide insights to improve the codebase.

To perform the testing, this application has used Postman, Mocha, and Chai. Postman is an open-source desktop application to perform API testing. It is a very handy tool to test the API as it provides clear results and errors while performing the testing. Below is an example of register route testing in Postman in figure 19.

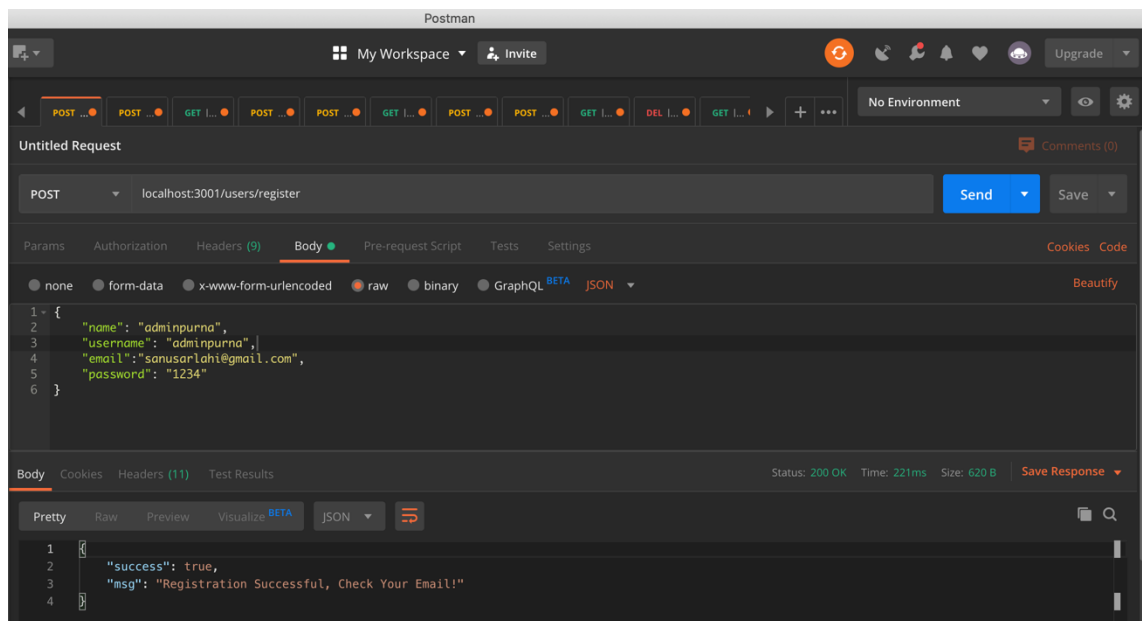
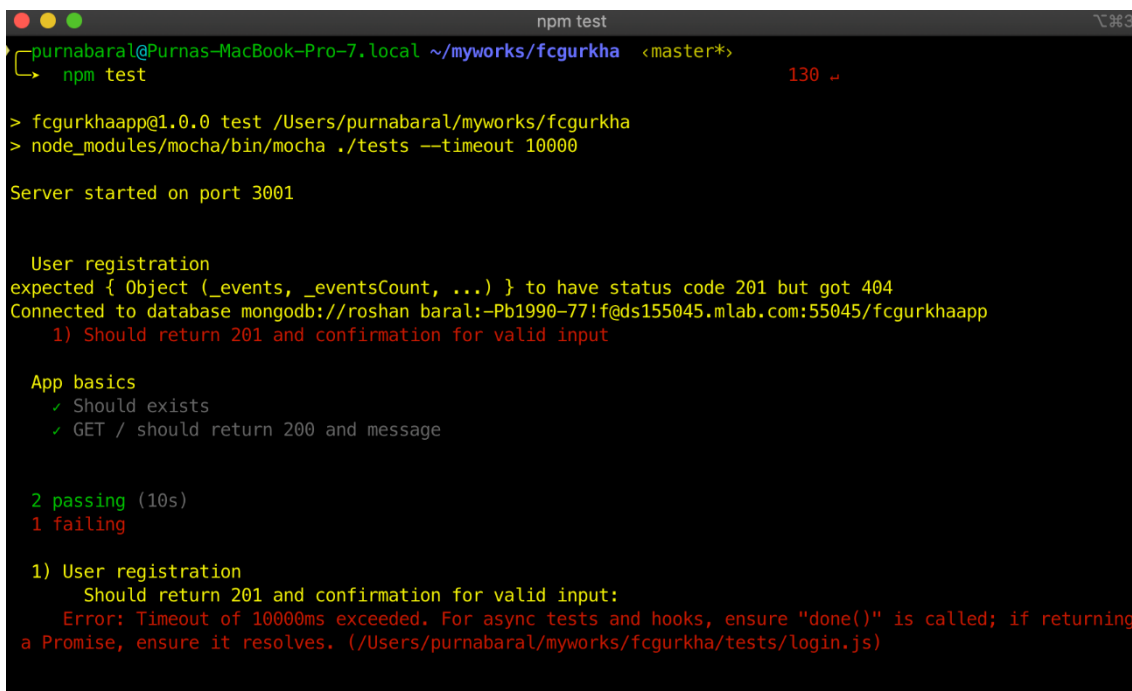


Figure 23. API testing with Postman.

Mocha and Chai have been used to perform the unit testing of the codebase inside the application. Mocha is a JavaScript testing framework to do asynchronous testing. It is simple, efficient, and flexible. It can be installed via npm (Node Package Manager). Another component for testing is Chai, which is an assertion library. It can be easily

paired with Mocha to produce effective test results. Below is an example of Mocha and Chai testing and results in figure 20.



```
npm test
130 ↵

> fcgurkhaapp@1.0.0 test /Users/purnabara1/myworks/fcgurkha
> node_modules/mocha/bin/mocha ./tests --timeout 10000

Server started on port 3001

  User registration
  expected { Object (_events, _eventsCount, ...) } to have status code 201 but got 404
  Connected to database mongodb://roshan baral:-Pb1990-77!f@ds155045.mlab.com:55045/fcgurkhaapp
  1) Should return 201 and confirmation for valid input

  App basics
    ✓ Should exists
    ✓ GET / should return 200 and message

  2 passing (10s)
  1 failing

  1) User registration
       Should return 201 and confirmation for valid input:
     Error: Timeout of 10000ms exceeded. For async tests and hooks, ensure "done()" is called; if returning a Promise, ensure it resolves. (/Users/purnabara1/myworks/fcgurkha/tests/login.js)
```

Figure 24. Mocha and Chai testing results.

The GitHub repository of the application for the thesis has also implemented Snyk. It is a tool to check the weakness and vulnerabilities in the dependencies i.e. the package that has been installed via npm. It constantly checks the package dependencies and updates or fixes those packages according to the necessity. It also ensures the security inside the application by preventing the vulnerabilities in dependencies. Figure 21 is a code implementation of mocha and chai testing.


```

var expect = require("chai").expect;
var app = require("../server");
var request = require("supertest");
const crypto = require("crypto");
const chai = require("chai");

const http = require("chai-http");
chai.use(http);

describe("User registration", () => {
  it("Should return 201 and confirmation for valid input", (done) => {
    //mock valid user input
    const new_user = {
      name: "test6",
      username: "test5",
      password: "1234",
      email: "sanusarlahi@gmail.com",
      role: "player",
      isVerified: false,
      verifyEmailToken: crypto.randomBytes(20).toString("hex"),
      verifyEmailTokenExpires: Date.now() + 1 * 60 * 60 * 60 * 1000,
    };
    //send request to the app
    chai
      .request(app)
      .post("/register")
      .send(new_user)
      .then((res) => {
        //assertions
        expect(res).to.have.status(201);
        done();
      })
      .catch((err) => {
        console.log(err.message);
      });
  });
});

describe("App basics", () => {
  it("Should exists", () => {
    expect(app).to.be.a("function");
  });

  it("GET / should return 200 and message", (done) => {
    //send request to the app
    chai
      .request(app)
      .get("/")
      .then((res) => {
        //assertions
        expect(res).to.have.status(200);
        expect(res.body.message).to.contain("invlaid endpoint");
        done();
      })
      .catch((err) => {
        console.log(err.message);
      });
  });
});

```

Figure 25. Mocha and Chai test code.

9 CLOUD HOSTING

Running the application in the local machine is not enough. It should be deployed or hosted somewhere so that users can access via the web address. It should be constantly monitored and maintained to ensure the better user experience. For this thesis application, Heroku cloud hosting has been used. Heroku is a cloud-based application hosting service provider. It is simple, user-friendly, and cost-effective too. The process of deployment starts by installing Heroku CLI in the local machine. Then the application is connected to Heroku by pushing the application to Heroku via CLI (Cloud Application Platform | Heroku, 2020). Before pushing to the Heroku, the environment variables should be set. Heroku provides the log for the hosting process. If something is missing or have some errors, then it shows that in the log.

10 CONCLUSION

This thesis provides an insight to develop and implement role-based user access control in MERN stack applications. The technical part of this thesis has been developed from scratch to be a live application deployed and hosted in the Heroku cloud. It describes the customizable roles and logic behind the access control according to the user needs. This is not the only way of implementing role-based access control in MERN stack applications, but it is one of the many ways to implement it.

10.1 Results

The technical implementation of this thesis delivers a full-stack application developed in the MERN stack to implement role-based user access control. The thesis has described the impact of the MERN stack in the modern application development scene. The frontend has been developed using Class-based React components along with the Bulma CSS framework. The application state has been managed within the Components itself. The backend is based on the REST API. This backend REST API can be shared with different platforms such as mobile applications, which can be developed from almost the same codebase by using React Native framework. Another important possibility is that the same application can be converted to the PWA according to the requirement.

Role-based user access control is all about authentication, verification of users, and protecting the resources from unwanted users. This application has solely focused on implementing modern encrypting algorithms according to recent industry standards. Besides this, it has strongly followed the security protocols for web applications. The application developed for this thesis will be deployed in the production mode very soon after adding more features, some design enhancements, and rigorous testing. This thesis can be taken as a reference for the beginners to intermediate level developers who wants to learn and explore MERN stack real-time.

REFERENCES

- AltexSoft. 2018. *The Good And The Bad Of Javascript Full Stack Development*. [online] Available at: <<https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-javascript-full-stack-development/>> [Accessed 16 March 2020].
- Apress. 2017. *Why MERN?*. [online] Available at: <<https://www.apress.com/gp/blog/all-blog-posts/why-mern/12056000>> [Accessed 18 April 2020].
- BcryptJS. 2017. *Bcryptjs*. [online] Available at: <<https://www.npmjs.com/package/bcryptjs>> [Accessed 15 March 2020].
- Blockchain Simplified. 2019. *What Is MERN Stack?*. [online] Available at: <<https://blockchainsimplified.com/blog/what-is-mern-stack/>> [Accessed 5 March 2020].
- Bocasay. 2020. *How Do MERN Stack Technologies Work Together?*. [online] Available at: <<https://www.bocasay.com/how-does-the-mern-stack-work/>> [Accessed 2 April 2020].
- Diakogiannis, K., 2018. *16.C React Routing And Single Page Apps*. [online] GoConqr. Available at: <https://www.goconqr.com/c/74455/course_modules/113576-single-page-vs-multi-page-architecture#> [Accessed 6 March 2020].
- Dunna, K., 2016. *Implement Access Control In Node.Js*. [online] Medium. Available at: <<https://blog.nodeswat.com/implement-access-control-in-node-js-8567e7b484d1>> [Accessed 13 March 2020].
- Dvpro.com. 2018. *Old Fashioned Web Development - DVPRO | DVPRO*. [online] Available at: <<https://www.dvpro.com/services/painless-website/old-fashioned-web-development>> [Accessed 6 April 2019].
- Emery, C., 2016. *A Brief History Of Web Development*. [online] Techopedia.com. Available at: <<https://www.techopedia.com/2/31579/networks/a-brief-history-of-web-development>> [Accessed 15 April 2020].
- Geekflare. 2019. *7 Tools To Scan Node.Js Application For Security Vulnerability - Geekflare*. [online] Available at: <<https://geekflare.com/nodejs-security-scanner/>> [Accessed 2 April 2020].
- Google Trends. 2020. *Web Libraries And Frameworks Trend*. [online] Available at: <<https://trends.google.com/trends/explore?cat=31&q=Vue.js,React,Angular>> [Accessed 13 April 2020].

- Guru99.com. 2018. *Nosql Tutorial: Learn Nosql Features, Types, What Is, Advantages*. [online] Available at: <<https://www.guru99.com/nosql-tutorial.html>> [Accessed 3 May 2020].
- Heroku.com. 2020. *Cloud Application Platform | Heroku*. [online] Available at: <<https://www.heroku.com/>> [Accessed 3 May 2020].
- Iteachrecruiters.com. 2019. *Common Web Application Architecture Explained For Recruiters*. [online] Available at: <<https://www.iteachrecruiters.com/blog/common-web-application-architecture-explained-for-recruiters/>> [Accessed 14 April 2020].
- Jwt.io. 2017. *JWT.IO*. [online] Available at: <<https://jwt.io/>> [Accessed 13 March 2020].
- MDN Web Docs. 2019. *About Javascript*. [online] Available at: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript> [Accessed 10 April 2019].
- Medium. 2016. *Single-Page Application Vs. Multiple-Page Application*. [online] Available at: <<https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58>> [Accessed 5 February 2020].
- Morgan, A., 2017. *The Modern Application Stack – Part 1: Introducing The MEAN Stack | Mongodb Blog*. [online] MongoDB. Available at: <<https://www.mongodb.com/blog/post/the-modern-application-stack-part-1-introducing-the-mean-stack>> [Accessed 16 March 2020].
- Npmjs.com. 2013. *Node Package Manager*. [online] Available at: <<https://www.npmjs.com/>> [Accessed 5 February 2020].
- Passport.js. 2017. *Passportjs Library*. [online] Available at: <<http://www.passportjs.org/>> [Accessed 20 March 2020].
- Patel, D., 2019. *Switching To The Modern Day MERN Stack!*. [online] Medium. Available at: <<https://medium.com/nybles/switching-to-the-modern-day-mern-stack-574bb478fc64>> [Accessed 20 February 2020].
- Peerbits. 2019. *It'S High Time To Say Goodbye To LAMP Stack And Hi To MEAN Stack*. [online] Available at: <<https://www.peerbits.com/blog/lamp-stack-vs-mean-stack.html>> [Accessed 12 March 2020].
- Reactjs.org. 2013. *Rendering Elements – React*. [online] Available at: <<https://reactjs.org/docs/rendering-elements.html>> [Accessed 15 May 2019].
- Statista. 2019. *Topic: Internet Usage Worldwide*. [online] Available at: <<https://www.statista.com/topics/1145/internet-usage-worldwide/>> [Accessed 4 April 2019].

- Tecla.io. 2019. *The Inside Scoop On MERN Stack | TECLA*. [online] Available at: <<https://www.tecla.io/blog/the-inside-scoop-on-mern-stack/>> [Accessed 4 July 2019].
- Website design company in Kolkata, India - Intlum. 2018. *Web Development History - Progress Of Web In Detail - Intlum*. [online] Available at: <<https://www.intlum.com/blog/web-development-history/>> [Accessed 3 March 2019].
- Wilson, E., 2018. *MERN Quick Start Guide*. [online] Google Books. Available at: <https://books.google.fi/books?id=HnxeDwAAQBAJ&pg=PA7&redir_esc=y#v=onepage&q&f=false> [Accessed 10 March 2020].

10.1.1 Error! No text of specified style in document.

