A Complete MERN Stack Developed Website Using MongoDB, Express JS, React JS, Node JS & Firebase

by

Rehana Akter Reyana
ID: CSE1901016036

Jannatul Ferdousi
ID: CSE1901016037

Md. Sadman Shakib
ID: CSE1901016153

Md. Imran Hossen
ID: CSE1901016188

Supervised by
Mohammad Naderuzzaman

Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SONARGAON UNIVERSITY (SU)
January 2023

APPROVAL

The Project titled “A Complete MERN Developed Website Using MongoDB, Express JS, React JS, Node JS & Firebase” submitted by Rehana Akter Reyana (CSE1901016036), Jannatul Ferdousi (CSE1901016037), Md. Sadman Shakib (CSE1901016153) and Md. Imran Hossen (CSE1901016188) to the Department of Computer Science and Engineering, Sonargaon University (SU), has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering and approved as to its style and contents.

Board of Examiners

Mohammad Naderuzzaman
Assistant Professor,
Department of Computer Science and Engineering
Sonargaon University (SU)

Supervisor

(Examiner Name & Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 1

(Examiner Name & Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 2

(Examiner Name & Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 3

DECLARATION

We hereby declare that the work presented in this report is the outcome of the investigation performed by us under the supervision of **Mohammad Naderuzzaman, Assistant Professor,** Department of Computer Science and Engineering, Sonargaon University, Dhaka, Bangladesh. We reaffirm that no part of this project has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned

(Mohammad Naderuzzaman)
Supervisor

Signature

Rehana Akter Reyana
ID: CSE1901016036

Jannatul Ferdousi
ID: CSE1901016037

Md. Sadman Shakib
ID: CSE1901016153

Md. Imran Hossen
ID: CSE1901016188

ABSTRACT

We have developed an effective niche e-commerce website using MongoDB as the backend database and the MERN stack for frontend development, which we believe we are using for the first time in our department.

It is a specific kind of e-commerce site, named niche. The business-to-consumer aspect of product commerce (e-commerce) is the most visible business use of the world wide web. The primary goal of an e-commerce site is to sell goods online. This project deals with developing an e-commerce, or basically niche website for drone sales. The technology used to implement this project is called MERN stack development. The MERN stack is a collection of technologies that enables faster web and application development with MongoDB Database Server because it is a document-oriented NoSQL database used for high volume data storage. This type of project maintains a lot of data, so MongoDB is very suitable for this project and others like it. Also, MongoDB is very popular for small database projects because of its speed and efficiency when loading data. The main purpose of using the MERN stack is to develop web applications using MongoDB with only Javascript. This is because the four technologies that make up the technology stack are all JS (Javascript)-based.

MongoDB provides official driver support for all the popular languages like C, C++, C#, Net, Go, Java, Node.js, Perl, PHP, Python, Motor, Ruby, Scala, Swift, are all examples of programming languages. As a result, anyone can use any of these languages to create an application or website. Nowadays, there are so many companies that use MongoDB to store their large amounts of data, like Facebook, Nokia, eBay, Adobe, Google, etc. On the MongoDB server, it is allowed to run multiple databases.

Our website provides drone selling services to the user with a catalog of different drones available in the store. The system is implemented using a 3-tier approach, with a backend MongoDB database, a middle tier of backend Node.js server sites with Firebase authentication, and a web browser as the front-end client site. In order to develop a niche type of e-commerce website, a number of technologies must be studied and understood. These include multi-tiered architecture, server and client site scripting techniques, both admin and user panels, and implementation technologies such as Node.js, React, Javascript, JQuery, Bootstrap, HTML5, relational databases (MongoDB), and an authentication system (such as Firebase). This is a project with the objective of developing a basic website where a consumer is provided with a shopping cart application and also learning about the technologies used to develop such an application. This document will discuss each of the underlying technologies required to create and implement an e-commerce website.

ACKNOWLEDGMENT

At the very beginning, we would like to express my deepest gratitude to the Almighty Allah for giving us the ability and the strength to finish the task successfully within the schedule time.

We are auspicious that we had the kind association as well as supervision of **Mohammad Naderuzzaman**, Assistant Professor, Department of Computer Science and Engineering, Sonargaon University whose hearted and valuable support with best concern and direction acted as necessary recourse to carry out our project.

We would like to convey our special gratitude to **Prof. Dr Md Alamgir Hossain,** Dean, Faculty of Science and Engineering for his kind concern and precious suggestions.

We are also thankful to all our teachers during our whole education, for exposing us to the beauty of learning.

Finally, our deepest gratitude and love to my parents for their support, encouragement, and endless love.

LIST OF ABBREVIATIONS

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CLI	Command Line Interpreter
CSS	Cascading Style Sheets
DBMS	Database Management System
DOM	Document Object Model
FTP	File Transfer Protocol
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IP	Internet Protocol
JSON	JavaScript Object Notation
MERN	MongoDB Express ReactJS Node.js
MVC	Model View Controller
ORM	Object Relational Mapping
REST	Representational State Transfer
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SPA	Single Page Application
SQL	Structured Query Language
TCP	Transmission Control Protocol
URL	Uniform Resource Locator
XML	Extensible Markup Language

TABLE OF CONTENTS

Title			Page No.
APPROVAL……			i
DECLARATION ……..……..			ii
ABSTRACT ……...…….			iii
ACKNOWLEDGEMENT ……..……			iv
LIST OF ABBREVIATIONS ……..……			v
CHAPTER 1			1-3
INTRODUCTION			
1.1	Introduction ……..…….....		1-2
1.2	Main Purpose of this Project...…….....		3
CHAPTER 2			4-5
RELATED TECHNOLOGIES ANALYSIS			
2.1	JavaScript…….….....		4
2.2	HTML……..……..		4
2.3	CSS……...		4
2.4	React Framework……..		4
2.5	MongoDB…….		4
2.6	Firebase…….		5
2.7	Node js……..		5
2.8	Software Requirment……		5
CHAPTER 3			6-37
MERN STACK DEVELOPMENT			
3.1	Introduction……..		6-11
3.2	Web application Development…….		6-12
	3.2.1	Front-end and Back-end……..	7-9
	3.2.2	RESTful API……...	9-10
	3.2.3	Single Page Application……..	10-11
3.3	Full Stack JavaScript……		12
	3.3.1	JavaScript…….	12-13
	3.3.2	Rise of Full Stack JavaScript……	13-15
3.4	MERN Stack Development……..		15-23
	3.4.1	Node.js…….	15-18
	3.4.2	Express js…….	18-19
	3.4.3	MongoDB……	19-22
	3.4.4	React js……	22-23
3.5	Implementation of MERN Development ……		23-
	3.5.1	Project Description……..	22-24
	3.5.2	Development Environmental Setup……..……...	24-26
	3.5.3	Imlementation……..	27-37
CHAPTER 4			38-42
JAVASCRIPT FUNCTIONALITIES AND APPLICATION			

4.1 Introduction & Funtionalities ……..... 38
4.1.1 Basic Function, Method, Features & Versions…….…..... 38-41
4.1.2 OOP in Javascript & Uses…….…….…..... 41
4.2 Application …….…… 41-42
4.2.1 Javascript in Web Application…….…….…..... 42

CHAPTER 5 43-45
NODE JS BACKEND PLATFORM
5.1 Introduction …….…… 43
5.2 Principle Methods in Projects …….. 43-45

CHAPTER 6 46-52
MONGODB DATABASE & FIREBASE
6.1 Introduction …….…… 46
6.2 MongoDB Features…… 46
6.3 Basic MongoDB Example…… 46-47
6.4 Key Components of MongoDB Architecture…… 47-48
6.5 Data Modelling in MongoDB……... 48-49
6.6 Applications of MongoDB in our Project……... 49-50
6.7 Firebase Authentication & features …….…… 50-52

CHAPTER 7 USE CASE & ER DIAGRAM 53
7.1 Use Case……. 53-55
7.2 ER Diagram…… 656
CHAPTER 8 CONCLUSION AND FUTURE ASPECTS 57
8.1 Conclusion ……...……..…... 57
8.2 Future Aspect ……..…….…… 57-58

REFERENCES …….……..…….. 58-60

LIST OF TABLES

Table No.	Title	Page No.
1	Node JS	17
2	MongoDB	20
3	Key Components of MongoDB Architecture	47
4	Use Case	55

LIST OF FIGURES

Figure No.	Title	Page No.
1	Web Application Development	8
2	RESTful API	10
3	Single Page Application	11
4	Node JS	15
5	Node JS (Event loop)	16
6	Express JS	19

7	MongoDB	21
8	MongoDB Server	26
9	Application Folder Structure	27
10	Package.json File	28
11	App.js File	29
12	Model Data of Course	30
13	Drone.js Controller	32
14	Drone Page	33
15	Details Page of a Drone to buy	34
16	Admin Panel	35
17	Responsive Design	36
18	Features of Javascript	40
19	Node JS Backend Platform	44
20	Basic MongoDB Example	47
21	Difference Between MongoDB & RDMS	49
22	ER Diagram	56

CHAPTER 1

INTRODUCTION

1.1 Introduction:

MongoDB is a scalable, flexible NoSQL document database platform designed to overcome the relational databases approach and the limitations of other NoSQL solutions. MongoDB is well known for its horizontal scaling and load balancing capabilities, which has given application developers an unprecedented level of flexibility and scalability.

MongoDB Atlas is the leading global cloud database service for modern applications. Using Atlas, developers can deploy fully managed cloud databases across AWS, Azure, or Google Cloud. Best-in-class data security and privacy standards practices means that developers can rest easy knowing that they have instant access to the availability, scalability, and compliance they require for enterprise-level application development.

As of 2023, MongoDB has been downloaded over 30 million times with over 730,000 MongoDB University registrations. There are drivers for 10+ languages, with dozens more added by the community. Best of all, MongoDB is completely free to use.

With MongoDB, we use MERN Stack Development (MongoDB Express React Node) is one of several variations of the MEAN stack (MongoDB Express Angular Node), where the traditional Angular.js front-end framework is replaced with React.js. Other variants include MEVN (MongoDB, Express, Vue, Node), and really any front-end JavaScript framework can work.

MERN stands for MongoDB, Express, React, Node, after the four key technologies that make up the stack.

- MongoDB — document database
- Express(.js) — Node.js web framework
- React(.js) — a client-side JavaScript framework
- Node(.js) — the premier JavaScript web server

Express and Node make up the middle (application) tier. Express.js is a server-side web framework, and Node.js is the popular and powerful JavaScript server platform. Regardless of which variant you choose, MERN is the ideal approach to working with JavaScript and JSON all the way through.

Using this technology, we created a niche website that sold drones and provided drone information. A niche website is also a kind of e-commerce. A niche site is any website that caters to a specific audience or topic. It can be about anything: health, business, relationships, travel, food, fashion, animals, or the more obscure

E-commerce, also known as e-business or electronic business, is simply the sale and purchase of services and goods over an electronic medium, like the Internet.

It also involves electronically transferring data and funds between two or more parties. Simply put, it is online shopping as we commonly know it. an e-commerce website, by definition, is a website that allows you to buy and sell tangible goods, digital products, or services online. Trade, be it barter exchange or the buying and selling of goods and services, has been prevalent for centuries. No one can be self-sufficient. This highlights the importance of demand and supply for goods and services. Transactions have been going on all over the world for centuries, locally and across locations. Keeping the same concept in mind, now think electronic.

However, also bear in mind that, with the whole world going online, data privacy laws have become increasingly stringent. And before you begin an eCommerce venture, you should be aware of all the legal policies required for your eCommerce website**.** So, we are trying to make an niche website which sell any kinds of Drone, latest types of drone, get an exclusive idea what drone it is and how it works.

It is built with cutting-edge technology and is simple to understand and use. With this type of technology, like the MongoDB database and easy UI interface, the fastest hosting technology can be used to make any other official e-commerce, niche website, or web application for a startup company. This is the main innovation of this project. We can use the same systematic approach to build any kind of e-commerce project for any company.

1.2 Main Purpose of this Project:

Nowadays, data maintenance is a very difficult problem on every platform. So, by constructing this site with the MongoDB database and the MERN stack, we are attempting to solve this type of problem for big data projects—any projects that have encountered data-related and development issues. we are attempting to create an e-commerce (niche) website that is simple to use, simple to understand, and easily accessible to customers due to its minimalistic design.
Most people in our country have no idea how a drone works or operates. We are attempting to introduce to customers and the general public the functions of a drone and how useful they are in our modern era.

So, those are the the main goal of our project. We can use the same systematic approach to build up any kind of e-commerce project or any functional web application for any company.

CHAPTER 2

RELATED TECHNOLOGIES ANALYSIS

2.1 JavaScript:

JavaScript is the most widely used programming language in the world, making it an excellent choice for programmers. When you learn Javascript, you can use it to create powerful front-end and back-end applications utilizing Javascript-based frameworks like jQuery and Node.JS.Javascript is available; it's built into every current web browser, so there's no need to set up a specific environment to study it. Javascript is supported by Chrome, Mozilla Firefox, Safari, and every other browser available today. Javascript enables you to create stunningly gorgeous and lightning-fast websites. You may create a website with a console-like appearance and feel to provide the finest Graphical User Experience to your visitors.

2.2 HTML:

The HTML language was utilized to create headers, titles, tables, and other elements in this web application for global dissemination. HTML stands for Hyper Text Markup Language, which is a markup language used on the World Wide Web. HTML is a computer language that may be used with any other programming language to create a web application. It contains systems for style sheets, scripts, and frames built in. To make web applications more accessible, HTML is employed.

2.3 CSS:

CSS is a style sheet Layout, colors, and fonts may all be separated from the text. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, allow multiple web pages to share formatting by specifying the relevant CSS in a separate.css file, reducing complexity and repetition in structural content, and permit the.css file to be stored to improve page load speed between the pages that share the file and its formatting.

2.4 React Framework:

Reacts is a JavaScript toolkit that allows you to create reusable user interface components. React.js is a JavaScript library for building user interfaces that originated from an internal Facebook project. In addition to the implementation of Web Components, React.js also has a virtual DOM mechanism. Virtual DOM is a highlight of React.js.Virtual DOM establishes an abstraction layer on the basis of DOM. Any changes we make to data and state will be automatically and efficiently synchronized to the virtual DOM, and finally synchronized to the DOM in batches. The introduction of virtual DOM helps to improve the performance of the system.

2.5 MongoDB:

MongoDB database: Use MongoDB database to implement persistent data support. Different from the traditional relational database, MongoDB is a non-relational database, a document-oriented database, with weak consistency, which can obtain faster access speed; storing data in the form of documents can obtain data more quickly; no need to perform the tedious transaction operation is

simpler. In this system, in order to meet the user's frequent data operations and store a large amount of user information, it is necessary to obtain data quickly and reduce the response time of the system, so a simple storage structure and convenient database operations are required. Therefore, choose to use MongoDB as the database for this platform.

2.6 Firebase:

Google Firebase is a Google-backed application development software that enables developers to develop iOS, Android and Web apps. Firebase provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment.

2.7 Node js:

Node.js is a framework for easily creating fast and scalable network applications based on Chrome's JavaScript runtime. Node.js is lightweight and efficient because to its event-driven, non-blocking I/O approach, which is ideal for data-intensive real-time applications that operate across multiple devices.

2.8 Hardware & Sofware Requirments:

1. Operating System: 64-bit Microsoft® Windows® 7-10.

2. Running Environment: Visual CODE.

3. Frontend: React, Java-script, JQUERY CSS3, and Bootstrap 5.

4. Server Site as Backend: NODE JS, Express JS.

5. Database: MongoDB, Firebase.

Others Tools: Adobe Photoshop

CHAPTER 3

MERN STACK DEVELOPMENT

3.1 Introduction:

The MERN (MongoDB, Express, React.js, Node.js) stack was one of the early open-source stacks that epitomized this shift towards SPAs and the adoption of NoSQL. React.js, is a front-end framework based on the model-view-controller (MVC) design pattern, anchored this stack. MongoDB, a very popular NoSQL database, was used for persistent data storage. Node.js, a serve side JavaScript runtime environment, and Express, a web server built on Node.js, formed the middle tier, or the web server. This stack is arguably the most popular stack for any new web application these days.

MERN utilizes JavaScript for easier and quicker full stack web application development. This easy-to-understand full stack JavaScript framework includes, Libraries, Frameworks and Databases

The architecture of the MERN stack:

1. Client layer: It is the top tier of the MERN stack that represents React.js. It is accountable for creating dynamic client-side web and mobile apps. React enables you to reuse the code and handle errors efficiently. It is like a one-size-fits-all solution, helping you to manage lists, forms and events.

2. Server layer: It is the second tier of the MERN stack that consists of two components – Express.js and Node.js. It is accountable for managing HTTP requests and routing URLs. This layer acts as a bridge between the client tier and the database tier.

3. Database layer: It is the third and last tier of the MERN stack, responsible for storing your app data in MongoDB. This layer stores data in the form of a table i.e., rows and columns. You can retrieve the stored data effortlessly and efficiently.

Advantages of choosing MERN stack for modern web/mobile apps:

1. Quick and efficient than the traditional servers, making it apt for small apps
2. More responsive to user inputs, offering a better user experience
3. MVC architecture keeps the presentation details separated from the business logic
4. Full stack development provides support to both frontend and backend app development
5. A fast and resilient web server saves time spent on troubleshooting input/output and CPU issues

3.2 Web Application Development:

3.2.1 Front-end and Back-end:

Web application development is a combination of front-end and back-end development. Front-end web development, also known as client-side development, involvesthe practice of creating a graphic user interface (GUI) for clients (users) so that the application can be interacted with by the users. It involves the use of primary web technologies. and tools such as HTML, CSS, and JavaScript. [2]

HTML is a markup language that provides the structure for a web page. It defines how a web page would look like, so it can be considered the skeleton of any web application. CSS, on the other hand, is a style sheet language that provides style and visual enhancements.

JavaScript is the most advanced language. among these technologies. HTML DOM (Document Object Model) is also used. manipulation to provide a dynamic interface to users. Moreover, it provides an interactive interface to the users by creating pop-up messages, validating form inputs and changing the layout based on events like user input or mouse clicks. All these technologies are controlled by the browser to provide a front-end web interface. [3]

Back-end web development, also known as server-side development, involves the development of computer programs and databases to serve the client. A web application Java did not require a front-end in its early days, only a functional server-side application. was enough for it to be considered a web application. Several changes have been made in this field since then. Today's sophisticated web applications cannot run without both the front-end and back-end services Back-end technologies usually consist of the programming languages such as PHP, Ruby, Python, Java, Node.js, and different frameworks Web application development encompasses all the web technologies related to the frontend and the back-end. In addition, a web application must ensure communication between the client and the server with the use of different communication protocols. Protocols are a set of rules for exchanging messages in a communication network. Protocols vary with different tasks and layers. HTTP (HyperText Transfer Protocol), TCP/IP (Transmissio Control Protocol/ Internet Protocol), FTP (File Transfer Protocol), SMTP (Simple Mail Transfer Protocol), SOAP (Simple Object Access Protocol) and REST (RepresentationalState Transfer) are some common examples of the protocols used in web applications. [4.] a web application, in its most elementary form, sends an HTTP request to a server to establish connection, and the server sends an HTTP response to the client. A typical example of communication in a web application is illustrated in figure 1.

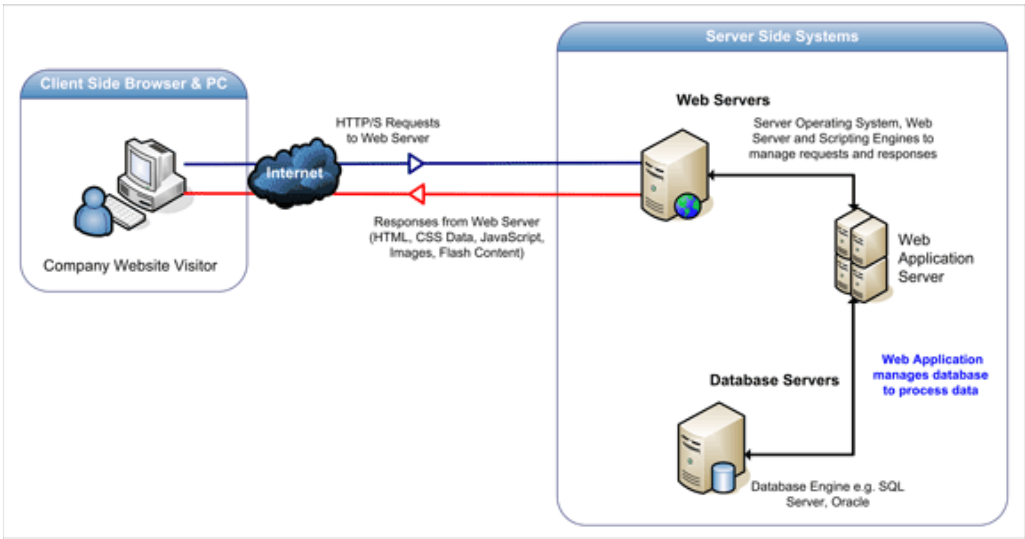


Figure: 1

Figure 1, Illustrates the communication among the three layers of a web application model. The first layer is a client-side web browser, the second layer is a server-side dynamic content generator, and the third layer is a database server. A user sends an initial request using the HTTP protocol through the browser over the internet to the server. The web server then processes the request by accessing the database server and retrieving the requested data. The web server then sends the response to the user over the internet through the browser. The response usually contains the data requested by the user. [4]

Building a solid database management system (DBMS) is essential for storing information a process in the creation of web applications. Data can be created, saved, and updated by users using DBMS. search for data in a web application, and query data. Relational databases and other types non-relational, and. Traditional databases are relational databases, also called SQL databases. databases, which use rows and columns to organize data into tables. the term "relational" refers to a relationship that exists between tables. database'. The three most widely used relational databases are SQL Server, MySQL, and Oracle Database.[5] NoSQL databases, commonly referred to as non-relational databases, store data in different unlike tables. NoSQL databases store data in a key-value pair, graph, document, column, or multi-model depending on the database selected for application development. MongoDB, HBase, Cassandra, Redis, and Riak are some of the examples of popular NoSQL databases. Relational databases are generally used in enterprise applications which can handle large related data, whereas, non-relational databases, which are flexible and scalable in nature, are used in data-driven real-time web applications with rapidly increasing data.[5]

These days, a strong framework for web application development is just as important as a good database. In actuality, web frameworks are used more frequently than native programming languages to create complex web applications. Web frameworks, often referred to as "online application frameworks," are created to make the effort of developing web applications easier by including facilities for database access, templates, session management, and code reuse. JavaScript frameworks like Ember.js, Backbone.js, React.js, and Angular.js, and Knockout.js are the most popular frameworks in the front-end development. Whereas, PHP frameworks (Laravel, Symfony,

CakePHP), JavaScript frameworks (Node.js, Meteor, Express), Rails, Pyramid, ASP.NET, Java EE, Spring, and Django are the most popular frameworks in the backend development. [5]

3.2.2 RESTful API:

In order to establish web services, web development uses the REST architectural style. REST only outlines the foundational ideas for client-server web services and communication. It does not contain a list of guidelines (protocols) for developing web services. Any APIs (application programming interfaces) for online services that are created with RESTful APIs, or simply REST APIs, are what REST architectures are known as. REST offers excellent distributed computer systems' performance, scalability, and dependability. There are several restrictions on what qualifies as a REST application. However, specific implementations of REST APIs must abide by at least four key design tenets. [6]

• Use of HTTP methods: REST APIs are required to explicitly follow the HTTP methods. To access a resource on the server, they must use GET, and to make an HTTP POST, PUT is used to add or update a resource, and DELETE is used to remove a resource.
• Stateless communication: For client and server communication to be successful, each request from the client must provide all the information necessary for the server to process it. The server shouldn't need any saved information to fulfill the request..
• Use of directory-structure like URIs: REST APIs must use the URIs that are straightforward, properly structured, and easily understood, such as
http://www.myservice.org/discussion/topics/{topic}
• Data transfer in XML or JSON: The data transferred between the client and the servic exchange must be in XML or JSON format so the data are properly structuredand readable. [6]

In addition, REST web services (APIs) must have a clear separation of client-side logic and server-side logic. A uniform interface separates clients and servers, which allows developers to work on the individual part of web application and improve one without affecting another. Clients and intermediaries should be able to cache server responses to avoid reuse of stale data in response to future requests. Clients also cannot assume a direct connection to the server. In most cases, intermediaries between the client and the server serve the request-response cycle. Furthermore, in REST web services, server may temporarily extend the client by transferring logic to the client. [6]

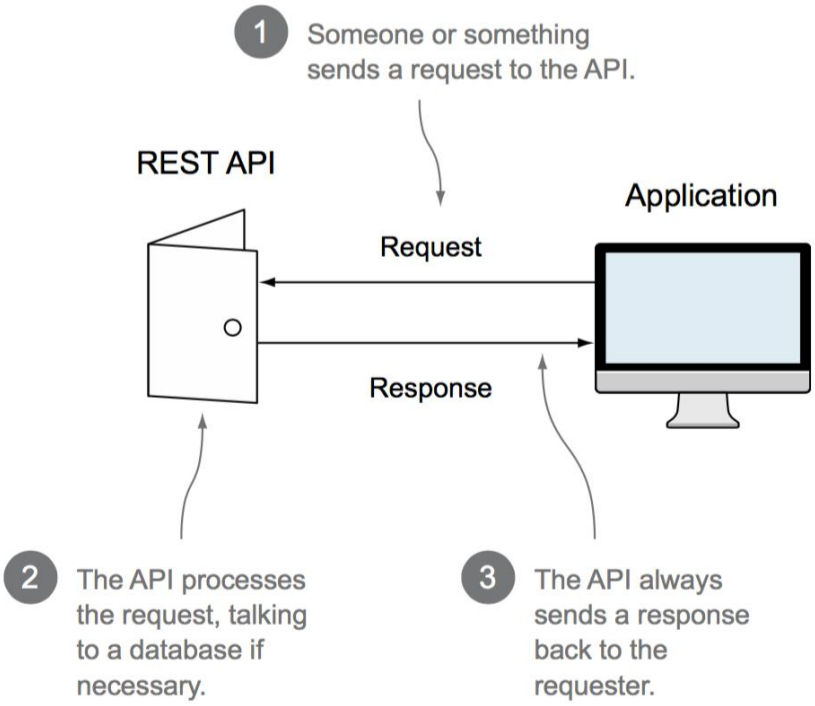


Figure: 2

Figure 2 illustrates the communication between a client-side application and a REST API. The client does not know the implementation of the server and does not directly communicate with the server. The REST API takes all the incoming HTTP requests from the client, processes them, and sends HTTP responses.

3.2.3 Single Page Application:

Single Page Application (SPA) is a web application which fits into a single web page. In contrast to the traditional full page loads, an SPA loads all the resources required to navigate through web application on the first page load. It then dynamically changes the contents as the user intera with the application, so no full page request will ever be made again. However, URLs are updated in the address bar of the browser with a hash tag following the name of the resources accessed.

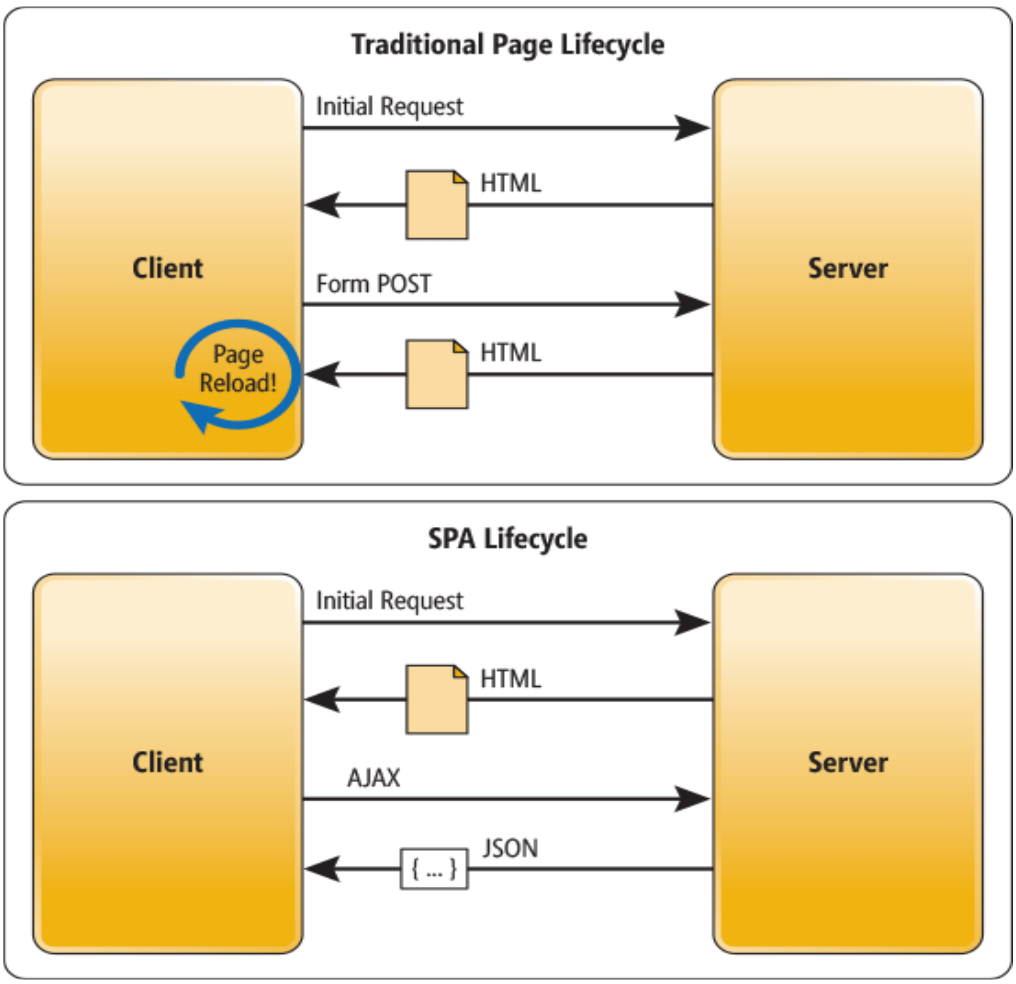


Figure 3: Comparison of traditional page lifecycle and SPA lifecycle [24]

Figure 3 illustrates the distinction between the lifecycle of a traditional web page and an SPA web page. In traditional web applications, every time when a client sends a request to a server, the server will respond by rendering a new HTML page. All the subsequent requests to the server are also processed in a similar fashion and every time a new page is loaded on the client's browser. In an SPA application, after the first page loads, all interactions between the client and the server happen with the AJAX (Asynchronous JavaScript and XML) calls, which in return send data in JSON (or XML) form. The browser then uses the JSON data to update the page dynamically, without any page reloads. [24]

3.3 Full Stack JavaScript:

3.3.1 JavaScript:

JavaScript is a cross-platform, light-weight, object-oriented scripting language designed primarily for adding interactivity to modern web pages and web applications. It is one of the three main components of the front-end technologies besides HTML (providing structure to the web page) and CSS (providing style to the web page). It was developed by an American programmer Brendan Eich at Netscape in 1995. Originally called by the names 'Mocha' and 'LiveScript', Javascript got its current name in December 1995 with the release of the third beta version. [25]

JavaScript was submitted to ECMA (European Computer Manufacturer Association) for standardization so that other browsers could implement it on the basis of the standardization. ECMA international then published the standard JavaScript language specifications under the name ECMAScript. The current JavaScript specification is based on the ECMAScript 2016, which was released in June 2016. [25]

JavaScript provides rich features to create dynamic and interactive interfaces in clientside applications. It can detect the user's browser and OS (Operating System), and performs platform-specific operations. Originally termed as an interpreted language, JavaScript can be executed without preliminary compilation by the browser so it can perform simple calculations on the client-side. It is often used to validate the user's input in forms send the data asynchronously with the AJAX. It also performs HTML DOM manipulation to provide dynamic interface. It is an object-oriented programming language and supports several built-in objects with inheritance. Thus, JavaScript enables developers to add dynamic features to static HTML pages, control multimedia and add animations.Moreover, JavaScript provides several third-party APIs and libraries to facilitate the task of building dynamic web pages. One of the most popular JavaScript library is JQuery. [25]

JQuery is a library based on JavaScript, developed by an American programmer John Resig. All the functions that can be programmed in traditional JavaScript can be performed in JQuery, but with much simpler methods. Therefore, a function that needs several lines of codes to be written in JavaScript can be easily written in few lines using JQuery. It is almost supported by all modern browsers and does not require extra plugins or extensions to run. One of the main features of JQuery is DOM manipulation. DOM is a tree-like structural representation of the elements in an HTML page. JQuery syntax makes it easy to navigate throughout these elements, find and manipulate them. For example, it can be used to find all elements with a certain property in a web page, change the elements' properties or make them respond to certain events like mouse clicks. This task is difficult to perform in pure JavaScript but JQuery's syntax makes it easier. [26] For example, in order to change the background colour of the body in an HTML page,

one writes the following function in JavaScript:

```
function changeBackground(color) {
document.body.style.background = color;
}
onload="changeBackground('red');"
```

whereas, the same task can be performed in one line of code using the following JQuery function:

```
$('body').css('background', '#ccc');
```

Moreover, JQuery provides simpler functions to create animations, handle different events, and develop AJAX applications. AJAX (Asynchronous JavaScript and XML) is a web development technique to create asynchronous web applications. With the use of AJAX, asynchronous communication can be made between the client and the server. AJAX uses XMLHttpRequest object to communicate with the server, and sends and receives information in XML, JSON, HTML, or even in text file format. XMLHttpRequest is an API that allows asynchronous communication between the client and the server without the need of a full page reload. Hence, AJAX allows to update a certain portion of a web page without page refresh (reload). [27]

3.3.2 Rise of Full Stack JavaScript:

JavaScript in its primary days earned a bad reputation because of the lack of performance and its incompatibilities with the prominent browsers of that time. However, things started to change after large browser vendors invested time and money in improving the language. JavaScript, finally, became the de facto standard of the client-side scripting. While JavaScript remained prominent in the client-side, several new technologies were introduced in the server-side such as PHP, JAVA, .NET, Ruby, and Python. Developers

started to realize that use of two separate languages in the development of the client and the server was complicating the tasks of web programming. Several attempts were made to unify the two sides by creating client components on the server and compiling them into JavaScript (for example: ASP.NET web forms and GWT), but they failed. The only solution to this problem was the implementation of JavaScript on the server-side, and Node.js was introduced. Node.js is actually the backbone of Full Stack JavaScript web development. It finally put the power of JavaScript on the server with the idea of non-blocking programming paradigm. Node.js became popular in a short time due to its easy-to-use components. It allowed the developers to quickly set up a server and start building applications on top of it. Several frameworks started to emerge to facilitate Node.js implementation such as Express and Connect.js. Express became the most prominent one. Node.js ecosystem continued to expand and a package manager like ‘npm’ was introduced. While Node.js was invading the server, NoSQL database started to gain popularity in the field of database. MongoDB, a NoSQL database, was introduced with the concept of storing data in Binary JSON. Due to the use of JSON, which is a JavaScript way of storing data, MongoDB became the preferred database for Node.js applications. JavaScript also started to evolve on the client-side and new frameworks like AngularJS, Backbone. js, and ReactJS started to take over the use of traditional JavaScript, JQuery, and AJAX for building Single Page Applications. When all the prominent tools to build a Full Stack JavaScript application were ready, additional tools based on JavaScript were introduced

to improve the development process such as Mocha.js and Chai.js for application testing, Gulp.js and Grunt.js for automation of build tasks.

Node.js eco-system, with addition of React JS on the front-end, Express as the backendframework, and MongoDB as database raised a new term in web application development called MERN (MongoDB, Express.js, ReactJS, Node.js) stack. The word MERN stack is used to refer Full Stack JavaScript but it is still a subset of the term Full Stack JavaScript. Node.js, Express, and MongoDB are the prominent members of the Full Stack JavaScript. AngularJS, however, can be replaced with Backbone.js, AngularJS, or Ember.js to meet the developers' requirements. Nevertheless, the term Full Stack JavaScript is generally used to refer to the MERN stack.

3.4 MERN Stack Development:

3.4.1 Node.js:

Node.js is a software platform which helps to build asynchronous and event-driven network applications. It contains built-in HTTP server libraries which allow developers to create their own web server and build highly scalable web applications on top of it. The V8 JavaScript runtime engine used by Node.js is the same engine used in Google's Chrome browser. [7] The V8 engine complies the codes directly to the native machine code leaving out the interpreter and byte code execution process, which gives Node.js a huge boost in performance [8]. In addition to the V8 engine, Node.js is composed of several components as shown in figure 4.

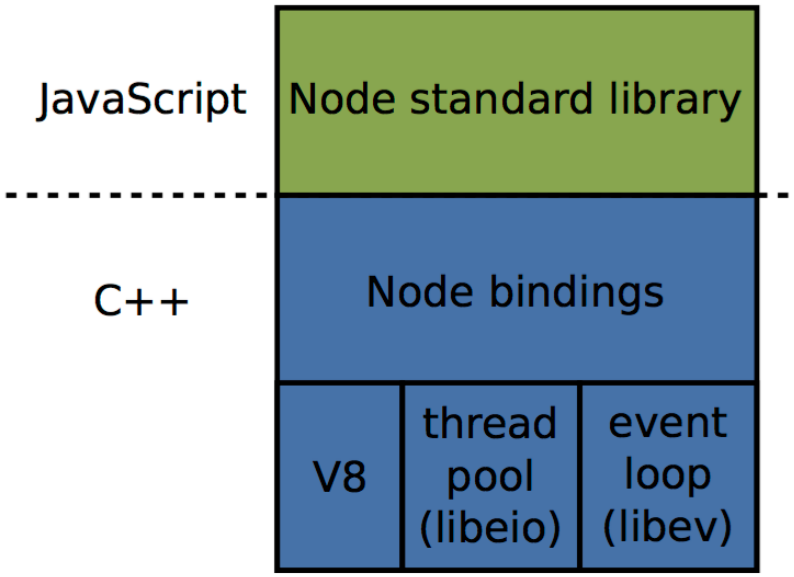


Figure 4

Figure 4 illustrates the architecture of Node.js. Node.js is composed of Node standard library at the top, thin C++ node bindings in the middle, and V8 engine, libeio and libev at the bottom. Node standard JavaScript library exposes operating system features to the application, while the C++ bindings expose the core APIs of the underlying elements to JavaScript. The V8 engine provides the run-time environment for the application, and libeio handles the thread pool to make asynchronous (non-blocking) I/O calls to libev, the event loop. [8] The asynchronous, non-blocking I/O feature of Node.js plays an important role in resource management and performance enhancement of Node.js applications. Unlike other mainstream servers like Apache and IIS, Node.js uses single-threaded, non-blocking I/O operation. What this means is that instead of running each session (request) in a separate thread and providing an associated amount of RAM for each session, Node.js uses a single thread to execute all requests and implements an event loop to avoid blocking of I/O. In a multi-threaded server, as the number of threads increase, the server overhead (scheduling and memory footprints) increases resulting into a slow performance of the overall system. Node.js uses an event-driven and non-blocking I/O approach

to handle all the requests to the server. [10;11]

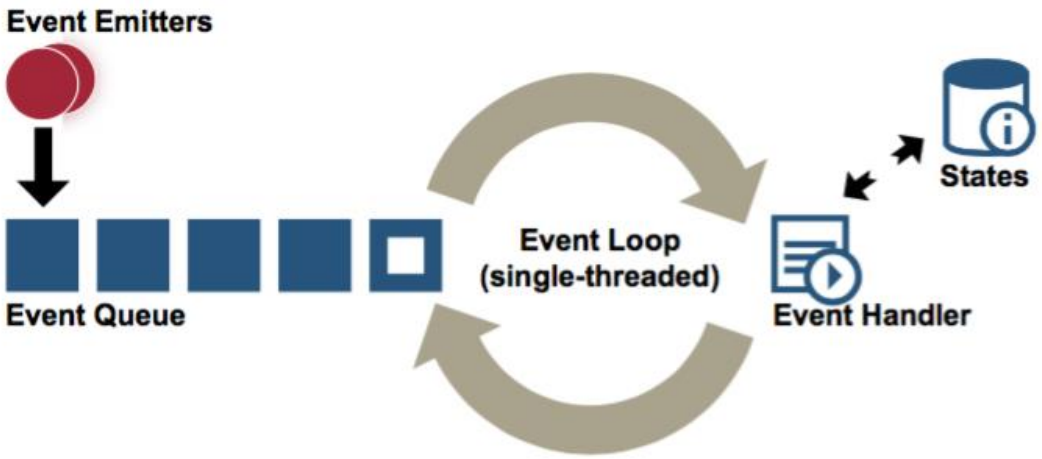


Figure 5

In Figure 5, Node.js creates an event-loop with event handlers for all requests. When an I/O operation occurs, the associate handler is queued up for execution and a callback function emits an event after the I/O operation is completed. In the mean-time, other I/O operations keep running outside the event loop of the server. Thus, Node.js performs the I/O operations asynchronously and does not block any script execution, allowing the event loop to respond to other requests. [10.] One common use of a callback function displaying the non-blocking I/O operation of Node.js is shown in listing 1.

```
var fs = require("fs");
fs.readFile('input.txt', function (err, data) {
if (err) return console.error(err);
console.log(data.toString());
});
console.log("Reading file");
```

Listing 1. Node.js non-blocking code example: Listing 1 illustrates the non-blocking code example of Node.js. The program is trying to perform three operations: read a file ‘input.txt’, print the file’s data in console, and print message “Reading file” in console. The function readFile() executes first in the order of appearance but the program does not wait for file-reading function to complete. Once the function starts to read the file, it passes control to execute the next instruction immediately so the program prints “Reading file” before printing the file’s data. Once the file I/O is complete without any error, it will call the callback function function (err, data) and returns the file data as parameters, then prints the data below the ‘Reading file’ message in the console. Hence, there is no blocking of I/O operations and all the operations execute asynchronously. Node.js has

a rich development eco-system with several compatible libraries and package managers. One of the main package managers that comes pre-bundled with Node.js during installation is npm. Npm is run from the Command Line Interpreter (CLI) and it manages all the dependencies for the Node.js applications. Instead of manually downloading and configuring the JavaScript modules for use in the application, npm provides a simpler alternative. The names of the modules and dependencies can be included with their version numbers in 'package.json' file inside the folder structure, and the modules are downloaded issuing a simple 'npm install' command from the CLI. [12,37-42.] The npm automatically handles all the downloading process and save all the named modules in the 'node-modules' folder. The modules can be updated by changing their version number in the 'package.json', and can be easily distributed by uploading a single 'package.json' file to the server instead of uploading the large 'node-modules' folder. After the update or upload, issuing the 'npm install' command installs the specific modules and dependencies. Therefore, Node.js applications are light-weight, flexible and easily sharable. [12,43-47.]

Node.js is actually the foundation of the MEAN & MERN stack. All the other technologies work on its foundation. Node.js introduces the power of JavaScript on the server side allowing the developers to create both server side and client side logic in one single language 'JavaScript'. One of the major advantages of Node.js over other server-side platforms is the built-in event loop. The event loop is used by all the available modules and libraries in the Node.js which makes the I/O operations efficient. [10] This Node.js feature maintains the speed and efficiency of the overall system.

Web development technology	Calculate value of Fibonacci	Mean requests per second [#/sec]	Mean time per request [ms]
Node.js	Fib (10/ 20/ 30)	2491.77/ 1529.4/ 58.85	0.401/ 0.654/ 16.993
Python-Web	Fib (10/ 20/ 30)	633.68/ 209.89/ 2.9	1.578/ 4.764/ 345.307
PHP	Fib (10/ 20/ 30)	2051.22/ 168.8/ 1.78	0.488/ 5.942/ 560.553

Table 1

Table 1 illustrates the performance results of the Node.js in comparison with Python-Web and PHP when calculating Fibonacci (10/20/30). The study was conducted by the researchers in Peking University, China and results were published on the technical report by IEEE. It is clearly seen that the Node.js is better in handling requests and performing calculations than the other two technologies. When calculating the small Fibonacci of 10, Node.js and PHP, both perform well, whereas Python-Web lags behind. Node.js handles about 2500 requests per second taking 0.401ms time to handle each request. PHP falls short by only few numbers but the gap in their performances increases drastically when the calculation tends to become complex. When the task reaches to the calculation of Fibonacci of 30, both PHP and Python-Web perform poorly, almost processing 2 or 3 requests per second, and taking 345ms and 560ms to process each request respectively. In contrast to them, Node.js still maintains the quality by processing requests 20 times higher than PHP and taking a relatively short time, about 17ms. [13] Although Node.js is an ideal choice for a scalable, data-driven, I/O intensive applications, it is not a perfect solution for all applications. It uses JavaScript so it is more efficient

when used with other JavaScript-based technologies. The use of single-thread to handle all requests is ideal for some cases but it is not a good feature for intensive data computing applications. Moreover, Node.js uses tight coupling between the web server and the web application so all the classes are dependent on each other. Such tightly coupled system is hard to maintain since a problem in one area causes the whole system to fail. Node.js also does not work well with the relational databases. All these factors must be considered before choosing Node.js as a development platform for any application. Studies have shown that Node.js is ideal for real-time data-driven web applications in collaboration with push technology and web sockets. [8]

3.4.2 Express JS:

Express is a node module which provides a minimal and flexible framework for Node.js web applications. It works on top of the core node modules without hiding any of the features of Node.js. In addition, it provides robust and clean functions to add to the node modules so the development of Node.js application using Express is far easier than using the native node modules. [14.] Due to the simplicity of Express, it has been adopted by large companies such as MySpace, Paypal, Apiary.io, Persona, and Ghost. The use of Express framework on top of Node.js helps to maintain clarity of the code. It also makes module integration easy to handle, and provides a solution structure for applications. [15, 142.] Express is installed using the npm package manager issuing “npm install express” command [14].

An Express server is made up of three building blocks: router, routes, and middleware. A web server’s core functionality depends on its excellent routing methods. In a clientserver communication, a client requests some resources from the server, the server locates the resources and responds by sending the resources to the client. This is the core functionality of a web server and it requires excellent routing methods to serve the request. Express makes this tedious job really easy by allowing developers to create routes in simple structure. A route in Express is a combination of a HTTP verb and a path. The HTTP verb is generally one of the four HTTP methods: GET, POST, PUT and DELETE, and the path is the location of the resource (URI). [15,144-146.] A basic route in Express is created as below:

app.METHOD (PATH, HANDLER) where:

• app is an instance of express
• METHOD is an HTTP request method
• PATH is a route path (URI)
• HANDLER is the function which executes when the route is matched.

Middleware in Express are the functions that have the pattern function (req, res, next). The req is the incoming request from the client, res is the response from the server, and next is the callback function. Therefore, middleware functions execute any code inside it, handle request and response objects, end request-response cycle, and call the next middleware function. A current middleware function must always call the next middleware function, even in the case of incomplete request-response cycle to

16

avoid request hanging. [14;15] A simple Express web server containing all three building

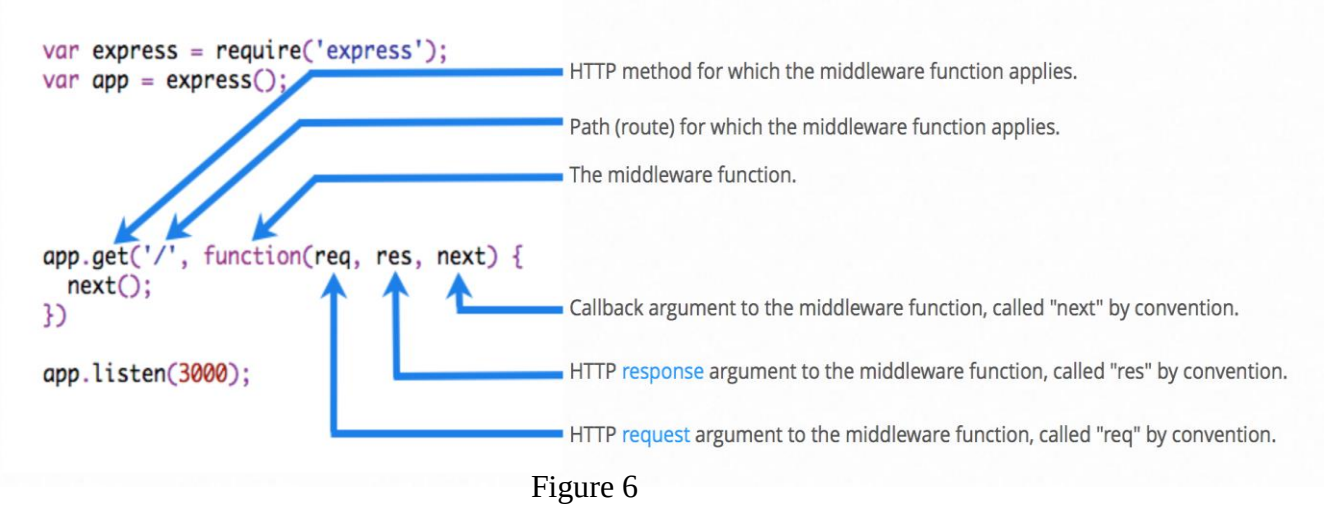


Figure 6

Figure 6 illustrates the three building blocks of an Express application: router, route, and middleware. The first two lines uses Node.js require() method to load express module in the application by creating the app object. The third line is a simple router with a route to '/' location and a middleware function. The application listens to port 3000 for any request. Express JS also provides a simple application generator tool for providing structure to our Node.js application. It can be installed from the CLI by issuing 'npm install express-generator' command. It also provides the options for creating template engine to write HTML codes. The application generator, by default, creates jade templates for the views, but it also supports Handlebars, EJS, Pug, and Mustache. The application structure created by the Express generator has separate directory for routes, views, and public-rendering files. However, the application structure is just one of many ways to structure an Express application. It can be easily modified during the application development process to meet the requirements of the application. [14]

3.4.3 MongoDB:

MongoDB is an open-source, non-relational, document database. It deviates from the need of creating Object Relational Mapping (ORM), for rapid application development. [16.] Unlike the relational databases, it does not contain columns and rows. However, the concept of rows still exits in MongoDB but it is called a document. The document defines both itself and the data it contains. A document is a set of fields and it can contain complex data such as lists, arrays, or an entire document. Each document contains an ID field which can be used as a primary key for a query operation. A set of documents is called a collection and MongoDB holds a set of collections. The format in which the MongoDB stores the data is called BSON, which stands for binary JSON. Since JSON is the JavaScript way of storing data, MongoDB works perfectly with the applications built with JavaScript stack. [11,13-15.] A basic example of a MongoDB document is illustrated in listing 2.

```
{
"firstName": "Homer",
"lastName": "Simpsons",
```

_id: ObjectId("52279effc62ca8b0c1000007")
}
Listing 2. Example of a MongoDB document
Listing 2 illustrates a code snippet from a MongoDB collection. The document stores the first and last names of a customer. Unlike traditional relational databases, MongoDB does not hold a data set corresponding to a set of columns, instead it uses the concept of name-value pair to store data [11,14]. The _id is the unique identifier (primary key) for that set of data (document). There are some variations in the naming terms of relational database and MongoDB, illustrated in table 2.

MySQL	MongoDB
Database	Database
Table	Collection
Index	Index
Row	BSON Document
column	BSON Field
Join	Embedded documents and linking
Primary key	Primary key
Group By	Aggregation

Table 2

As illustrated in table 2, in MongoDB, some MySQL terms like table is called collection, and row is called BSON document. Instead of Join operation, MongoDB embeds sub documents inside the main document and provides links to the sub documents. MongoDB, like MySQL uses unique identifier (primary key) for each document so that it is easy to query and find the data. MongoDB supports insert, query, update, and delete operations like any other databases. [17;11]

One of the features that makes MongoDB stand out against the traditional databases is the inclusion of dynamic schema. Collections in MongoDB have different schema and the documents within the same collection can have as many different schema and shapes as required. This feature enables developers to start storing data in the database with any consideration of the database structural design. The documents' keys and values can be changed and updated when required since there is no pre-defined rule governing the data type validation. [18,7-12]

Other important features of MongoDB are auto-sharding and replication. Since the data are stored in a document, they can be stored across multiple locations. As the size of databases increase, a single machine may not be able to store data and handle readwrite operations. MongoDB solves this problem by allowing horizontal scaling, meaning that the data are distributed to multiple servers and all servers can work together to support data growth and provide efficient read-write operations. Likewise, the data can also be replicated to another data server so that the data are always available in case one server fails. [16.] This allows to build highly scalable and efficient data servers in comparison to relational databases such as MySQL and Oracle databases.

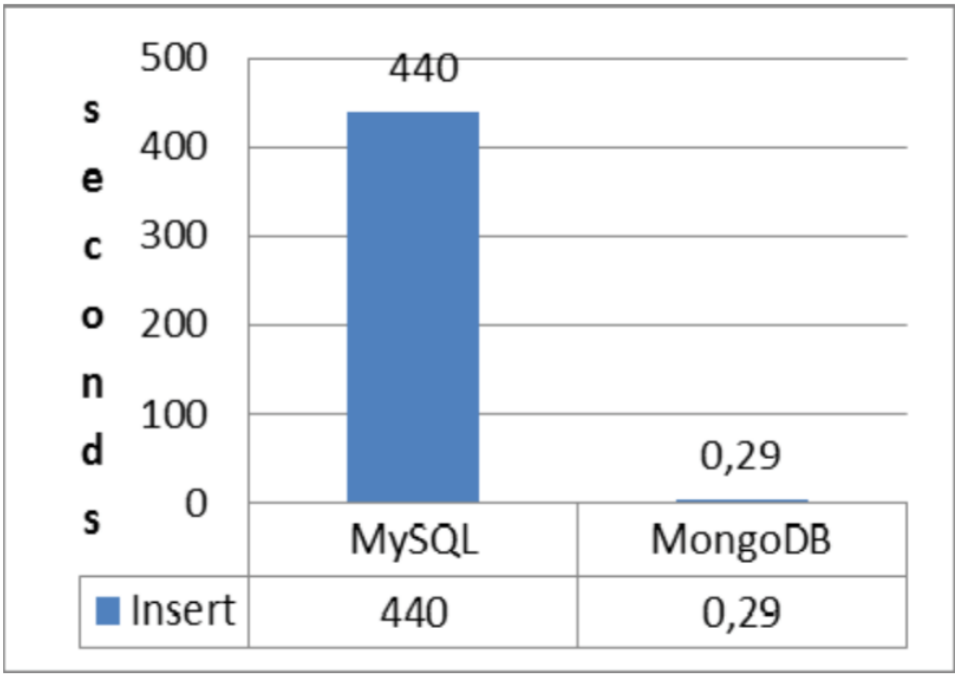


Figure: 7

Figure 7 illustrates the time taken by MySQL and MongoDB when running a script to insert 10,000 users' data. MySQL which uses traditional database approach took significantly longer time than MongoDB. It shows that MySQL took 440 seconds to insert 10,000 users, while MongoDB only took 0.29 seconds to perform the same task. It proves that the MongoDB is excellent in performing large read-write operations. [17]
Although MongoDB is good at performing majority of tasks, it has its drawbacks. MongoDB can not take multiple operations as one transaction. If any operation in one transaction fails, it will cause the entire operation to fail. It also can not perform the join operations like MySQL database so it is not a good choice in an application where there are multiple relationships among the data. The data has to be searched and updated in multiple documents at once and all operations need to be tied in a single transaction to ensure that the data is updated in all collections. Transactional databases work better in such cases than the non-transactional database like MongoDB. Although the flexibility of MongoDB to allow any sort of data is good in some cases, most applications still need certain kind of structure to their data to work properly. To solve this problem, Mongoose was created by the company behind MongoDB. [11,15]
Mongoose is a MongoDB data modelling for Node.js. It was created to solve the problem of writing complex data validation, casting, and business logic in MongoDB. It provides simple, elegant, data-modeling feature to the application. Using mongoose, one can define what kind of data can be in a document and what data must be in a document. In technical term, it provides data validation rules, query building functions, and business logic to the application data. Moreover, it provides an entire set of new features to use on top of MongoDB. It can manage the connections to MongoDB database, as well as

read, write, and save data. It also allows only valid data to be saved in MongoDB by providing validation rules at the schema level. [11,15]

3.4.3 React JS:

React.js, more commonly known as React, is a free, open-source JavaScript library. It works best to build user interfaces by combining sections of code (components) into full websites. Originally built by Facebook, Meta and the open-source community now maintain it. One of the good things about React is that you can use it as much or as little as you want! For example, you can build your entire site in React or just use one single React component on one page. [19]

React.js is built using JSX – A combination of JavaScript and XML. Elements are created using JSX, then use JavaScript to render them on your site. While React has a steep learning curve for a junior developer, it's quickly shaping into one of the most popular and in-demand JavaScript libraries.

React is considered a JavaScript library rather than a framework, whereas the other options we'll consider today are considered frameworks. It helps to think of a library as a tool that developers could use in any project and a framework as a whole design.
There has been a massive rise in the popularity of React.js, as found in the studies by State Of JS. This is partially due to its flexibility and speed of development but also helped by the fact that it is supported by Meta, making developers and companies feel safe in their decision to use React. Therefore, the demand for React developers is very high. As a result, there is a wide range of jobs for developers who know how to use React.

Vue.js vs React.js
So why would you choose React instead of Vue.js, or vice versa? The big bonus to React is that it's maintained by Meta – a tech giant. Such strong support from a main player in the tech world provides React the stability and long-term support that most libraries just don't have. This gives developers the confidence that React won't be deprecated in the near future, and developments will continue to improve it.
On the other hand, Vue was built by one developer, Evan You. Evan is a former Google employee who also worked on AngularJS. He aimed to combine the best approaches found in Angular, Ember, and React with other features to speed up the process of development. While Vue is considered a truly open-source project, it fully relies on the community, contributors, and crowdfunding to progress, so it lacks React's stability. [20]

The main differentiator between React and Vue.js is syntax. Vue uses HTML templates, with the option to write in JSX, whereas React solely uses JSX, combining the HTML, CSS, and JavaScript together. Due to this, the separation that Vue provides between HTML, CSS, and JavaScript makes Vue a more beginner-friendly library with an easier learning curve than React. Vue is also incredibly well-documented, making it easy to find answers to questions that a developer might have.

App.jsx: A Simple React Class

```
const contentNode = document.getElementById('contents');
class IssueList extends React.Component {
render() {
return (
<div>This is a placeholder for the issue list.</div>
);
}
}
ReactDOM.render(<IssueList />, contentNode); // Render the component inside 
the content Node
```

If you refresh your browser, you should see the placeholder text. Now, let's examine what happened here in a little more detail.

The render() method is something that that the React framework calls when it needs to display the component. There are other methods with special meaning to React that can be implemented, called the Lifecycle functions, which provide hooks into various stages of the component formation and events. I'll discuss them in later chapters. But render() is one that must be present; otherwise you'll have a component that has no screen presence.

Within the render() method, we're supposed to return a native component instance, or an instance of a component defined by us. In this case, we returned a element, just like the first Hello World example used an <h1> element. We don't really need the brackets around the but it's convention, and it helps readability when rendering amore complex or nested set of elements.

3.5 Implementation of MERN Development:

3.5.1 Project Description:

The web application provides minimalistic features of an online drone sales platform. An admin can add drones, provide drone descriptions, embed drone images, manage drones, check the order list, manage orders, make new admins, and manage reviews. A user can simply browse the drones and request to buy with the necessary information. The application provides the basic foundation of an online shopping platform, which can be extended later as a full e-commerce website with payment methods.After analysing the requirements, the following interfaces were implemented.

• An interface for admin to create drones, add description, embed image URLs, and other drone's materials.
• An interface for user to browse drones, view drones' teaser videos, and request to buy the drone providing the email address and other necessary information.
• An interface for admin to manage the drones, users and orders.

In addition to the above requirements, the application should also meet the following best practices set by the standard MERN application:

- The application should be built with the MVC architecture.
- The application should be a Single Page Application (SPA).
- The application should consist of REST APIs for data communication.
- The application should follow the responsive mobile-first design principles.

Based on the requirements, the application was built in the MVC architecture. There is a complete separation between the server-side and the client-side logic. The model is implemented in the server, and the view and the controller are implemented in the client. Moreover, the application does not communicate directly to the server, but uses REST APIs to feed the data. The application uses the routes created in ReactJS to navigate throughout the web pages but all data are served by updating a single page. This SPA REST API approach and the MVC architecture of the application are illustrated.

Request-response flow in MVC application:

It illustrates the MVC architecture of the application. It also illustrates the routing mechanism the application uses to access specific web pages in the application. A request from a visitor is routed to the controller. The controller, if needed, makes a request to the model and the model responds to the controller. The controller then sends a response to the view and the view renders the page on the visitor's browser.

Its illustrates a common MERN stack architecture that the application follows. A REST API is built using MongoDB, Express and Node.js, and a Single Page Application (SPA) is created using AngularJS. The REST API then feeds JSON data to the Reacts SPA on the browser.

3.5.2 Development Environment Setup:

The application was developed in HP pavilion 2018 running the operating system OS windows 10.11.6. Besides the hardware, several programs and tools were down loaded and configured to set up an environment for the MERN application.

Web Storm IDE (Integrated Development Environment):
WebStorm IDE is a lightweight and powerful IDE which facilitates the development of a MERN application. It has built-in git support and embedded CLI. It has also built-in support for the latest technologies such as Node.js and AngularJS. It can be downloaded from the official website: https://www.jetbrains.com/webstorm.
Node.js: Node.js is the first program to be downloaded since its provides the platform on which a MERN application is built. It can be downloaded from the website: https://nodejs.org/en/download.
After the download and installation, it can be checked by issuing node –v command from the terminal or CLI as illustrated in this code

```
helios:~ anuj$ node -v
v4.5.0
helios:~ anuj$
```

Node.js version can check by this code in the command promote to view the Node.js version installed on the computer. The version used in the application is 4.5.0. It confirms that Node.js is properly installed in the development machine,

Express: After the installation of Node.js, a root directory is made and inside the directory, express is installed issuing command from the terminal npm install –g express. An express generator can also be installed to provide simple structure to the application. All the node modules and dependencies were installed using npm install command.

MongoDB and Mongoose: MongoDB is installed from the official web site: https://www.mongodb.com/download center#community. After locating the downloaded folder, MongoDB server can be started with simple command from the terminal: mongod as illustrated in figure 13.

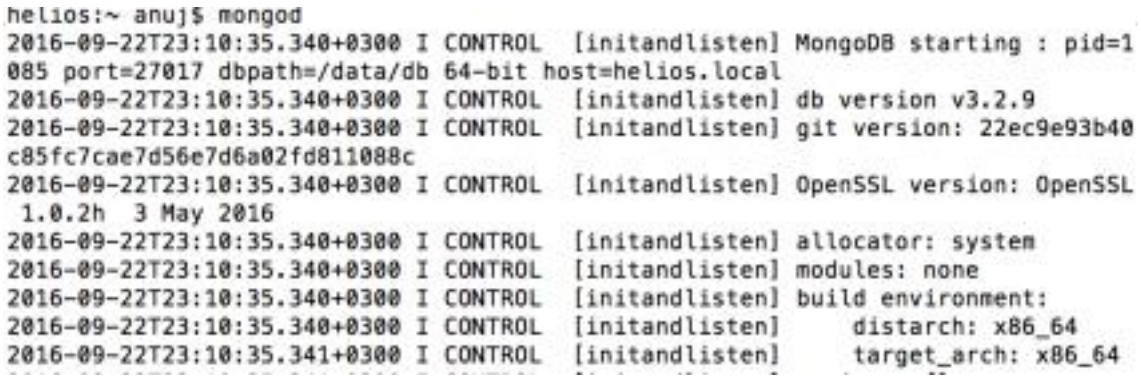


Figure 8. Starting the MongoDB server

As illustrated in figure 8, MongoDB can be started with the single command mongod. Other information about the MongoDB can also be seen in the figure. Mongoose was installed using the command: npm install mongoose.

Nodemon: Nodemon is a utility that monitors the changes in the source code during development and automatically restarts the server. It is downloaded using the command: npm in stall –g nodemon, and implemented using the nodemon command.

3.5.3 Implementation:

ReactJS, React Routes, JQuery, Bootstrap (front-end packages): There are several ways to download the front-end packages. In this application, bower was used to download and configure the front-end package. Bower is the package manager for the front-end, whereas npm is the package manager for the back-end. All the modules and dependencies were downloaded using npm and bower. After the installation and onfiguration of all the required tools and programs, the development process was started.

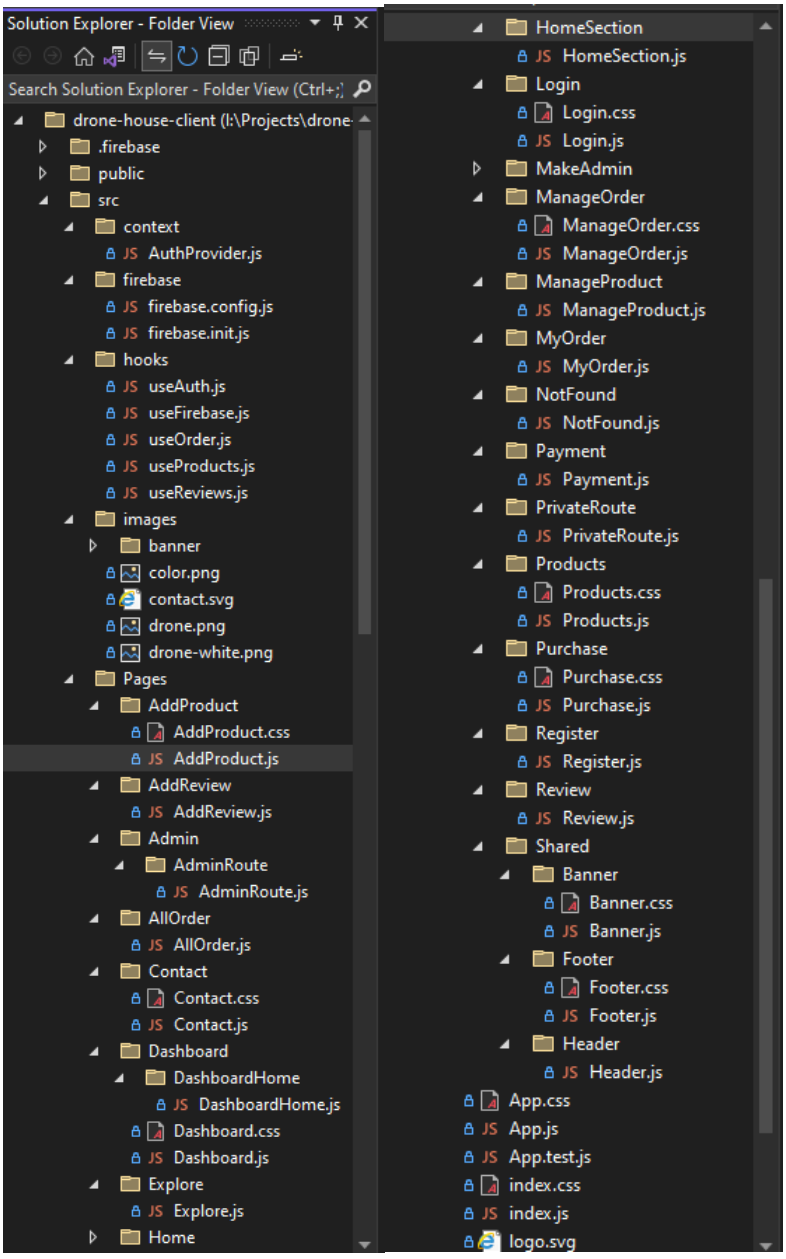


Figure 9. Application folder structure

Figure 9 illustrates an overview of the application's folder structure as seen in Web Storm IDE. The 'package. json' file contains all the dependencies required to run the ap plication. The 'app.js' file (~/meanapp/app.js) contains all the Express middleware, and API routes of the application. The 'www' file contains the command to start server at port 3000. There is a clear division between the server-side logic and the client-side logic. All the client-side scripts are placed under the 'public' directory. The models, views and controllers are separated as per the MVC architecture. The data models are placed in

the server-side, whereas views and controllers are placed on the client-side. All the server-side dependencies are installed in 'node_modules' folder and all the client-side dependencies including ReactrJS are installed in 'bower components' folder.

package.json: The 'package.json' file contains all the important information about the application and the dependencies required to build and run the application.

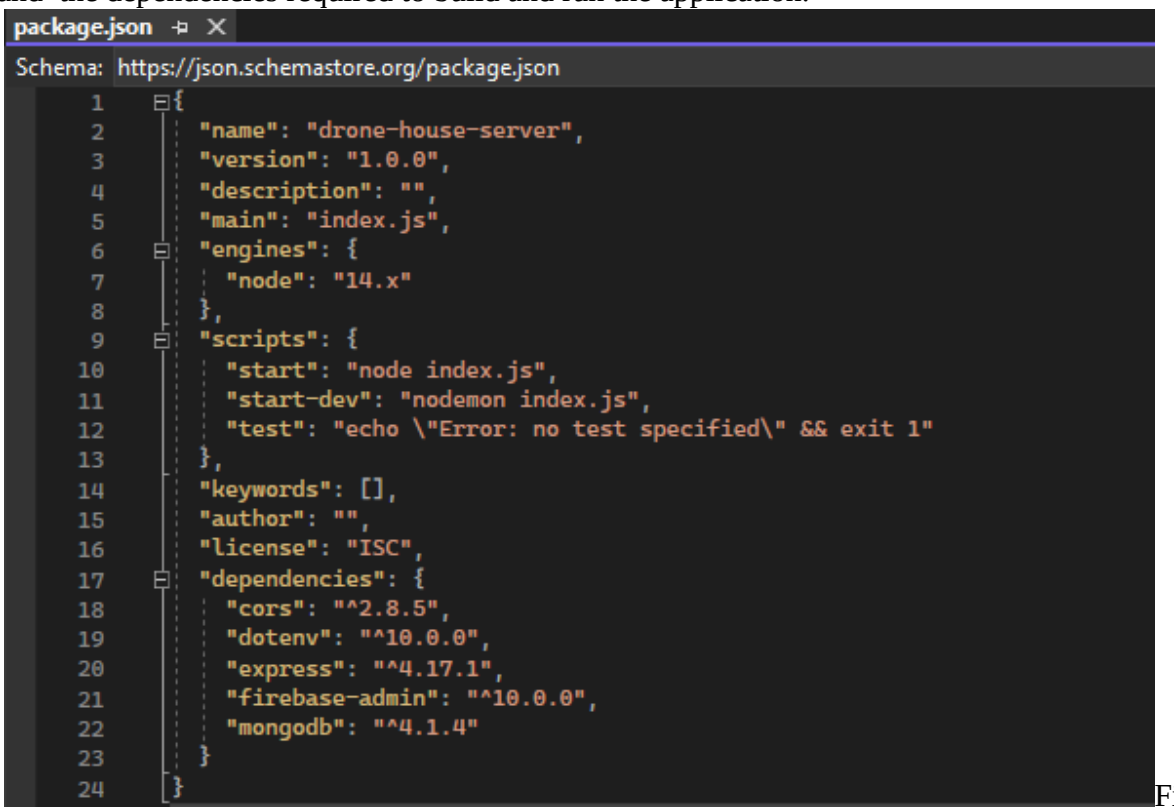


Figure 10. package.json file

Figure 10 illustrates the 'package.json' file's contents which provides overall information about the application. It contains the application name, version, and all the dependen cies. On line 6, the code shows that the application needs the scripts on /bin/www file to start the server and it can be started with the node ./bin/www command.

app.js (Server-side): The app.js file is the main configuration file of the Express application. It configures the server and imports all the modules to run the application.

App.js

Miscellaneous { } "App"

```
1   import { BrowserRouter as Router,Switch,Route } from 'react-router-dom';
2   import './App.css';
3   import Home from './Pages/Home/Home';
4   import Explore from './Pages/Explore/Explore';
5   import Dashboard from './Pages/Dashboard/Dashboard';
6   import Login from './Pages/Login/Login';
7   import Register from './Pages/Register/Register';
8   import AuthProvider from './context/AuthProvider';
9   import PrivateRoute from './Pages/PrivateRoute/PrivateRoute';
10  import Purchase from './Pages/Purchase/Purchase';
11  import NotFound from './Pages/NotFound/NotFound';
12  import Footer from './Pages/Shared/Footer/Footer';
13
14
15  function App() {
16    return (
17      <AuthProvider>
18        <Router>
19        <Switch>
20            <Route exact path="/">
21              <Home></Home>
22            </Route>
23            <Route path="/home">
24              <Home></Home>
25            </Route>
26            <Route path="/explore">
27              <Explore></Explore>
28            </Route>
29            <PrivateRoute path="/dashboard">
30              <Dashboard></Dashboard>
31            </PrivateRoute>
32            <Route path="/login">
33              <Login></Login>
34            </Route>
35            <Route path="/register">
36              <Register></Register>
37            </Route>
38            <PrivateRoute path="/purchase/:name">
39              <Purchase></Purchase>
40            </PrivateRoute>
41            <Route path="*">
42              <NotFound></NotFound>
43            </Route>
44        </Switch>
45        <Footer></Footer>
46      </Router>
47      </AuthProvider>
48    );
49  }
50
51   export default App;
```

Figure 11. app.js file

Figure 11 illustrate the contents of app.js file. On line 1, it imports the express framework to get started. It imports node modules for handling paths, serving favicon, logging re quest and responses, and handling forms and cookies. Line 16 shows the connection to our MongoDB server, where 'meanapp' is the name of our database. Line 20 tells the application to use public folder for rendering views. The app.use() function tells the application to use the parameters given inside the brackets.

Models: After creating the database, and configuring Mongoose and Express (See Figure 16, line 8, 16, and 17), model classes were created using Mongoose. A simple schema for the course is illustrated in figure 17 below.

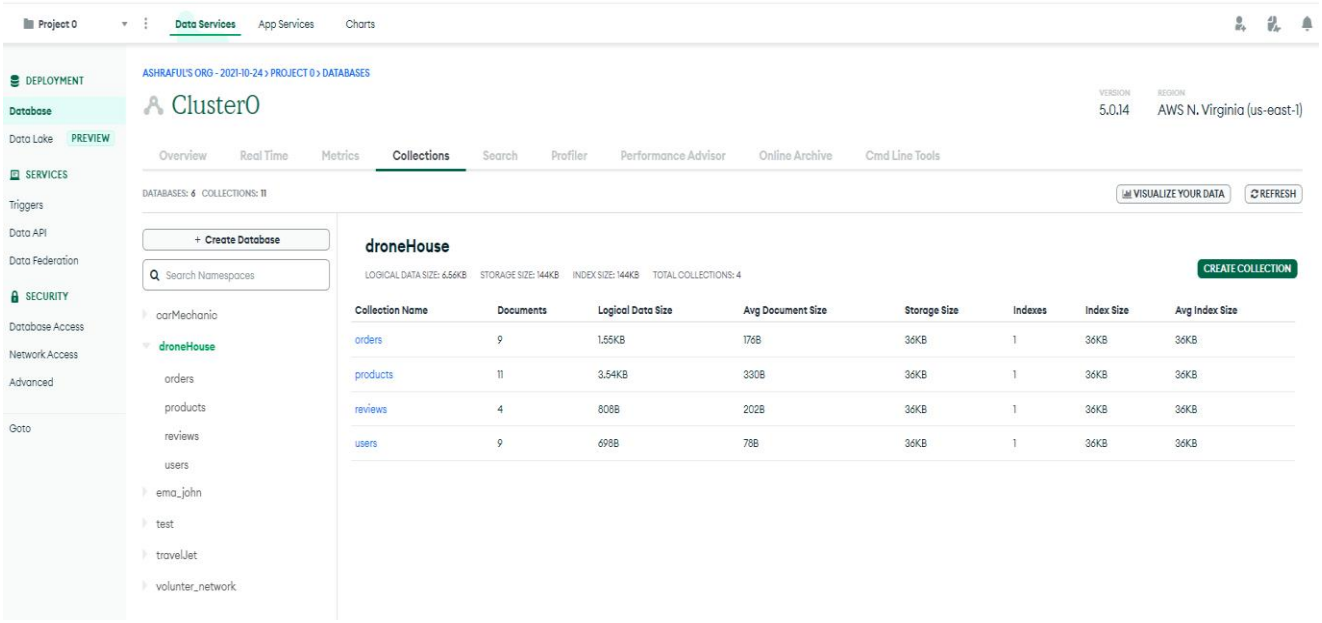


Figure 12. Model data of course

Figure 12 illustrates the model class for the course. The schema is designed using mon goose and it is exported for creating REST APIs.

REST APIs: The routing mechanism in Express helps to create REST APIs for the application. Gen erally, in a MEAN application, REST APIs are built to feed the data to the client using Node.js, Express, and MongoDB.

```
app.get('/api/v1/products', function(req, res){
 products.getProducts(function(err, courses){
                                    if(err){
                                    throw err;
                                    }
                                    res.json(product);
                                    });
                                    });

                                   app.get('/api/v1/products/:_id',          function(req,
                                   res){          product.getProductById(req.params._id,
                                   function(err,  products){
                                    if(err){


                                    throw err;
                                    }
                                    res.json(products);
```

```
});
});

app.post('/api/v1/ products ', function(req, res){
var courses=req.body;
products.addproducts
(products,function(err,products){  if(err){
throw err;
}
res.json(products);
});
});

app.put('/api/v1/products/:_id',     function(req,
res){  var id=req.params._id;
var products =req.body;
products.updateProducts
(id,products,{},
function(err, products){
if(err){
throw err;
}
res.json(products);
});
});


app.delete('/api/v1/  products  /:_id',  function(req,
res){  var id=req.params._id;
product.removeProduct
(id,function(err, products){  if(err){
throw err;
}
res.json(products);
}); });
```

Listing 4. REST APIs for course: Listing 4 illustrates the code snippet for creating REST APIs for the courses. The HTTP methods GET, POST, PUT, DELETE were used to perform the various operations at the endpoints. The URI follows the path root-api/api/v1/ representing the version one of the REST APIs. When a user provides the route (URI) such as http://lo calhost:3000/api/v1/courses, the getCourses() function is invoked which will return all the courses in JSON if there is no error.

Public directory (Client-side scripting): The public directory contains all the files and scripts needed to provide the routes (URIs) and render HTML pages. The ngRoute module routes the Angular application to differ ent pages without page reloads to make a Single Page Application. The routes are de fined in the public/app.js file.

Routes in ReactJS: It illustrates the routing mechanism in AngularJS. The routes refer to the pages to perform CRUD operations. The first line of code adds ngRoute as a dependency in the application module and then uses $routeProvider to configure the routes. The contents provided by the routes are put inside a ng-view directive in index.html. The ng-view directive then acts as a place holder for all the available views, thus creating a Single Page Application. When the user provides one of the routes, the related controller is invoked and the view is rendered inside the ng-view directive of the index.html with out page reloads.

Controllers in ReactrJS: The ReactJS controller module is used to consume the REST APIs when the user provides the appropriate routes (URIs) and shows the results to the user.

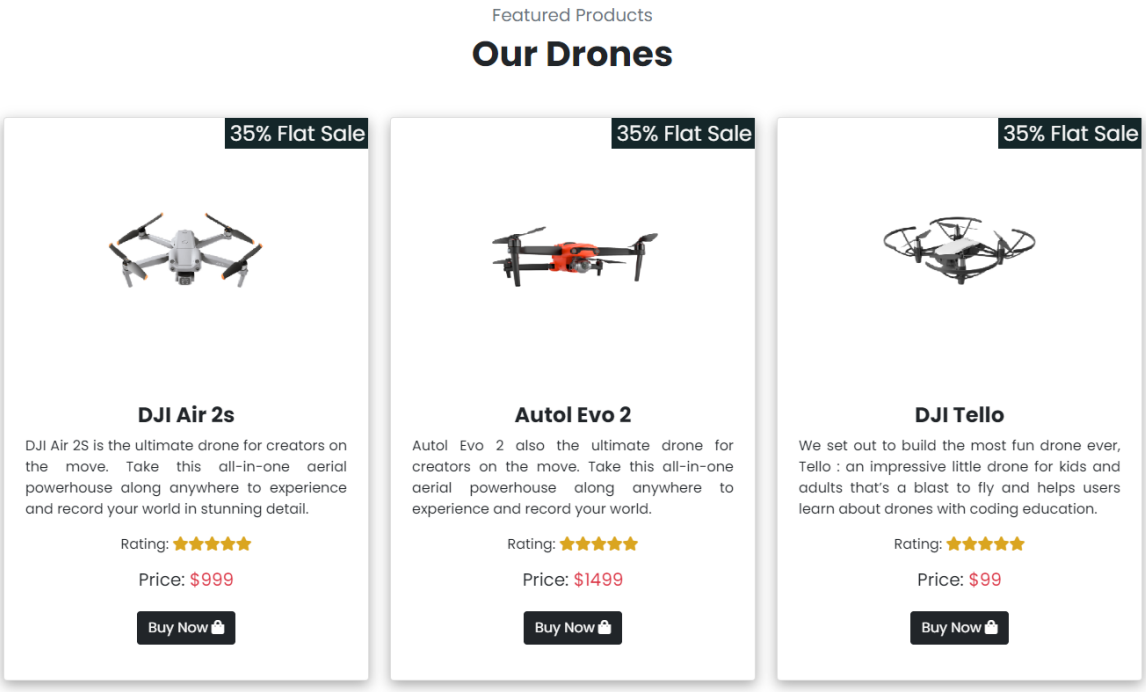


Figure 13. Drones.js controller

Figure 13 illustrates a code snippet from the Course.js controller class. The code snippet illustrates the GET request method for all courses. The controller module uses the $http.get method to consume the REST service at the given URI (/api/v1/courses). It assigns the JSON returned back from the API to the $scope.courses, which sets a model

object named courses. AngularJS then binds courses object to the applica tion's DOM rendering it on the user's browser.

Results and Discussion: The implementation of the MERN stack in real-life web application development was
carried out successfully. The prototype application followed all the best practices of developing a MERN application. A RESTful API was created on the server-side using

Node.js, Express, and MongoDB, and a Single Page Application was developed on the client-side using React.js. Moreover, the application is responsive and fits well on all screen sizes
RESTful APIs:
To illustrates the RESTful APIs built using the Node.js, Express, and MongoDB.
Given the URI 'localhost:3000/api/v1/courses', the GET method renders all the courses' information on the browser.

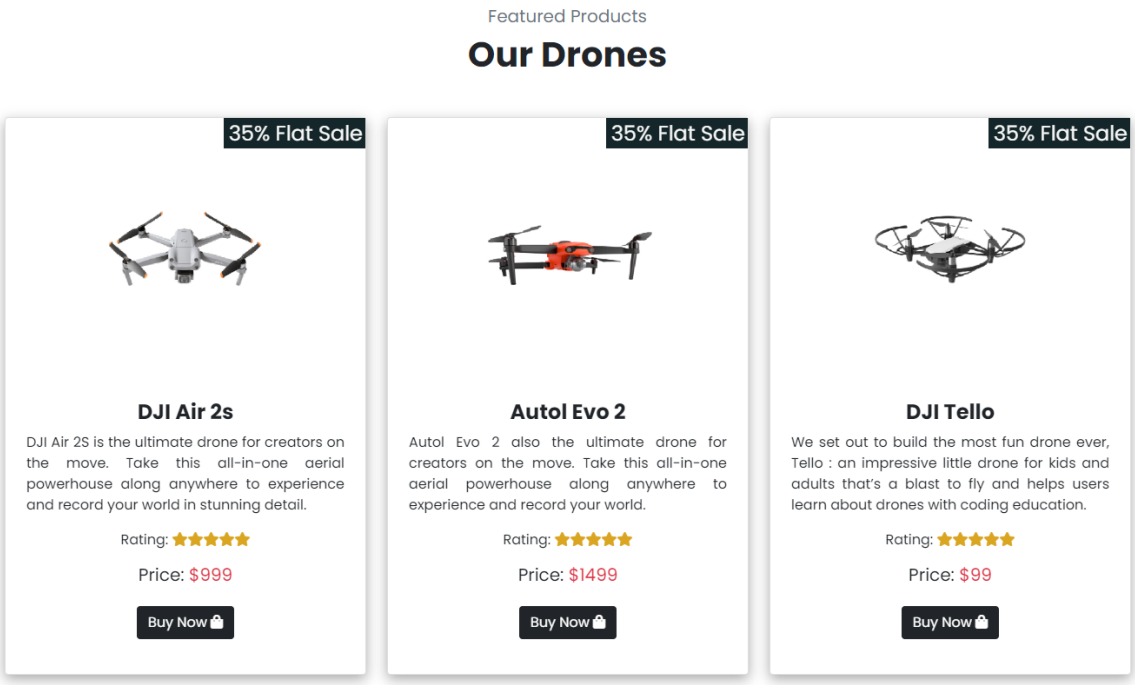


Figure 14. Drone page
Figure 14 illustrates the course page of the application. The application has a menu for navigating to different pages but all the views are rendered inside the ng-view directive in the index.html file. The URL contains # (hash) sign, which indicates the SPA's routing mechanism. The application downloads all the resources necessary in the first request and keeps updating the views later.

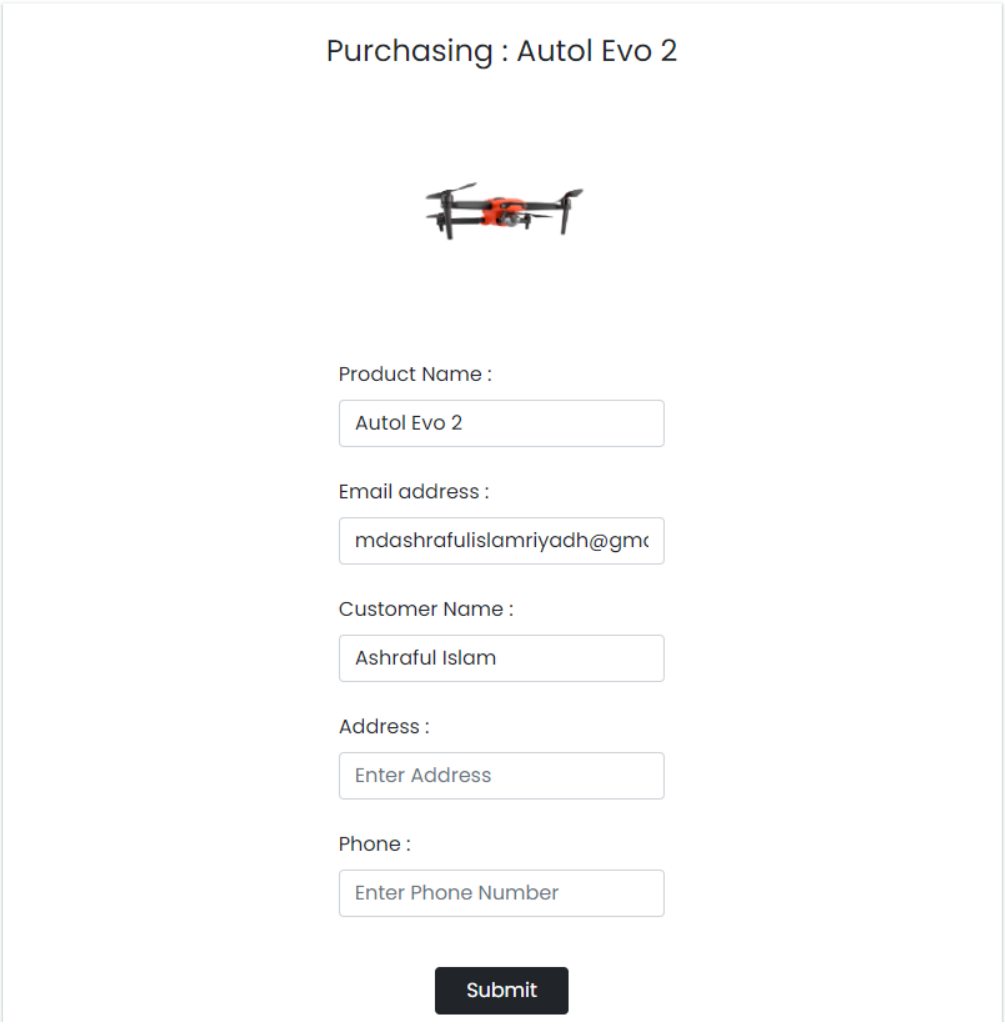


Figure: 15

Figure 15. Detail page of a drone to buy

Figure 15 illustrates the details of a course having the id shown in the URI. A user can browse the courses in the course page and click the link to go to the details. However, the page itself is not reloaded, the application will request the certain courses details from the REST API and JSON data sent back from the API is updated in the page. Using this interface, users can view the video and request for the full video playlist.

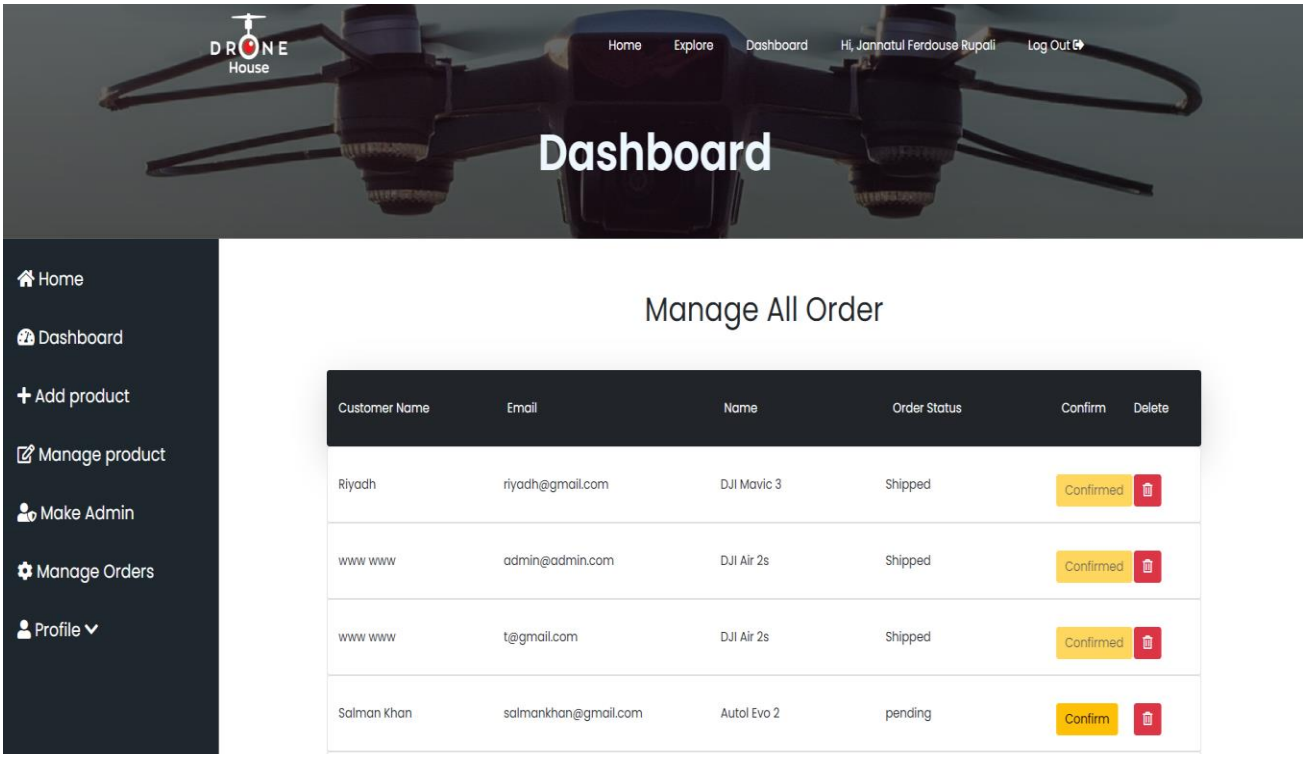


Figure 16. Admin panel

Figure 16 illustrates a simple admin panel to perform the CRUD operations. The admin can manage the resources of the web application from this page.

The application was designed using Bootstrap's grid layout so the application follows the best responsive design principles.

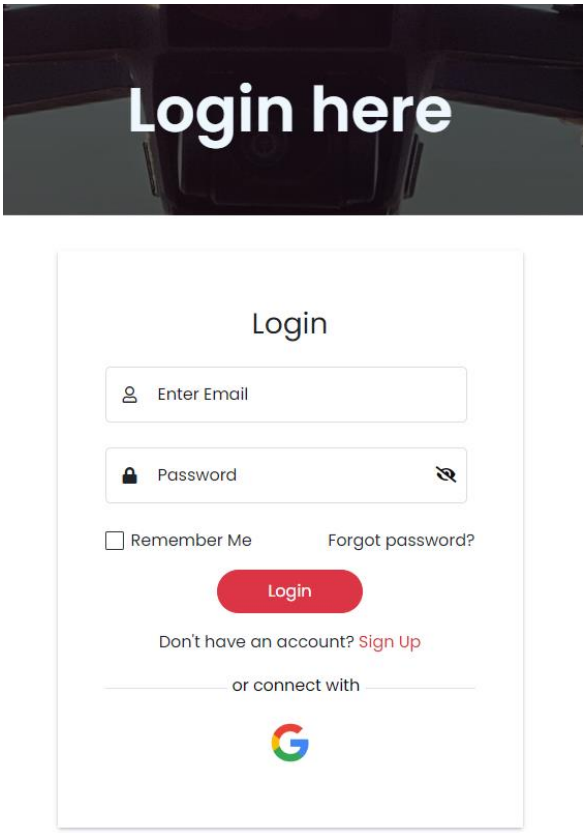


Figure 17. Responsive design

Figure 17 illustrates the responsive design of the application. Since the application was developed in a local environment, it was tested on a browser by minifying the width of the browser.

MERN stack indeed proved itself as one of the best solutions to bring all the technologies together for modern web application development. The MEAN approach of creating RESTful APIs on the server-side and a Single Page Application on the client-side clearly separates the client and server implementation. Since the client does not communicate directly to the server, future changes on the client-side application will not affect the server. Similarly, hybrid applications and native applications can be developed and they can run perfectly having access to the RESTful APIs.

In spite of all the benefits, a few problems were faced during the application development. The first problem was storing the data in MongoDB. Since MongoDB uses a non-relational database model, it was difficult to keep a relation between the data and perform related queries. In some cases, documents were embedded inside another document and in other cases, the references to other documents were included by using an array of referring object ids. These practices can lead to data repetition and unbounded arrays,

which might cause a great problem in large enterprise applications with huge amounts of related data.

The second problem was with the initial loading time of the SPA. Since the SPA downloads all the resources at the first load, it took quite a long time for the application to load at first. The loading time might be longer if the web application uses several heavy graphics and multimedia. Although the development of REST APIs is good in long-term development, it is still an additional task while trying to develop a small web application within a short time. Instead, the client can directly access the data from the database.

REST APIs should be implemented in large-scale web applications to minimize the coupling between the client and server, and to serve different types of clients (Web, Android, iOS, Hybrid). This loose coupling allows developers to update the server and client-side components independently in the future without affecting each other.

Since the application was developed to show the implementation of each component of the MERN stack and was created from scratch, it took a good deal of time just to set up the working environment and directories. The application development process can be accelerated by using different seed projects such as mean.io and mean.js, which provide ready-to-use boilerplate codes and authentication. Similarly, scaffolding tools and code generators like yeoman can be used to simplify the development process.

CHAPTER 4

Javascript Funtionalities & Application

4.1 Introduction & Functionalities:

4.1.4 Basic Function, Method, Features & Versions:

JavaScript is a cross-platform, object-oriented scripting language used to make webpages interactive (e.g., having complex animations, clickable buttons, popup menus, etc.). There are also more advanced server side versions of JavaScript such as Node.js, which allow you to add more functionality to a website than downloading files (such as realtime collaboration between multiple computers). Inside a host environment (for example, a web browser), JavaScript can be connected to the objects of its environment to provide programmatic control over them. JavaScript contains a standard library of objects, such as Array, Date, and Math, and a core set of language elements such as operators, control structures, and statements. Core JavaScript can be extended for a variety of purposes by supplementing it with additional objects; for example: Client-side JavaScript extends the core language by supplying objects to control a browser and its Document Object Model (DOM). For example, client-side extensions allow an application to place elements on an HTML form and respond to user events such as mouse clicks, form input, and page navigation. Server-side JavaScript extends the core language by supplying objects relevant to running JavaScript on a server. For example, server-side extensions allow an application to communicate with a database, provide continuity of information from one invocation to another of the application, or perform file manipulations on a server. This means that in the browser, JavaScript can change the way the webpage (DOM) looks. And, likewise, Node.js JavaScript on the server can respond to custom requests sent by code executed in the browser.

JavaScript and Java

JavaScript and Java are similar in some ways but fundamentally different in some others. The JavaScript language resembles Java but does not have Java's static typing and strong type checking. JavaScript follows most Java expression syntax, naming conventions and basic control-flow constructs which was the reason why it was renamed from LiveScript to JavaScript.

Functions are one of the fundamental building blocks in JavaScript. A function in JavaScript is similar to a procedure—a set of statements that performs a task or calculates a value, but for a procedure to qualify as a function, it should take some input and return an output where there is some obvious relationship between the input and the output. To use a function, you must define it somewhere in the scope from which you wish to call it.

See also the exhaustive reference chapter about JavaScript functions to get to know the details.

Defining functions

Function declarations

A function definition (also called a function declaration, or function statement) consists of the function keyword, followed by:

The name of the function.

A list of parameters to the function, enclosed in parentheses and separated by commas.

The JavaScript statements that define the function, enclosed in curly brackets, { /* … */ }.

For example, the following code defines a simple function named square:

```
function square(number) {
  return number * number;
}
```

The function square takes one parameter, called number. The function consists of one statement that says to return the parameter of the function (that is, number) multiplied by itself. The statement return specifies the value returned by the function:

```
return number * number;
```

Features:

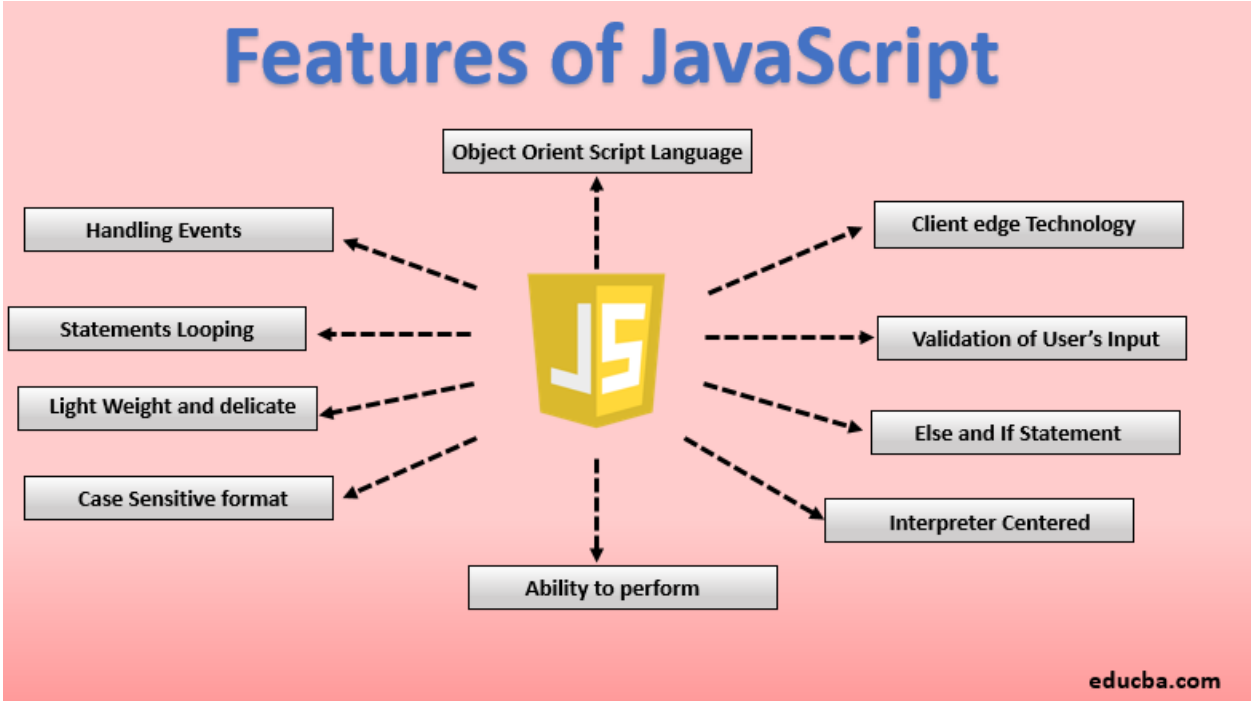


Figure: 18

JavaScript is mainly a client edge script language. This object-centered script language most commonly used for designing web pages which are a standalone language developed in Netscape. It gives the user extra control over the browser with potential in the creation of new functions in scripts. This scripting language features case sensitive input with the detection of the user's browser and operating system. JavaScript is mostly used by validation in client edge technology. The following are features of JavaScript:

1. Object-Centered Script Language
2. Client edge Technology
3. Validation of User's Input
4. Else and If Statement
5. Interpreter Centered
6. Ability to perform In Built Function
7. Case Sensitive format
8. Light Weight and delicate
9. Statements Looping

Version:
ES6 stands for ECMAScript 6.

ECMAScript was created to standardize JavaScript, and ES6 is the 6th version of ECMAScript, it was published in 2015, and is also known as ECMAScript 2015. [21]

Object-oriented Programming:

Object-oriented Programming treats data as a crucial element in program development and doesn't allow it to flow freely around the system. It ties data more securely to the function that operates on it and protects it from accidental modification from an outside function. OOP breaks down a problem into several entities called objects and builds data and functions around these objects.

4.1.2 OOP Concepts in JavaScript:

Objects

Objects are the basic run-time bodies in an object-oriented framework. They may represent a place, a person, an account, a table of data, or anything that the program needs to handle. Objects can also represent user-defined data such as vectors, time, and lists. Consider two objects, “customer” and “account” in a program. The customer object may send a message requesting the bank balance.

Classes

We know that objects hold the data and the functions to manipulate the data. However, the two can be bound together in a user-defined data type with the help of classes. Any number of objects can be created in a class. Each object is associated with the data of type class. A class is therefore a collection of objects of similar types.

For example, consider the class “Fruits”. We can create multiple objects for this class -

Fruit Mango;

This will create an object mango belonging to the class fruit.

Encapsulation

Encapsulation is the wrapping up/binding of data and function into a single unit called class. Data encapsulation is the most prominent feature of a class wherein the data is not accessible to the outside world, and only those functions wrapped inside the class can access it. These functions serve as the interface between the object’s data and the program.

Inheritance

The phenomenon where objects of one class acquire the properties of objects of another class is called Inheritance. It supports the concept of hierarchical classification. Consider the object “car” that falls in the class “Vehicles” and “Light Weight Vehicles”.

In OOP, the concept of inheritance ensures reusability. This means that additional features can be added to an existing class without modifying it. This is made possible by deriving a new class from the existing one.

4.2 Application:

4.2.1 Javascript in Web Application:

Web application development is the combination of the front-end and the back-end development. Front-end web development, also known as client-side development, involves the practice of creating Graphical User Interface (GUI) for clients (users) so that the users can interact with the application. It involves the use of primary web technologies and tools such as HTML, CSS, and JavaScript. [2.]

HTML is a mark-up language which provides the structure to a web page. It defines how a web page would look like so it can be considered the skeleton of any web application. CSS, on the other hand, is a style sheet language which provides style and visual enhancements to the documents written in HTML. JavaScript is the most advanced language among these technologies. It performs HTML DOM (Document Object Model) manipulation to provide a dynamic interface to users. Moreover, it provides an interactive interface to the users by creating pop-up messages, validating form inputs, and changing the layout based on events like user-input or mouse clicks. All these technologies are controlled by the browser to provide a front-end web interface. [3.]

Back-end web development, also known as server-side development, involves the development of computer programs and databases to serve the client. A web application in its primary days did not need to have a front-end but a functioning server-side application was enough for it to be considered a web application. Several changes have been made in this field since then. Today's sophisticated web applications cannot run without both the front-end and back-end services. Back-end technologies usually consist of the programming languages such as PHP, Ruby, Python, Java, Node.js, and different frameworks. Web application development encompasses all the web technologies related to the frontend and the back-end. In addition, a web application must ensure communication between the client and the server with the use of different communication protocols. Protocols are a set of rules for exchanging messages in a communication network. Protocols vary with different tasks and layers. HTTP (HyperText Transfer Protocol), TCP/IP (Transmission Control Protocol/ Internet Protocol), FTP (File Transfer Protocol), SMTP (Simple 3 Mail Transfer Protocol), SOAP (Simple Object Access Protocol) and REST (Representational State Transfer) are some common examples of the protocols used in web applications. [4.]

A web application, in its most elementary form, sends an HTTP request to a server to establish connection, and the server sends an HTTP response to the client.

CHAPTER 5

Node JS Backend Platform

5.1 Introduction:

Node.js is a JavaScript runtime used to create scalable server-side and networking applications via **virtual private servers**. It offers non-blocking input/output (I/O) operations and is built on event-driven, asynchronous architecture to help developers create various projects efficiently and easily.

We use Node JS as backend platform of our project in MERN Development. In Chapter 3, we see that how we implement Node JS in our project. In this chapter, we disscused how Node JS use in different platform, principle method and its feature.

Compared to other platforms, Node.js has a particular workflow. It operates as a single process, which means it doesn't create a new thread for every request. A thread is a set of instructions that need to be performed by the server.

Node.js employs non-blocking I/O operations – when a client sends a request to the web server, the single-threaded event loop picks it up and sends it to a worker thread for processing.

Instead of blocking the thread and wasting CPU resources by waiting for a response, Node.js will continue working on the next task. This way, it can handle a massive amount of concurrent requests.

That said, Node.js is not suitable for CPU-intensive tasks as they could prevent the main thread from handling other requests, effectively blocking it.

5.2 Principle Method in Project:

As previously discussed, the Node.js thread doesn't wait for a response and moves on to process a subsequent request. On the asynchronous architecture, the event loop operates in an event-driven manner. Once it has received a response from the previous API call, it puts the reply in the event queue.

The event loop will finish all prior and current requests before executing a callback function to send the server's response to the client.

Since Node.js uses a single-threaded event loop, it can deal with multiple requests simultaneously with faster execution times, and less resource usage.

In comparison, synchronous architecture software performs one task at a time. Therefore, the event loop will only move to the next one if the previous task is finished.

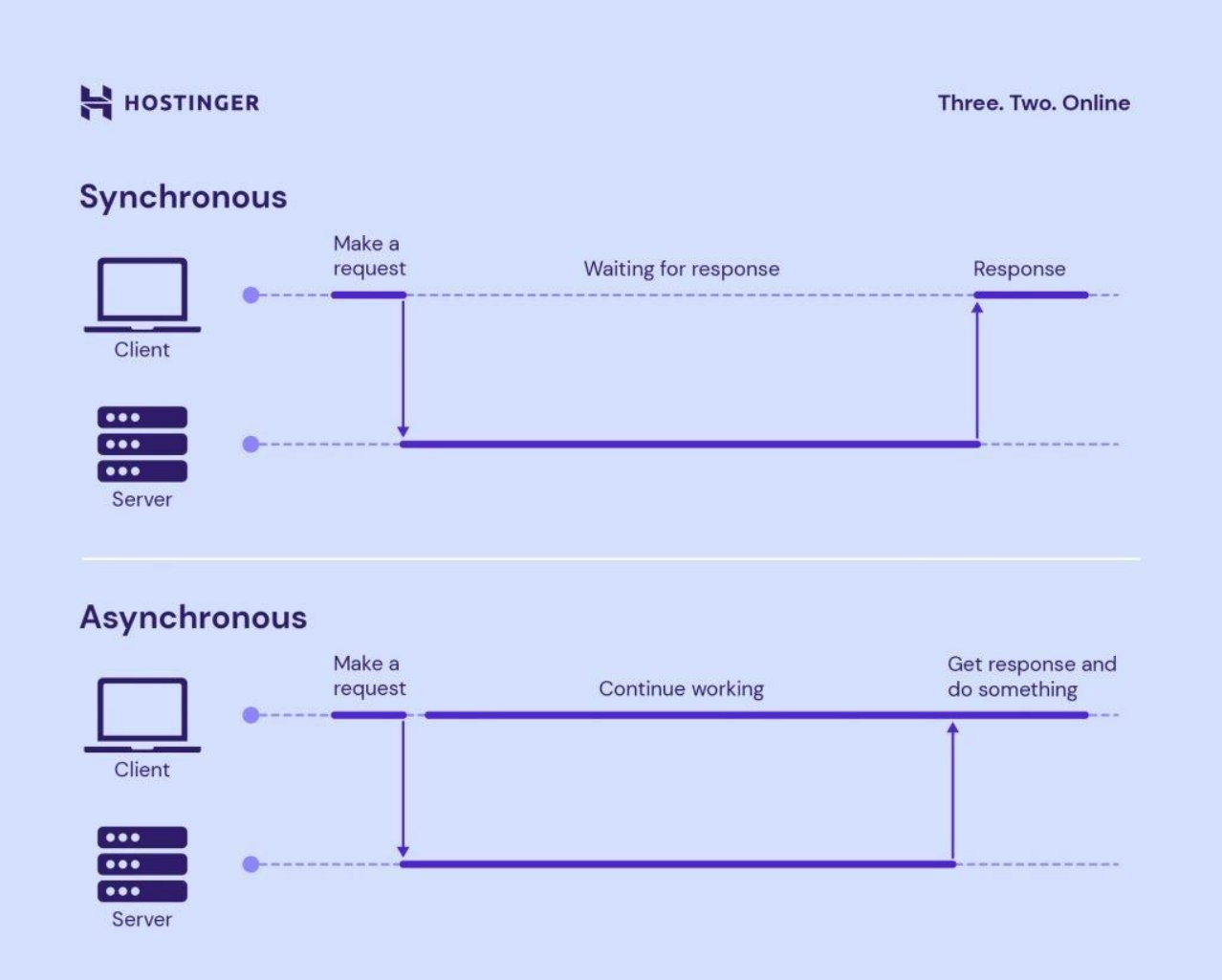


Figure:19

Node.js Use Cases & Feature:
Node.js is a good solution to perform data-intensive tasks or real-time analytics since it has asynchronous architecture and non-blocking I/O features. Some popular use cases include:

- **Real-time chatting.** Node.js can work with data-intensive programs like chat apps as it handles I/O tasks effectively. It uses push technology over web sockets, allowing two-way communication between servers and clients. As a result, the server doesn't need to maintain separate threads for each open connection due to Node.js' single-threaded asynchronous architecture.
- **Data streaming.** Node.js has built-in modules that support data streaming, which allows the creation of readable and writable streams. Node.js will help queue data and distribute it asynchronously without blockages or interruptions. It is a good option for streaming features that can process files while they're being uploaded.

- **Server-side proxies.** Node.js can manage a massive amount of concurrent connections using a non-blocking approach. It can be employed as an effective server-side proxy which collects data from various third-party resources. In some cases, Node.js is used to build client-side apps to manage assets and proxying/stubbing API requests.
- **System monitoring dashboards.** Using the event loop feature of Node.js, you can create a web-based dashboard to check the status of multiple services asynchronously. Both internal and public service statuses can be reported live and in real-time.
- **REST API.** Node.js features a number of packages such as Express.js and Koa.js which can be used to build applications. It can speed up the API integration process and be the basis for a light and fast REST API.
- **Single Page Applications (SPAs).** Entire SPAs are loaded on a single page for a desktop-app-like experience. As Node.js can handle asynchronous calls efficiently among heavy I/O operations, it allows SPAs to have seamless data updates without refreshing the page.

It's also important to note that Node.js supports both frontend and backend development. Here are the reasons why it works for both:

- **Reusable code.** Multiple components of Node.js can be reused for both the backend and frontend.
- **High efficiency.** Using Node.js can reduce context-switching between multiple programming languages.

Node.js vs npm:
While Node.js is a JavaScript runtime environment, the **Node Package Manager or npm** is a large part of the Node.js ecosystem.
It is a package manager that allows the JavaScript and Node.js communities to publish and share their node modules with other people. This makes the application development process faster and more efficient.
npm consists of two parts:

- A command-line interface (CLI) tool for downloading and publishing packages.
- An online repository that stores JavaScript and Node.js packages.

npm hosts millions of downloadable packages which are organized in multiple categories. It is considered to be the world's largest software registry.
By default, npm comes with every Node.js installation.

Node.js is an open-source, cross-platform JavaScript runtime environment used to develop scalable server-side and networking applications. It uses the Chrome V8 JavaScript execution engine.

CHAPTER 6

MONGODB DATABASE & FIREBASE

6.1 Introduction:

MongoDB is a document-oriented NoSQL database used for high-volume data storage. Instead of using tables and rows as in traditional relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs, which are the basic unit of data in MongoDB. Collections contain sets of documents and functions, which are the equivalent of relational database tables. MongoDB is a database that first appeared in the mid-2000s.

6.2 MongoDB Features:

1. Each database contains collections which in turn contains documents. Each document can be different with a varying number of fields. The size and content of each document can be different from each other.
2. The document structure is more in line with how developers construct their classes and objects in their respective programming languages. Developers will often say that their classes are not rows and columns but have a clear structure with key-value pairs.
3. The rows (or documents as called in MongoDB) doesn't need to have a schema defined beforehand. Instead, the fields can be created on the fly.
4. The data model available within MongoDB allows you to represent hierarchical relationships, to store arrays, and other more complex structures more easily.
5. Scalability – The MongoDB environments are very scalable. Companies across the world have defined clusters with some of them running 100+ nodes with around millions of documents within the database

6.3 Basic MongoDB Example:

The below example shows how a document can be modeled in MongoDB.

1. The _id field is added by MongoDB to uniquely identify the document in the collection.
2. What you can note is that the Order Data (OrderID, Product, and Quantity) which in RDBMS will normally be stored in a separate table, while in MongoDB it is actually stored as an embedded document in the collection itself. This is one of the key differences in how data is modeled in MongoDB.

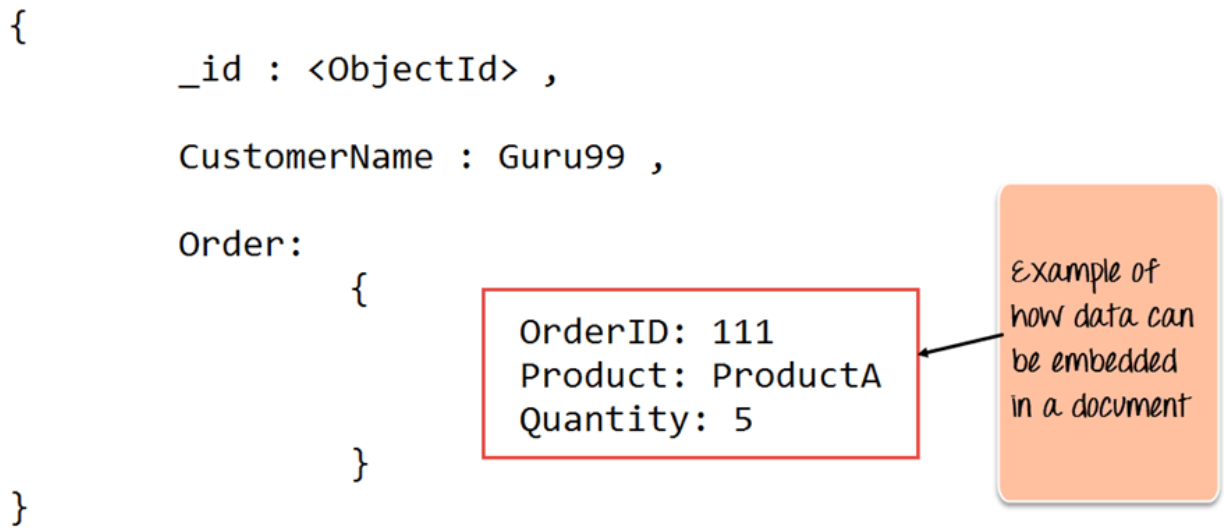


Figure: 20 (Basic MongoDB Example)

6.4 Key Components of MongoDB Architecture:

Below are a few of the common terms used in MongoDB

1. **_id** – This is a field required in every MongoDB document. The _id field represents a unique value in the MongoDB document. The _id field is like the document's primary key. If you create a new document without an _id field, MongoDB will automatically create the field. So for example, if we see the example of the above customer table, Mongo DB will add a 24 digit unique identifier to each document in the collection.

_Id	CustomerID	CustomerName
563479cc8a8a4246bd27d784	11	Guru99
563479cc7a8a4246bd47d784	22	Trevor Smith
563479cc9a8a4246bd57d784	33	Nicole

Table 3

2. **Collection** – This is a grouping of MongoDB documents. A collection is the equivalent of a table which is created in any other RDMS such as Oracle or MS SQL. A collection exists within a single database. As seen from the introduction collections don't enforce any sort of structure.
3. **Cursor** – This is a pointer to the result set of a query. Clients can iterate through a cursor to retrieve results.

4. **Database** – This is a container for collections like in RDMS wherein it is a container for tables. Each database gets its own set of files on the file system. A MongoDB server can store multiple databases.
5. **Document** – A record in a MongoDB collection is basically called a document. The document, in turn, will consist of field name and values.
6. **Field** – A name-value pair in a document. A document has zero or more fields. Fields are analogous to columns in relational databases.The following diagram shows an example of Fields with Key value pairs. So in the example below CustomerID and 11 is one of the key value pair's defined in the document

7. **JSON** – This is known as JavaScript Object Notation. This is a human-readable, plain text format for expressing structured data. JSON is currently supported in many programming languages.

Just a quick note on the key difference between the _id field and a normal collection field. The _id field is used to uniquely identify the documents in a collection and is automatically added by MongoDB when the collection is created.

6.5 Data Modelling in MongoDB:

As we have seen from the Introduction section, the data in MongoDB has a flexible schema. Unlike in SQL databases, where you must have a table's schema declared before inserting data, MongoDB's collections do not enforce document structure. This sort of flexibility is what makes MongoDB so powerful.

When modeling data in Mongo, keep the following things in mind

1. What are the needs of the application – Look at the business needs of the application and see what data and the type of data needed for the application. Based on this, ensure that the structure of the document is decided accordingly.
2. What are data retrieval patterns – If you foresee a heavy query usage then consider the use of indexes in your data model to improve the efficiency of queries.
3. Are frequent inserts, updates and removals happening in the database? Reconsider the use of indexes or incorporate sharding if required in your data modeling design to improve the efficiency of your overall MongoDB environment.

Difference between MongoDB & RDBMS
Below are some of the key term differences between MongoDB and RDBMS

RDBMS	MongoDB	Difference
Table	Collection	In RDBMS, the table contains the columns and rows which are used to store the data whereas, in MongoDB, this same structure is known as a collection. The collection contains documents which in turn contains Fields, which in turn are key-value pairs.
Row	Document	In RDBMS, the row represents a single, implicitly structured data item in a table. In MongoDB, the data is stored in documents.
Column	Field	In RDBMS, the column denotes a set of data values. These in MongoDB are known as Fields.
Joins	Embedded documents	In RDBMS, data is sometimes spread across various tables and in order to show a complete view of all data, a join is sometimes formed across tables to get the data. In MongoDB, the data is normally stored in a single collection, but separated by using Embedded documents. So there is no concept of joins in MongoDB.

Figure: 21

Apart from the terms differences, a few other differences are shown below

1. Relational databases are known for enforcing data integrity. This is not an explicit requirement in MongoDB.
2. RDBMS requires that data be normalize first so that it can prevent orphan records and duplicates Normalizing data then has the requirement of more tables, which will then result in more table joins, thus requiring more keys and indexes.As databases start to grow, performance can start becoming an issue. Again this is not an explicit requirement in MongoDB. MongoDB is flexible and does not need the data to be normalized first.

6.6 Applications of MongoDB in our project:

Below are the few of the application as to why one should start using MongoDB

1. Document-oriented – Since MongoDB is a NoSQL type database, instead of having data in a relational type format, it stores the data in documents. This makes MongoDB very flexible and adaptable to real business world situation and requirements.
2. Ad hoc queries – MongoDB supports searching by field, range queries, and regular expression searches. Queries can be made to return specific fields within documents.

3. Indexing – Indexes can be created to improve the performance of searches within MongoDB. Any field in a MongoDB document can be indexed.
4. Replication – MongoDB can provide high availability with replica sets. A replica set consists of two or more mongo DB instances. Each replica set member may act in the role of the primary or secondary replica at any time. The primary replica is the main server which interacts with the client and performs all the read/write operations. The Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically switches over to the secondary and then it becomes the primary server.
5. Load balancing – MongoDB uses the concept of sharding to scale horizontally by splitting data across multiple MongoDB instances. MongoDB can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure.

6.7 Firebase Authentication & Features:

Google Firebase is a Google-backed application development software that enables developers to develop iOS, Android and Web apps. Firebase provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment.

Firebase use cases include:

- Create onboarding flows – developers can give users a quick, intuitive sign-in process using Firebase Authentication. They allow users to sign into their apps via their Google, Twitter, Facebook or GitHub accounts in less than five minutes. Developers can also track each step of their onboarding flows to enhance the user experience. Additionally, developers can use Google Analytics for Firebase to log events at each step of their onboarding flows, create funnels to determine where users are dropping off and use remote configuration to make changes to their apps to see how those changes affect conversions.
- Customize a "welcome back" screen – developers can use personalization to give every user the best experience by customizing the initial screen based on a user's preferences, usage history, location or language. Developers can define audiences based, in part, on user behaviors and show targeted content to each audience.
- Progressively roll out new features – developers can launch new features with minimal risk by first testing those features on a few users to see how they work and how users respond. Then, when developers are satisfied, they can roll out their apps to the rest of their users.
-

Firebase authentication:

Firebase authentication is a feature of user authentication that is provided by **Firebase** as its backend services. This is a token-based authentication system that provides easy integration with most of the platforms.

The best part is that it is a role-based system, which means you can create different roles for different users. Another great feature here is that this authentication is effortless to integrate with the API from Firebase, making this feature up and running easily in no time.

Firebase authentication advantages

Firebase authentication service comes with a lot of different advantages. Here are some of them.

- **Save time on developing web services methods for authentication**
  When you are building any web application, a mobile application, or a computer application for the system's security, you have to save the login credentials and other authentication means.
  This takes a lot of time because you have to implement different services for this. However, when you use Firebase authentication, you will not have to do a lot of coding. The information that needs to be stored will be stored in Firebase's system, and Firebase will efficiently handle every type of authentication service.

- **Analytics**
  One of the most important things in an application's development and future is how it is being improved. One of the best ways to improve the application is by making it better for the users in terms of user experience and features. Firebase authentication is also an exceptional platform in this case.
  It is because it provides detailed analytics about the users' information and interaction on your platform. This demographic information and analytics of users make it extremely efficient for improvement purposes.

Firebase authentication core features

Although this is a feature of Firebase backend services with a very narrow field of scope, it still comes with many core features. Here we will discuss all of these core features in detail.

- **Drop-in authentication solution**
  This is one of the best ways of adding an authentication feature to your system. This consists of a complete sign-in system for your application. The system uses email addresses and passwords for sign-in, and it works for all mobile and web apps.
  One of the best things here is that this feature is open-source. You can add this feature to your app using the authentication SDK from Firebase.
- **Federated identity provider integration.**
  Apart from credential authentication, one of the best ways for authentication and the most popular authentication is by using some other federated identity providers. By using the Firebase authentication services, you can let the users sign-in using the following platforms.

1. Google
2. Facebook
3. Twitter
4. GitHub
5.

- **Phone number authentication**
  Another amazing feature of Firebase authentication is that the users can be authenticated using their phone numbers by sending an SMS. It's a good feature, and it works well according to the situation. Also we can use Google Account(Gmail) verification directly and can log in or sign up easily by Google Account with its information.

- **Custom authentication system integration**
  If you want to use your application's existing authentication system, you can integrate it with Firebase authentication services to use the real-time database services from Firebase.

CHAPTER 7

USE CASE & ER DIAGRAM

7.1 USE CASE:

A use case is a methodology used in system analysis to identify, clarify and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. The method creates a document that describes all the steps taken by a user to complete an activity.

Business analysts are typically responsible for writing use cases and they are employed during several stages of software development, such as planning system requirements, validating design, testing software and creating an outline for online help and user manuals. A use case document can help the development team identify and understand where errors may occur during a transaction so they can resolve them.

Use Case in our project in the below:

Use Case 1	
Use Case Name:	User signup
Actors:	User and Database
Description:	Signing up user account
Precondition:	To create an account, the user must provide all of his personal information.(Google account authentication is available & auto generate data)
Basic Flow:	If the user provides all of the information and creates an account, he may login at any moment using those data..
Alternative Flow:	A pop-up notice will appear, prompting the user to complete all fields.
Post Condition:	The username and password can be used to log in at any time.

Use Case 2

Use Case Name:	Admin Signup.(Google account authentication is available & auto generate data)
Actors:	Admin and Database
Description:	Signing up admin account
Precondition:	User must give all his personal info for creating an admin account
Basic Flow:	If the administrator provides all of the information and creates an account, he may log in at any time using those data.
Alternative Flow:	A pop-up message will appear, requesting the administrator to complete all of the fields.
Post Condition:	With the login and password, the administrator can log in at any time.

Use Case 3	
Use Case Name:	User Login
Actors:	User and database

Description:	Add a New Product
Precondition:	Admin must login with his details and choose product
Basic Flow:	Admin must log in with his credentials and choose products whice is uplode and added on database.
Alternative Flow:	If the user or admines enters incorrect information about themselves, an error notice will appear.
Post Condition:	product will be added to the product page & user can see this

Description:	Login with user account
Precondition:	The user must enter his login and password from the registration page.
Basic Flow:	If the user enters the right information, the system will grant him access to his portal.
Alternative Flow:	A pop-up message will appear instructing the user to input their credentials.
Post Condition:	The user will be able to access his account.

Use Case 5	
Use Case Name:	Admin Panel
Use Case 6	
Actors:	Admin and database
Description:	Login with Admin account
Precondition:	The administrator must provide the login and password that he generated on the register page.
Basic Flow:	The system will grant access to the Admin's portal if he logs in with the necessary credentials.
Alternative Flow:	A pop-up message will appear, instructing the administrator to reenter the credentials.
Post Condition:	Admin will be granted access to his account.

Table 4

7.2 ER Diagram:

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

The ER Diagram of Our Project is:

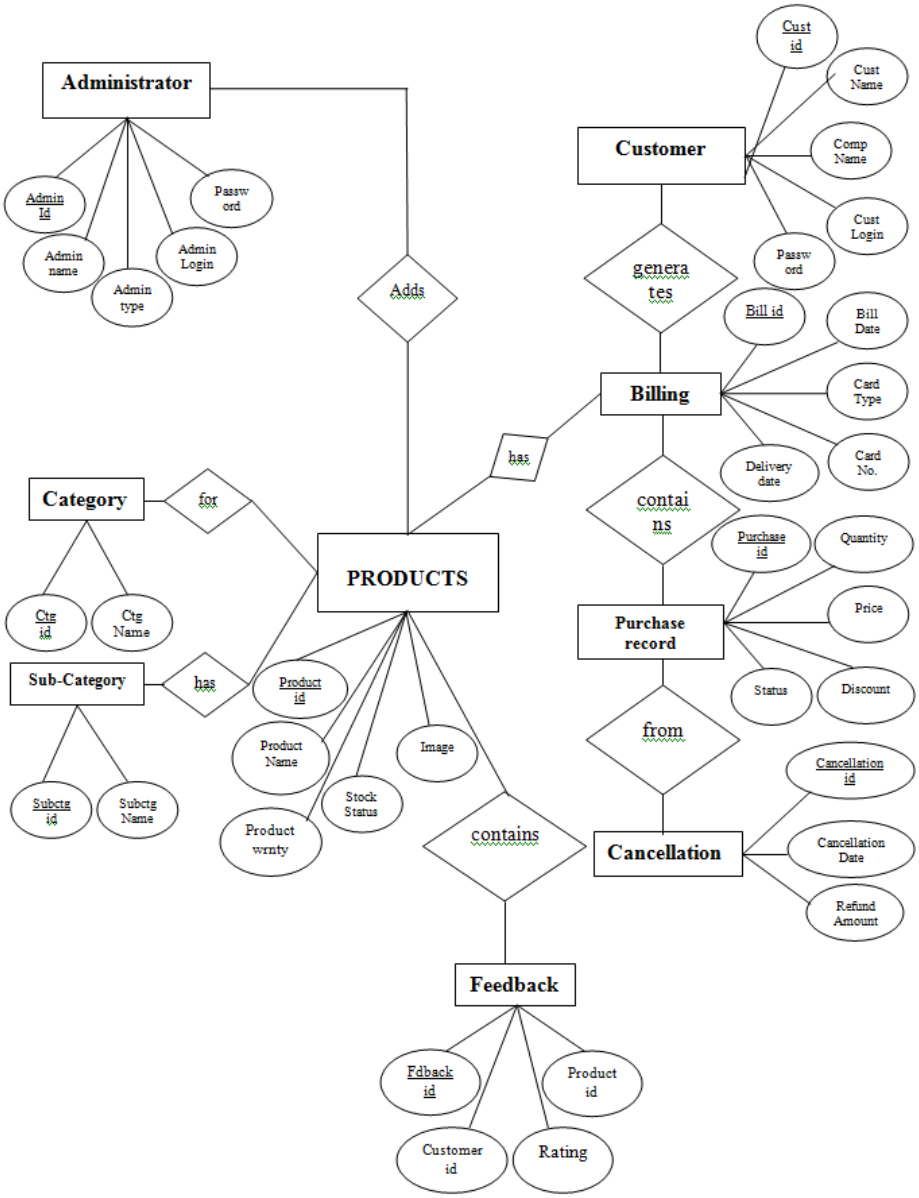


Figure 22

CHAPTER 8

CONCLUSION & FUTURE WORKS

8.1 Conclusion:

MERN Development is a very suitable platform nowadays, and its features are expanding on a day by day; it is also simple to use. With the MERN stack, we can ideally build any web application you want by learning just one language, Javascript. With the increased popularity of MongoDB that is a go-to database because of its scalability and flexible document schemas.

The goal of the project was to study the different components of MongoDB and MERN stack development. It is the most popular full-stack JavaScript framework, and we developed a web application based on it. It's such a great example of a MongoDB database-based website with MERN Development.We believe we are using it for the first time in our department. all the building blocks of the MERN stack: MongoDB, Express, React.js, and Node.js, were studied in great detail. The strengths of each component were analyzed and were also compared with those of the other technologies.

This website offers a digital equivalent of the shop manipulation method, which is advantageous to both users and store customers. The entire procedure is made available online, where people can browse and purchase different things. Regular users can log in using another option. the program that allows consumers to log in and view the status of their orders, as well as submit requests or ideas. Administrators have the option to log in and add other settings. item, analyze user behavior, user review, occasionally provide a discount, and also include details on other events for the customer.

8.2 Future aspect:

The eventual scope of the MERN Development will be extremely expansive. The project might eventually be made available on the web.

The following are the future scope items for the project:

- There is a lot of e-commerce out there that is still using old databases and technologies that are extremely slow to scale. This technology (MERN Stack Development and MongoDB) will definitely replace this and solve many kinds of problems.
- Gradually, every developer in the present day wants to build web applications utilizing these technologies since they can handle massive data stores, large feature systems, and flexibility.
- can be incorporated into the inventory management system in our site.
- Multiple branches may be added.
- can be added multilingually to this site.
- And many features can be added to this project to make it more robust.

- Add more different types of data in the different sections that can be used in the artificial intelligence or machine learning sectors.

REFERENCE

1 The MERN & MEAN Stack on OpenShift [online]. OpenShift developers' official website.
URL: https://developers.openshift.com/languages/nodejs/examplemeanstack.
Html

2 Dana Nourie. Java technologies for Web Applications [online]. Oracle official website
URL: http://www.oracle.com/technetwork/articles/java/webapps-1-138794.html

3 Hanin M. Abdullah, Ahmed M. Zeki. Frontend and Backend Web technologies in Social Networking Sites: Facebook as an Example. Advanced Computer Science Applications and Technologies (ACSAT), IEEE 3rd International Conference on 2014. p.85-89.

4 Web Applications: What are They? What of Them [online]. Acunetix official website
URL: http://www.acunetix.com/websitesecurity/web-applications/

5 Mimi Gentz. NoSQL vs SQL [online]. Microsoft Azure official website
URL: https://azure.microsoft.com/en-us/documentation/articles/documentdbnosql-vs-sql/

6 Rodriguez A. RESTful Web services: The basics [online]. IBM official website
URL: https://www.ibm.com/developerworks/library/ws-restful/

7 Nodejs.org official website [online]
URL: https://nodejs.org/en/

8 Erik Eloff, Daniel Torstensson. An Investigation into the Applicability of Node.js as a Platform for Web Service. Linköping: Institutionen för datavetenskap

9 Holmes S. Getting MEAN with Mongo, Express, Angular, and Node. Manning Publications

10 Benjamin Erb. Concurrent Programming for Scalable Web Architectures. Universität Ulm

11 MongoDB official website [online].
URL: https://docs.mongodb.com/manual/

12 Herron D. Node Web Development. 2nd ed. Packt Publiching Ltd

13 Kai Lei, Yining Ma, Zhi Tan. Performance Comparison and Evaluation of Web Development Technologies in PHP, Python and Node.js. Computational Science and Engineering (CSE), IEEE 17th International Conference on; 2014: p.661-668.

14 Express official website [online].
URL: https://expressjs.com

15 Ihrig CJ, Bretz A. Full Stack Development with MEAN. Cambridge, AUS: Site-Point Pty. Ltd

16 MongoDB documentation official website [online].
URL: https://docs.mongodb.com/getting-started/shell/introduction/

17 Györödi C, Györödi R, Pecherle G, Olah A. A comparative study:MongoDB vs. MySQL. Engineering of Modern Electric Systems (EMES), IEEE 13th International Conference on; 2015. p.1-6.

18 Chodorow K. MongoDB: The Definitive Guide. 2nd ed. Gravenstein Highway North, CA: O'Reilly Media Inc

19 React js Website
URL:https://codeinstitute.net/global/blog/what-is-react-js/#:~:text=js%3F-,React.,source%20community%20now%20maintain%20it
.
20 JAVASCRIPT Pro MERN Stack Full Stac Web App Development with Mongo, Express, React & Node

21 React.js Tutorial [online]. W3schools official website
URL: https://www.w3schools.com/REACT/default.asp

22 Single Page Application Tracking [online]. Google developers' official website
URL: https://developers.google.com/analytics/devguides/collection/analyticsjs/single-page-applications

23 Fernando Monteiro. Learning Single-Page Web Application Development. Packt Publishing

24 ASP.NET- Single-page Applications: Build Modern, responsive Web Apps with ASP:NET [online]. Microsoft developers' official website
URL: https://msdn.microsoft.com/en-us/magazine/dn463786.aspx

25 Mozilla developers' official website [online]
URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction

26 jQuery API [online]. JQuery official website. URL: http://api.jquery.com

27 What’s AJAX? [online]. Mozilla developers’ official website.
URL: https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started

28 Hernandez A. Init.js: A Guide to the Why and How of Full-Stack JavaScript
[online]. Toptal LLC developers’ official website. URL: https://www.toptal