JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

DESIGING WEB APPLICATION FRAMEWORK USING MERN STACK A COMPARATIVE REVIEW

PRADEEP SHARMA, Dr. VISHAL SHRIVASTAVA, Dr. AKHIL PANDEY, Er. ROBIL VARSHNEY
B.TECH. Scholar, Professor, Assistant Professor
Computer Science & Engineering
Arya College of Engineering & I.T. India, Jaipur

Abstract

Our purpose on this project is to increase a entire internet application framework that addresses the precise desires of the online barber enterprise. Based on the MERN stack, our device offers functions together with actual-time analytics of availability, ordering services, transparent pricing, analytics, and product descriptions geared toward providing suppliers improve the consumer experience Efficiency of our device in addressing the challenges faced by the barbershop quarter thru comparative gaining knowledge of and allow us to explore, we can ultimately introduce digital answers in this vicinity. This examine opens the door for destiny traits in internet utility development and presents insights into the ability for innovation in online barber applications.

Introduction

The service zone has gone through a exchange with the arrival of the digital age, and as a result, groups now want to streamline their operations and build a robust on line presence. The demand for online reserving, statistics get admission to, and seamless user studies has grown, and the conventional barber enterprise isn't any exception. This study sets out to create a web software framework this is particularly ideal to the requirements of the online barber offerings industry in reaction to this changing panorama.

This venture's essential intention is to create a reliable, consumer-pleasant, and powerful web software framework through utilizing the skills of the MERN stack, which consists of Express.Js, React.Js, MongoDB, and Node.Js. To create a complete on-line revel in for barbers and their clients, this framework will enable customers to effortlessly schedule appointments, check real-time availability, get entry to obvious pricing, examine evaluations, and discover product facts. Our purpose is to improve the convenience and ordinary experience of the usage of online barber offerings by means of integrating those crucial features into a single platform.

In addition, a radical comparative evaluation inspecting the framework's efficacy and overall performance towards other options might be supplied in this look at document. By assessing factors like protection, scalability, and consumer-friendliness, we hope to focus on the framework's advantages and shortcomings even as additionally supplying insightful evaluation to the online barber offerings area. This project highlights the enormous capacity for digitizing and enhancing conventional service sectors, while also creating a widespread contribution to the field of internet software improvement as a whole and performing as a catalyst for innovation within the barber offerings industry.

Methodology

The case looks at's methodology is a methodical approach especially designed for the advent and evaluation of the internet software software framework for on line barber offerings. The MERN stack, which consists of Express.Js for the server, React.Js for the front-surrender interface, MongoDB for database control, and Node.Js for server-component JavaScript execution, is the generation stack which have become determined on. These generation were determined on because they healthful the venture's goals and were adaptable and scalable.

A blended-strategies technique is used in the research layout, incorporating quantitative and qualitative information. Comprehensive qualitative research of the internet barber services market has become completed to pinpoint requirements

and issues. Using quantitative records series techniques together with surveys, person interactions, and usual performance indicators, the capability, usability, and overall performance of the framework have been all thoroughly assessed. Barbers and forestall customers every contributed to the facts collection gadget, ensuring a comprehensive evaluation of the software application's functionality and applicability to the intended audience.

A Comparative Review

In contrast to more general e-commerce and service booking frameworks, the research paper "Designing a Web Application Framework for Online Barber Services Using the MERN Stack" stands out for its specific focus on the online barber services sector. The paper successfully integrates MongoDB, Express.js, React, and Node.js using the MERN stack, producing a fluid and responsive web application. The framework places significant emphasis on aspects that are important to users, like the ability to schedule appointments, check availability in real-time, transparent pricing, reviews, and product descriptions. These features serve the needs of both clients and barbers, thereby improving the online barber services experience.

Furthermore, the framework's performance is evaluated against other solutions in the paper's comparative evaluation, which adds a great deal of value. This assessment covers aspects such as security, scalability, and user-friendliness, providing a thorough understanding of the advantages and shortcomings. The study project has the potential to stimulate innovation in the online barber services business in addition to meeting the industry's current needs. It highlights its dedication to remaining relevant and adaptable in the always changing digital ecosystem by outlining potential directions for further research and development in web application design for barbers. In conclusion, this research paper presents a customized web application framework that addresses a critical void in the barber services industry, emphasizing improving user experiences and providing a clear roadmap for further advancements.

Advantages of MERN STACK

JavaScript from End to End: JavaScript is used by the MERN stack to facilitate code reuse and streamline development processes for both front-end and back-end development.

Exceptionally Effective Development: MERN provides an abundant ecosystem of tools, modules, and libraries that enable quick development, simple maintenance, and smooth integration throughout the stack.

Complete-Stack Uniformity: By ensuring a unified and consistent development environment, MERN makes web application development, debugging, and scalability easier.

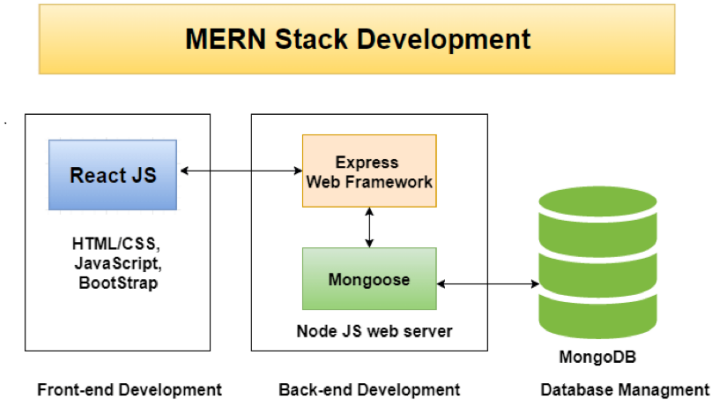


Fig: Work model of MERN

Case Study

A group of developers took on the task of creating a web application framework specifically suited to the demands of the expanding online barber services market. By utilizing the MERN stack—MongoDB, Express, React, and Node.js—the framework sought to transform the way customers interact with and access neighborhood barbers. A thorough examination of current online service frameworks, user surveys, and market research were all part of the research and development process. Clients now enjoy a smooth and improved experience thanks to the framework's successful implementation, which includes an effective and user-friendly booking system, real-time availability checks, transparent pricing, and a review platform. The case study highlights how technology-driven solutions can revolutionize established service sectors and highlights how flexible the MERN stack specializes in creating cutting-edge digital solutions to satisfy changing customer demands.

Challenges

1. Complex Data Management: There were challenges in storing, retrieving, and synchronizing data throughout the framework while managing a variety of data, such as client reviews, price, service specifics, and barber profiles.

2. Real-time Functionality: To ensure smooth and responsive user interactions, real-time features like availability checks and fast booking changes have to be implemented. This involved overcoming technical obstacles.

3. Security and Data Privacy: One of the main challenges in designing the framework to guard against potential breaches and data leaks was ensuring the security and privacy of sensitive user data, such as booking information and customer details.

Technologies in MERN

. MongoDB

Introduction to MongoDB

Data is being generated, evaluated, communicated, and then used to make business decisions at a far quicker speed compared to the past. To give an idea about the amount of data generated, IBM calculated in 2013 that over 90% of the data which was generated by the worlds was produced is the years 2011 and 2012. Generally, Relational Database Management Systems (RDBMS) have been used to handle vast volumes of data created by enterprise applications and to satisfy the storage requirements by various industries. They also allow for effective organized data manipulation. As a result, the concept of NoSQL was developed as an easier and more effective solution to handle massive data. NoSQL stands for "Not Only SQL," and it expands on the traditional method to data storage and retrieval. As the volume of data has increased exponentially and applications must handle millions of users simultaneously and process a huge volume of unstructured and complex data sets, a relational database model has serious limitations when it has to handle that huge volume of data. These limitations have led to the development of non-relational databases, also commonly known as NoSQL (Not Only SQL). This huge number of unstructured and complex data sets, typically indicated with the term Big Data, are characterized by a large volume, velocity, and variety, and cannot be managed efficiently by using relational databases, due to their static structure [6]. For this reason, software developers have also begun to consider NoSQL data storage solutions. In today's context of Big Data, the developments in NoSQL databases have achieved the right infrastructure which can very much be well-adapted to support the heavy demands of Big Data. NoSQL databases are extensively useful when they are needed to access and analyze huge amounts of unstructured data or data that are stored remotely on multiple virtual servers. A NoSQL database does not store information in the traditional relational format. NoSQL databases are not built on tables and, in some cases, they do not fully satisfy the properties of atomicity, consistency, isolation, and durability (ACID). A feature that is common to almost all NoSQL databases is that they handle individual items, which are identified by unique keys. Additionally, their structures are flexible, in the sense that schemas are often relaxed or free schemas. A classification that is based on different data models has been proposed in [6,8], it groups NoSQL databases into four major families, each based on a different data model: Key–value-stores databases (Redis, Riak, Amazon's DynamoDB, and Project Voldemort), column-oriented databases (HBase and Cassandra), document-based databases (MongoDB, CouchDB, and the document-based MySQL), and graph databases (Neo4j, OrientDB and Allegro Graph). From the several NoSQL databases that we have today, this paper focuses on document-based model databases,

choosing two well-known NoSQL databases, MongoDB and document-based MySQL, and analyzing their behavior in terms of the performance of CRUD operations. To perform performance analysis, a server application has been developed and presented in this paper. The application serves as a backend for streamlining the activity of small service providers, using the two document-based MongoDB and MySQL data-bases, with an emphasis on how to use query operations through which the CRUD operations are performed and tested, the analysis being performed on the response times of these for a data volume of up to 100,000 items. This article takes three papers into consideration and summarizes them. Each of the papers are a comparative study of MongoDB with other databases. Rows and columns which consist the table make up the relational databases. These columns in relational databases, contain the information for a given class and the rows in these databases, specify an instance of that data which is determined by the category. Examples of relational databases are MySQL, IBM, Microsoft Examples of non-relational databases are ArangoDB, CouchBase, MongoDB, Adabas. MySQL and Oracle are relational databases. Cassandra and MongoDB databases are non-relational databases.

WORKING OF MONGODB

MongoDB is an open-source document-oriented database. It is used to store a larger amount of data and also allows you to work with that data. MongoDB is not based on the table-like relational database structure but provides an altogether different mechanism for storage and retrieval of data, that's why known as NoSQL database. Here, the term 'NoSQL' means 'non-relational'. The format of storage is called BSON (similar to JSON format). Now, let's see how actually this MongoDB works? But before proceeding to its working, first, let's discuss some important parts of MongoDB – ● 1. Drivers: Drivers are present on your server that are used to communicate with MongoDB. The drivers support by the MongoDB are C, C++, C#, and .Net, Go, Java, Node.js, Perl, PHP, Python, Motor, Ruby, Scala, Swift, Mongoid. 2. MongoDB Shell: MongoDB Shell or mongo shell is an interactive JavaScript interface for MongoDB. It is used for queries, data updates, and it also performs administrative operations. 3. Storage Engine: It is an important part of MongoDB which is generally used to manage how data is stored in the memory and on the disk. MongoDB can have multiple search engines. You are allowed to use your own search engine and if you don't want to use your own search engine you can use the default search engine, known as WiredTiger Storage Engine which is an excellent storage engine, it efficiently works with your data like reading, writing, etc.

● MongoDB work in two layers –

1. Application Layer

2. Data layer

APPLICATION LAYER

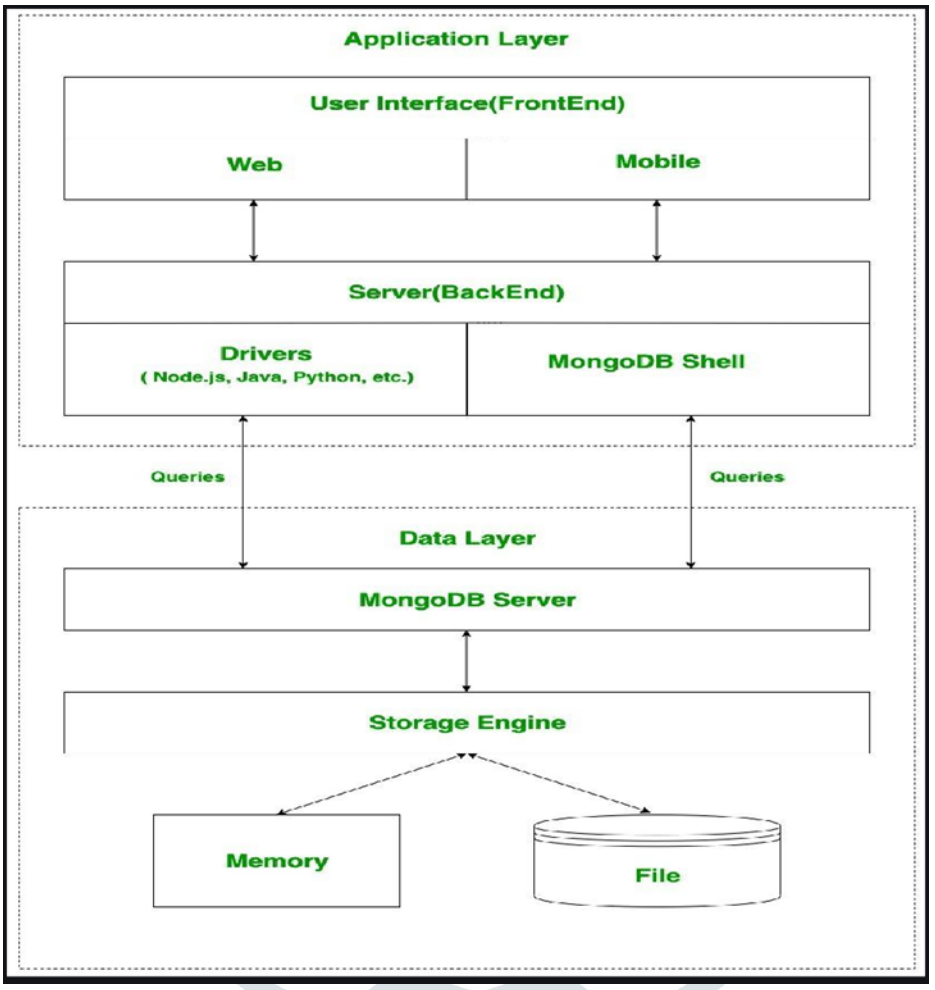
Application Layer is also known as the Final Abstraction Layer, it has two-parts, first is a Frontend (User Interface) and the second is Backend (server). The frontend is the place where the user uses MongoDB with the help of a Web or Mobile. This web and mobile include web pages, mobile applications, android default applications, IOS applications, etc. The backend contains a server which is used to perform server-side logic and also contain drivers or mongo shell to interact with MongoDB server with the help of queries.

DATA LAYER

These queries are sent to the MongoDB server present in the Data Layer. Now, the MongoDB server receives the queries and passes the received queries to the storage engine. MongoDB server itself does not directly read or write the data to the files or disk or memory. After passing

the received queries to the storage engine, the storage engine is responsible to read or write the data in the files or memory basically it manages the data.

The following image shows how the MongoDB works:



ADVANTAGES AND DISADVANTAGES OF MONGODB

Advantages of MongoDB

MongoDB offers several potential benefits:

- Schema-less. Like other NoSQL databases, MongoDB doesn't require predefined schemas. It stores any type of data. This gives users the flexibility to create any number of fields in a document, making it easier to scale MongoDB databases compared to relational databases.

- Document-oriented. One of the advantages of using documents is that these objects map to native data types in several programming languages., Having embedded

documents also reduces the need for database joins, which can lower costs.

- Scalability. A core function of MongoDB is its horizontal scalability, which makes it a useful database for companies running big data applications. In addition, sharding lets the database distribute data across a cluster of machines. MongoDB also supports the creation of zones of data based on a shard key.
- Third-party support. MongoDB supports several storage engines and provides pluggable storage engine APIs that let third parties develop their own storage engines for MongoDB.

- Aggregation. The DBMS also has built-in aggregation capabilities, which lets users run MapReduce code directly on the database rather than running MapReduce on Hadoop. MongoDB also includes its own file system called GridFS, akin to the Hadoop Distributed File System. The use of the file system is primarily for storing files larger than BSON's size limit of 16 MB per document. These similarities let MongoDB be used instead of Hadoop, though the database software does integrate with Hadoop, Spark and other data processing frameworks.

Disadvantages of MongoDB

Though there are some valuable benefits to MongoDB, there are some downsides to it as well.

- Continuity. With its automatic failover strategy, a user sets up just one master node in a MongoDB cluster. If the master fails, another node will automatically convert to the new master. This switch promises continuity, but it isn't instantaneous -- it can take up to a minute. By comparison, the Cassandra NoSQL database supports multiple master nodes. If one master goes down, another is standing by, creating a highly available database infrastructure.

- Write limits. MongoDB's single master node also limits how fast data can be written to the database. Data writes must be recorded on the master, and writing new information to the database is limited by the capacity of that master node.

- Data consistency. MongoDB does not provide full referential integrity using foreign-key constraints, which could affect data consistency.

- Security. In addition, user authentication isn't enabled by default in MongoDB databases. However, malicious hackers have targeted large numbers of unsecured MongoDB systems in attacks, which led to the addition of a default setting that blocks networked connections to databases if they haven't been configured by a database administrator.

2.2 Setting Up MongoDB

To embark on the MERN journey, it's essential to set up and configure MongoDB. We'll guide you through the installation process, whether you choose to deploy MongoDB locally or utilize cloud-based services.

2.3 CRUD Operations in MongoDB

Understanding how to perform CRUD (Create, Read, Update, Delete) operations is fundamental for working with any database. We'll dive into MongoDB's query language and demonstrate how to interact with data through practical examples. Whether you're retrieving documents, updating existing records, or deleting entries, this section equip you with the essential skills to manipulate data effectively in MongoDB.

Data Modeling in MongoDB

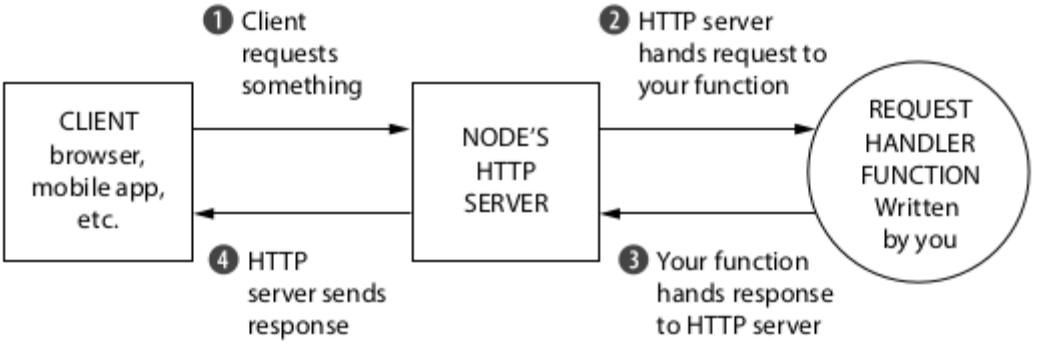
Efficient data modeling is crucial for building scalable and maintainable applications. MongoDB's flexible schema allows developers to design data structures that align with the application's requirements.

Express.js

Introduction to Express.js

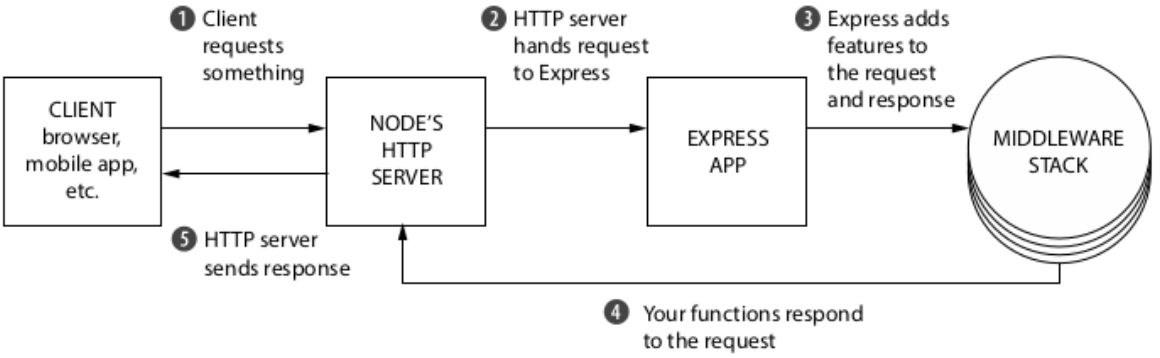
Express.js (Express) is a light web framework which sits on top of Node.js and it adds functionality like (middleware, routing, etc.) and simplicity to Node.js.

When creating a Node.js web application, we write a single JavaScript application which listens to requests from the browser, based on the request, the function will send back some data or an HTML web page.



A request handler is a JavaScript function which takes a request and sends an appropriate response.

Node.js APIs can get complex and writing how to handle a single request can end up being over 50 lines of code. Express makes it easier to write Node.js web applications.



Advantages of using Express with Node.js

- Express lets you take away a lot of the complexities of Node.js while adding helpful functions to a Node.js HTTP server.
- Instead of a large request handler function, Express allows us to handle requests by writing many small modular and maintainable functions.
- Express is *not opinionated*, meaning Express does not enforce any "right way" of doing things. You can use any compatible middleware, and you can structure the app as you wish, making it flexible.
- We can integrate with a template rendering engine (also called a view rendering engine in some articles) of our choice like Jade, Pug, EJS, etc.

A template engine enables you to use static template files and at runtime change the values of variables in those files.

- You can set up "middleware" for request processing.

Core parts of Express

Middleware

Middleware is a set of functions that sit between a raw request and the final intended route. Middleware functions have access to *all* the HTTP requests coming to the server. Middleware can handle tasks such as logging, sending static files, authorization, and session management, etc.

In Node.js, the request and response objects are passed to one function (request handler) that we write, in Express these objects are passed through a set of functions, called the **middleware stack**.

Express will start at the first function in the stack and execute in order down the stack.

Every function in the stack takes three arguments request, response and next. next is a function, that when called Express executes the next function in the stack. This is a subtle difference between middleware and a route handler which we saw above.

Routing

Express makes request handling easier by mapping requests to different request handlers. A request handler is a function which handles all the requests to a specific path with a specific HTTP method.

In the basic example above, we saw how to handle a GET request. As an application grows in size the routes grow as well as do the request handlers.

Template Engines

Websites are built with HTML, you can dynamically generate HTML pages using Express. Dynamically generated HTML pages are useful when you want to show real time data or change a page's details based on the user.

A template engine allows you to use static template files and at runtime replace variables in a template file with actual data.

There are different template engines available like Pug, Jade, and EJS. Let's see a basic template using EJS.

3.2 Getting Started with Express.js

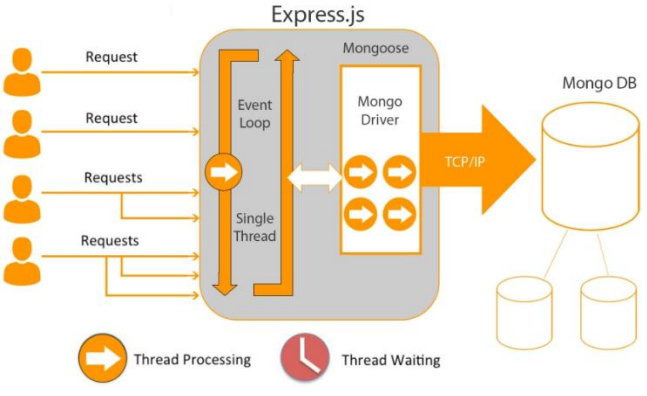
Express.js, a minimalist and flexible web application framework for Node.js, plays a pivotal role in building the backend of MERN applications. In this section, guide through the basics of Express.js, from setting up a new project to understanding the core concepts such as routing and middleware. You will gain hands-on experience in creating a simple Express.js server and handling HTTP requests.

Routing in Express

Routing is a fundamental aspect of any web application, and Express.js simplifies this process by providing a robust routing system. We will delve into the intricacies of routing in Express, demonstrating how to define routes, handle parameters, and structure your application for optimal code organization.

Middleware in Express

Middleware functions in Express enable you to execute code at different stages of the request-response cycle. This powerful feature allows for customization, authentication, and error handling. We'll explore the concept of middleware in Express, demonstrating how to create, use, and chain middleware functions effectively.



Understanding middleware is essential for implementing features such as authentication and request validation in your MERN applications.

3.5 Error Handling in Express

Handling errors gracefully is crucial for maintaining a robust and user-friendly application. This section guide you through the best practices of error handling in Express.js, covering topics such as creating custom error handlers, handling asynchronous errors, and implementing global error middleware.

Node.js

Introduction to Node.js

Node.js is an open-source and cross-platform JavaScript runtime environment. It is a popular tool for almost any kind of project!

Node.js runs the V8 JavaScript engine, the core of Google Chrome, outside of the browser. This allows Node.js to be very performant.

A Node.js app runs in a single process, without creating a new thread for every request. Node.js provides a set of asynchronous I/O primitives in its standard library that prevent JavaScript code from blocking and generally, libraries in Node.js are written using non-blocking paradigms, making blocking behavior the exception rather than the norm.

When Node.js performs an I/O operation, like reading from the network, accessing a database or the filesystem, instead of blocking the thread and wasting CPU cycles waiting, Node.js will resume the operations when the response comes back.

This allows Node.js to handle thousands of concurrent connections with a single server without introducing the burden of managing thread concurrency, which could be a significant source of bugs.

Node.js has a unique advantage because millions of frontend developers that write JavaScript for the browser are now able to write the server-side code in addition to the client-side code without the need to learn a completely different language.

In Node.js the new ECMAScript standards can be used without problems, as you do not have to wait for all your users to update their browsers - you oversee deciding which ECMAScript version to use by changing the Node.js version, and you can also enable specific experimental features by running Node.js with flags.

Differences between Node.js and the Browser

Both the browser and Node.js use JavaScript as their programming language. Building apps that run in the browser is a completely different thing than building a Node.js application. Even though it is always JavaScript, there are some key differences that make the experience radically different.

From the perspective of a frontend developer who extensively uses JavaScript, Node.js apps bring with them a huge advantage: the comfort of programming everything - the frontend and the backend - in a single language.

You have a huge opportunity because we know how hard it is to fully, deeply learn a programming language, and by using the same language to perform all your work on the web - both on the client and on the server, you're in a unique position of advantage.

Node.js with TypeScript

What is TypeScript?

TypeScript is a trendy open-source language maintained and developed by Microsoft. It is loved and used by a lot of software developers around the world.

Basically, it is a superset of JavaScript that adds new capabilities to the language. The most notable addition is static type definitions, something that is not present in plain JavaScript. Thanks to types, it's possible, for example, to declare what kind of arguments we are expecting and what is returned exactly in our functions or what's the exact shape of the object that we are creating. TypeScript is a powerful tool and opens a new world of possibilities in JavaScript projects. It makes our code more secure and robust by preventing many bugs before the code is even shipped - it catches problems during code development and integrates wonderfully with code editors like Visual Studio Code.

Setting Up a Node.js Server

Getting started with Node.js involves setting up a server to handle incoming requests and responses. This section guide you through the process of creating a basic Node.js server, covering topics such as handling HTTP requests, routing, and interacting with the file system.

Handling Asynchronous Operations

Asynchronous programming is a key feature of Node.js, allowing developers to handle concurrent tasks efficiently. We will explore asynchronous operations in Node.js, covering callbacks, Promises, and the async/await syntax. Understanding how to manage asynchronous code is crucial for building responsive and scalable applications, and this section provides practical examples and best practices.

NPM (Node Package Manager)

NPM is a package manager for Node.js that simplifies the process of installing, managing, and sharing dependencies in your projects. This section introduce you to NPM, covering topics such as installing packages, managing project dependencies, and creating your own NPM packages.

React

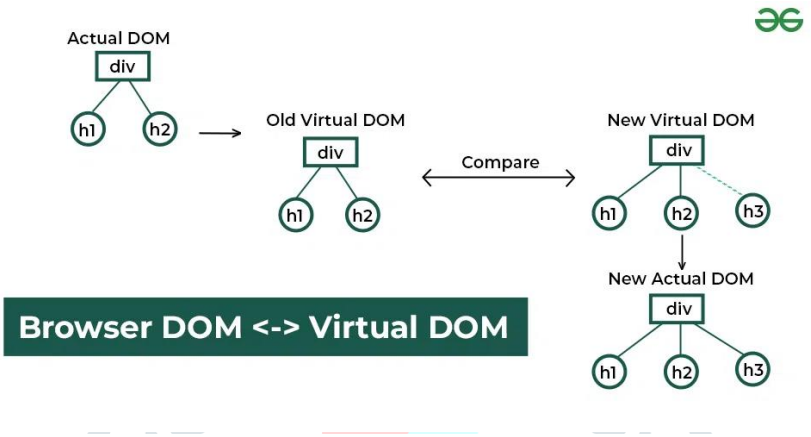
Introduction to React

React, developed by Facebook, is a powerful JavaScript library for building user interfaces. In this section, it provides a comprehensive introduction to React, exploring

its component-based architecture, virtual DOM, and the concept of declarative rendering. Understanding the core principles of React is essential for creating interactive and dynamic user interfaces in MERN stack applications.

How does ReactJS work?

React creates a virtual DOM in memory to update the browser's DOM. The virtual DOM will try to find the most efficient way to update the browser's DOM.



Unlike browser DOM elements, React elements are simple objects and are cheap to create. React DOM takes care of updating the DOM to match the React elements. The reason for this is that JavaScript is very fast and it's worth keeping a DOM tree in it to speed up its processing.

Although React was designed to be used in the browser, because of its design allows it to be used on the server with Node.js as well.

Components and Props

Central to React's philosophy is the concept of components—modular, reusable building blocks for user interfaces. This section delves into creating and managing React components, understanding the role of state and props, and exploring best practices for component composition.

State and Lifecycle

State management is a critical aspect of building dynamic and responsive applications. We will explore React's state and lifecycle methods, showcasing how to manage component state, handle lifecycle events, and create interactive user interfaces that respond to user actions.

React Router

Single-page applications often require navigation between different views. React Router is a powerful library that enables navigation and routing in React applications. We will guide you through setting up React Router, creating dynamic routes, and handling navigation events.

Results and Benefits:

The MERN stack-powered framework produced amazing outcomes, such as a smooth user interface and strong real-time functionalities, which eventually helped customers and barbers in the online barber services sector. This technologically advanced approach promoted business expansion and scalability in addition to increasing operational effectiveness.

Outcomes and Advantages:

1. Enhanced User Experience: By streamlining the reservation process and offering real-time availability checks, the web application framework greatly improved user experience, which raised customer satisfaction.

2. Effective Time Management and Streamlined Appointment Scheduling Benefited Barbers: Better time management and streamlined appointment scheduling increased business productivity and decreased no-show rates.

3. Transparency and Trust: By removing surprises and fostering a sense of confidence among customers, the transparent pricing module helped them make more assured service selection decisions.

4. Improved Service Quality: By enabling customers to provide comments, the review system encouraged barbers to continuously improve, which in turn raised the bar for service quality.

5. Growth and Scalability: The MERN stack-based architecture was made to be easily expanded and adjusted to the changing requirements of the online barber services sector and ensuring long-term growth and sustainability.

Future Scope

The website that uses the MERN stack to provide online barber services has a very promising future. There are many of chances for growth and development. Adding services like appointment reminders and real-time chat assistance can increase customer happiness and engagement. The platform may be made even more approachable and user-friendly with the creation of specialized mobile apps. The platform's reach and revenue potential can be increased through geographic expansion and the incorporation of e-commerce features. Furthermore, the application of AI, machine learning, and data analytics can yield insightful data that further customizes the user experience. Contributions to the platform's success and durability include making sure it supports multiple languages, putting strong security measures in place, and building a user community. All things considered, this website could develop into a broad, well-known platform that meets the demands of customers and barbers in the online barber services sector.

Conclusion

This paper indicates an in-depth analysis of the overall performance of MERN stack which is the present-day technology. It additionally studies the net improvement the use of one-of-a-kind technologies and frameworks. It includes the architecture of MERN stack, its additives and its implementation. It shows the Node.Js Frameworks and tools, also the Python vs. Node.Js comparative desk, Advantage of the usage of Node. We studied Express JS. Moreover, we additionally studies React and its additives, React Virtual DOM and its blessings. We also took the evaluate of Mongo DB and its comparison with Oracle database. Lastly it studies the blessings of MERN stack and the pinnacle brands the usage of it.

References

https://www.ijert.org/research/performance-optimization-using-mern-stack-on- web-application-IJERTV10IS060239.pdf

https://www.ijraset.com/research-paper/mern-full-stack-development

http://ijrra.net/Vol5issue1/IJRRA-05-01-26.pdf

https://www.irjmets.com/uploadedfiles/paper//issue_3_march_2023/34270/final/fin_irjmets1678887207.pdf