UtilitasMathematica

ISSN 0315-3681 Volume 122, 2025

Design and Development of a Student Industry Relations and Placement Web-App Using MERN Stack

Gowtham Rudru[1], MVBT Santhi[1]

[1]Department of Computer Science & Engineering,

Koneru Lakshmaiah Education Foundation, Vijayawada-India, 2301050051cse@gmail.com, santhi2100@gmail.com

Abstract

Architecting and Developing Web Portals have expanded rapidly with the advancement of digital innovation. Software-based solutions are developing at a rate comparable to that of hardware. The Internet and real-time-enabled electronic devices have reinforced the growing importance of performance. Regrettably, due to their widespread use and a long history of traditional technology development and maintenance, some of them do not meet current client activity standards. The MERN stack was developed to address this performance issue. This is an application of the web that allows seekers and posters to meet and look for jobs. This project ends with the building and deployment of a functional and production-ready web store application. The results of the application and possible improvements are also covered. This dissertation can serve as an introduction to the MERN stack for beginners along with those thrilled by the discovery of the technology framework.

Keywords

Internet-Based Software Development, MERN-Based Framework, Creative Solution, Transformation, Employment Seeking Activity, Job Opportunity Announcement, User Seeking and Offering Opportunities

INTRODUCTION

In the current era, the rate of advancement in technology is unprecedented. Simultaneously with the development of hardware, software technologies also advanced to fill the vacuum. As Internet-powered electronic devices become increasingly prevalent and have real-time capabilities, performance has emerged as a significant concern. Traditionally, web development was done using JAVA servlets, ASP.NET, or PHP. Although these technological tools are common and feature well-designed components with so many years of existence and a huge community, they also carry some restrictions regarding today's needs, such as performance. MERN stack, which is MongoDB, Express, React, and Node, has been created in the past few days as a more advanced alternative to solve the difficulty with this performance-related challenge

The fundamental aims of this dissertation are to describe and recognize the basic core principles and usages of each technological solution contained within the MERN stack and their alignment, along with their benefits of being employed as a total framework in the development of web-based applications. It was reached by the implementation of these new technologies and the actual use of a web application. The thought behind this web-based system was to consider the startup led by the writer's progenitors since individuals wanted to initiate a literary store. However, after they did their search, the writer realized how enormous merchants have been opening at an extraordinarily fast pace during the past few decades all over the world, giving customers a more convenient experience than physical stores. The merchant has changed the way of communication between corporations and clients permanently, which aids users to be connected with their preferred shops and labels anytime and anywhere they desire, as well as aids the outlets in reaching out with greater intensity to customers. It is predicted that the expansion of merchants for the next upcoming years is growing an at a remarkable speed alongside technological advancements. To satisfy this, the author has designed a product: an application for Merchant, an online bookstore, in which the startup will design its business plan.

The document outline followed this form. The opening section introduced the objective regarding the thesis and the technologies that will be employed. The basic ideas and theoretical underpinning of every technology of the stack were then laid out, along with examples for clarity. In the third place came the application development. This consisted of both front-end and back-end parts explained in detail and illustrated in depth. Finally, the paper analyzed the project as regards improvements that might be required for the resultant product.

I. RELATED WORK:

PROJECT SUMMARY:

The Job Portal utilizes the MERN Stack to deliver a dynamic web application, which removes the complexity from the flow of the job application process. It facilitates the user's role to be either applicant or recruiter and provides them with the option to create a web-based application account. This application supports a persistent user session, while the REST APIs remain secured via JWT token validation. Once authenticated, the recruiter is granted the ability to create, delete, or update job listings; manage applications by shortlisting, accepting, or rejecting candidates; review submitted résumés; and modify their own profile. On the other hand, applicants can explore available job opportunities, utilize fuzzy search with multiple filters, apply to positions by submitting a Statement of Purpose (SOP), track their applications, upload a profile image and résumé, and edit their personal profiles. Therefore, the

system functions as a comprehensive, unified platform for job application management.

THEORETICAL FRAMEWORK:-

The full-stack technology in use primarily is the MERN stack. This chapter offers a summary of the technologies utilized within the MERN stack along with the assistance of Mongoose, a supporting third-party library, in building the project.

THE MERN STACK:-

MERN has emerged as the most influential adaptation of the MEAN stack. Since its inception in 2013, the MEAN stack has gained popularity through the efforts of a team of engineers at MongoDB with the vision to create a JavaScript-centric framework for making application development simpler. The elements that make up the stack are open-source; here, MongoDB handles data storage, Express operates as the backend framework, Angular manages the frontend, and Node runs JavaScript on the server. This framework, Angular, can be substituted by the client-side library React. Along with other technologies in a MERN stack setup, React is able to be a supporting component to implement applications using JavaScript and JSON standards

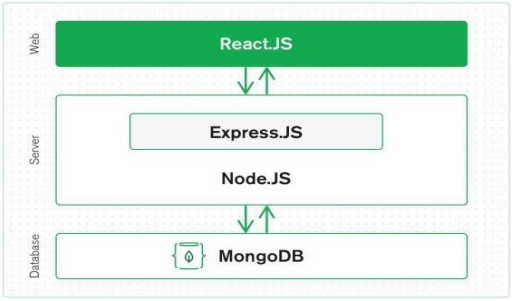


Figure 1. The structural design of the MERN technology stack

The MERN stack, as depicted above, is founded on the traditional three-tier design, proving its full-stack development capabilities. So, the three pieces of software Components of the MERN stack include the presentation layer is powered by React, the logic layer is built with Node.js and Express, and MongoDB provides the database support.

MongoDB is a community-driven open-source cross-platform NoSQL database primarily for applications and operations involving a lot of data, which do poorly in traditional relational databases. It employs document storage in binary-encoded JavaScript Object Notation. Its creators are Dwight Merriman, Eliot Horowitz, and Kevin Ryan, who gained quick popularity around mid-2000. It follows that, according to the design of Figure 2, that MongoDB employs collections and documents for data storage, rather than the table-row system of correlated databases.

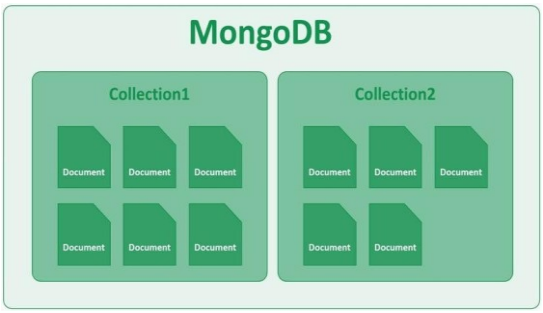


Figure 2. The architecture of MongoDB

MongoDB supports multiple document operations such as add, query, update, and delete. MongoDB is suited to many applications owing to a diverse range of values for field entries and flexible query languages. Also, its horizontal scaling with larger volumes of data makes it stand as one of the most successful NoSQL databases around the world.

Essential characteristics of MongoDB:

Scheme-free Database: It's a flexible schema model because it enables one collection to hold thousands of documents, and each document can have a varied Structure, content, and storage size. Accordingly, MongoDB bestows a huge flexibility upon databases due to this great feature.

• Document-oriented: Data resides within fields holding a very structured format in the form of key-value pairs instead of structured in a large table format of flexibility when compared to RDBMS

• Indexing: MongoDB documents are indexed by all fields, each with principal and subordinate indices. Accessing stored data becomes quite easy and quicker. Without proper Data pointers, the database has to search through Individual documents for the corresponding queries, and storing takes a lot of time and is inefficient.

• Horizontal Scalability: Sharding was horizontal scalability. Sharding is an operation where the distribution of data among multiple servers occurs. By means of a shard key, lots of data get divided into pieces and are then distributed in an even manner across shards that are composed of different physical servers. Furthermore, it will introduce new machines into a current database

• Replication: MongoDB has the advantage of replication. It means the data is duplicated in many places, and this is on a different server, helping in the defense of the database from hardware failure, thereby making the data recover from another one incase any server stops working

•Accumulation: operations involving the dataset are supported, which gives isolated or computed result with three aggregating techniques involved multi-stage data processing, map and reduce-based computation, one-step data summarization

.**EXPRESS:-**

Denoting 'E' as Express.js in MERN, Express is a highly light and multi-purpose web application toolkit developed on NodeJS. Resulting from its extensive support community, it has a rich set of functionality that can be implemented on the web and in mobile application development. Although a range of supporting tools bundled with its capabilities that

are employed for improved software development, Express doesn't impact the operational capability of NodeJS.

Express was started as recorded on 22 May 2010 authored by T.J. Holowaychuk, as per the GitHub repository. Subsequently, the rights of project management were purchased by Strong Loop in June 2014 until the company became owned by IBM on September 2015. Subsequently, since January 2016, the controlling authority of Express was taken over by NodeJS Foundation, and Express is presently the primary use case of the NodeJS environment.

Express.js serves as the middleware and routing solution that's designed to support all kinds of routing to a site. It's the one that falls in between request and response loops. Before sending a response back via controller actions and once after the request is received by the server, middleware gets called. Operations of middleware are usually called once or multiple times, maybe for authenticating the requests or to analyze the request payload. In Express, applications function by passing requests through a sequence of middleware functions. The first one is consistently called to begin the task pipeline processing of the solicitation. The output from the initial one can terminate processing the request to render content for the user or proceed to the next middleware function. This is done until the final middleware of the pipeline gets the input of the previous middleware.

REACT:-

In the context of the MERN stack, 'R' stands for React and is focused on developing the View Layer, renowned for all parts of an application's visible page. React is a multi-purpose, open-source JavaScript library employed in crafting interfaces constructed from reusable user interface elements. Being designed for large-scale, complex, real-time user interfaces and featuring built-in data binding, React has continued to stay at the leading edge of single-page application development and front-end utilities to meet contemporary programmer requirements on all fronts. It is the most extensively used framework for web development, surpassing other popular libraries and frameworks such as jQuery, Angular, and VueJS. Besides providing modular design code that saves period during development and decreases system bugs and faults, Reacts provides a lot of features required, which add up to the developers' attraction.

JSX:-

JSX is virtually an extension to JavaScript that resembles HTML. Since JSX compiles into regular JavaScript and takes advantage of all its facilities of, JSX is far faster than the traditional one. Despite JSX not being used mandatorily in the procedure for developing applications with React, it's a recommended because it simplifies developing tools for developers when markup components and event binding are needed. This grants the ability to avoidance of the segregation of structure and functionality into two different files because presentation logic and interaction handling may be put within those same modules to lead to more readable, maintainable codebases on websites.

VIRTUAL DOM:-

VDOM is short for Virtual DOM, and this signifies a theoretical representation related to the DOM with solutions constructed over the normal DOM. The abbreviation of DOM is Document Object Model, and it models the document in the form of an ensemble of various building blocks and objects used to navigate the structure, arrangement, and website content generated using programming languages. Tree-based layout of the Document Object Model Figure 3 shows the structure of DOM as an illustration.

While modifying the DOM is time-consuming because of re-rendering the element and its nested components updated with the latest data, the Virtual DOM updates and renders only the necessary components, thus fastening the rendering phase and increasing the performance. The working mechanism of the Virtual DOM must, therefore, be further discussed to realize why the virtual DOM performs efficiently and practical. Whenever a page is rendered using Virtual DOM, It consistently maintains the DOM's state tree hierarchy. In this case, instead of building a tree once more, the library takes a different approach if the UI needs changing. This is when the Virtual DOM is used by the React library, which enables it to run the calculations. Inside this domain, not involve the actual DOM. Thus, with each change of state in a component, React automatically tracks and updates it.

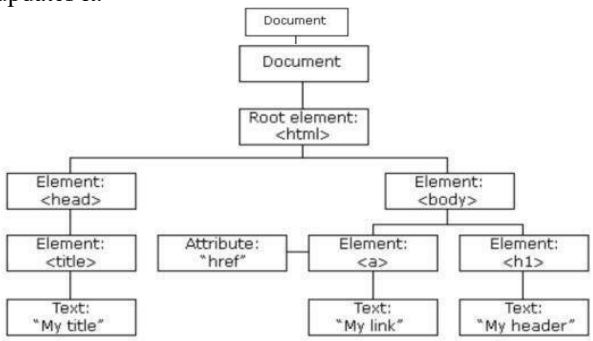


Figure 3. Tree-based layout of the Document Object Model

Comparing the Virtual DOM tree with the earlier one. That is done via a diffing algorithm, aiming at minimizing as many DOM operations/refreshes as possible which makes it a speed booster. Thus, the overall process is called Reconciliation

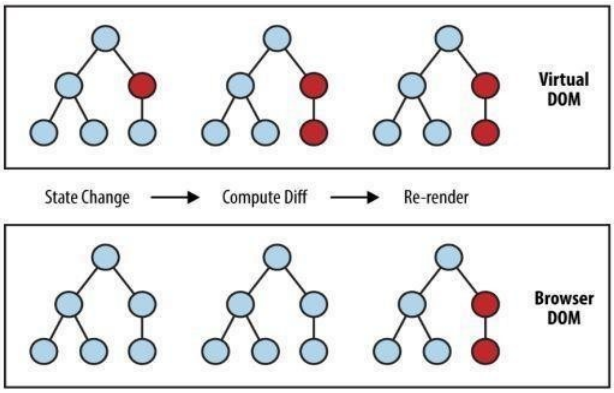


Figure 4. Variations in updating between Virtual

COMPONENTS:-

Components lie at the foundation of React, supporting the breakdown of complex interfaces into simpler, reusable chunks. There exist two forms of components in React: class-based and functional components. The concept of a functional component is what people feel is the simplest method of creating because A simple JavaScript function can be used to define it and return JSX. Class-based part is

created with the usage of ES6 class-based syntax and built-in library React class "Component".

Components can be referred to other components. Therefore, A parent component can include numerous child components, with no restriction on nesting depth or complexity. In addition, React requires that both functional and class components operate as pure functions, treating props as immutable. Props is the abbreviation for properties; it represents a collection of inputs received by the component as parameters. Next, the pure function determines the state where it runs a function that operates without changing its arguments. Hence, In React components act as pure functions of their inputs but never modifies its input and generates output consistent if props are passed multiple times.

HOOKS:-

Until React 16.8, all components in React were class-based on the fact that class-based components provide lifecycle methods for Maintaining state inside a component. Starting from React version 16.8, React has provided a new concept referred to as Hooks that provides a different means of accessing the state and additional functionality of a hook-enabled component. Using the hooks, the stateful logic of a component may be isolated and independently tested with the possibility of reuse without having any impact on the hierarchy of the component. Besides, Hooks enable developers to break down code for any single component into related functions instead of using life-cycle methods.

Two of the native React Hooks will be explained. The first is the use State hook, which enables managing state within a component by providing a React state property will keep its hold on. It takes a State hook, receives and provides two objects: the state in question and an update function for it. It reduces the complexity of initialization usage, and changing of the component state. The impact Hook, which is also called the utilize effect hook, is the second one. Utilization of the effect Hook facilitates developers in controlling the life cycles of the components. The impact Hook has addressed the issue of splitting related data and logic across multiple class lifecycle methods: e.g., component Did Update, component Did Mount, and component will unmount. That implies that the React component has the ability to deal with more than a single effect With the intention of maintain the issues of manipulating data isolated.

NODE:-

NodeJS is a JavaScript runtime built on an open-source, cross-platform framework for scalable application development. NodeJS is developed separately on the V8 runtime environment of Google Chrome, which is well established to run successfully without a browser. NodeJS provides efficient I/O handling through asynchronous and non-blocking operations so that the operation can be executed using a single-thread event loop and an event-driven approach can be followed. It enhances the performance and scalability of web applications, as shown in Figure 5 below:

As a result, it offers developers an alternative approach to meet the requirements of building Optimized and real-time software solutions.

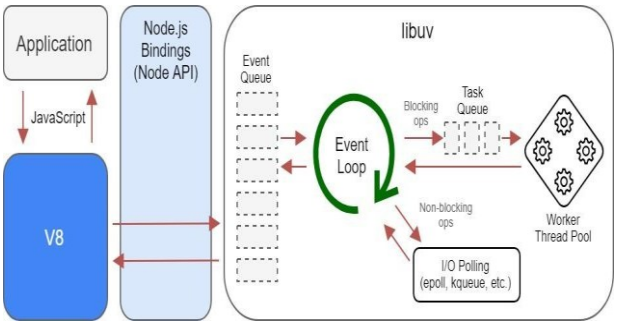


Figure 5. The Event Loop Mechanism in NodeJS

Node's package management tool (NPM):

The Node Package Manager (NPM) serves as the default package manager for NodeJS apps, allowing developers to manage packages and modules using the npm command-line tool. So, It simplifies operations that typically demand a lot of manual effort as it automatically takes care of the third-party packages. Therefore, the creator can save a lot of time as part of the development. Isaac Z. Schlueter introduced NPM to the public on January 12, 2010. It's also packaged with NodeJS; it installs other libraries and components needed for use in a NodeJS project. As of late March 2022, it has arround two million packages, and it stands as the biggest software registry globally

MONGOOSE:-

Mongoose is the ODM library that simplifies Node and MongoDB development. Mongoose manages data relationship, schema validation and serves as an intermediary between code objects and MongoDB object representations. Mongoose also provides a series of methods and functions that help in very efficiently facilitating interaction between MongoDB and NodeJS. Figure 6 shows how Mongoose, NodeJS, and MongoDB interact with each other.

According to Figure 6, Mongoose connects Node with MongoDB by applying object mapping techniques. Mongoose subsequently interacts with MongoDB through a Mongo Driver. As a result, data operations are made possible through the integration of Mongoose, NodeJS, and MongoDB.

Mongoose, just like any other ODM library, begins by declaring a schema. On the Mongoose documentation site, the schema declares the organization of data and the casting of properties using object-level functions, multi-field indexes, class-level methods, and middleware functions. Having completed the initial step, the schemas previously declared will be then used for mapping MongoDB collections and declaring the shape of the documents containing the data stored within each collection. Programmers must also perform the second step, which is the definition of a Mongoose model. Models consist of schema developers whose main function is the creation and MongoDB document schema validation Other significant features are the querying, removal, and modification of a document within the database.

Figure 6. How Mongoose Connects NodeJS and MongoDB

III - PRESENT METHODS FOR IDENTIFYING MOVING OBJECTS

SYSTEM ANALYSIS DESIGN

Project Overview:-

The MERN stack Merchant website project specification outlines the system's key requirements, functionalities, and features in detail. It provides a comprehensive overview of the proposed Web platform, encompassing its design and user interface, operational capabilities, Protective measures, and efficiency expectations.

General Requirements:

The project involves building the Merchant website using the MERN stack framework, In other words, MongoDB, Express.js, React, and Node.js Your site needs to be responsive, simple to operate, and intuitive. The site should provide a clean and minimalist user interface in accordance with the company image and brand. Your website must be multi-currency and multi-language to support sales worldwide.

Functional Requirements:

A Merchant site should possess the following functionalities and features:

User Account Management: The site management must accommodate all user account management, including personal details and shipping addresses to ship to and favorite payment methods.

Merchant inventory: The website should include a comprehensive product catalog featuring detailed information, images, pricing, and client feedback. The system must allow users to browse and refine assets using different factors, including cost, type, and manufacturer.

Shopping Cart**:** The web platform is designed to permit users to store products in their shopping cart, select visible selected products, and proceed to checkout.

Payment Gateway Integration**:** Integration with a trusted and secure payment gateway is essential to ensure the safety and efficiency of online payments.

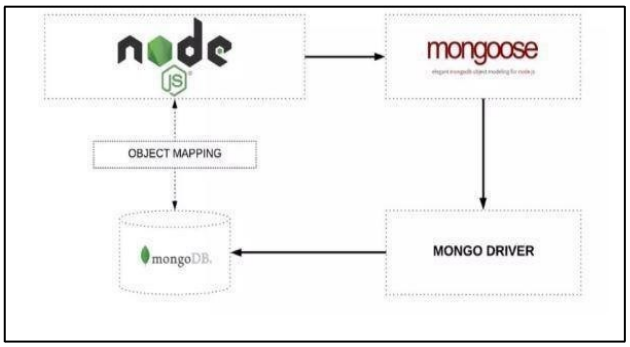
Order Management**:** The platform should support comprehensive order management features such as real-time tracking, efficient processing, and fulfillment handling.

Customer Service**:** The platform should include robust customer service tools, including real-time chat support, email communication, and an accessible FAQ page to address common queries.

Security Requirements:

A Merchant site should incorporate essential security components such as:

SSL Encryption: The platform should incorporate SSL



encryption to ensure the security of user data and prevent unauthorized access.

Efficiency Requirements**:** The system ss obligated to meet the listed criteria to ensure optimal performance of the Merchant platform.

Fast Load Time**:** The system must be optimized for rapid loading speeds to enhance user interaction and reduce bounce rates.

Scalability**:** The system must support horizontal and vertical scaling to handle increased user loads and operational demands over time.

Reliability**:** The platform should ensure high availability and dependable performance to facilitate uninterrupted access for users and transactions around the clock.

The MERN stack-based Merchant website project specification outlines the system's requirements, features, and core functionalities. It emphasizes a responsive layout, a user-friendly system interface, payment integration, and optimized load-up speed to ensure a fluid user experience. Based on this requirement, organizations can build and deploy a secure vendor platform, utilize the MERN stack, and reach the entity's across the globe.

IV - IMPLEMENTATION

Seeker:
In this module, the job seeker registers by providing basic information such as full name, email address, and password. Upon successful registration, the seeker can log in to the platform, browse job postings based on specific criteria—such as full-time or part-time roles—and proceed to apply for relevant positions.

Poster:
Here, the job Poster registers by providing basic information such as full name, email address, and password. User approval testing is the last stage of any project, and significant effort is supplied by the client. Furthermore, it validates that the system is functional according to the functional specifications.
Test results: Each test case described met the required criteria with distinction. No defects were discovered. After completing the registration process, the job poster logs in to the platform, submits job-related details through a form, and gains access to view the list of applicants who have shown interest in the posted jobs.

Steps to Create the Project

Creating a project folder
mkdir job post-app
cd job post-app
Backend Setup
Create a folder within the Merchant-app folder i.e. server
mkdir server cd server
Initialize the Express project and install the required dependencies.
npm init -y
npm i express cors mongoose

Folder Structure(Backend):
The updated dependencies in package.json file of the backend will look like this: "dependencies": {
"cors": "^2.8.5",

"express": "^4.18.2",

"mongoose": "^8.0.4",

"nodemon": "^3.0.2"
}

V - TESTING

System testing: The primary goal of system analysing is to identify defects within the software. It serves as a trial process to uncover any potential errors or vulnerabilities in the system components. Testing ensures that individual parts, sub-assemblies, complete assemblies, or the final product operate as intended. It involves executing the software to verify that it aligns with both specified design requirements and user goals while also ensuring it maintains stability and avoids critical failures. Various testing methodologies are used throughout the development process, each designed to address specific validation and verification needs..

White box testing: White box analysis is a method in which the internal structure is fully exposed for testing purposes, logic, and source code of the software solutions being tested. This approach allows for testing internal operations that are not accessible through black box testing methods. It is particularly useful for validating code paths, logic flows, and internal functions to ensure they perform as expected.

Black box testing: Black box testing involves testing software without knowledge of its internal structure or code. The focus is on giving inputs to the software and checking if it produces the expected outputs, based on its requirements. The test cases are typically derived from these functional specifications. The software is treated as a 'black box,' and interactions happen only through the user interface or API.

Acceptance testing: User Acceptance Testing serves as the final review in the software testing lifecycle and involves active participation from the end users. Its primary purpose is to verify that the system meets the defined functional specifications and operates correctly in practical situations. This testing phase confirms whether the software is ready for deployment and aligns with user needs and business objectives.

Test results: Every test case above passed with flying colors. No defects were found.

Testing methodologies:

The methodologies applied in testing are:

1. Unit testing.
2. Interface testing.
3. End-User tesing
4. Output Verification
5. validation testing.

Unit testing:

Unit testing targets the smallest functional units of software, typically individual modules. It verifies specific control paths within a module to guarantee thorough testing and optimal error identification. Each module is tested independently to confirm that it operates correctly according to design specifications. This includes validating both processing logic and error-handling routines, along with checking interface consistency.

Interface testing:

Interface testing resolves two problem verifications and program building issues. Following the building of software integration, several high-level tests are then performed. Essentially, the ultimate goal of such testing is taking unit-tested modules and developing the structure of the program as set by design

The following represents integration testing types;

1) Top-down integration: It is an incremental method used to construct a program's structure by integrating modules starting from the top of the control hierarchy—typically the main module. Subordinate modules are added progressively using either a depth-first or breadth-first strategy. As integration progresses, testing begins at the top level, and stub modules are gradually replaced with actual components as the process moves downward.

2) Bottom-up integration

This approach begins construction and analysis with the foundational modules in the program structure. Since modules are constructed bottom-up, all processing they require below any level is always present, and thus the elimination of stubs.

After completing the following steps, a bottom-up integration strategy should be employed:

* Lower-tier components are arranged inside functional groups that carry out particular operations within the system.

* Driver, therefore, the test control program is coded to manage input values and output responses.

* The clustered components are verified.

* Drivers stripped and grouped modules are merged, moving up the solution

The bottom-up strategy analyzes every individual component and, thereafter, incorporates each module alongside the primary section and functional tests.

End-User testing:

User approval is a vital aspect that determines the overall success of any system. For the system in question, user acceptance was evaluated during the development phase by maintaining regular communication with potential users and

incorporating feedback-driven changes as needed. The interface is designed to be user-friendly and intuitive, making it accessible even for individuals with no prior experience using similar systems.

Output Verification:

After conducting the validation test, the second thing to be done is output verification of the invented system. No system is of any benefit in case it fails to generate the desired results in a structured format. Requesting from the users regarding the required setup, the output is tested by them being printed or on-screen by the system being analyzed. Thus, the output is analyzed in terms of two dimensions – one on-screen and another printed.

Validation testing:

The system validates the following fields:

Text field**:** It allows characters only up to its defined limits. In certain tables, alphanumeric input is required for text fields, and in other tables, alphabetic. Whenever an incorrect entry has been performed, an error message is displayed.

Numeric Field**:** This field accepts only numeric values ranging from 0 to 9. If any non-numeric character is entered, an error message is displayed. Each module is tested independently to ensure it functions correctly and performs its intended task. Sample data is used during these test runs to verify the module's behavior. Once individually tested, these modules are integrated to form a complete system. Testing involves running the program with actual data to detect potential defects based on the output. The testing procedure must be organized to verify that each requirement is assessed separately. A successful evaluation identifies flaws caused by invalid inputs and produces results that highlight errors within the platform.

Test data setup:

This type of test data covers all the necessary testing. Preparing the test data is a crucial step in system testing. After completing the test data preparation, the specific test data is utilized for testing the specific system in question. During testing that system with test data again new errors come before it, so by using these above steps testing errors get corrected and those error corrections also make a note to use for further time.

Utilizing real test data:

Real test data refers to information gathered from organizational records. When the platform is not fully built, then programmers or analysts request users to enter a sample of their typical operational data. The platform user utilizes such data as a means of selective testing of the system. At times, programmers or analysts obtain representative live data from stored files and request them to be keyed in by them. Gathering live data in sufficient quantities for comprehensive testing can be challenging. Moreover, while realistic data simulates the system's behavior for typical processing needs, often relies on the assumption that the input live data is representative of real-world usage. However, such data usually fails to cover all possible combinations or input formats that the system might encounter. This reliance on typical values results in an incomplete and less reliable system test, as it overlooks edge cases and unusual inputs — the very scenarios highly probable to lead to platform failures.

Using artificial test data:

There are test data for the aim of testing only: both are merely generated so that all potential format and value combinations are covered. In effect, the generated data, readily available from the data-generating system utility within the information systems division, means that every authentication process and program flow can be verified through testing.

The most effective testing programs utilize synthetic data produced by external tools rather than the program authors. Often, a different group of testers takes a test plan from the system's specification. All the requirements specified in the software requirements document were fulfilled by the virtual private network package, which was then approved.

User training:

Upon each new arrival system being designed, the users need to be trained so that they are aware of the functionality of the system consequently that the system can be used effectively by the people whom the system is primarily built to serve. Regarding this, the task's regular workflow was demonstrated to the potential users. Its functionality is extremely easy, and initially, the prospective users are people who have a good understanding of computers, the functionality of this system is extremely easy.

Maintenance: This includes a wide range of activities, such as correcting code and design errors.

In the long term, we will require less maintenance. During this process of development of the system, we have described the user's requirement more effectively. Based on the specifications, this platform is designed to be as satisfactory to the requirements as is possible. Following the advancement of technology, various additional features may become feasible in line with the further requirement. The design and coding are user-friendly and simple to learn. and therefore will make maintenance easier.

Testing strategy :

A platform test plan combines test cases and design methods into a well-organized sequence of steps to ensure effective software development. The test plan must align with test preparation, test case creation, test execution, and subsequent information gathering and analysis. A comprehensive software testing plan should incorporate both low-tier analysis to confirm that fewer sections of the code script are implemented correctly and high-tier tests to validate that key system functions meet client specifications. Software analysis plays a crucial role in quality assurance in software development and serves as the final validation of specification design and coding. Testing presents an interesting paradox in the software development life cycle. Consequently, various tests are conducted on the proposed system before it is ready for customer approval testing.

System testing:

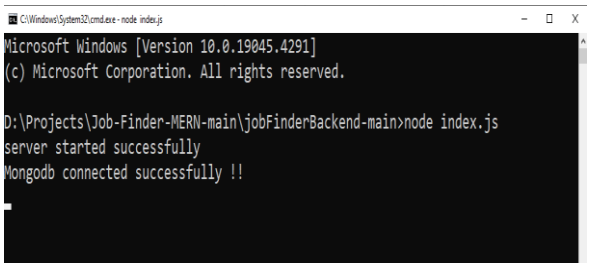
Software, after certification, should be integrated into additional components of the system (e.g., Hardware, people, database). System analysis confirms that all components are correct and that platform function efficiency as a whole is executed. System testing furthermore checks to see whether there are any differences between the system and its original purpose, existing requirements, and technical documentation.

Unit testing:

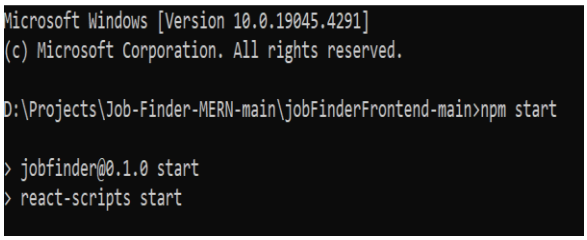
In unit testing, each module is tested based on the specified requirements done in the design phase of the module. Unit testing plays a key role in checking the code written in the coding stage; hence, its ultimate aim is to check the module-level logic. Under the guidance of an in-depth overview of the design, critical control paths are checked to see whether there are any faults within the module boundary or not. This is done in the programming phase itself. In this phase of testing, all the modules were found to be operating in the Expected Output From The Module mode seamlessly. With time, there will be continuous progress in technology. For integration into the technical infrastructure, most of the units of the network architecture are envisioned to be generic, hence making it feasible for future projects to use or communicate with them. There are numerous opportunities for developing and extending this project in the future.

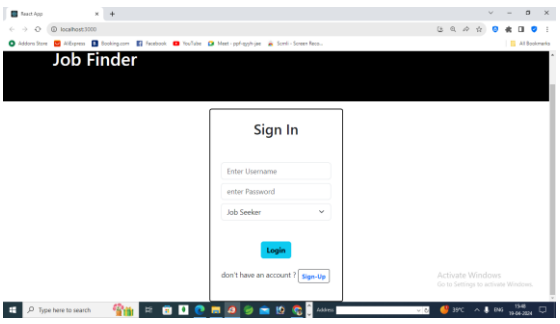
VI - Outcomes and Deliverables

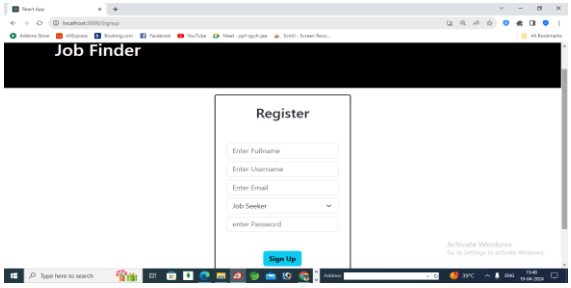
Back-end code execution steps:



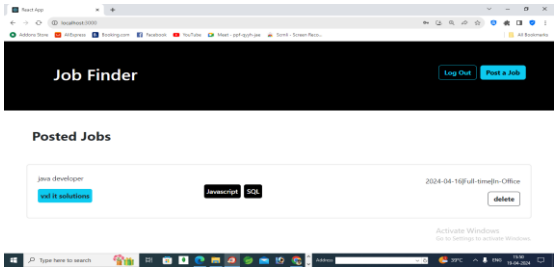
Front-end code execution steps:

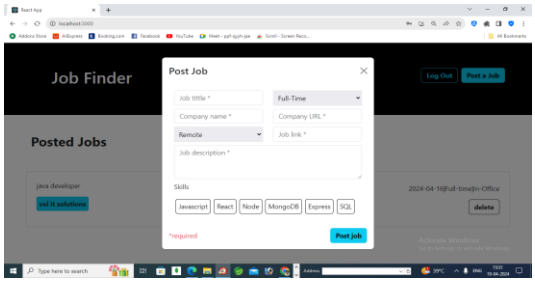


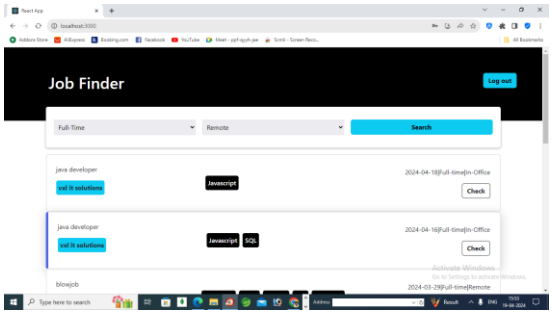


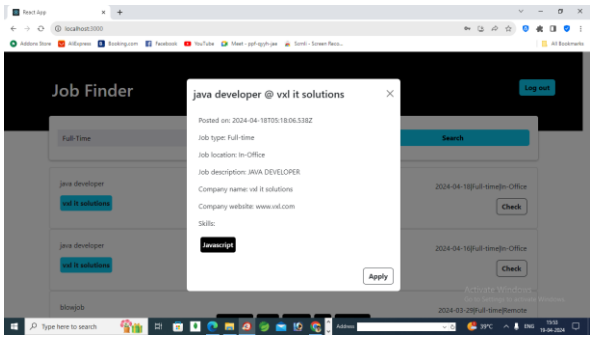


Poster login:









VII - CONCLUSION:

Our experience with the MERN stack has been more than a learning experience as far as acquiring skills and knowledge is involved; it has encouraged us to try newer technologies and go further into new things that can be completed via web development. In the coming days, we are looking forward to enhancing and growing our Merchant review app, according to user data feedback, and incorporating new features with the aim of improving the user experience. With the set MERN stack, nothing is now impossible, and it's time for us in general. The MERN stack offered an all-encompassing framework for developing a robust Merchant review application, effectively demonstrating the harmonious integration of its core technologies. From handling database operations and implementing server-side logic to designing the front-end interface, each layer of the stack worked seamlessly together so well that the application became cohesive and well-rounded. Looking back at our journey, we can see that learning and adaptation are essential in this ever-changing world of web development. Our experience with the MERN stack has been greater than a learning experience as far as gaining skills and knowledge are concerned it has motivated us to explore newer technologies and venture further into new things that may be possible through web development.

In the future, we look forward to improving and expanding our Merchant review app, taking into consideration the feedback from the users and adding new features to enhance the user experience. With the established MERN stack, nothing is now impossible, and it's time for us.

References

[1] Patel, K., & Shah, R. (2023). A Comprehensive Study on Building a Merchant Management System using MERN Stack. International Journal of Web Development and Technologies, 10(3), 45-52.

[2] Sharma, S., & Gupta, A. (2022). Design and Implementation of a Merchant Booking System using MERN Stack. Journal of Web Engineering & Technology, 7(2), 88-95.

[3] Nguyen, H., & Le, T. (2021). Developing a Merchant Review Platform with MERN Stack: A Case Study. International Journal of Advanced Web Studies, 6(1), 12-19.

[4] Khan, A., & Kumar, S. (2020). MERN Stack Application for Online Food Ordering and Delivery System. Journal of Emerging Technologies in Web Intelligence, 5(4), 187-194.

[5] Gupta, R., & Singh, M. (2019). Building a Real-Time Merchant Management System using MERN Stack. Journal of Web Development & Application, 4(3), 56-63

[6] Sharma, R., & Verma, A. (2018). MERN Stack Implementation for Merchant Reservation and Management System. International Journal of Web Engineering and Development, 9(2), 30-38.

[7] Patel, N., & Shah, P. (2017). Design and Development of a Merchant Menu Management System using MERN Stack. Journal of Web Technologies and Applications, 12(4), 55-63.

[8] Gupta, S., & Singh, V. (2016). Implementing a Merchant Feedback System with MERN Stack. International Journal of Advanced Web Technologies and Applications, 3(1), 18-25.

[9] Khan, M. A., & Ali, S. (2015). MERN Stack-Based Application for Merchant Order Management and Tracking. Journal of Web Application Development, 8(3), 47-55.

[10] Sharma, A., & Gupta, R. (2014). Building a Dynamic Merchant Website using MERN Stack. International Journal of Web Application Engineering, 7(2), 82-89.