SUMMER INTERNSHIP TRAINING REPORT

FULL STACK DEVELOPMENT(MEAN)

A Report is submitted

In Partial Fulfillment of the Requirements For

the Degree of

BACHELORS OF TECHNOLOGY

In

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

By

PRADYUMN SETH

(4rd YEAR)

UNIVERSITY ROLL NO. : 1900100100105

to the faculty of

UNITED COLLEGE OF ENGINEERING AND RESEARCH

DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW

DATE OF SUBMISSION:22th DECEMBER,2022

ACKNOWLEDGEMENT

I would like to earnestly acknowledge the sincere efforts and valuable time given by Our teacher Mr. Dilip Kumar and Mr. Vijay Kumar Dwivedi for their valuable guidance and feedback has helped me in completing this project.

Also, I would like to mention the support system and consideration of my parents who have always been there in my life.

I extend my warm gratitude and regards to everyone who helped me during my Summer Training.
Thanks a lot.

ABSTRACT

The main purpose of this effort is to present a brand-new environment for practising some of the most broadly used – both client- and server- side – web technologies. It is about a web-based, access-free, educational platform, which provides a user-friendly interface, illustrative graphics and supporting material, as well. Full stack development platforms are rarely met online, as most of them are usually oriented towards either front or back- end development and focus on specific programming languages without offering an overview of actual, integrated projects. This research also involves evaluating existing teaching methods, scanning and comparing some of the most popular, educational web platforms and, furthermore, discovering simple techniques and efficient approaches to reach valuable programming resources for both students and self-learners. The paper places particular emphasis on the recognition of the applications' key features and the variety of programming tools that promote learning and skill enhancement. Moreover, it discusses the roles of tutors and learners, while suggesting a learning path for novice developers. Given the fact that computer science courses often require exceptional practices, this study aims at encouraging active, self-motivated and self-paced learning.

CONTENT

DECLARATION

I hereby declare that the work presented in this report entitled "Full Stack Development Using MEAN", was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

I have given due credit to the original authors/sources for all the words, ideas, diagrams, computer program that are not my original contribution.

I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

PRADYUMN SETH

UNIVERSITY ROLL NO. : 1900100100105

COMPUTER SCIENCE AND ENGINEERING

Pradyumn Seth

(Candidates Signature)

INTRODUCTION

With the recent evolution in web technologies, there has never been a more exciting time for developers and technologists around the globe to build modern web applications. Web development is not anymore confined in the realm of pure HTML (HyperText Markup Language), CSS (Cascading Style Sheets), and JavaScript on the front-end, and PHP/Perl on the back-end. There is a plethora of new languages, web frameworks and tools to choose for any web application development. Although the rise of web technologies has helped to ease the application development process, it has created confusion among developers to select a perfect technology stack to start with.

For any web application development, it is important to choose a correct technology stack which allows rapid prototyping, constant iteration, code reuse, maximum efficiency, and robustness. It is also important that the technology stack is easy to learn and understand by the developers working on the front-end and the back-end. Thus, the concept of Full Stack JavaScript was developed. Originally web development was based on the LAMP (Linux, Apache, MySQL, PHP/Perl) stack and Java (Java EE, Spring), which consists of different programming languages. JavaScript solved this multi-language paradigm by introducing MEAN (MongoDB, Express, AngularJS, Node.js) stack which is based on a single language 'JavaScript'.

This thesis was carried out as a research project on the topic 'Full Stack JavaScript'. The main objective of the thesis was to study the different components of the most popular Full Stack JavaScript framework, MEAN stack, and build a prototype application based on it. The prototype application was developed only to illustrate the implementation of the MEAN stack, so the report focuses more on the practical implementation of the MEAN stack's components in the application than the application itself. Furthermore, the report analyses the strengths and weaknesses of the MEAN stack, and suggests the context where to use it and where to avoid it. The report also suggests the best design principles and architecture to follow when developing a MEAN stack web application.

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 throughout the web application on the first page load. It then dynamically changes the contents as the user interacts 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.

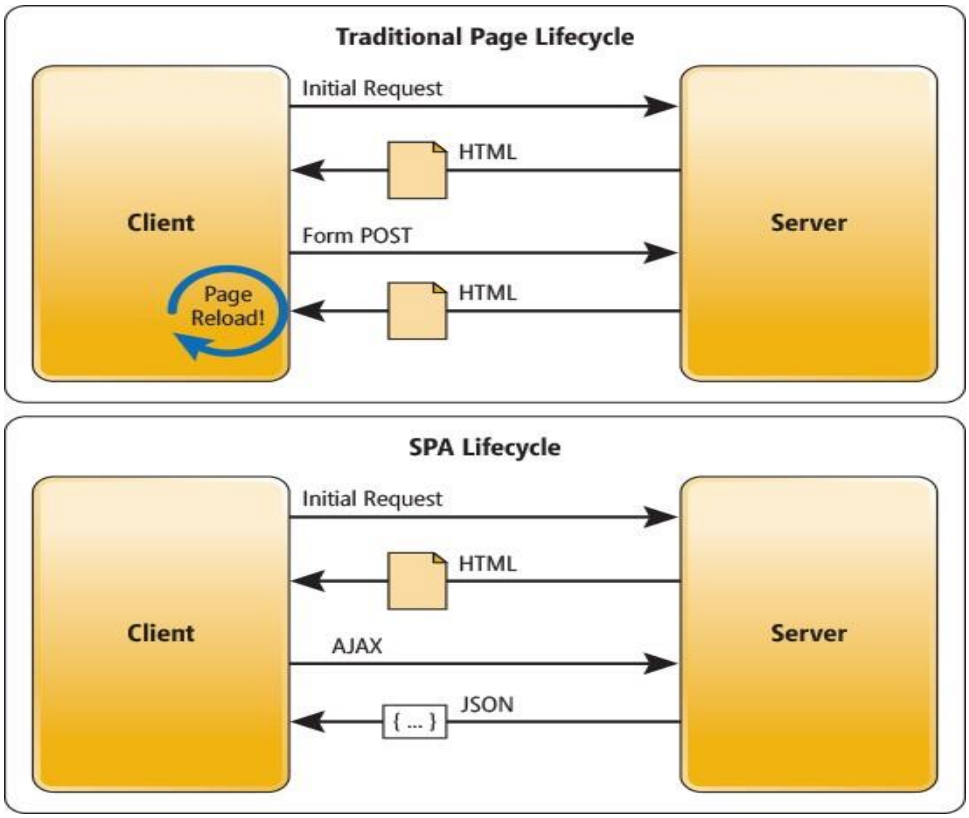


Fig 1 : Comparison of traditional page lifecycle and SPA lifecycle

Figure 1 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.

SPAs use AJAX (to interact with the server), HTML templates, MVC frameworks, and JavaScript to perform most of the navigation works on the front-end. Modern front-end JavaScript frameworks such as AngularJS, Ember.js, React, and Meteor.js have simplified the tasks of creating SPAs by providing rich DOM manipulation and two way data binding features. SPAs provide a rich interface and fluid user experience. Moreover, SPAs make users feel like they were interacting with a desktop application.

MEAN Stack : 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.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. In addition to the V8 engine, Node.js is composed of several components as shown in figure 4.

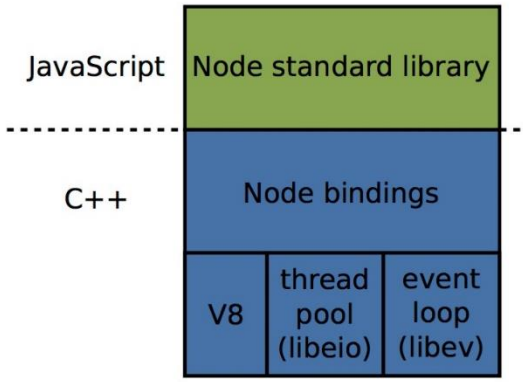


Fig 2 : Node.js Architecture

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.

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.

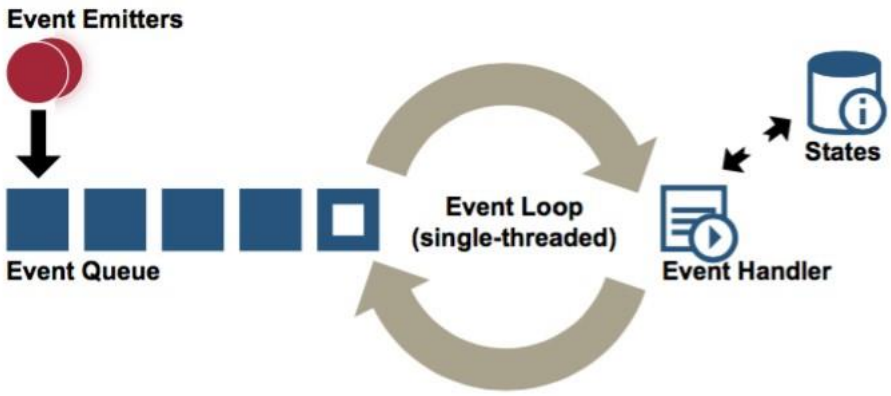


Fig 3 : Node.js processing model

In figure 3, 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.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.

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.

Node.js is actually the foundation of the MEAN 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.This Node.js feature maintains the speed and efficiency of the overall system.

Table 1. Performance benchmark of Node.js, Python-Web, and PHP

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 illustrates the performance results of the Node.js in comparison with PythonWeb 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.

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.

MEAN Stack : Express

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. 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.Express is installed using the npm package manager issuing "`npm install express`" command.

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 client-server 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). 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 avoid request hanging.A simple Express web server containing all three building blocks is shown in figure 4.

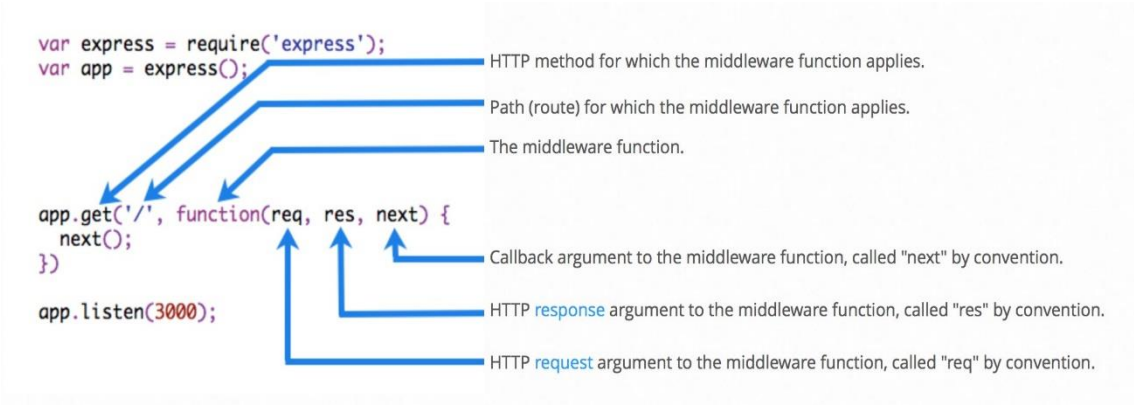


Fig 4 : Express Web Server

Figure 4 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 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.

MEAN Stack : 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. 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.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.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.

Table 2. Comparison of MySQL and MongoDB terms

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

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.

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.

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 read- write 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.This allows to build highly scalable and

efficient data servers in comparison to relational databases such as MySQL and Oracle databases.

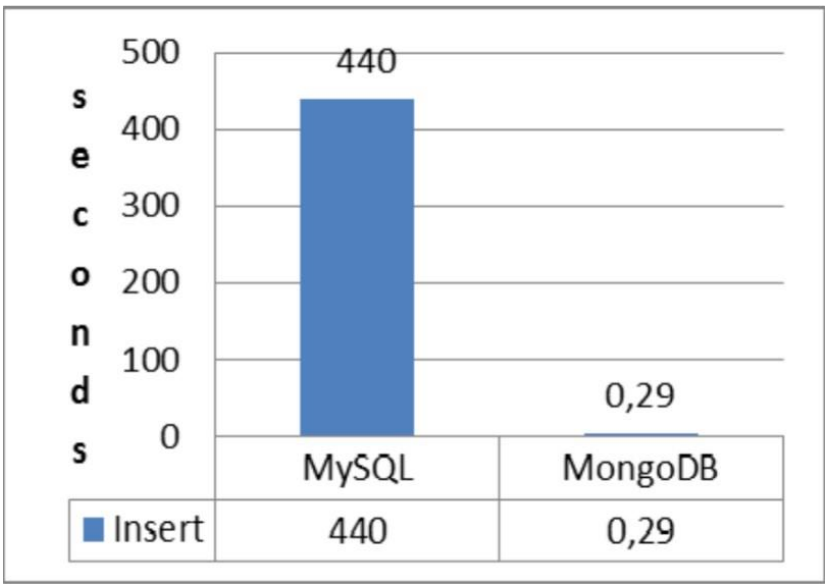


Fig 5 : MySQL vs MongoDB insert

Figure 5 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.

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.

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.

MEAN Stack : AngularJS

AngularJS is a JavaScript framework for building front-end of web applications. It is designed to build a Single Page Application with the introduction of MVC (Model View Controller) architecture to the front-end.AngularJS framework extends HTML functionalities by providing different elements and attributes, which help to build large web application with ease. It extends HTML with `ng-directives`, binds the HTML input controls to the application data with `ng-app` directive, and binds the application data to the view with `ng-bind` directive. A simple HTML template with AngularJS is shown in listing 3.

```
<html lang="en-US">

<script

src="http://ajax.googleapis.com/ajax/libs/a
ngularjs/1.4.8/angular.min.js"></script>

<body>

<div ng-app="">

 <p>Name: <input type="text" ng-model="name"></p>

 <p ng-bind="name"></p>
```

```
</div>

</body>

</html>
```

Listing 3. AngularJS example

In listing 3, AngularJS is added to the page with a `<script>` tag so AngularJS starts immediately when the page is loaded. The actual implementation starts inside the `<div>` container where AngularJS is initialised by `ng-app` directive. The `ng-model` directive then binds the value of input to the variable 'name', and finally `ng-bind` directive binds `<p>` element's innerHTML to the variable 'name'. Hence, the name typed by the user in the input field is displayed inside the `<p>` element. It is common practice to use AngularJS expression instead of `ng-bind` directive. For example in the above case, `<p ngbind="name"></p>` can be replaced with `<p>{{name}}</p>`.

One of the important features of AngularJS is its two-way data binding mechanism. The two-way data binding feature allows synchronization of data between the model and the view. Model refers to some JavaScript variables, and view is the HTML container where the model data is displayed. Whenever data in the model changes, the view will be updated immediately, and whenever the view component changes, the model data will be updated as well. This feature eliminates the need of tedious DOM manipulation in order to find and change the data in the DOM tree.

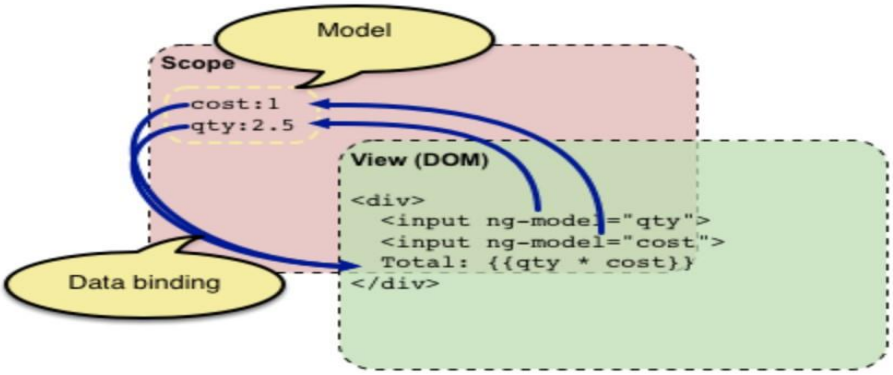


Fig 6 : Two-way data binding in AngularJS

Figure 6 illustrates the two-way data binding feature of an AngularJS application. The view contains two text box inputs: cost and qty with ng-model attributes attached to them. The ng-model binds the input values to the scope that holds the model data. In one-way data binding, data is bound in only one direction. Once the view is rendered, any changes in the model data or view components are not automatically reflected in the view. If a user changes the values of cost and qty to get a new total, the total will not be automatically updated in the view. In AngularJS application, if the user changes the values of cost and qty, the new total will be displayed immediately in the view because of the live view provided by the two-way data binding.

Another important feature of the AngularJS is the introduction of MVC architecture on the front-end. In front-end web development, view and controller are generally placed in one place. Because of the automatic synchronization of model and view in AngularJS, controller can be kept outside of the view as shown in figure 7.

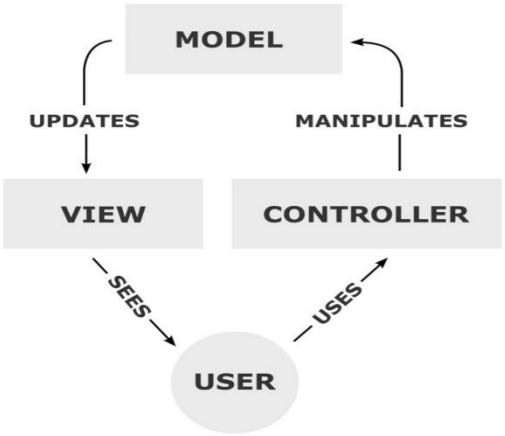


Fig 7 : MVC architecture in AngularJS

Figure 7 illustrates the MVC architecture in AngularJS. The model defines the application at a data layer, the view provides the visual presentation, and the controller manipulates the model data. Due to the two-way data binding between the model and the view, the controller focuses only on the model data, and the view reflects back any changes done in the controller from the model.

AngularJS was created with the concept of SPAs from its early days of development, so it provides excellent routing support to create SPAs. One can create multiple views for multiple URLs and AngularJS loads the appropriate view in the main view according to the URL requested. Since there are no redirects, AngularJS takes a minimal amount of time to load the pages. This feature provides good user experience.

AngularJS applications are also embeddable, testable, and injectable. They can easily be embedded with other applications and work perfectly with other technologies. Dependencies are automatically handled by the AngularJS injector subsystem by using different services and factory methods. This allows to create loosely coupled application, so individual components of the application can be tested in isolation.

PROJECT DESCRIPTION

This is a web application by which any user can manage tasks by creating his account. This project has various small parts like creating personal account, creating new lists, creating new tasks, editing the lists, editing the tasks, deleting the lists, deleting the tasks, etc.

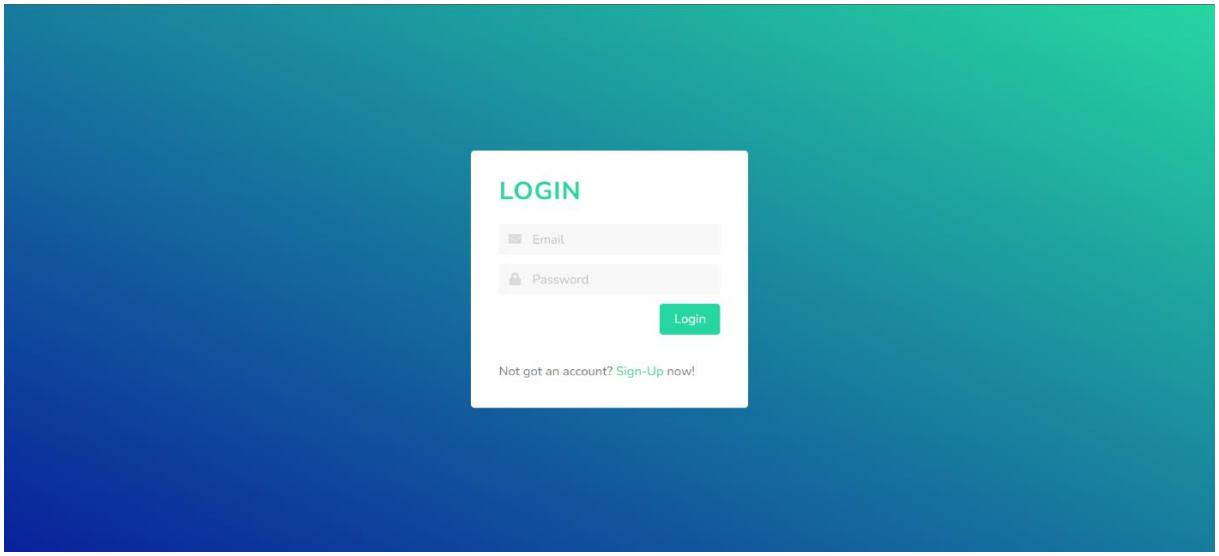
It is very useful for daily routine task management in order to manage time and work life balance.

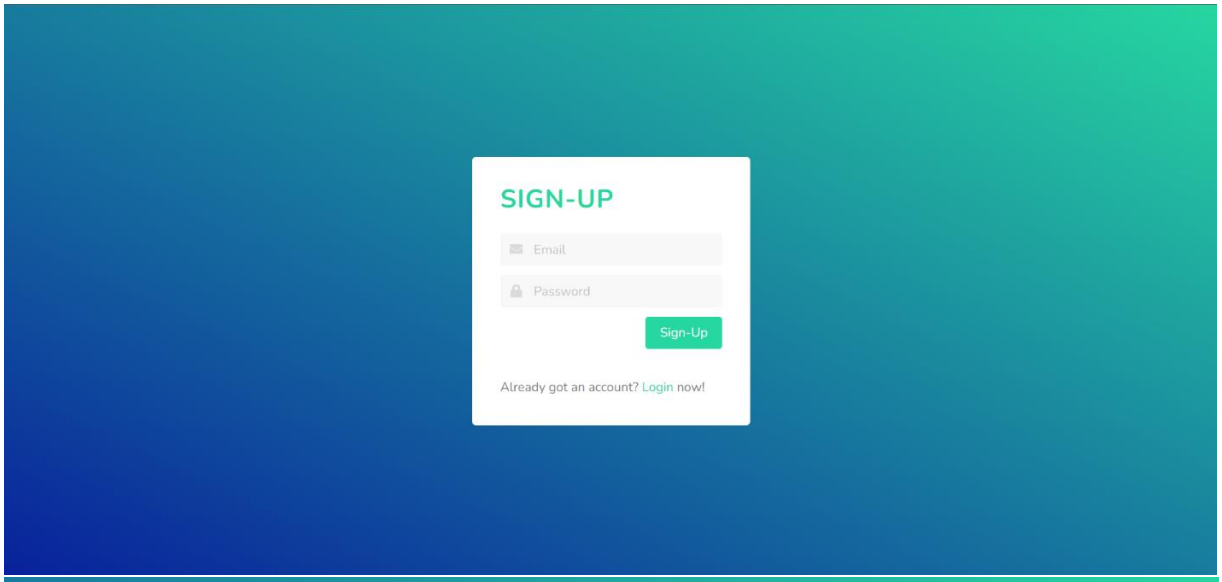
This is a Full Stack Development(MEAN) project. In this project, I have used different technologies like MongoDB, Expess, AngularJS, Node.JS for implementation of this project.

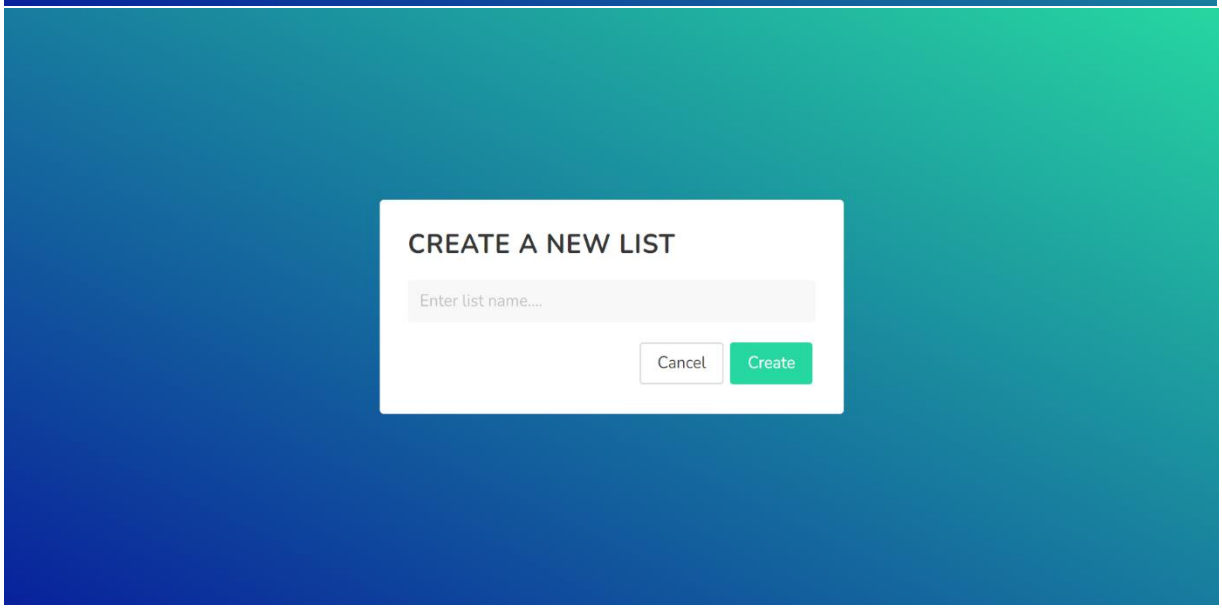
Project Aim :

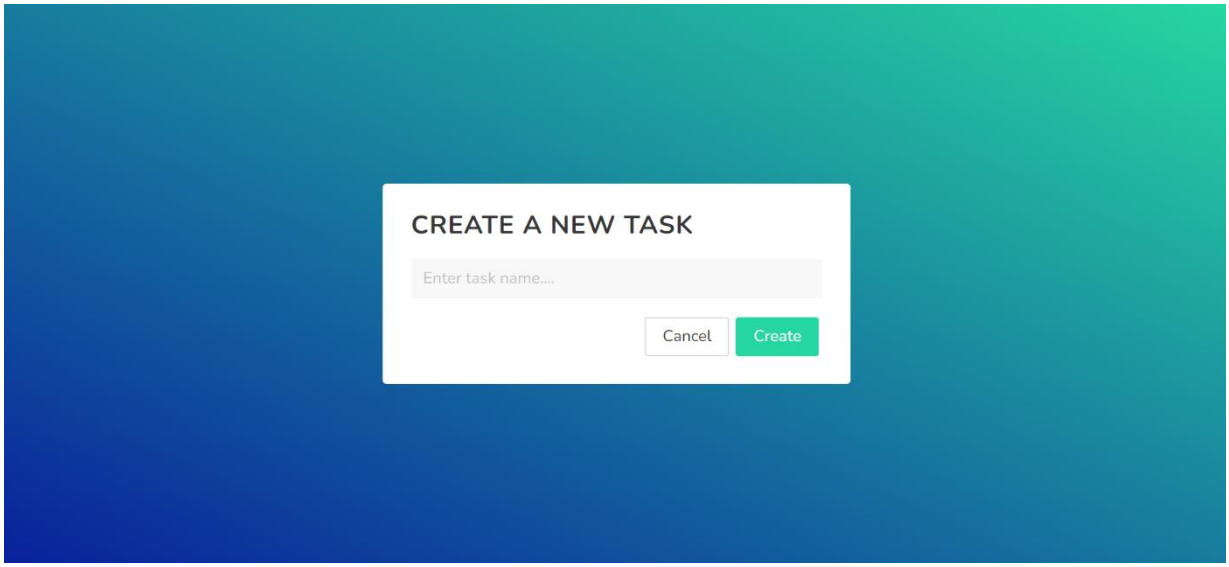
- It satisfy the user requirements
- Easy to understand by the user
- Easy to operate and manage tasks
- Have a Good User Interface
- Expandable

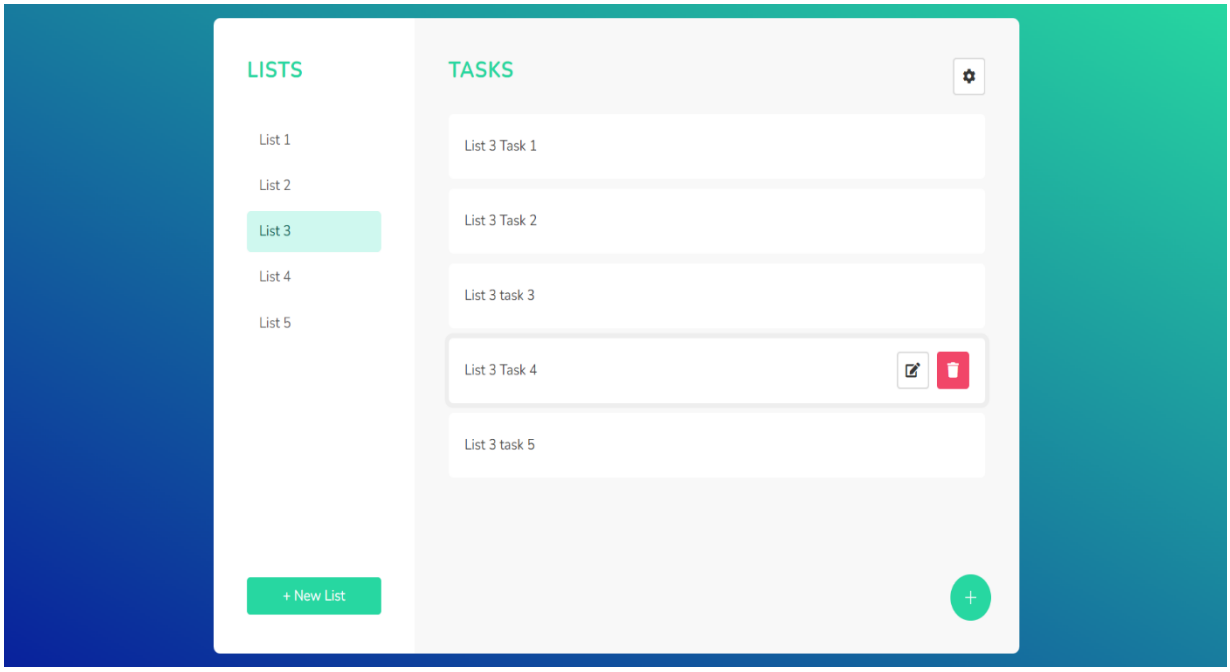
SNAPSHOT OF THE PROJECT

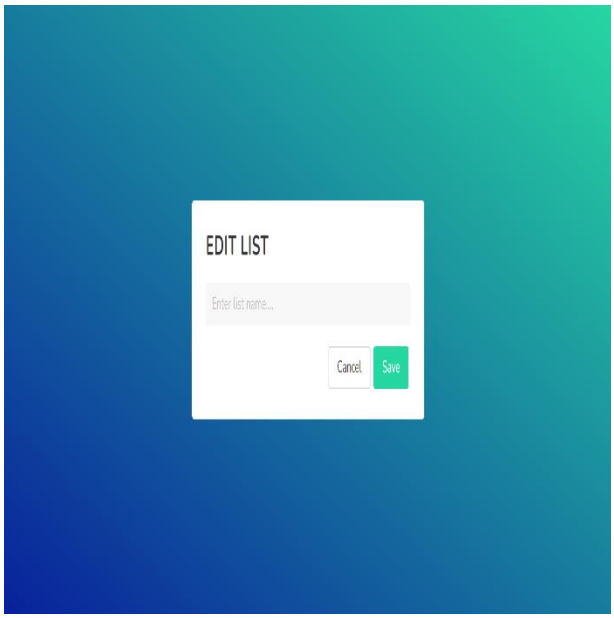


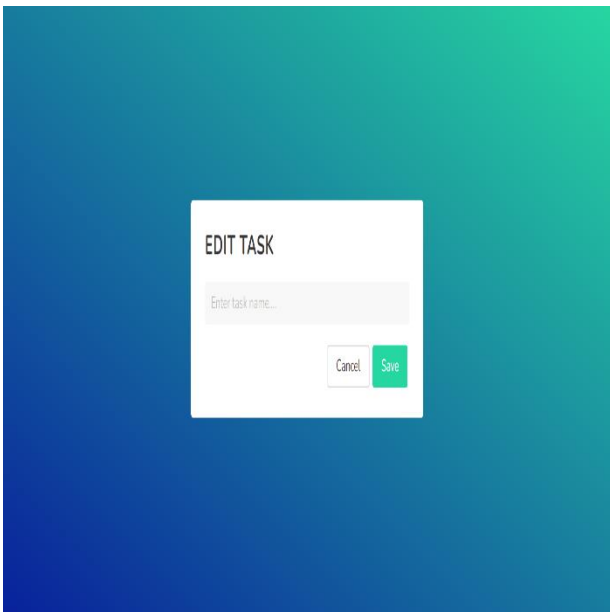





CREATE A NEW TASK
Enter task name....
Cancel
Create


LISTS
List 1
List 2
List 3
List 4
List 5
+ New List
TASKS
List 3 Task 1
List 3 Task 2
List 3 task 3
List 3 Task 4
List 3 task 5


EDIT LIST
Enter list name....
Cancel
Save


EDIT TASK
Enter task name....
Cancel
Save

REFERENCE

- **SEARCH ENGINE** : GOOGLE CHROME
  - GEEKS FOR GEEKS
  - INTERVIEWBITS
  - JAVATPOINT
  - PROGRAMIZ

- **ONLINE PLATFORM** : YOUTUBE, E-BOOKS

CONCLUSION

Here, I have come to the end Seminar report on the Full Stack Development(MEAN). I tried my best to include all the necessary points that I have learnt during the summer training. Some of the information I have referred to some books. This Summer training report contains information of for developing the new technologies tool and to enhance our coding skills . It will be interesting and may be even knowledgeable.