**Develop Full-Stack web application using MERN Stack -** [Ravindu Wijerathne](https://medium.com/@ravinduwijerathne629?source=post_page-----47538c48458f--------------------------------) May 15, 2022

MERN stack is a JavaScript based web development stack that is used for easier and faster develop web applications. MERN stack includes 4 main technologies there are Mongo DB, Express, React, and Node. Mongo DB is NoSQL based database that we can use to store our data easily. In this project, we are using the MongoDB Atlas version which is an online version of MongoDB. Mainly, we are importing a package called mongoose to our project to easily handle our database operation. When we are developing full-stack web applications using MERN stack we use node js for the backend and react js for the frontend. Node js was made to basically liberate JavaScript from the browser and allow programmers to use JavaScript o interact directly with the hardware of a computer. Express is a node framework, and it gives lots of functionalities for organize and structure our code. Express specially build for write less repetitive code when you are building web applications and it’s specially made for web developers. For the frontend part, we use React. React is a JavaScript library for building user interfaces. React works by creating JSX files. JSX files are files where we have got HTML right in the body of a JavaScript file.

We used Visual Studio Code as our development IDE. Visual Studio Code is a code editor redefined and optimized for building and debugging modern web and cloud applications.

First, we have to create our backend server. We imported the project folder to the vs code and type the command **npm init** to initialize npm. Npm is the package manager for external modules. After initialize npm it automatically generates a file called package.json. In that file, we can see all the packages that we installed under the dependencies.

After that, we installed some npm modules. They are cors, express, mongoose, nodemon, concurrently and dotenv.

**CORS**

CORS is a node.js package for providing a Connect/Express middleware that can be used to enable CORS with various options.

**npm i cors**

**Express**

Express is Fast, unopinionated, minimalist web framework for node.

**npm i express**

**Mongoose**

Mongoose provides a straight-forward, schema-based solution to model your application data.

**npm i mongoose**

**Nodemon**

Nodemon is a utility depended on by over 1.5 million projects, that will monitor for any changes in your source and automatically restart your server.

**npm i nodemon**

**package.json file**

Text

Description automatically generated

After that, we can create file called server.js and import necessary packages. we imported mongoose, cors, and express packages. then called the connect method comes with mongoose package and pass the relevant uri and options to that method.

We separately create file called .env and that file contains MongoDB URI like secret pieces of information. That file also we have imported into server.js file. After import express package we used listen(), use() like methods, and specify the specific port number for our server.

**server.js file**

Text

Description automatically generated

Then we created a folder called model inside our backend directory. Inside the model folder, we can create files for include necessary schemas.

**Example (create a schema for staff members)**

Require the mongoose package and create a new schema relevant to staff information and finally, we have to export that schema because the controllers related to this schema need to import this.

Text

Description automatically generated

Then we created another folder called controllers inside our backend directory to add our controllers. In this folder, we created the necessary files relevant to each section of our app. For example staff part we create a file called staff.controller.js inside the controller folder. That file includes all the methods related to the staff members like add, delete, update, get, and validate staff members. For those methods we have to connect to the database because of that we imported the previously exported staff model inside this staff.controller.js file. Like in the model file we have to export the relevant method inside this controller file.

**Staff.controller.js file**

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Next, we created the routes folder inside the backend directory, and we create the necessary routes files inside this folder. For example, we created staff.route.js file inside the routes folder and firstly imported the necessary methods related to staff which was exported from the staff.controller.js file.

Then we can map those imported methods with the necessary get, post, put, and delete requests routes. After that When necessary requests come to these routes it correctly maps the relevant method inside the controller.

We have to export these routes and in the server,js file we specify the relevant route inside app.use() method.

**Staff.route.js file**

Text

Description automatically generated

Then we can move to the frontend part. First, we need to create a folder for the fronend directory. Then we can go inside that folder and we can install necessary packages.

React is a JavaScript library for building user interfaces.

**npm i react**

React package for working with the DOM.

**npm i react-dom**

Blazing fast, zero configuration web application bundler.

**npm i parcel**

React Router DOM is an npm package that enables you to implement dynamic routing in a web app.

**npm i react-router-dom**

Promise based HTTP client for the browser and node.js.

**npm i axios**

Text

Description automatically generated

Then create index.html file and specify id=”root” inside div tag and link index.js file and necessary css files.

Text

Description automatically generated

Create index,jsx file and import necessary packages and get the id “root” and render relevant components to that targeted root element.

A screenshot of a computer screen

Description automatically generated with medium confidence

After that we created necessary components. We used functional components to develop our frontend.

Using React Hooks and props we managed to handle our components according to our requirements.

With help of axios package, we send necessary http requests to the backend to manage our data with the database.

Add relevant routes for the components inside App.js file

Text

Description automatically generated

**AllUsers component code and user interface**

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

7

7