MERN Stack

MongoDB : A NoSQL database that stores data in flexible, JSON like documents . MongoDB is known for its scalability and ease of use.

Express.js : A web application framework for Node.js , designed for building web applications and APIs . It simplifies the development of server side code.

React : A javascript library for building user interfaces , particularly single page applications . React allows developers to create reusable UI components.

Node.js : A javascript runtime built on Chromes engine . It enables Server side scripting with javascript, making it possible to use javascript for both frontend and backend development .

Role of Each

MongoDB :

Role : Database

Advantages : Scalability , high performance , flexible schema , and rich query capabilities .

Express.js :

Role : Backend Framework

Advantages : Simplifies server side development , lightweight , and ~~&~~ integrates seamlessly with Node.js .

React :

Role : Frontend Library

Advantages : Efficient rendering with a virtual DOM, reusable components, and a large ecosystem of tools and libraries .

Node.js :

Role : Server Environment

Advantages : Non - blocking , event driven architecture , high performance and a vast ecosystem of modules .

1. Setting up the Backend

1. Initialize a new Node.js project

2. Install Express.js and other dependencies

```
npm install express mongoose cors
```

3. Create basic server

```
Const express = require ('express');
Const mongoose = require ('mongoose');
Const cors = require ('cors');
Const item = require ('./items');
Const app = express ();

app. use (cors ());
~~app~~
app. use ( express . json());
app. use ( '/items', item);
mongoose . Connect ( 'mongodb :// localhost : 27017/myapp')
app. get ( '/', (req, res) => {
        res . send ('Hello, MERN Stack');
});
```

4. Create MongoDB Schema

```
const mongoose = require('mongoose');
const ItemSchema = new mongoose.Schema({
    name: {
        type: String,
        required: true,
    },
});
module.exports = mongoose.model('Item', ItemSchema);
```

5. Set up Routes

```
Const express = require('express');
Const router = express.Router();
Const Item = require('./models/Item);
router.get('/', (req, res) => {
try {
    const items = await Item.find();
    res.json(items);
}
catch (err){
    res.json({ message: err.message });
});
module.exports = router;
```

2. Setting up Frontend

1. Create react application

```
npx create-react-app my-app
```

2. Install Axios for HTTP request
      npm install axios

3. Create a Component to display

```
import React, { ~~useEff~~ useEffect, useState } from 'react';
import axios from 'axios';
const ItemList = () => {
        const [items, setItems] = useState([]);

    useEffect(() => {
        axios.get('http://localhost:5000/items')
        .then(response => setItems(response.data))
        .catch(err => console.log(err));
    }, []);

    return ( <div>
            <h1> Items </h1>
            <ul>
              { items.map(item => (
                    <li key={item._id}>
                        {item.name} </li> ))}
              </ul>
              </div>
        );
    };
    export default ItemsList;
```

4. Add the component to App:

```
const App = () => { return (<div className="App">
        <ItemsList /> </div>
      );
    };
  export default App;
```