

MOOC COURSE SUMMER INTERNSHIP

on

(MERN STACK BOOTCAMP)

Submitted by

**Name:-SARVESH KUMAR
RegistrationNO:1210310
Program Name:-MCA**

**School of Computer Application
Lovely Professional University, Phagwara**

(June – July 2021)

Acknowledgment

The MOOC Course Summer Internship opportunity I had with **LearnCodeOnline** was a great chance for learning and professional development. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it. I am also grateful for having a chance to learn from professionals who led me through this internship period.

I express my deepest thanks to Training and Placement Coordinator, School of Computer Application, Lovely Professional University for allowing me to grab this opportunity. I choose this moment to acknowledge his contribution gratefully by giving necessary advices and guidance to make my internship a good learning experience.

[Student Name-Sarvesh kumar]

[Registration No-1210310]

Internship Certificate
(As given by MOOC or Organization in original)



Table of contents

Sr. No.	Description	Page No.
1	Introduction of the Course	4-9
2	Technical Learnings from the course	10-14
3	Introduction of Mini Project or Technical Assignments	15-16
4	Details of Mini Project	15-29
4.1	Interfaces Designed (If any)	15-29
4.2	Code snippets (If any)	16-29
5	Grade sheet of assignments/ marks card from the MOOC	30
6	Bibliography or References	31

Introduction of the Course

A group of technologies known as the MERN stack make it possible to construct applications more quickly. Worldwide, developers use it. The fundamental goal of the MERN stack is to create apps that solely use JavaScript. This is due to the fact that all four of the technologies that comprise the technology stack are JS-based. Therefore, the backend, frontend, and database may be operated simply if one is familiar with JavaScript (and JSON).

MERN Stack Full Form

MERN Stack is a compilation of four different technologies that work together to develop dynamic web apps and websites.

It is a contraction for four different technologies as mentioned below:

M - MongoDB

E – Express.JS

R – React.JS

N – Node.JS

Considering you've downloaded and configured all the four aforementioned technologies, you need to know how to create a new project folder to get started with the MERN stack. Next, you need to open the folder on the CMD (or terminal) and enter the following command to initialize a Package.json file:

npm init

```
C:\Windows\System32\cmd.exe

C:\Users\JasRaj\Desktop\expressor>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

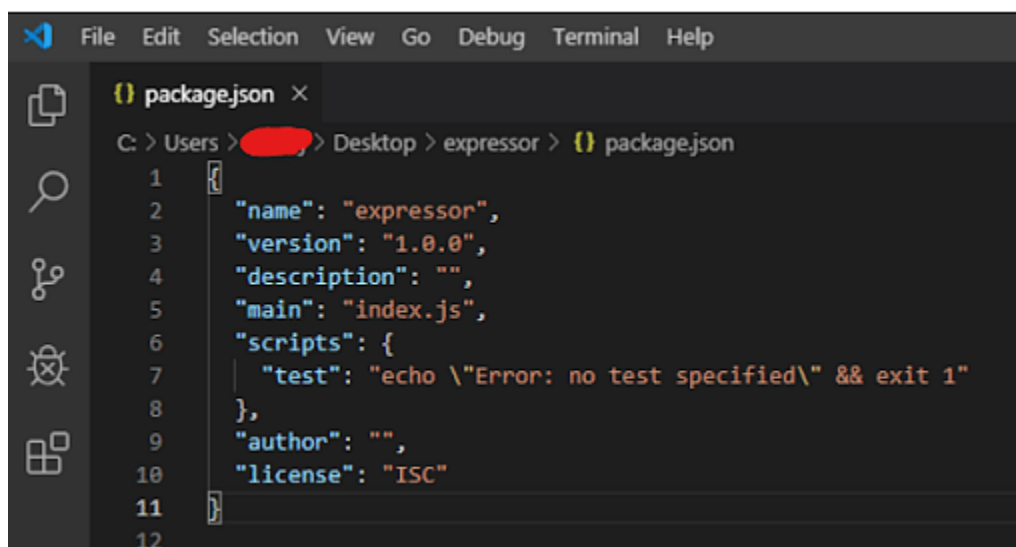
Press ^C at any time to quit.
package name: (expressor)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\JasRaj\Desktop\expressor\package.json:

{
  "name": "expressor",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this ok? (yes)

C:\Users\JasRaj\Desktop\expressor>
```

A standard JSON package looks as shown below:



```
File Edit Selection View Go Debug Terminal Help

package.json x
C:\Users\JasRaj\Desktop\expressor> {} package.json
1 {
2   "name": "expressor",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1"
8   },
9   "author": "",
10  "license": "ISC"
11 }
12
```

we can install modules by using:
`npm install module_name -save`

MERN Stack Components

There are four components of the MERN stack. Let's discuss each of them one by one.

- The first component is MongoDB, which is a [NoSQL](#) database management system.
- The second MERN stack component is ExpressJS. It is a backend web application framework for NodeJS.
- The third component is ReactJS, a JavaScript library for developing UIs based on UI components.
- The final component of the MERN stack is NodeJS. It is a JS runtime environment, i.e., it enables running JavaScript code outside the browser.

About these MERN Stack components.

➤ MongoDB

MongoDB is a NoSQL [DBMS](#) where data is stored in the form of documents having key-value pairs similar to JSON objects. MongoDB enables users to create databases, schemas, and tables. It offers the Mongo shell that provides a JS interface for deleting, querying, and updating the records.

➤ ExpressJS

ExpressJS is a NodeJS framework that simplifies writing the backend code. It saves you from creating multiple Node modules. For keeping the code precise, ExpressJS offers a range of middleware.

➤ **ReactJS**

ReactJS is a JS library that allows the development of user interfaces for mobile apps and SPAs. It allows you to code JavaScript and develop UI components. The JS library uses virtual DOM for doing everything.

➤ **NodeJS**

NodeJS is an open-source JavaScript runtime environment that allows users to run code on the server. It comes with the node package manager or npm, enabling users to select from a wide selection of node modules or packages. Being developed on the Chrome JavaScript Engine enables Node to execute code faster.

Why Should You Work With MERN Stack?

There are many good reasons to use the MERN Stack. For example, it allows the creation of a 3-tier architecture that includes frontend, backend, and database using JavaScript and JSON.

MongoDB, which is the base of the MERN stack, is designed to store JSON data natively. Everything in it, including CLI and query language, is built using JSON and JS. The NoSQL database management system works well with NodeJS and thus, allows manipulating, representing, and storing JSON data at every tier of the application.

It comes in a variant called MongoDB Atlas that further eases database management by offering an auto-scaling MongoDB cluster on any cloud provider and with just a few clicks.

Express is a server-side framework that wraps HTTP requests and responses and makes mapping URLs to server-side functions easy. This perfectly complements the ReactJS

framework, a front-end JS framework for developing interactive UIs in [HTML](#) while communicating with the server.

As the two technologies work with JSON, data flows seamlessly, making it possible to develop fast and debug easily. To make sense of the entire system, you need to understand only one language, i.e., JavaScript and the JSON document structure.

Use Cases of MERN

Like other popular web stacks, it is possible to develop whatever you want in MERN. Nonetheless, it is ideal for cloud-based projects where you require intensive JSON and dynamic web interfaces. A few examples of purposes where MERN is used are:

- **Calendars and To-do Apps**

A calendar or a to-do app is a rudimentary project that can tell you a lot about the mechanics of the MERN stack. You can design the frontend, i.e., the interface of the calendar or to-do app using ReactJS. The data to be stored, accessed, modified, shown in the to-do app is made possible using MongoDB.

- **Interactive Forums**

Another suitable use case for MERN is an interactive forum, which can be a social media platform or a website that allows users to share messages and communicate. The topic of the interactive forum may or may not be predefined.

- **Social Media Product**

An interactive forum is just one use of the MERN stack for social media. These include ads, posts, a mini web app embedded in the social media page, etc.

Conclusion

Contemporary web developers with enough experience in React and JS prefer developing intuitive applications using the MERN stack. To know in-depth about using the MERN stack, you can opt for the [Post-Graduate Program In Full-Stack Web Development](#) by Simplilearn. Offered by Caltech CTME, Full-stack Web Development Program is a descriptive online bootcamp that includes 25 projects, a capstone project, and interactive online classes. In addition to the MERN stack, the course also details everything you need to become a full-stack developer.

Technical Learnings from the course

Following are the technical learning from this course.

Frontend-React

Backend-Node/Express

Database-MongoDB

Version control-Git

Operating systems-Linux/mac/windows

FRONT-END

REACTJS

ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front end library responsible only for the view layer of the application. It was created by **Jordan Walke**, who was a software engineer at **Facebook**. It was initially developed and maintained by Facebook and was later used in its products like **WhatsApp** & **Instagram**. Facebook developed ReactJS in **2011** in its newsfeed section, but it was released to the public in the month of **May 2013**. Today, most of the websites are built using MVC (model view controller) architecture. In MVC architecture, React is the 'V' which stands for view, whereas the architecture is provided by the Redux or Flux.

A ReactJS application is made up of multiple components, each component responsible for outputting a small, reusable piece of HTML code. The components are the heart of all React applications. These Components can be nested with other components to allow complex applications to be built of simple building blocks. ReactJS uses virtual DOM based mechanism to fill data in HTML DOM. The virtual DOM works fast as it only changes individual DOM elements instead of reloading complete DOM every time.

To create React app, we write React components that correspond to various elements. We organize these components inside higher level components which define the

application structure. For example, we take a form that consists of many elements like input fields, labels, or buttons. We can write each element of the form as React components, and then we combine it into a higher-level component, i.e., the form component itself. The form components would specify the structure of the form along with elements inside of it.

Why learn ReactJS?

Today, many JavaScript frameworks are available in the market (like angular, node), but still, React came into the market and gained popularity amongst them. The previous frameworks follow the traditional data flow structure, which uses the DOM (Document Object Model). DOM is an object which is created by the browser each time a web page is loaded. It dynamically adds or removes the data at the back end and when any modifications were done, then each time a new DOM is created for the same page. This repeated creation of DOM makes unnecessary memory wastage and reduces the performance of the application.

BACKEND

NODEJS

Node.js is a cross-platform runtime environment and library for running JavaScript applications outside the browser. It is used for creating server-side and networking web applications. It is open source and free to use. Many of the basic modules of Node.js are written in JavaScript. Node.js is mostly used to run real-time server applications.

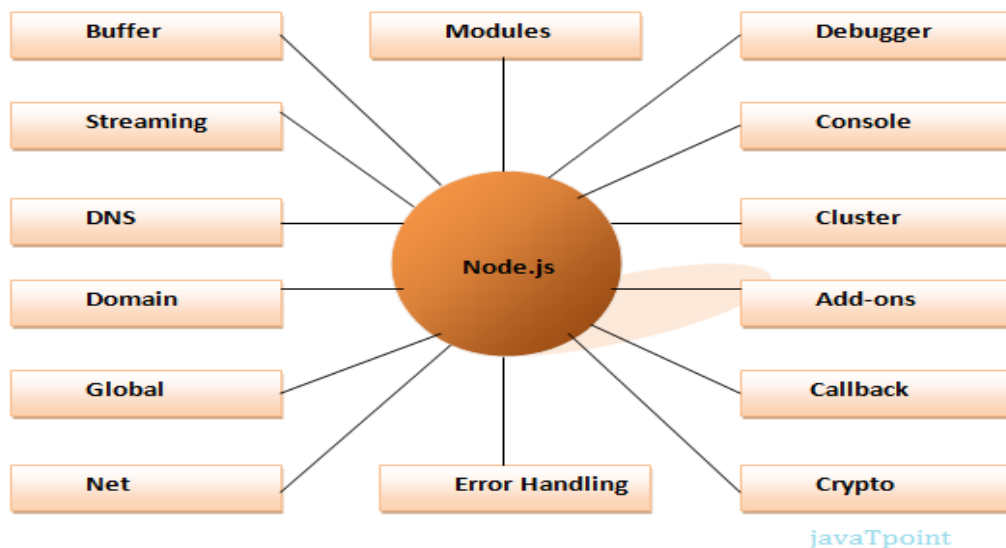
The definition given by its official documentation is as follows:

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.?

Node.js also provides a rich library of various JavaScript modules to simplify the development of web applications.

1. **Node.js** = **Runtime** Environment + JavaScript Library

The following diagram specifies some important parts of Node.js:



Features of Node.js

Following is a list of some important features of Node.js that makes it the first choice of software architects.

1. **Extremely fast:** Node.js is built on Google Chrome's V8 JavaScript Engine, so its library is very fast in code execution.
2. **I/O is Asynchronous and Event Driven:** All APIs of Node.js library are asynchronous i.e. non-blocking. So a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call. It is also a reason that it is very fast.
3. **Single threaded:** Node.js follows a single threaded model with event looping.
4. **Highly Scalable:** Node.js is highly scalable because event mechanism helps the server to respond in a non-blocking way.
5. **No buffering:** Node.js cuts down the overall processing time while uploading audio and video files. Node.js applications never buffer any data. These applications simply output the data in chunks.

6. **Open source:** Node.js has an open source community which has produced many excellent modules to add additional capabilities to Node.js applications.
7. **License:** Node.js is released under the MIT license.

EXPRESSJS

Express is a fast, assertive, essential and moderate web framework of Node.js. You can assume express as a layer built on the top of the Node.js that helps manage a server and routes. It provides a robust set of features to develop web and mobile applications.

Let's see some of the core features of Express framework:

- It can be used to design single-page, multi-page and hybrid web applications.
- It allows to setup middlewares to respond to HTTP Requests.
- It defines a routing table which is used to perform different actions based on HTTP method and URL.
- It allows to dynamically render HTML Pages based on passing arguments to templates.

Why use Express

- Ultra fast I/O
- Asynchronous and single threaded
- MVC like structure
- Robust API makes routing easy

DATABASE

MongoDB

MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++. This tutorial will give you great understanding on MongoDB concepts needed to create and deploy a highly scalable and performance-oriented

database

MongoDB provides high availability with replica sets. A replica set consists of two or more copies of the data. Each replica-set member may act in the role of primary or secondary replica at any time. All writes and reads are done on the primary replica by default. Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically conducts an election process to determine which secondary should become the primary. Secondaries can optionally serve read operations, but that data is only eventually consistent by default.

If the replicated MongoDB deployment only has a single secondary member, a separate **daemon** called an *arbiter* must be added to the set. It has a single responsibility, which is to resolve the election of the new primary.¹ As a consequence, an idealized distributed MongoDB deployment requires at least three separate servers, even in the case of just one primary and one secondary.

VERSION CONTROL

GIT

Git tutorial provides basic and advanced concepts of Git and GitHub. Our Git tutorial is designed for beginners and professionals.

Git is a modern and widely used **distributed version control** system in the world. It is developed to manage projects with high speed and efficiency. The version control system allows us to monitor and work together. **Git** is an **open-source distributed version control system**. It is designed to handle minor to major projects with high speed and efficiency. It is developed to co-ordinate the work among the developers. The version control allows us to track and work together with our team members at the same workspace. Git is the foundation of many services like **GitHub** and **GitLab**, but we can use Git without using any other Git services. Git can be used **privately** and **publicly**. Git was created by **Linus Torvalds** in **2005** to develop Linux Kernel. It is also used as an important distributed version-control tool for **the DevOps**. Git is easy to learn, and has fast performance. It is superior to other SCM tools like Subversion, CVS, Perforce, and ClearCase.

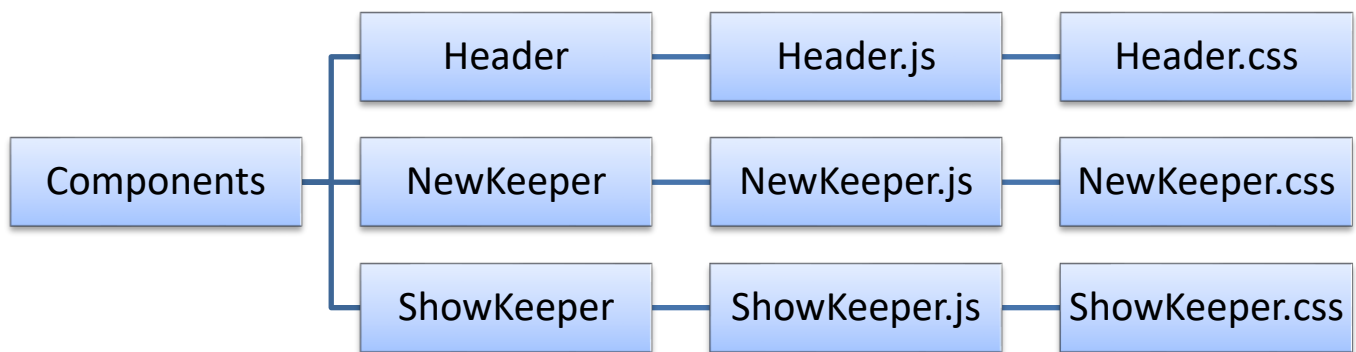
Introduction of Mini Project or Technical Assignments (Notes-Keeper-App)

This project **Notes-Keeper-App** has been developed on Mern Stack The main objective for developing this project was to keep your daily note and plans online. This project helps in daily Routines.

Details of Mini Project

The objective of the Project on **Notes-Keeper-app**:

Mini project flow chats



- **Header:-** Header components for header where our miniproject name reflected.
- **NewKeeper:-** In NewKeeper that is a input form where two input field and one add button are there one input field for title and another one for description, where user can first give the title and descriptions and click to add.
- **ShowKeeper:-** In Showkeeper that input user can add in the Newkeeper that reflect in the black field.

Technology Used in the project The-Note-keeper

React JS: The User Interface has been designed in React JS

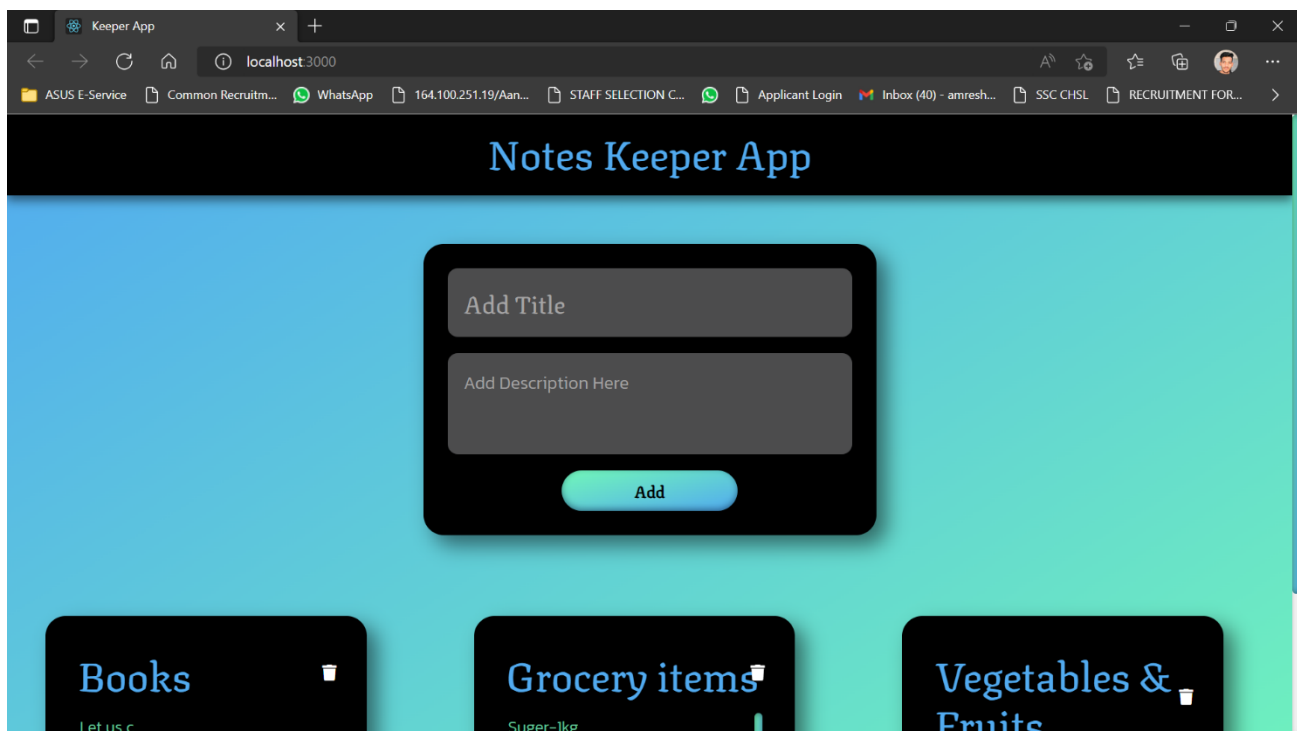
Node Js: used for developing the premier JavaScript web server

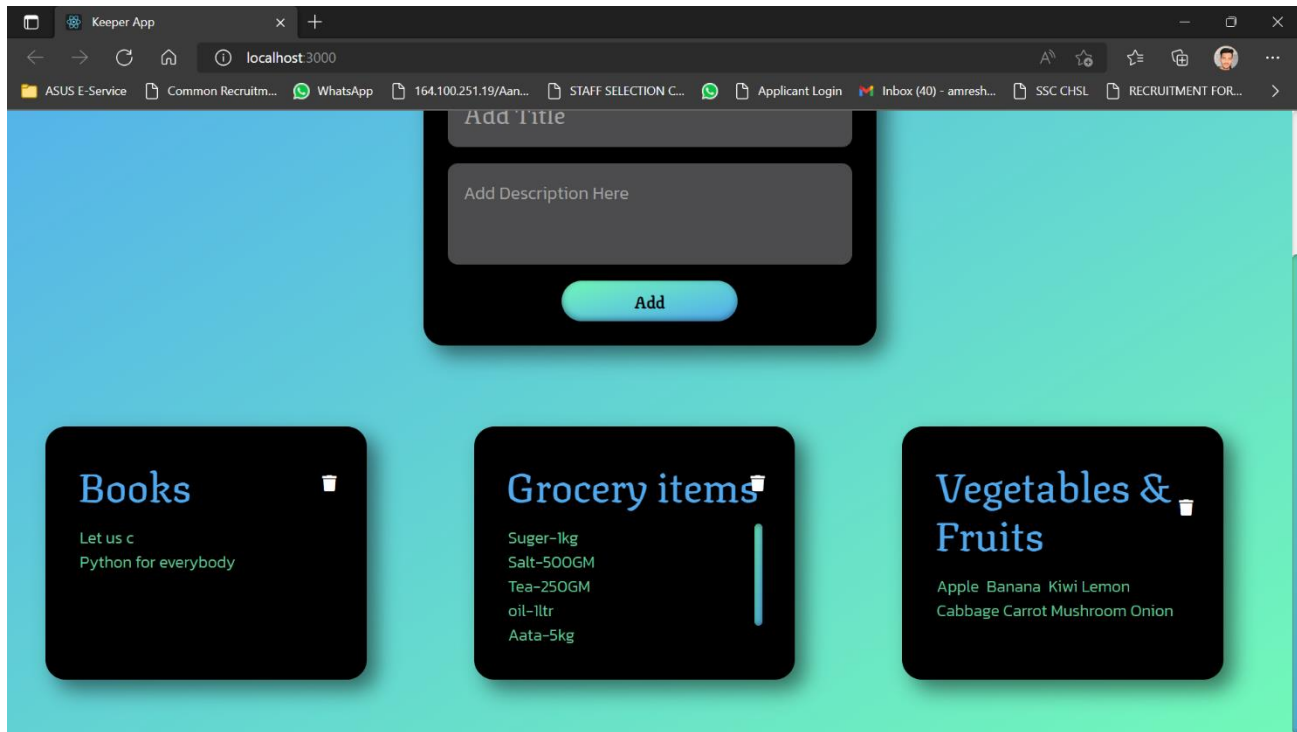
Express Js: used for developing Node.js web framework

MongoDB: the database has been used as a database for the project

DataBase:- For database we are using mongoDB that's can run on localhost:27017 that can connect the frontend part to the database.

Interfaces Designed





Code snippets

For front-end:-

Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Keeper App</title>
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link
href="https://fonts.googleapis.com/css2?family=Kanit:wght@300&family=Texturina&
display=swap" rel="stylesheet">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.css"
/>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css" integrity="sha512-
iBBXm8fw90+nuLcSK1bmrPcLa0OT92x01BIsZ+ywDWZCvqswGccV3gForBv0z+8dLJgyAHlR35VZc2
oM/gI1w==" crossorigin="anonymous" />
  </head>
```

```

<body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
</body>
</html>

```

Src-->App.js

```

import './App.css'
import Header from './components/header/header'
import AddKeeper from './components/addkeeper/addkeeper'
import ShowKeeper from './components/showkeeper/showkeeper'
import { useState, useEffect } from 'react'
import axios from 'axios'

function App() {

  const [ keeperList, setKeeperList ] = useState([])

  useEffect(() => {
    axios.get("http://localhost:3001/api/getAll")
      .then(res => setKeeperList(res.data))
  }, [])

  return (
    <div className="App">
      <Header />
      <AddKeeper keeperList={keeperList} setKeeperList={setKeeperList} />
      <ShowKeeper keeperList={keeperList} setKeeperList={setKeeperList} />
    </div>
  );
}

export default App;

```

App.css

```

.App {
  min-height: 800px;
  background: linear-gradient( to bottom right, rgb(82, 171, 240) , rgb(114, 248, 185) );
}

```

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';

ReactDOM.render(
  <App />,
  document.getElementById('root')
)
```

Index.css

```
* {
  margin: 0;
}

/* modified scrollbar scrollbar*/
*::-webkit-scrollbar {
  width: 8px;
}

*::-webkit-scrollbar-track {
  -webkit-box-shadow: inset 0 0 6px rgba(0,0,0,0.3);
  border-radius: 10px;
}

*::-webkit-scrollbar-thumb {
  border-radius: 10px;
  -webkit-box-shadow: inset 0 0 6px rgba(0,0,0,0.5);
  background: linear-gradient( to bottom right, rgb(114, 248, 185) , rgb(82, 171, 240) );
}
```

Src→components→addkeeper

Addkeeper.js

```
import React, { useState } from "react"
import "./addkeeper.css"
import axios from "axios"

const AddKeeper = ({ setKeeperList }) => {

  const [keeperObj, setKeeperObj] = useState({
    title: "",
    description: ""
  })

  const handleChange = e => {
    const { name, value } = e.target
    setKeeperObj({
      ...keeperObj,
      [name]: value
    })
  }

  const add = () => {
    if(keeperObj.title) {
      axios.post("http://localhost:3001/api/addNew", keeperObj)
        .then(res => setKeeperList(res.data))
      setKeeperObj({
        title: "",
        description: ""
      })
    }
  }

  return (
    <div className="addKeeper">
      <input
        className="inputBox titleInput"
        type="text"
        name="title"
        autoComplete="off"
        placeholder="Add Title"
        onChange={handleChange}
        value={keeperObj.title}
      />
    </div>
  )
}
```

```

        />
        <textarea
            className="inputBox description"
            name="description"
            placeholder="Add Description Here"
            onChange={handleChange}
            value={keeperObj.description}
        />
        <div className="addButton" onClick={add}>Add</div>
    </div>
)
}

export default AddKeeper

```

addkeeper.css

```

.addKeeper {
    display: flex;
    flex-direction: column;
    background: rgb(0, 0, 0);
    padding: 1rem;
    border-radius: 20px;
    margin: 3rem auto;
    width: fit-content;
    box-shadow: rgba(0, 0, 0, 0.5) 10px 10px 20px;
}

.inputBox {
    border: none;
    border-radius: 10px;
    margin: 0.5rem;
    padding: 1rem;
    background: rgb(76, 76, 77);
    outline: none;
}

.inputBox::placeholder {
    color: rgb(167, 166, 166);
}

.titleInput {
    font-size: 1.5rem;
}

```

```

    color: rgb(82, 171, 240);
    font-family: 'Texturina', serif;
}

.description {
    font-size: 1.1rem;
    font-family: 'Kanit', sans-serif;
    min-height: 100px;
    min-width: 400px;
    color: rgb(102, 201, 156);
    resize: none;
}

.addButton {
    color: black;
    font-weight: bold;
    background: linear-gradient(to bottom right, rgb(114, 248, 185) , rgb(82, 171, 240) );
    margin: 0.5rem auto;
    padding: 0.5rem 4.5rem;
    font-family: 'Texturina', serif;
    border-radius: 20px;
    box-shadow: rgba(50, 50, 93, 0.5) 0px 50px 100px -20px, rgba(0, 0, 0, 0.5) 0px 30px 60px -30px, rgba(10, 37, 64, 0.5) 0px -2px 6px 0px inset;
    cursor: pointer;
}

```

Src→components→showkeeper

Showkeeper.js

```

import React from "react"
import "./showkeeper.css"
import axios from "axios"

const ShowKeeper = ({ keeperList, setKeeperList }) => {

    const deleteKeeper = (id) => {
        axios.post("http://localhost:3001/api/delete", { id })
    }
}

```

```

        .then(res => setKeeperList(res.data))
    }

    return (
      <div className="showKeeper row">
        {
          keeperList.map( keeper => (
            <div className="keeperCard col-md-3" key={keeper._id}>
              <h1 className="title">
                {keeper.title}
                <i className="deleteIcon fa fa-trash" aria-
hidden="true" onClick={() => deleteKeeper(keeper._id)} ></i>
              </h1>
              <textarea
                className="descriptionBox"
                value={keeper.description}
                readOnly />
            </div>
          ))
        }
      </div>
    )
  }
}

export default ShowKeeper

```

showkeeper.css

```

.showKeeper {
  width: 100%;
  display: flex;
  justify-content: space-around;
}

.keeperCard{
  display: flex;
  flex-direction: column;
  background: rgb(0, 0, 0);
  padding: 2rem;
  border-radius: 20px;
  width: 300px;
  min-height: 250px;
}

```



```

margin: 2rem;
position: relative;
box-shadow: rgba(0, 0, 0, 0.5) 10px 10px 20px;
}

.title {
margin-bottom: 1rem;
font-family: 'Texturina', serif;
color: rgb(82, 171, 240);
display: flex;
align-items: center;
}

.descriptionBox{
display: flex;
width: 100%;
resize: none;
flex: 1;
color: rgb(102, 201, 156);
font-family: 'Kanit', sans-serif;
outline: none;
background: black;
border: none;
}

.deleteIcon {
color: white;
position: absolute;
right: 30px;
font-size: 1rem;
cursor: pointer;
}

```

Src→components→Header

Header.js

```
import React from "react"
```

```
import "../header.css"

const Header = () => {
  return (
    <div className="header">
      <h1>Notes Keeper App</h1>
    </div>
  )
}

export default Header
```

header.css

```
.header {
  width: 100%;
  display: flex;
  padding: 0.75rem;
  background: black;
  color: rgb(82, 171, 240);
  justify-content: center;
  font-family: 'Texturina', serif;
  box-shadow: rgba(0, 0, 0, 0.5) 0px 5px 10px;
}
```

Package.json

```
{
  "name": "keeper-app",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^0.27.2",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
  }
}
```

```
"build": "react-scripts build",
"test": "react-scripts test",
"eject": "react-scripts eject"
},
"eslintConfig": {
  "extends": [
    "react-app",
    "react-app/jest"
  ]
},
"browserslist": {
  "production": [
    ">0.2%",
    "not dead",
    "not op_mini all"
  ],
  "development": [
    "last 1 chrome version",
    "last 1 firefox version",
    "last 1 safari version"
  ]
}
}
```

For back-end

Index.js

```
import express from "express"
import cors from "cors"
import mongoose from "mongoose"

const app = express()
app.use(express.urlencoded())
app.use(express.json())
app.use(cors())
```

```

mongoose.connect("mongodb://localhost:27017/mykeeperAppDB", {useNewUrlParser:
true, useUnifiedTopology: true}, () => console.log("DB connected"))

const keeperSchema = mongoose.Schema({
  title: String,
  description: String
})

const Keeper = new mongoose.model("Keeper", keeperSchema)

app.get("/api/getAll", (req, res) => {
  Keeper.find({}, (err, keeperList) => {
    if(err){
      console.log(err)
    } else {
      res.status(200).send(keeperList)
    }
  })
})

app.post("/api/addNew", (req, res) => {
  const { title, description } = req.body
  const keeperObj = new Keeper({
    title,
    description
  })
  keeperObj.save( err => {
    if(err){
      console.log(err)
    }
    Keeper.find({}, (err, keeperList) => {
      if(err){
        console.log(err)
      } else {
        res.status(200).send(keeperList)
      }
    })
  })
})
})

```

```

app.post("/api/delete", (req, res) => {
  const { id } = req.body
  Keeper.deleteOne({ _id: id }, () => {
    Keeper.find({}, (err, keeperList) => {
      if(err){
        console.log(err)
      } else {
        res.status(200).send(keeperList)
      }
    })
  })
})

app.listen( 3001, () => {
  console.log("Backend created at port 3001")
})

```

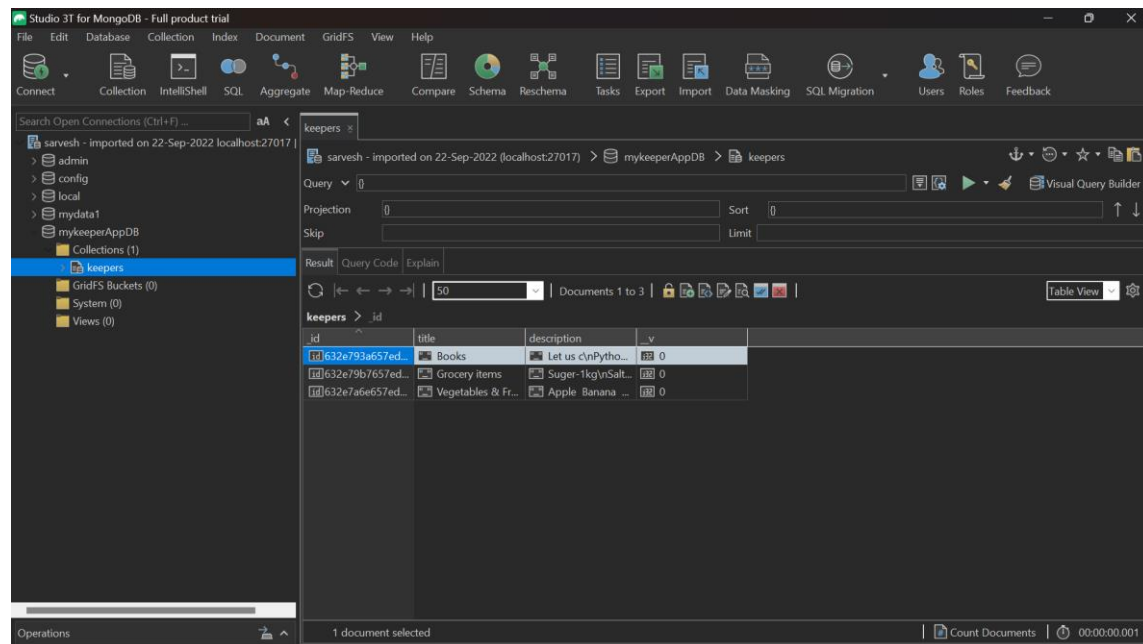
Package.json

```

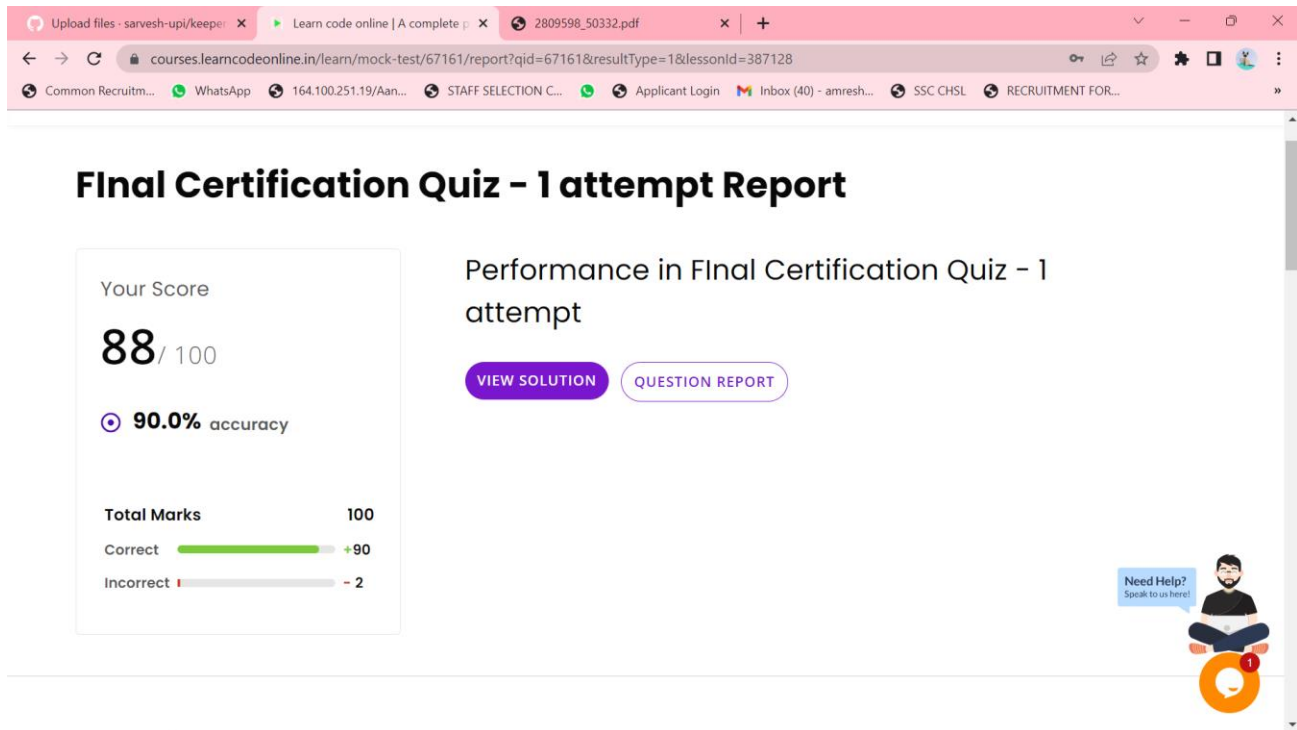
{
  "name": "keeper-app-backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "type": "module",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.18.1",
    "mongoose": "^6.6.1",
    "nodemon": "^2.0.20"
  }
}

```

mongoDB :-



Marks card from the MOOC



Learning Outcomes

- ✓ Understand the syntax, semantics, and idioms of the Javascript programming language.
- ✓ Gain confidence in dynamic & responsive web designing.
- ✓ Get practical exposure to the various designs and components.
- ✓ Work with HTML5, CSS, JavaScript and Bootstrap.
- ✓ Manipulate DOM elements with the help of JavaScript.
- ✓ Understand advantages and disadvantages of using React.
- ✓ Understand functional components, state components, child components, lifecycle, and routing in React.
- ✓ Understand and work with Node.js execution model, events, streams, APIs etc.
- ✓ Get a certificate on successful completion of the course.

REFERENCE

<https://courses.learncodeonline.in/learn/home/Full-Stack-MERN-Bootcamp/section/71641/lesson/435149>

<https://codingthesmartway.com/the-mern-stack-tutorial-building-a-react-crud-application-from-start-to-finish-part-1/>

<https://www.codingninjas.com/courses/full-stack-web-dev-mern>
