



MERN Stack CRUD Application

Dr. Selva kumar S

BMS College of Engineering

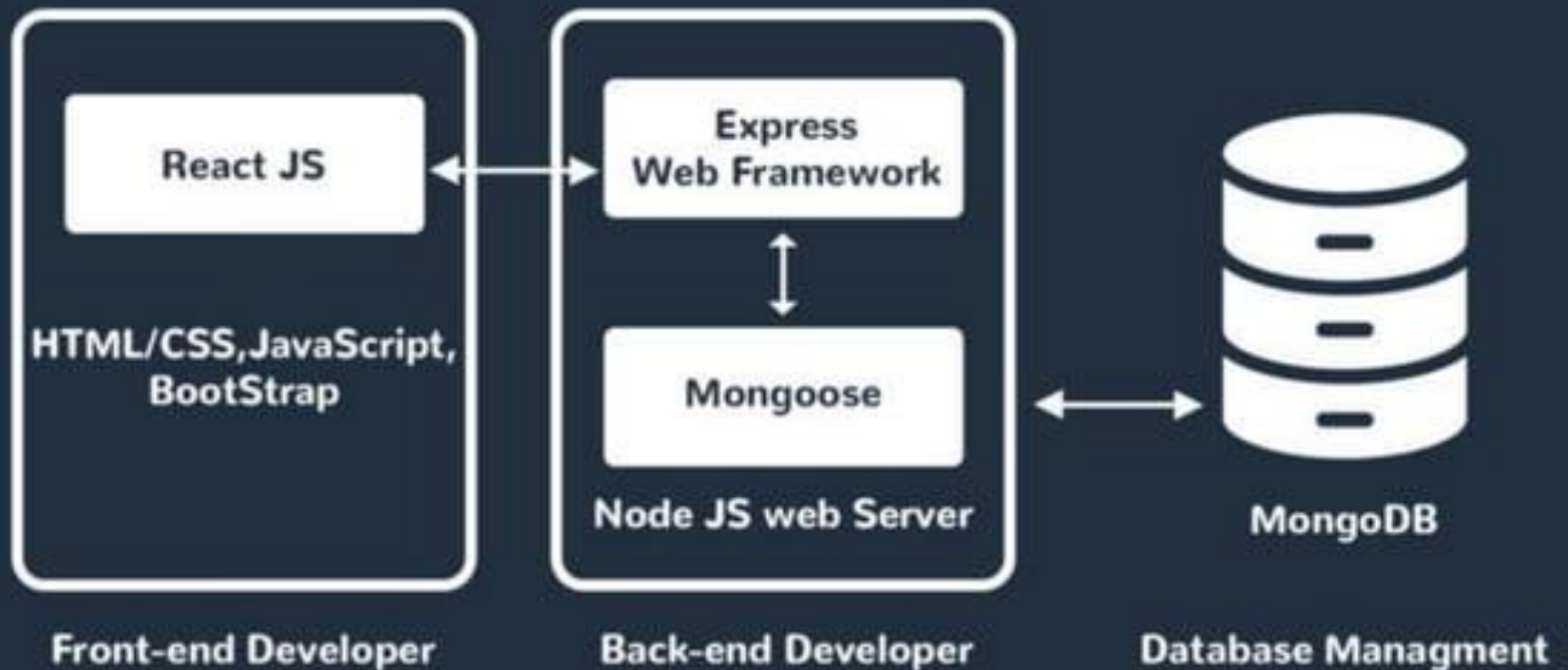
Prerequisites for MERN stack

- HTML
- CSS
- JavaScript
- Knowledge of database.

Introduction

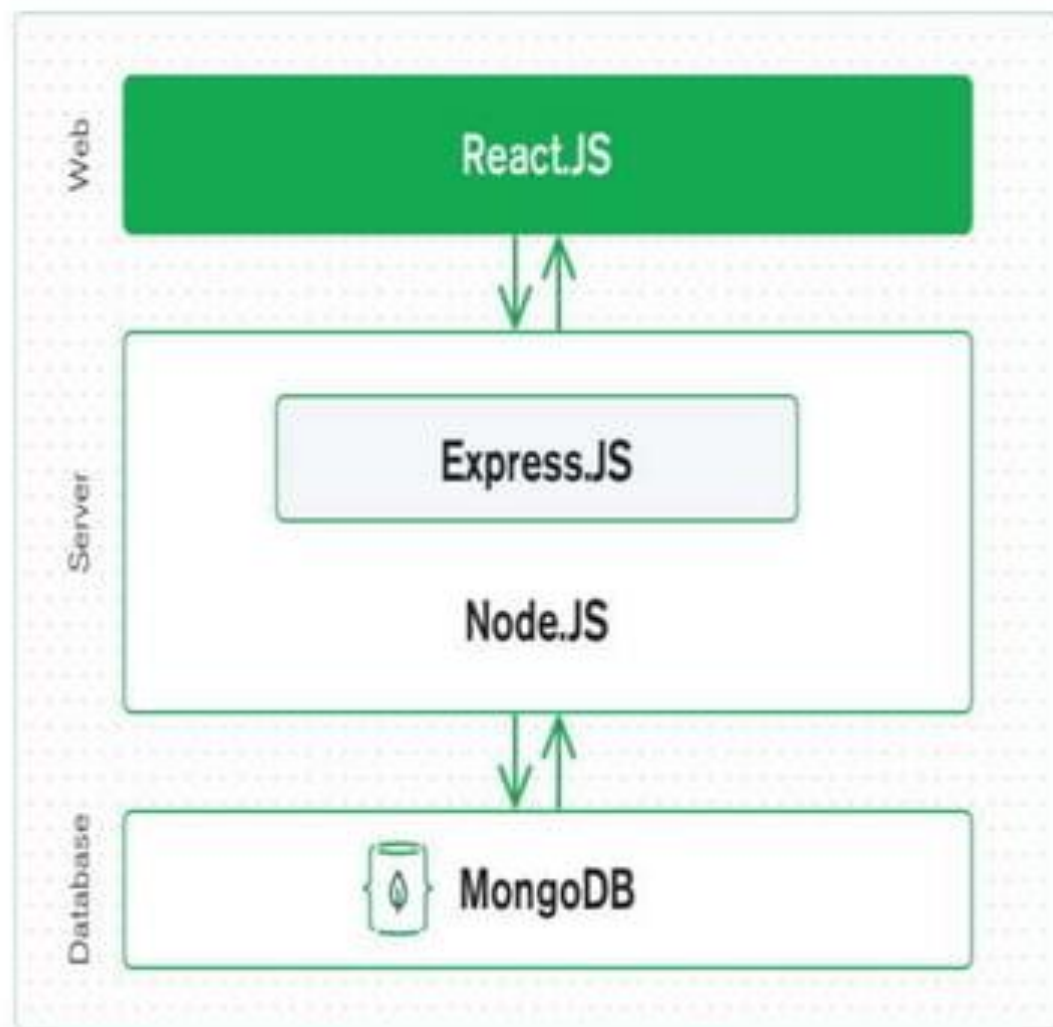
- MERN stands for MongoDB, Express, React, Node, after the four key technologies that make up the stack that is used for easier and faster deployment of full-stack web applications.
- MongoDB - document database
- Express(.js) - Node.js web framework
- React(.js) - a client-side JavaScript library
- Node(.js) - the premier JavaScript web server

MERN Stack Development



How does the MERN Stack work?

- The MERN architecture allows you to easily construct a 3-tier architecture (frontend, backend, database) entirely using JavaScript and JSON.





mongoDB®

MongoDB

- NoSQL database used for high-volume data storage.
- Open-source document-oriented database.
- MongoDB is written in C++.
- It stores data in JSON format.
- Can be easily used with Node.
- MongoDB uses BSON to query database.
- Documents containing key-value pairs are the basic units of data in MongoDB.



Database Concepts

Tabular (Relational)	MongoDB
Database	Database
Table	Collection
Row	Document
Index	Index
Join	\$lookup
Foreign Key	Reference

Why MongoDB?

- **Fast** – Being a document-oriented database, easy to index documents. Therefore a faster response.
- **Scalability** – Large data can be handled by dividing it into several machines.
- **Use of JavaScript** – MongoDB uses JavaScript which is the biggest advantage.
- **Schema Less** – Any type of data in a separate document.
- Data stored in the form of JSON.
- **Simple Environment Setup** – Its really simple to set up MongoDB.



NodeJS

- JavaScript run-time environment built on Chrome's V8 JavaScript.
- Node.js allows you to run JavaScript on the server.
- It is free & open source, written in C++.
- Ryan Dahl developed Node.js in 2009. He embedded C++ code with Chrome's V8 Engine and gave the name as Node.js.
- Node.js runs single-threaded, non-blocking, asynchronous programming, which is very memory efficient.

Where to use NodeJS?

- Back-end services such as APIs.
- Highly scalable, data-intensive and real-time apps.
- I/O bounds applications.
- Single page applications.

Where not to use NodeJS?

Node.js is not used in CPU-intensive apps which requires calculations done by CPU.

Express



ExpressJS

- Flexible Node.js framework that provides robust set of features for web for web and mobile application.
- It provides easy routing of requests based on HTTP methods and URLs.
- It allows to set up middlewares to respond to HTTP Requests.
- Allows to dynamically render HTML Pages based on passing arguments to templates.



ReactJS

- A JavaScript library for building user interfaces for web and mobile applications.
- React is used to build single-page applications.
- React allows us to create reusable UI components.
- React-router to handle the front-end routing.
- React was created by Jordan Walke, a Software Engineer at Facebook.

Why to use React?

- JSX (JavaScript XML) makes it easier and simpler to write React components.
- ReactJS supports Components. These components also promote code reusability and make the overall web application easier to understand and debug.

How does React work?

- Instead of manipulating the browser's DOM directly, React creates a Virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM.
- React finds out what changes have been made, and changes **only** what needs to be changed.

REAL and VIRTUAL DOMs





IMPLEMENTATION

Add +

Name	Email	Age	Action
sks kumar	sks@gmail.com	40	Update Delete
selva kumar	selva.cse@bmsce.ac.in	42	Update Delete

Add User

Name

Enter Name

Email

Enter Email

Age

Enter Age

Submit

Update User

Name

selva kumar

Email

selva.cse@bmsce.ac.in

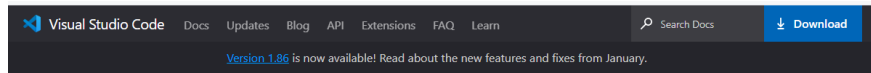
Age

42

Update

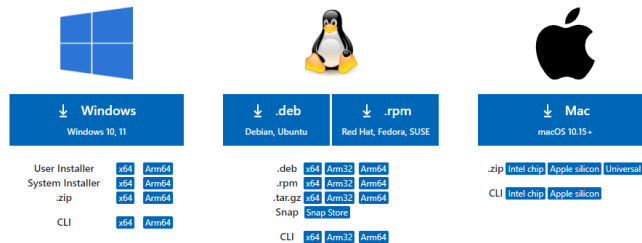
Setup

- <https://code.visualstudio.com/download>

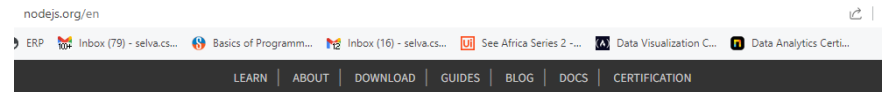


Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.



- <https://nodejs.org/en>



Node.js® is an open-source, cross-platform JavaScript runtime environment.

New security releases available ↗

Download Node.js®

20.11.1 LTS
Recommended For Most Users

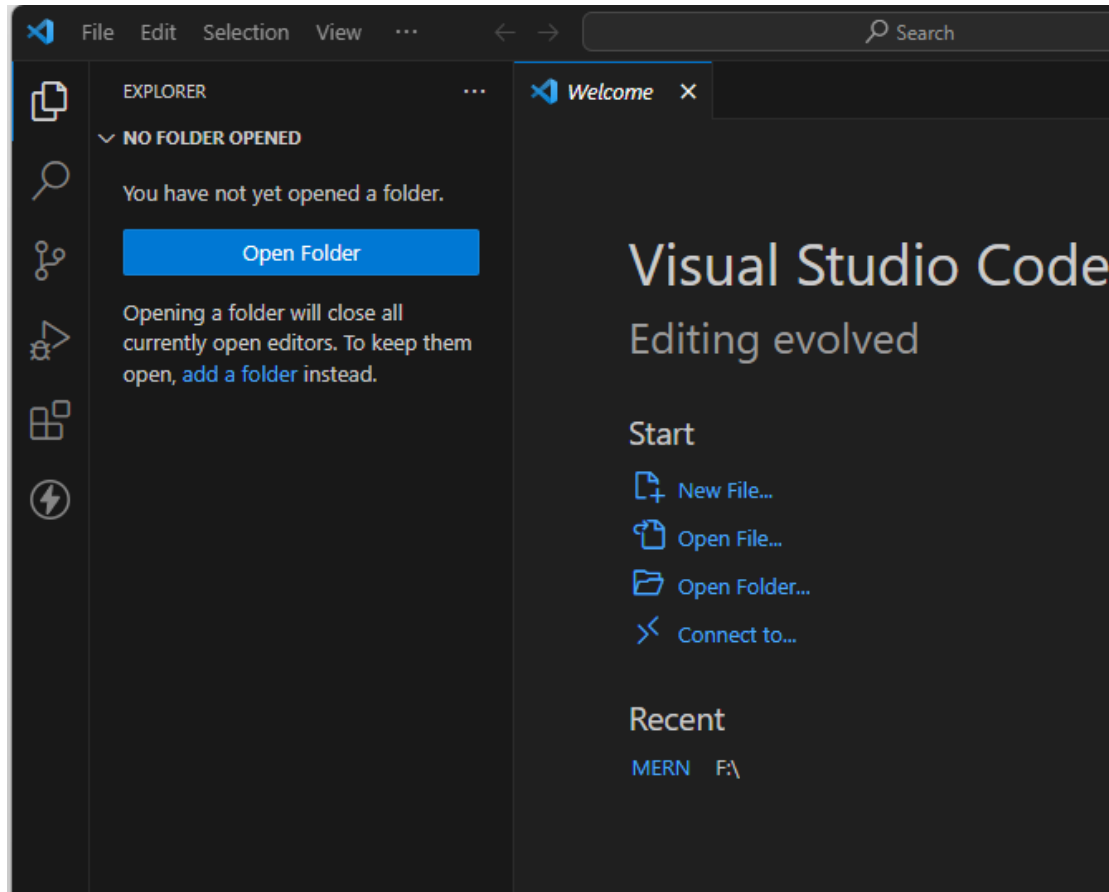
21.6.2 Current
Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#) | [Other Downloads](#) | [Changelog](#) | [API Docs](#)

For information about supported releases, see the [release schedule](#).

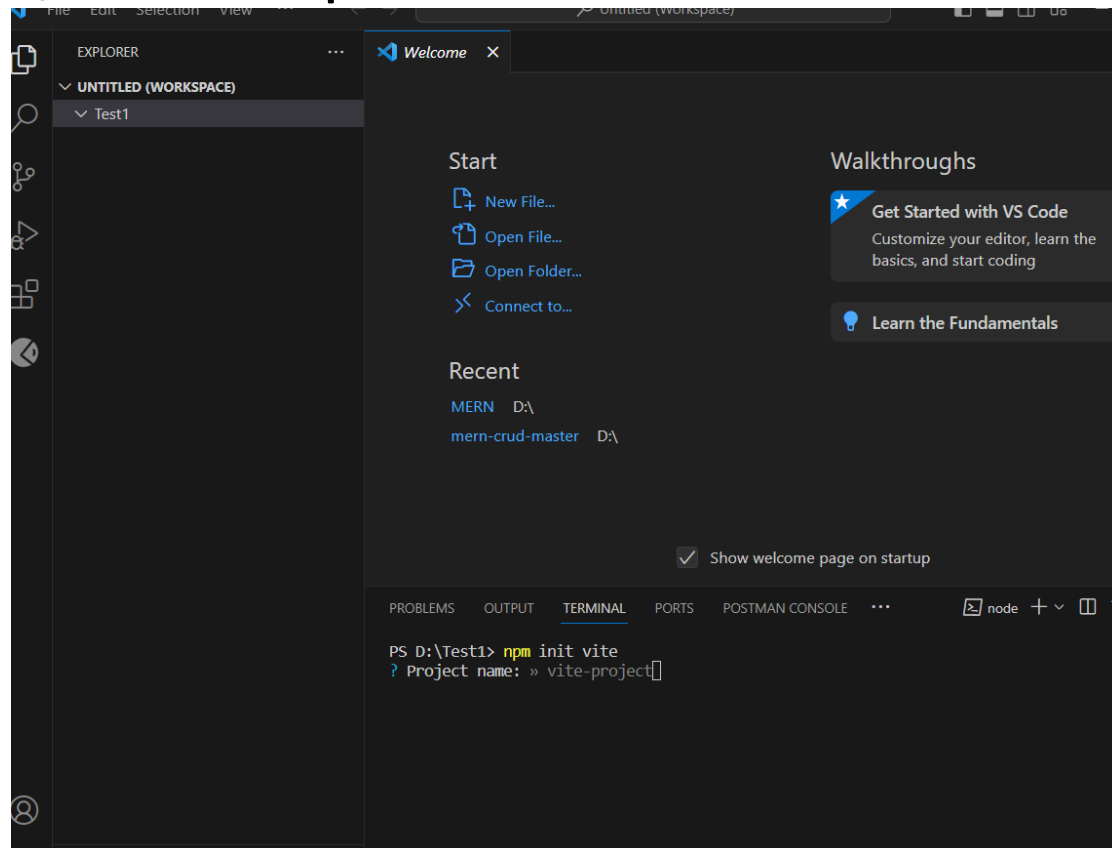
Open VS Code

- Add a folder MERN or Open Folder



Create react app

- go to terminal (Cntl + Shift + `)
 - Type below command to create react app
- d:\MERN\Demo> npm init vite



Create react app

- Enter project name : client
- select React
- select JavaScript

```
PROBLEMS OUTPUT TERMINAL PORTS POSTMAN CONSOLE ...
PS D:\Test1> npm init vite
✓ Project name: ... client
✓ Select a framework: » React
? Select a variant: » - Use arrow-keys. Return to submit.
  TypeScript
  TypeScript + SWC
>  JavaScript
   JavaScript + SWC
   Remix ↗
```

```
PROBLEMS OUTPUT TERMINAL PORTS POSTMAN CONSOLE ...
✓ Project name: ... client
✓ Select a framework: » React
✓ Select a variant: » JavaScript

Scaffolding project in D:\Test1\client...

Done. Now run:

  cd client
  npm install
  npm run dev

PS D:\Test1> 
```

```
PROBLEMS OUTPUT TERMINAL PORTS POSTMAN CONSOLE ...
PS D:\Test1> npm init vite
✓ Project name: ... client
? Select a framework: » - Use arrow-keys. Return to submit.
  Vanilla
  Vue
>  React
   Preact
   Lit
   Svelte
   Solid
   Qwik
   Others
```

```
EXPLORER
└─ UNTITLED (WORKSPACE)
   └─ Test1
      └─ client
         ├── public
         ├── src
         ├── .eslintrc.cjs
         ├── .gitignore
         ├── index.html
         ├── package.json
         ├── README.md
         └─ vite.config.js
```

Create react app

- Now run
 - cd client
 - npm install
 - npm run dev

```
PS D:\Test1> cd client
PS D:\Test1\client> npm install

added 273 packages, and audited 274 packages in 24s

98 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS D:\Test1\client> npm run dev
```

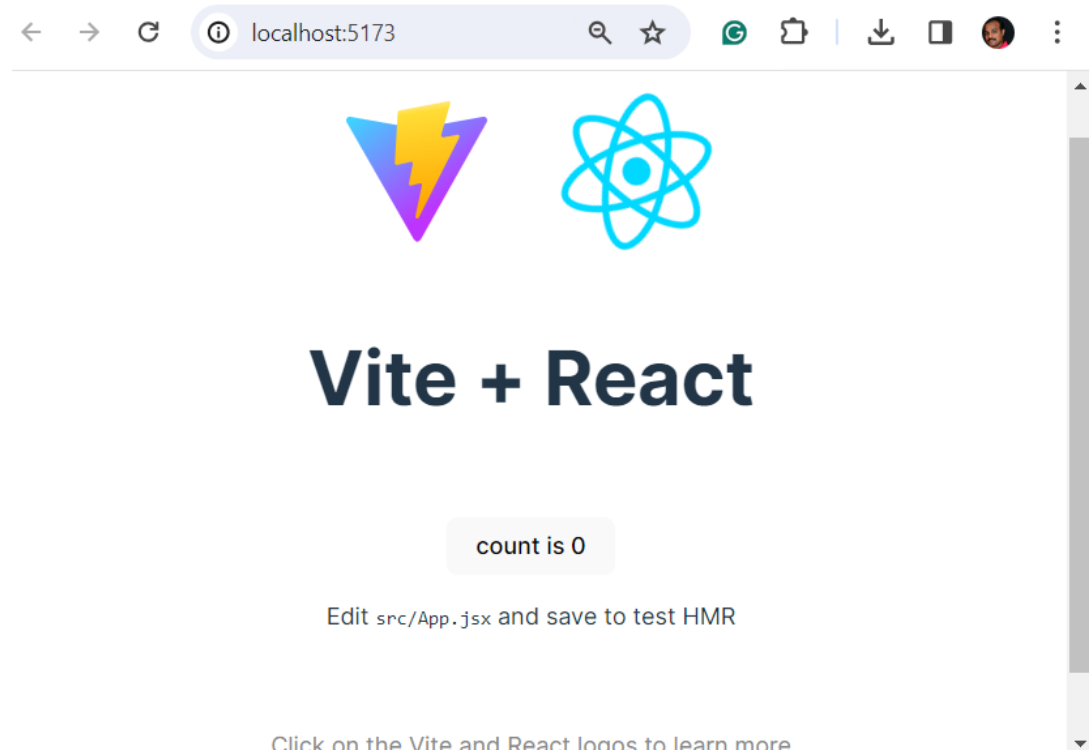
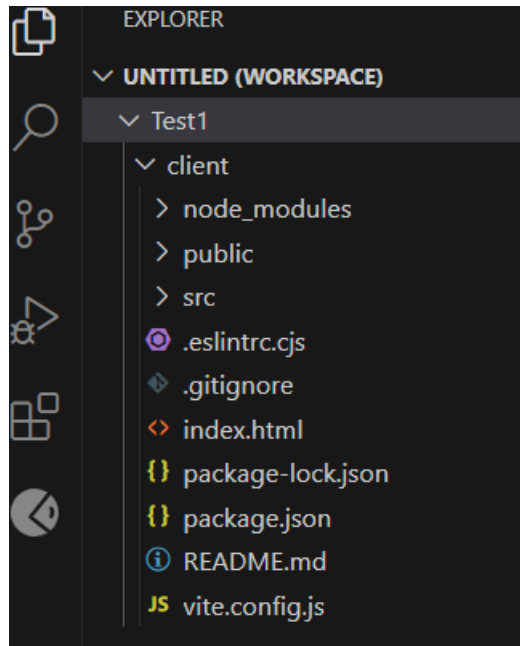
PROBLEMS OUTPUT TERMINAL PORTS

VITE v5.1.4 ready in 302 ms

→ Local: <http://localhost:5173/>

→ Network: use --host to expose

→ press h + enter to show help



Install other packages

- `npm install bootstrap axios react-router-dom`



AXIOS

 **React Router**



FRONT-END DEVELOPMENT

Client page development

- go to client->src folder
 - remove index.css
- go to main.jsx
 - remove the path of index.css

App.jsx

```
Test2 > client > src > App.jsx > App
1  import { useState } from 'react'
2  import './App.css'
3
4  function App() {
5    const [count, setCount] = useState(0)
6
7    return (
8      <>
9        <div>
10         Users
11        </div>
12      </>
13    )
14  }
15
16
17  export default App
18
```

localhost:5173

Users

Create pages

- Create the below files in src folder:
 - Users.jsx file
 - CreateUser.jsx file
 - UpdateUser.jsx file

Users.jsx

- Users.jsx file:

```
import React from "react";  
function Users () {  
  return(  
    <div>  
      Users  
    </div>  
  )  
}
```

```
export default Users
```

CreateUser.jsx

- CreateUser.jsx

```
import React from "react";  
function CreateUser () {  
  return(  
    <div>  
      Create Users  
    </div>  
  )  
}  
  
export default CreateUser
```

UpdateUser.jsx

- similiary updateuser.jsx

```
import React from "react";  
function UpdateUser () {  
  return(  
    <div>  
      Update Users  
    </div>  
  )  
}
```

```
export default UpdateUser
```

App.jsx

- Add Router details:

```
import { useState } from 'react'
import './App.css'
import {BrowserRouter, Routes, Route} from 'react-router-dom'
import 'bootstrap/dist/css/bootstrap.min.css'
import Users from './Users'
import CreateUsers from './CreateUser'
import UpdateUsers from './UpdateUser'
```

```
function App() {
  const [count, setCount] = useState(0)

  return (
    <div>
      <BrowserRouter>
        <Routes>
          <Route path="/" element={<Users />}></Route>
          <Route path="/create" element={<CreateUsers />}></Route>
          <Route path="/update/:id" element={<UpdateUsers />}></Route>
        </Routes>
      </BrowserRouter>
    </div>
  )
}
```

```
export default App
```

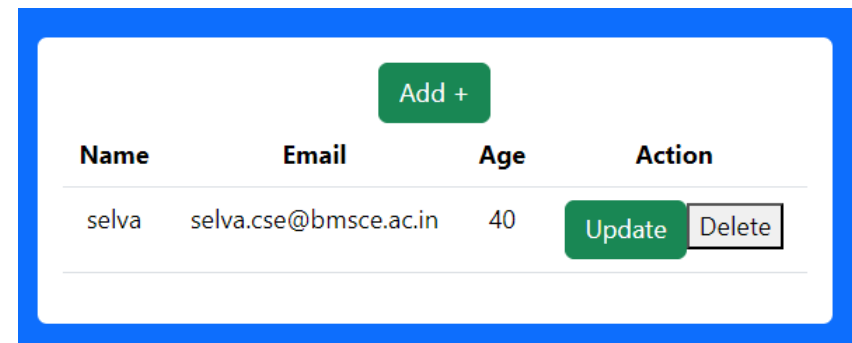
Add html code in Users.jsx

```
import React, { useState } from "react";
import { Link } from "react-router-dom";
```

```
function Users () {
  const [users, setUsers] = useState([
    { Name: "selva", Email: "selva.cse@bmsce.ac.in", Age: 40 }
  ])
```

```
  return(
    <div className="d-flex vh-100 bg-primary justify-content-center align-items-center">
      <div className="w-50 bg-white rounded p-3">
        <Link to="/create" className="btn btn-success">Add +</Link>
        <table className="table">
          <thead>
            <tr>
              <th>Name</th>
              <th>Email</th>
              <th>Age</th>
              <th>Action</th>
            </tr>
          </thead>
          <tbody>
            {
              users.map((user)=>{
                return <tr>
                  <td>{user.Name}</td>
                  <td>{user.Email}</td>
                  <td>{user.Age}</td>
                  <td>
                    <Link to="/update" className="btn btn-success">Update</Link>
                    <button>Delete</button>
                  </td>
                </tr>
              })
            }
          </tbody>
        </table>
      </div>
    </div>
  )
}
```

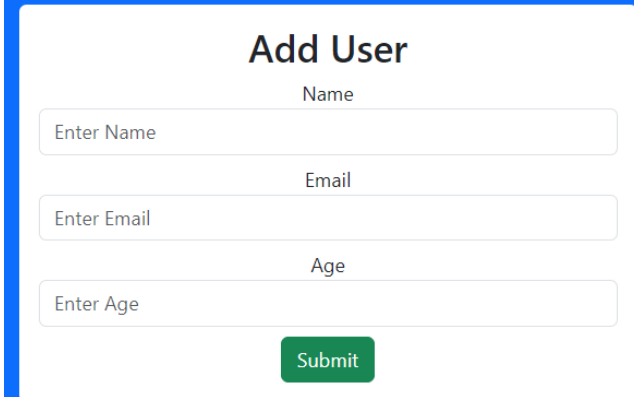
```
export default Users;
```



Add html code in the CreateUser.jsx

```
import React from "react";
function CreateUser () {
  return(
    <div className= "d-flex vh-100 bg-primary justify-content-center align-items-center">
      <div className='w-50 bg-white rounded p-3'>
        <form>
          <h2> Add User</h2>
          <div className="mb-2">
            <label htmlFor="">Name</label>
            <input type="text" placeholder="Enter Name" className="form-control" />
          </div>
          <div className="mb-2">
            <label htmlFor="">Email</label>
            <input type="text" placeholder="Enter Email" className="form-control" />
          </div>
          <div className="mb-2">
            <label htmlFor="">Age</label>
            <input type="text" placeholder="Enter Age" className="form-control" />
          </div>
          <button className="btn btn-success">Submit</button>
        </form>
      </div>
    </div>
  )
}

export default CreateUser;
```



Add User

Name
Enter Name

Email
Enter Email

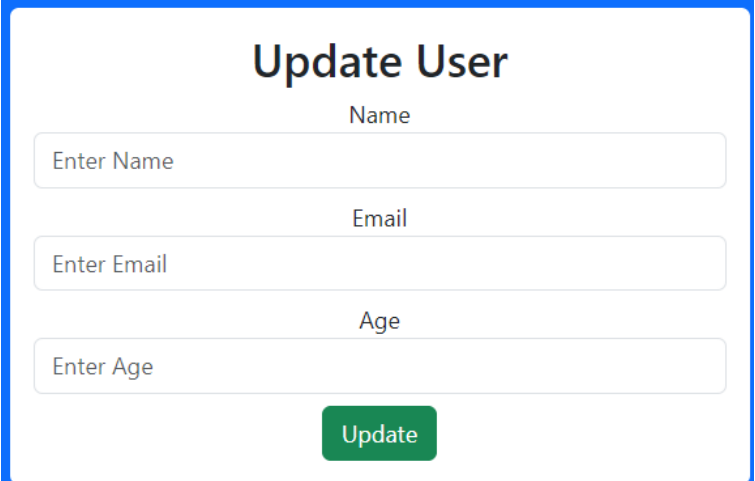
Age
Enter Age

Submit

Add html code in UpdateUser.jsx

```
import React from "react";
function UpdateUser () {
  return(
    <div className= "d-flex vh-100 bg-primary justify-content-center align-items-center">
      <div className='w-50 bg-white rounded p-3'>
        <form>
          <h2> Update User</h2>
          <div className="mb-2">
            <label htmlFor="">Name</label>
            <input type="text" placeholder="Enter Name" className="form-control" />
          </div>
          <div className="mb-2">
            <label htmlFor="">Email</label>
            <input type="text" placeholder="Enter Email" className="form-control" />
          </div>
          <div className="mb-2">
            <label htmlFor="">Age</label>
            <input type="text" placeholder="Enter Age" className="form-control" />
          </div>
          <button className="btn btn-success">Update</button>
        </form>
      </div>
    </div>
  )
}

export default UpdateUser;
```



Update User

Name

Enter Name

Email

Enter Email

Age

Enter Age

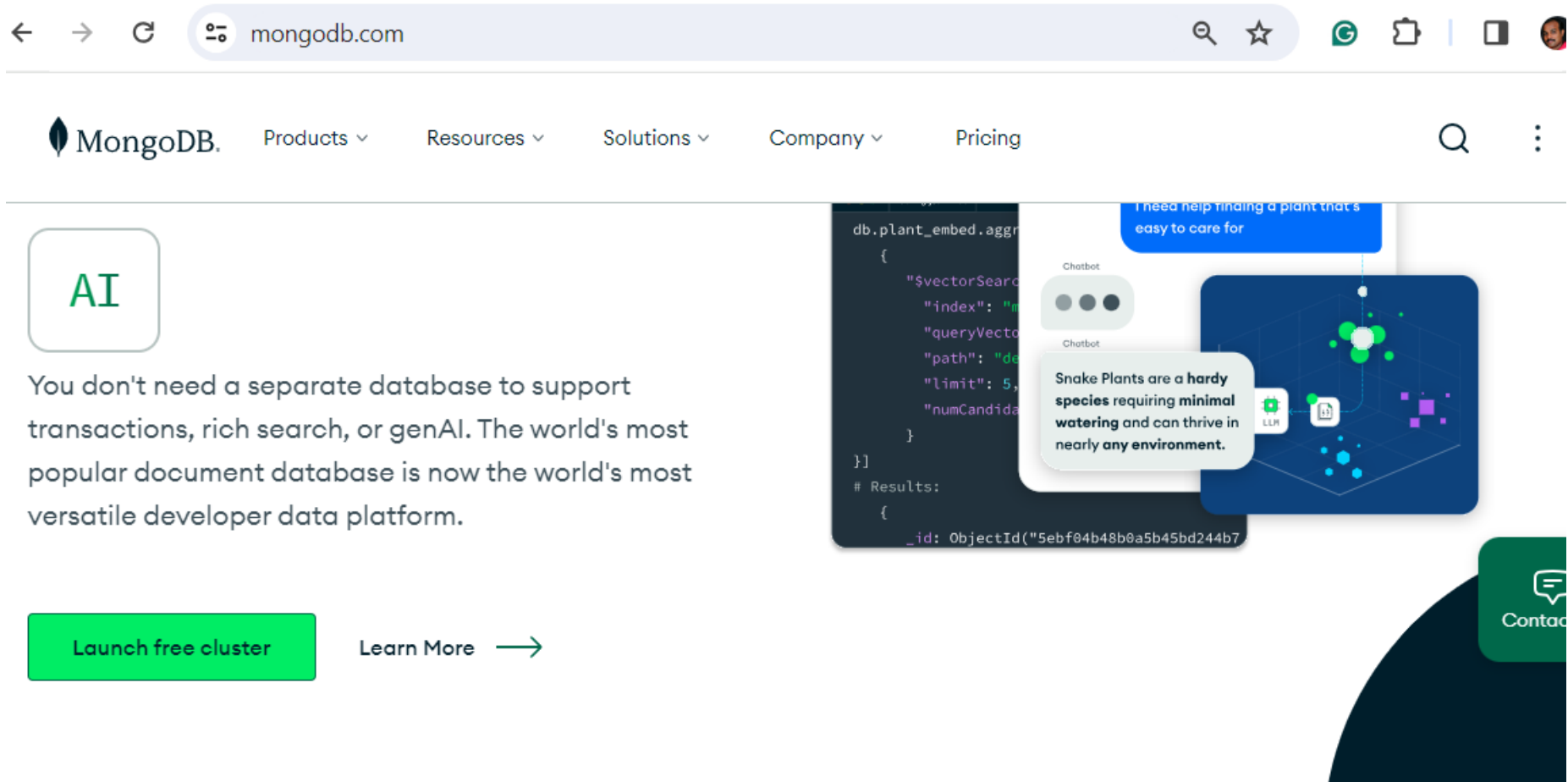
Update



BACK-END

MongoDB. database

- <https://www.mongodb.com/>



The screenshot shows the MongoDB website homepage. At the top is the MongoDB logo and navigation links: Products, Resources, Solutions, Company, and Pricing. Below the navigation bar, there's a section titled "AI" with the text: "You don't need a separate database to support transactions, rich search, or genAI. The world's most popular document database is now the world's most versatile developer data platform." To the right of this text is a large graphic featuring a code editor with MongoDB aggregation queries, a chatbot interface with a user query "I need help finding a plant that's easy to care for" and a response "Snake Plants are a hardy species requiring minimal watering and can thrive in nearly any environment.", and a network diagram. At the bottom left is a green button labeled "Launch free cluster" and a link "Learn More" with a right-pointing arrow. At the bottom right is a green button labeled "Contact Us" with a speech bubble icon.

← → ↻ 🏠 mongodb.com 🔍 ☆ 🌱 📋 📱 👤

MongoDB. Products ▾ Resources ▾ Solutions ▾ Company ▾ Pricing 🔍 ⋮

AI

You don't need a separate database to support transactions, rich search, or genAI. The world's most popular document database is now the world's most versatile developer data platform.

Launch free cluster

Learn More →

```
db.plant_embed.aggregate(
  {
    "$vectorSearch": {
      "index": "plant_embedded",
      "queryVector": [0.1, 0.2, 0.3, 0.4, 0.5],
      "path": "description",
      "limit": 5,
      "numCandidates": 100
    }
  }
)

# Results:
{
  "_id": ObjectId("5ebf04b48b0a5b45bd244b7")
}
```

I need help finding a plant that's easy to care for

Chatbot

Snake Plants are a hardy species requiring minimal watering and can thrive in nearly any environment.

LLM

Contact Us

Setup MongoDB

Sign up

See what Atlas is capable of for free

 Sign up with Google

Overview

Real Time

Me

DATABASES: 2 COLLECTIONS: 2

+ Create Database

 Search Namespaces

▼ crud

| users

CLUSTERS > CREATE NEW CLUSTER

Create New Cluster

Serverless

Dedicated

Shared

For production applications with sophisticated workload requirements. Advanced configuration controls.

Network isolation, end-to-end encryption, and fine-grained access controls.
On-demand performance advice, including index and schema suggestions.

Global Cluster Configuration

Cloud Provider & Region

AWS, Hyderabad (ap-south-2) ^

aws

 Google Cloud

A

Database Deployments

 Find a database deployment...

 ClusterO

Connect

View Monitoring

Browse Collections

...

SELVA KUMAR'S ORG - 2022-11-02 > PROJECT 0 > DATABASES

 ClusterO

VERSION

6.0.13

REGION

AWS Mumbai (ap-south-1)

- Overview
- Real Time
- Metrics
- Collections
- Atlas Search
- Profiler
- Performance Advisor
- Online Archive
- Cmd Line Tools

DATABASES: 2 COLLECTIONS: 2

 VISUALIZE YOUR DATA

 REFRESH

+ Create Database

 Search Namespaces

- crud
- users
- selva-BMS

crud.users

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 180B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 36KB

- Find
- Indexes
- Schema Anti-Patterns 0
- Aggregation
- Search Indexes

INSERT DOCUMENT

Filter 

Type a query: { field: 'value' }

Reset

Apply

Options ▾

QUERY RESULTS: 1-2 OF 2

```
_id: ObjectId('65d3081aff91c75b2e32f5bf')
name: "sks kumar"
email: "sks@gmail.com"
age: 40
__v: 0
```

Connect to Cluster0



Connecting with MongoDB Driver

1. Select your driver and version

We recommend installing and using the latest driver version.

Driver

Version

Node.js

5.5 or later

2. Install your driver

Run the following on the command line

```
npm install mongodb
```



[View MongoDB Node.js Driver installation instructions.](#)

3. Add your connection string into your application code

☐ View full code sample

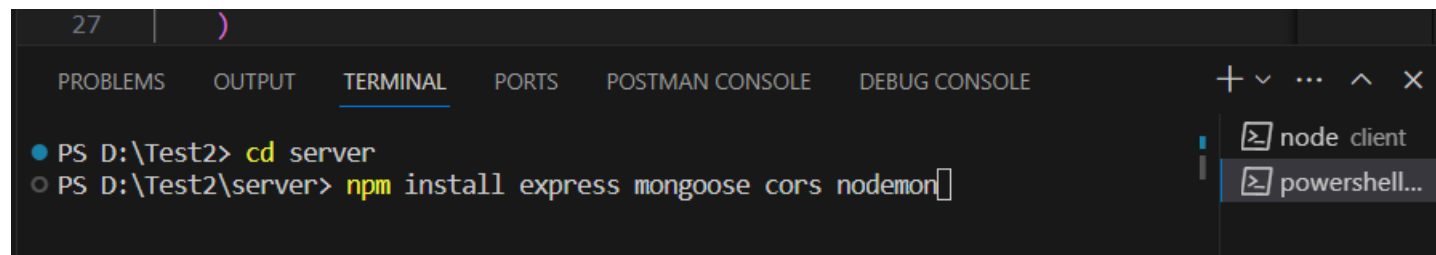
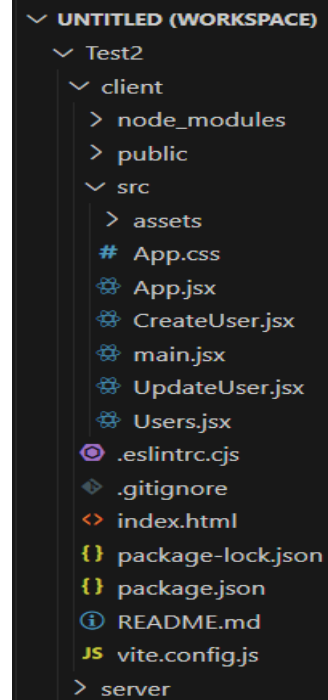
```
mongodb+srv://selvaBMS:<password>@cluster0.eli2h0j.mongodb.net/?  
retryWrites=true&w=majority
```



Replace **<password>** with the password for the **selvaBMS** user. Ensure any option params are [URL encoded](#).

Setup Server

- create server folder
- go to new terminal
d:\MERN\server> npm init -y
- Install express framework, mongoose database, cors to connect front-end with backend, nodemon to sync.
- d:\MERN\server> npm install express mongoose cors nodemon



Package.json

add the nodemon in package.json

```
"scripts": {  
  "test": "echo \"Error: no test specified\" && exit  
1",  
  "start": "nodemon index.js"  
}
```

Create index.js

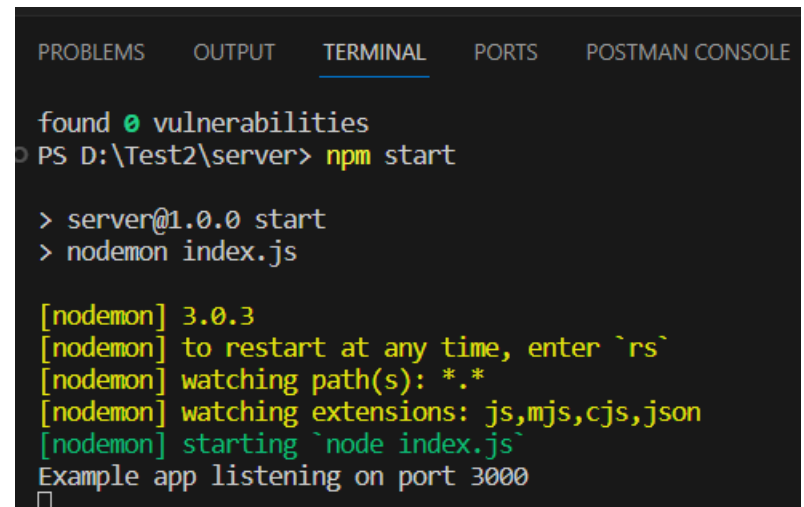
- Create index.js inside the sever folder and setup express server:

```
const express = require('express')
const app = express()
const port = 3000
```

```
app.get('/', (req, res) => {
  res.send('Hello World!')
})
```

```
app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

- Start the server
- d:\MERN\server> npm start



The screenshot shows a VS Code interface with a terminal window open. The terminal has tabs for PROBLEMS, OUTPUT, TERMINAL, PORTS, and POSTMAN CONSOLE. The TERMINAL tab is active. The terminal output shows a security scan result: 'found 0 vulnerabilities'. Below this, the command 'PS D:\Test2\server> npm start' is entered. The output continues with 'server@1.0.0 start' and 'nodemon index.js'. Then, the nodemon version '3.0.3' is displayed, followed by instructions: 'to restart at any time, enter `rs`'. The next lines show the paths and extensions being watched: 'watching path(s): *.*' and 'watching extensions: js,mjs,cjs,json'. Finally, it says 'starting `node index.js`' and 'Example app listening on port 3000'. A cursor is visible at the bottom of the terminal.

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  POSTMAN CONSOLE

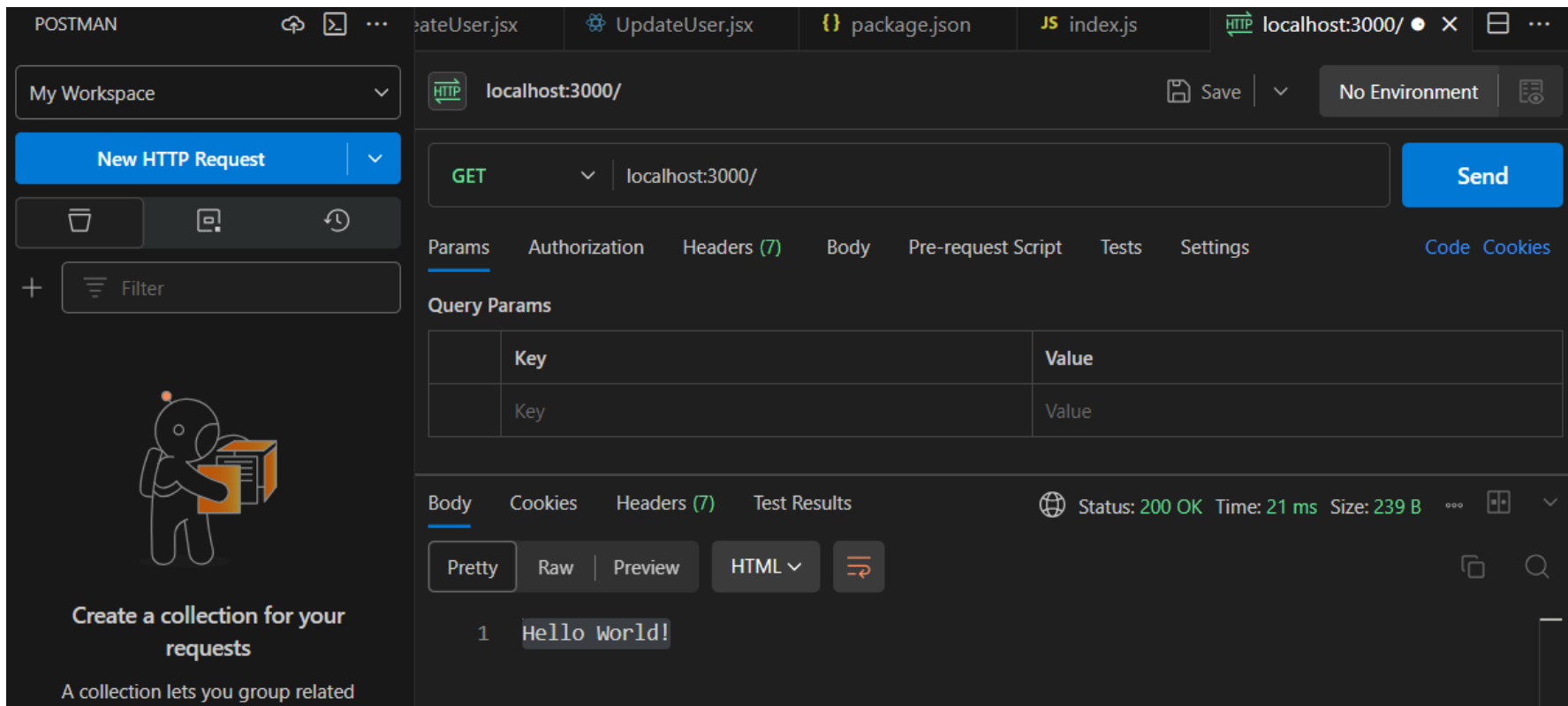
found 0 vulnerabilities
PS D:\Test2\server> npm start

> server@1.0.0 start
> nodemon index.js

[nodemon] 3.0.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
Example app listening on port 3000
```

Check the api with Postman

- Install Postman in VS code and test api



The screenshot displays the Postman application interface. On the left sidebar, there's a 'My Workspace' dropdown, a 'New HTTP Request' button, and a 'Filter' input. Below this is a cartoon character holding boxes with the text 'Create a collection for your requests' and 'A collection lets you group related'. The main panel shows a GET request to 'localhost:3000/'. The 'Headers' tab is active, showing 7 headers. The 'Body' tab is also visible, showing a 'Hello World!' response. The status bar at the bottom indicates 'Status: 200 OK', 'Time: 21 ms', and 'Size: 239 B'.

POSTMAN

My Workspace

New HTTP Request

Filter

Create a collection for your requests

A collection lets you group related

localhost:3000/

GET localhost:3000/

Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Code Cookies

Query Params

Key	Value
Key	Value

Body Cookies Headers (7) Test Results

Status: 200 OK Time: 21 ms Size: 239 B

Pretty Raw Preview HTML

1 Hello World!

Create Model

- Create model folder inside server folder.
- Create new file users.js

```
const mongoose = require('mongoose')
```

```
const UserSchema = new mongoose.Schema({  
  name: String,  
  email: String,  
  age: Number  
})
```

```
const UserModel = mongoose.model("users",UserSchema)  
module.exports = UserModel
```

Access mongoDB

- Modify index.js

```
const express = require('express')
const mongoose = require('mongoose')
const cors = require('cors')
const UserModel = require('./models/Users')
```

```
const app = express()
const port = 3000
app.use(cors())
app.use(express.json())
//mongoose.connect("mongodb+srv://selvaBMS:<password>@cluster0.eli2h0j.mongodb.net/<Database
name>?retryWrites=true&w=majority")
```

```
//mongoose.connect("mongodb+srv://selvaBMS:<password>@cluster0.eli2h0j.mongodb.net/crud?retryWrites=true&w=m
ajority")
mongoose.connect("mongodb://localhost:27017/crud")
```

```
app.post("/createuser", (req,res) => {
  UserModel.create(req.body)
  .then(users => res.json(users))
  .catch(err => res.json(err))
})
```

```
app.listen(port, () => {
  console.log('Server is running')
})
```

Test api through postman

The image shows a Postman interface on the left and a MongoDB interface on the right. In Postman, a POST request to `localhost:3000/createuser` is shown with a JSON body: `{ "name": "test", "email": "test", "age": 33 }`. The MongoDB interface on the right shows the results of the database operation, displaying two documents in the `users` collection.

Postman Request Details:

- Method: POST
- URL: `localhost:3000/createuser`
- Body (raw):

```
1 { "name": "test",
2   "email": "test",
3   "age": 33 }
```

MongoDB Query Results:

STORAGE SIZE: 36KB | LOGICAL DATA SIZE: 251B | TOTAL DOCUMENTS: 3 | INDEXES

Find | Indexes | Schema Anti-Patterns 0 | Aggregation

Filter [🔗](#) | Type a query: { field: 'value' }

Results:

```
_id: ObjectId('65d609cdafabf78511e37d51')
name: "selva kumar"
email: "selva.cse@bmsce.ac.in"
age: 42
__v: 0
```



```
_id: ObjectId('65d61ea632bd8560dab3a0bb')
name: "test"
email: "test"
age: 33
__v: 0
```

Integrate front-end with back-end

modify front-end createUser.jsx

```
=====
import React, { useState } from "react";
import {useNavigate } from 'react-router-dom';
import axios from 'axios';

function CreateUser () {

  const [name, setName]= useState()
  const [email, setEmail]= useState()
  const [age, setAge]= useState()
  const navigate = useNavigate()

  const Submit = (e) => {
    e.preventDefault();
    axios.post("http://localhost:3001/createuser", {name,email,age})
      .then(result => console.log(result))
      .catch(err => console.log(err))
    navigate('/')
  }

  return(
    <div className= "d-flex vh-100 bg-primary justify-content-center align-items-center">
      <div className='w-50 bg-white rounded p-3'>
        <form onSubmit={Submit}>
          <h2> Add User</h2>
          <div className="mb-2">
            <label htmlFor="">Name</label>
            <input type="text" placeholder="Enter Name" className="form-control"
              onChange={(e) => setName(e.target.value)}/>
          </div>
          <div className="mb-2">
            <label htmlFor="">Email</label>
            <input type="text" placeholder="Enter Email" className="form-control"
              onChange={(e) => setEmail(e.target.value)}/>
          </div>
          <div className="mb-2">
            <label htmlFor="">Age</label>
            <input type="text" placeholder="Enter Age" className="form-control"
              onChange={(e) => setAge(e.target.value)}/>
          </div>
        </form>
      </div>
    </div>
  )
}
```

localhost:5173/create

Add User

Name

newuser

Email

newuser@gmail.com

Age

22

Submit

✖ GET <chrome-extension://invalid/> <contentScript.bundle.js:261757>
net::ERR_FAILED

[CreateUser.jsx:15](#)
▶ {data: {...}, status: 200, statusText: 'OK', headers: AxiosHeaders, conf
 ig: {...}, ...}

```
_id: ObjectId('65d620d6a018a7f716b68c7c')
name: "newuser"
email: "newuser@gmail.com"
age: 22
__v: 0
```

Get api

```
index.js
=====
const express = require('express')
const mongoose = require('mongoose')
const cors = require('cors')
const UserModel = require('./Users')

const app = express()
const port = 3001
app.use(cors())
app.use(express.json())
//mongoose.connect("mongodb+srv://selvaBMS:<password>@cluster0.eli2h0j.mongodb.net/crud?retryWrites=true&w=majority")
mongoose.connect("mongodb://localhost:27017/crud")

app.get('/', (req,res) =>{
  UserModel.find({})
  .then(users => res.json(users))
  .catch(err => res.json(err))
})

app.post("/createuser", (req,res) => {
  UserModel.create(req.body)
  .then(users => res.json(users))
  .catch(err => res.json(err))
})

app.listen(port, () => {
  console.log('Server is running')
})
```

Display details

- Retrieve details from MongoDB and list the items.
- changes in front-end to display the get data in Users.jsx

```
import React, { useState, useEffect } from "react";
import { Link } from "react-router-dom";
import axios from 'axios'
```

```
function Users () {
  const [users, setUsers] = useState([])

  useEffect (()=>{
    axios.get('http://localhost:3001')
      .then(result => setUsers(result.data))
      .catch(err => console.log(err))
  },[])

  return(
    <div className="d-flex vh-100 bg-primary justify-content-center align-items-center">
      <div className='w-50 bg-white rounded p-3'>
        <Link to="/create" className='btn btn-success'>Add +</Link>
        <table className='table'>
          <thead>
            <tr>
              <th>Name</th>
              <th>Email</th>
              <th>Age</th>
              <th>Action</th>
            </tr>
          </thead>
          <tbody>
            {
              users.map((user)=>{
                return <tr>
                  <td>{user.name}</td>
                  <td>{user.email}</td>
                  <td>{user.age}</td>
                  <td>
                    <Link to="/update" className='btn btn-success'>Update</Link>
                    <button className='btn btn-danger'>Delete</button>
                  </td>
                </tr>
              })
            }
          </tbody>
        </table>
      </div>
    </div>
  )
}
```

Add +			
Name	Email	Age	Action
sks kumar	sks@gmail.com	40	<button>Update</button> <button>Delete</button>
selva kumar	selva.cse@bmsce.ac.in	42	<button>Update</button> <button>Delete</button>
test	test	33	<button>Update</button> <button>Delete</button>
newuser	newuser@gmail.com	22	<button>Update</button> <button>Delete</button>

Update function

- add id in the app.jsx

```
<Route path='/update/:id' element={<UpdateUsers />}></Route>
```

- modify user.jsx : pass id in the update button

```
<Link to={`/${update}/${user._id}`} className='btn btn-success'>Update</Link>
```

- modify index.js:

```
app.get('/getuser/:id', (req,res) =>{  
  const id = req.params.id;  
  UserModel.findById({_id:id})  
  .then(users => res.json(users))  
  .catch(err => res.json(err))  
})
```

```
app.put('/updateuser/:id', (req,res) =>{  
  const id = req.params.id;  
  UserModel.findByIdAndUpdate({_id:id},  
    {name: req.body.name,  
    email: req.body.email,  
    age: req.body.age})  
  .then(users => res.json(users))  
  .catch(err => res.json(err))  
})
```

Integrate update with back-end

```
import React, {useState, useEffect} from "react";
import { useParams, useNavigate } from 'react-router-dom';
import axios from 'axios';

function UpdateUser () {

  const {id} = useParams()
  const [name, setName]= useState()
  const [email, setEmail]= useState()
  const [age, setAge]= useState()
  const navigate = useNavigate()

  useEffect (()=>{
    axios.get('http://localhost:3001/getuser/'+id)
    .then(result => {console.log(result)
      setName(result.data.name)
      setEmail(result.data.email)
      setAge(result.data.age)})

    .catch(err => console.log(err))
  }, [])

  const Update = (e) => {
    e.preventDefault();
    axios.put("http://localhost:3001/updateuser/"+id, {name,email,age})
    .then(result => console.log(result))
    .catch(err => console.log(err))
    navigate('/')
  }

  return(
    <div className= "d-flex vh-100 bg-primary justify-content-center align-items-center">
    <div className='w-50 bg-white rounded p-3'>
      <form onSubmit ={Update}>
        <h2> Update User</h2>
        <div className="mb-2">
          <label htmlFor="">Name</label>
          <input type="text" placeholder="Enter Name" className="form-control"
            value ={name} onChange={(e) => setName(e.target.value)}/>
        </div>
        <div className="mb-2">
          <label htmlFor="">Email</label>
          <input type="text" placeholder="Enter Email" className="form-control"
            value ={email} onChange={(e) => setEmail(e.target.value)}/>
        </div>
        <div className="mb-2">
          <label htmlFor="">Age</label>
          <input type="text" placeholder="Enter Age" className="form-control"
            value ={age} onChange={(e) => setAge(e.target.value)}/>
        </div>
        <button type="submit" class="btn btn-primary">Update</button>
      </form>
    </div>
    </div>
  )
}
```

Delete function

- Users.jsx file: --> add onclick event
- `<button className='btn btn-danger' onClick={(e) => handleDelete(user._id)}>Delete</button>`

--> Add handleDelete below this code

```
useEffect (()=>{  
  axios.get('http://localhost:3001')  
    .then(result => setUsers(result.data))  
    .catch(err => console.log(err))  
}, [])
```

-->

```
const handleDelete =(id) => {  
  axios.delete('http://localhost:3001/deleteuser/'+id)  
    .then(res=> console.log(res))  
    .catch(err => console.log(err))  
  navigate('/')  
}
```

Delete Api

- Goto index.js --> write delete api

```
app.delete('/deleteuser/:id', (req,res) =>{  
  const id = req.params.id;  
  UserModel.findByIdAndDelete({_id:id})  
    .then(res => res.json(res))  
    .catch(err => res.json(err))  
  
})
```

References

- <https://www.mongodb.com/>
- <https://expressjs.com/>
- <https://reactjs.org/>
- <https://nodejs.dev/>

Thank You