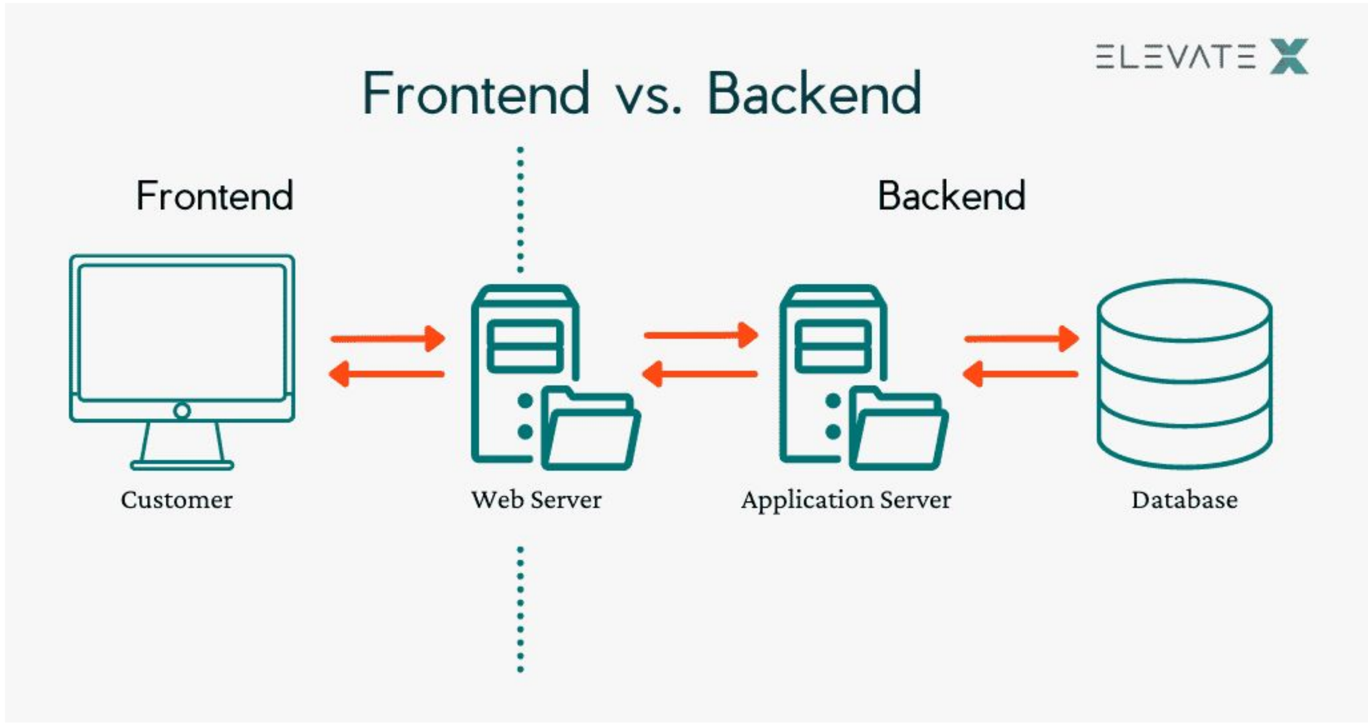
Express

By: Puskar Adhikari

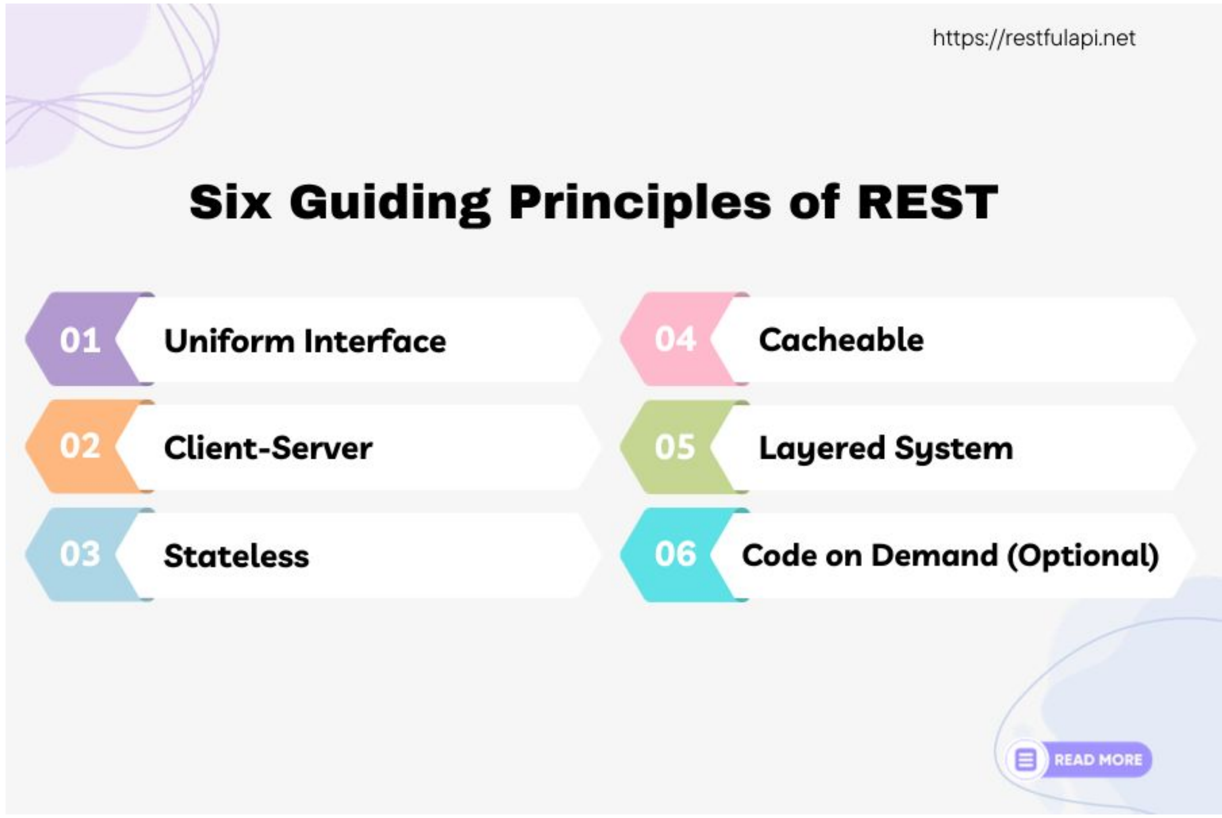
Backend

- Angular/React are just the frontends/UI.
- We need backend for:
  - User authentication
  - Database operations
  - APIs (fetching/sending data)
  - Security & business logic



REST APIs

- REST API stands for Representational State Transfer API.
- It is a type of API (Application Programming Interface) that allows communication between different systems over the internet.
- REST APIs work by sending requests and receiving responses, typically in JSON format, between the client and server.



https://restfulapi.net

Six Guiding Principles of REST

01 Uniform Interface
02 Client-Server
03 Stateless
04 Cacheable
05 Layered System
06 Code on Demand (Optional)

REST APIs

Common HTTP Methods Used in REST API

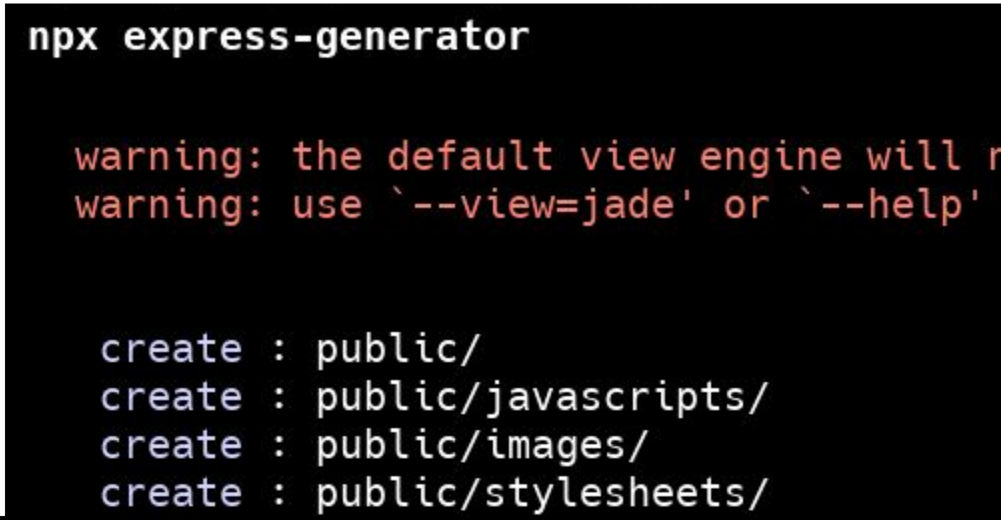
- In HTTP, there are five methods that are commonly used in a REST-based Architecture, i.e., POST, GET, PUT, PATCH, and DELETE.
- These correspond to create, read, update, and delete (or CRUD) operations, respectively.
- There are other methods that are less frequently used, like OPTIONS and HEAD.

OBJECTIVE

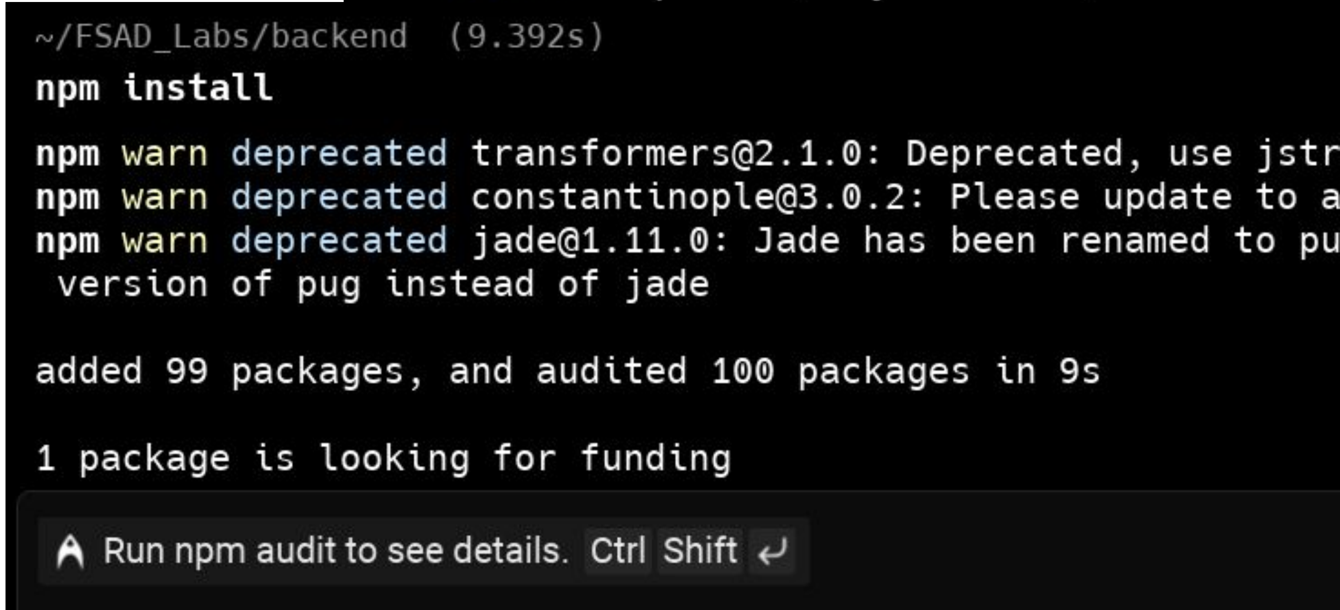
Building a Simple REST API using Node.js and Express

Installation

- Make a new folder backend
- cd backend/
- npx express-generator
- Install all packages using npm install



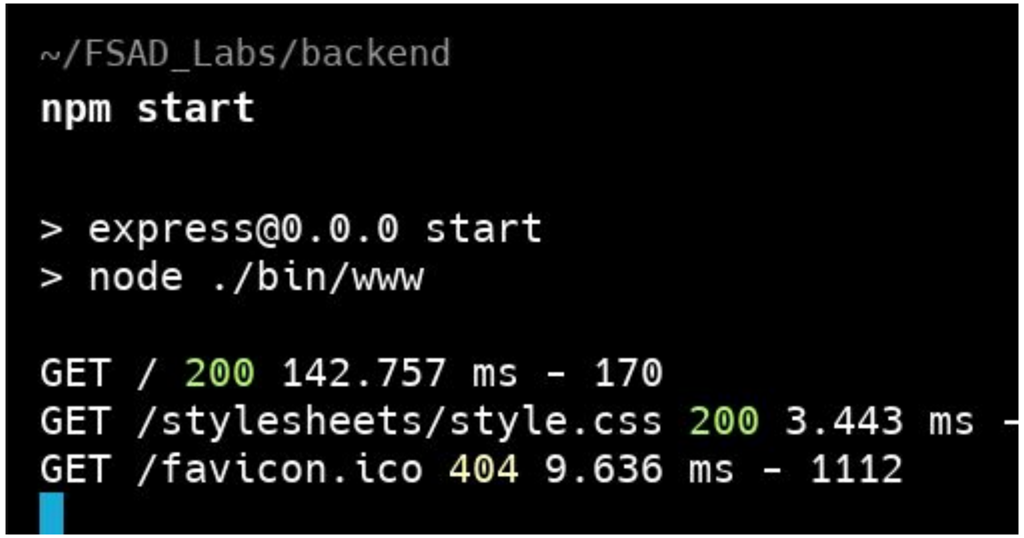
```
npx express-generator

  warning: the default view engine will r
  warning: use `--view=jade' or `--help'

   create : public/
   create : public/javascripts/
   create : public/images/
   create : public/stylesheets/
```



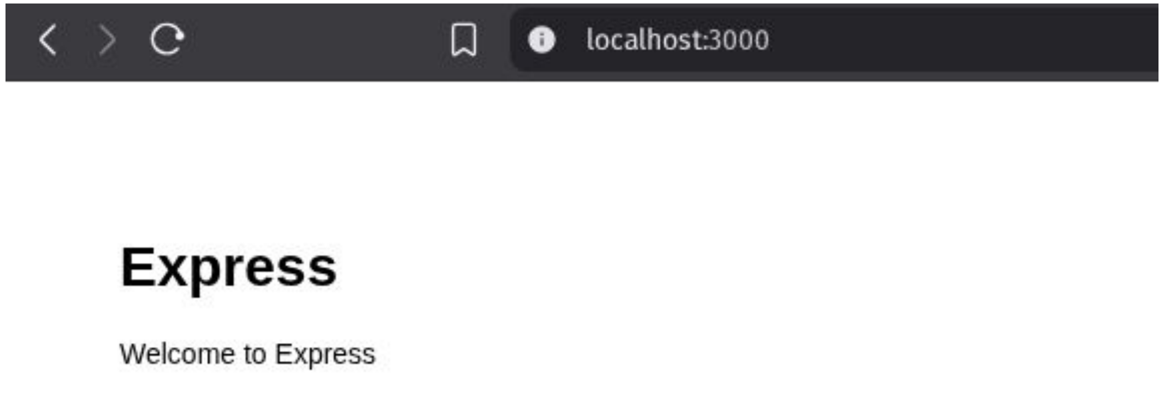
```
~/FSAD_Labs/backend  (9.392s)
npm install

npm warn deprecated transformers@2.1.0: Deprecated, use jstr
npm warn deprecated constantinople@3.0.2: Please update to a
npm warn deprecated jade@1.11.0: Jade has been renamed to pu
 version of pug instead of jade

added 99 packages, and audited 100 packages in 9s

1 package is looking for funding
```

Express App

- Start the express server using npm start
- Visit http://localhost:3000



```
~/FSAD_Labs/backend
npm start

> express@0.0.0 start
> node ./bin/www

GET / 200 142.757 ms - 170
GET /stylesheets/style.css 200 3.443 ms -
GET /favicon.ico 404 9.636 ms - 1112
```

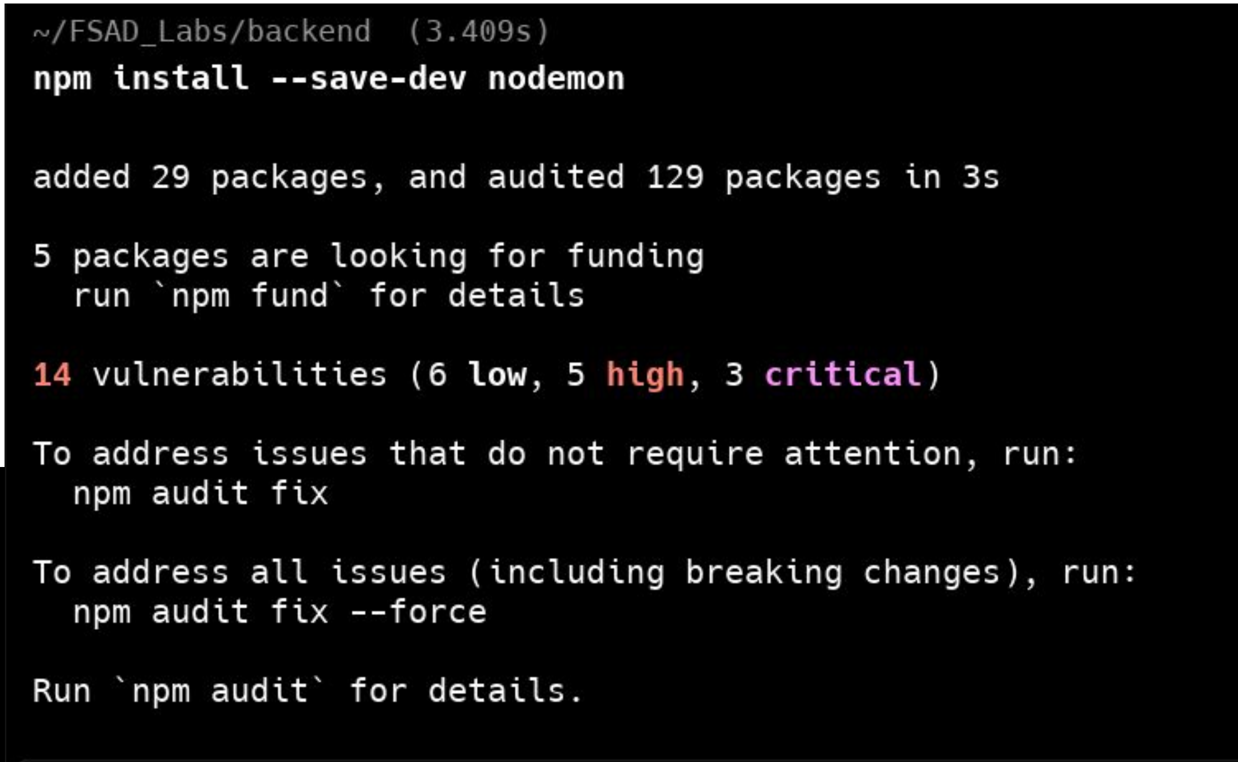


Express

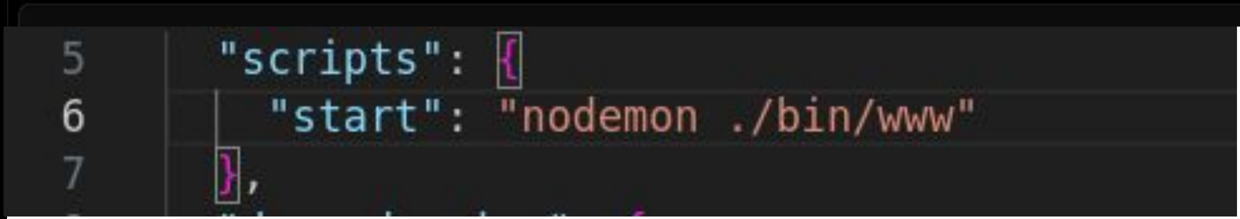
Welcome to Express

Express App

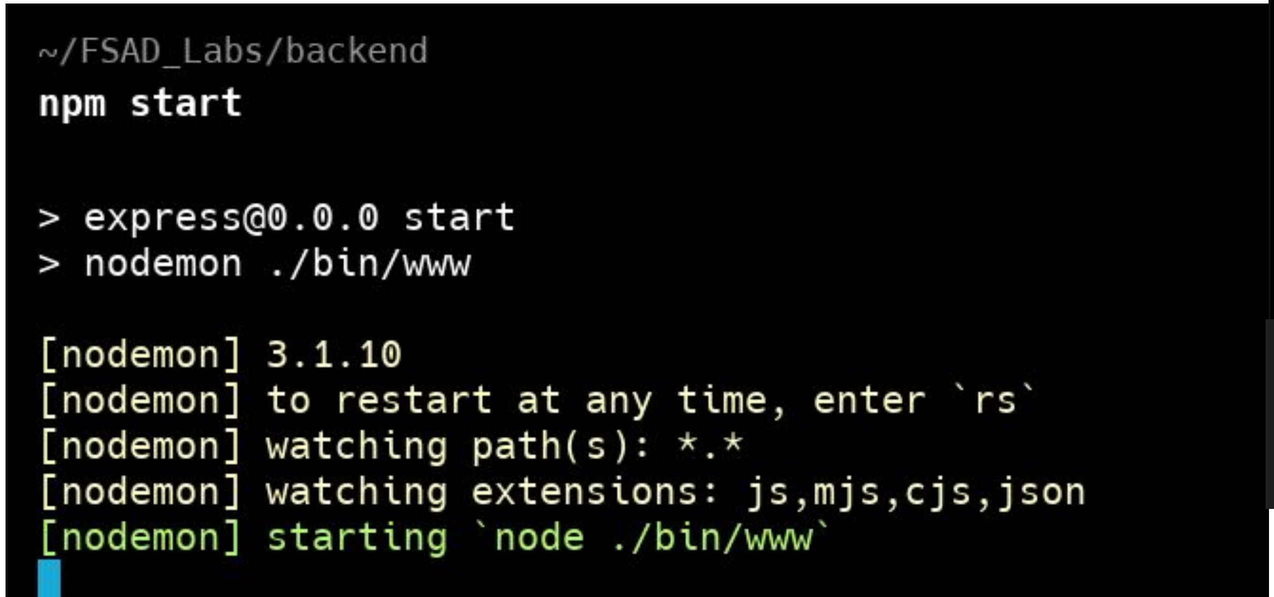
- Install nodemon for hot reload
- npm install –save-dev nodemon
- Modify start script in package.json as shown in the image.
- npm start



```
~/FSAD_Labs/backend  (3.409s)
npm install --save-dev nodemon

added 29 packages, and audited 129 packages in 3s

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

14 vulnerabilities (6 low, 5 high, 3 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```



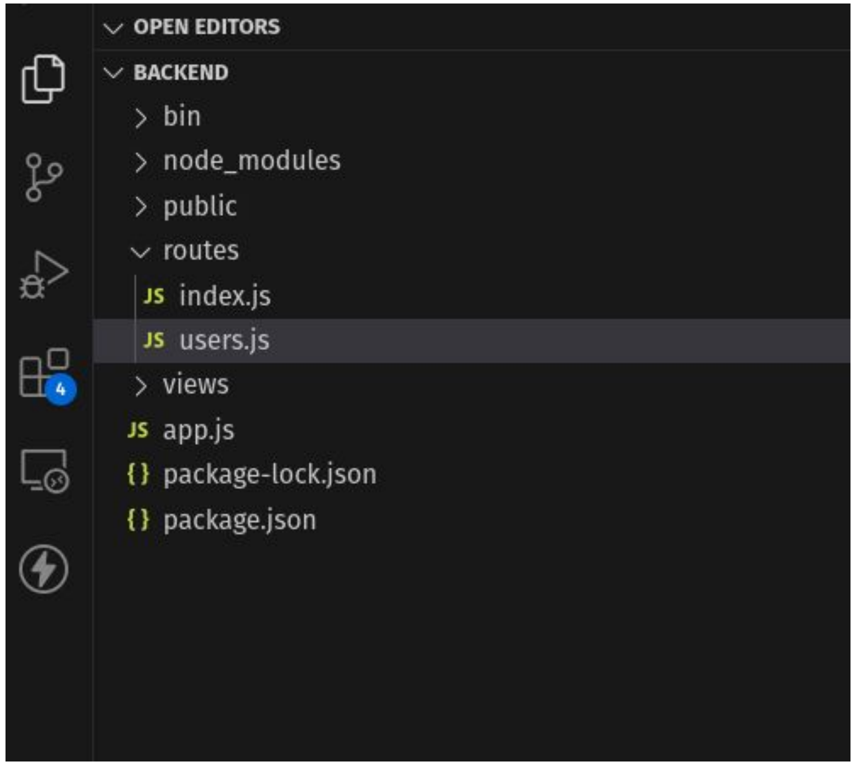
```
"scripts": {
  "start": "nodemon ./bin/www"
},
```



```
~/FSAD_Labs/backend
npm start

> express@0.0.0 start
> nodemon ./bin/www

[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node ./bin/www`
```

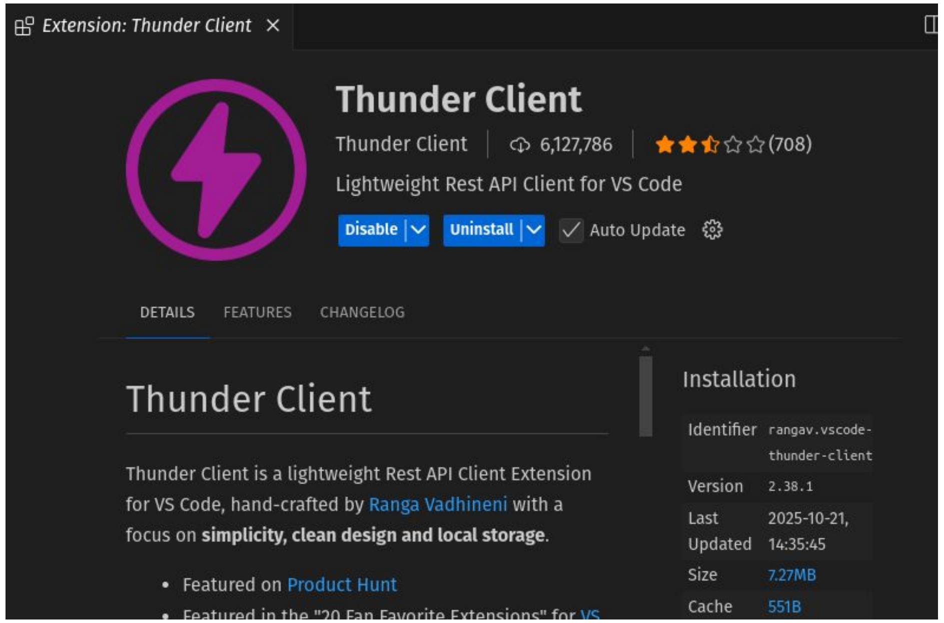
Project Structure

```
backend/
│
├── app.js          → main Express app
├── bin/
│   └── www          → starts the server
├── routes/
│   ├── index.js     → main route (/)
│   └── users.js     → /users route
├── public/         → static files
├── package.json
└── node_modules/
```


OPEN EDITORS
BACKEND
bin
node_modules
public
routes
index.js
users.js
views
app.js
package-lock.json
package.json

Testing APIs

- Install a testing application such as Postman or a simple vs-code extension like Thunder Client.
- Feel free to use curl or another packages as well if you are comfortable with.



Thunder Client

Thunder Client | 6,127,786 | (708)

Lightweight Rest API Client for VS Code

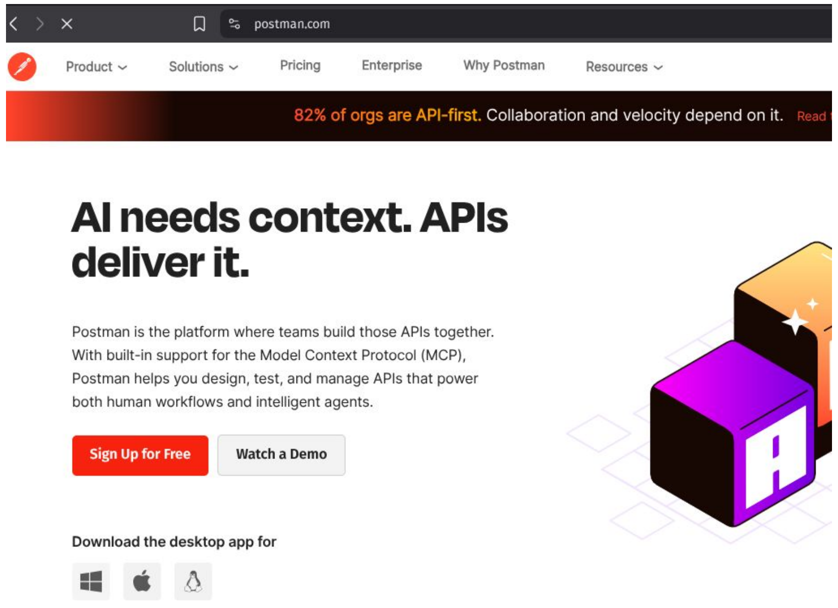
Thunder Client

Thunder Client is a lightweight Rest API Client Extension for VS Code, hand-crafted by Ranga Vadhineni with a focus on **simplicity, clean design and local storage**.

- Featured on Product Hunt
- Featured in the "20 Fan Favorite Extensions" for VS

Installation

Identifier	rangav.vscode-thunder-client
Version	2.38.1
Last Updated	2025-10-21, 14:35:45
Size	7.27MB
Cache	551B



82% of orgs are API-first. Collaboration and velocity depend on it. Read

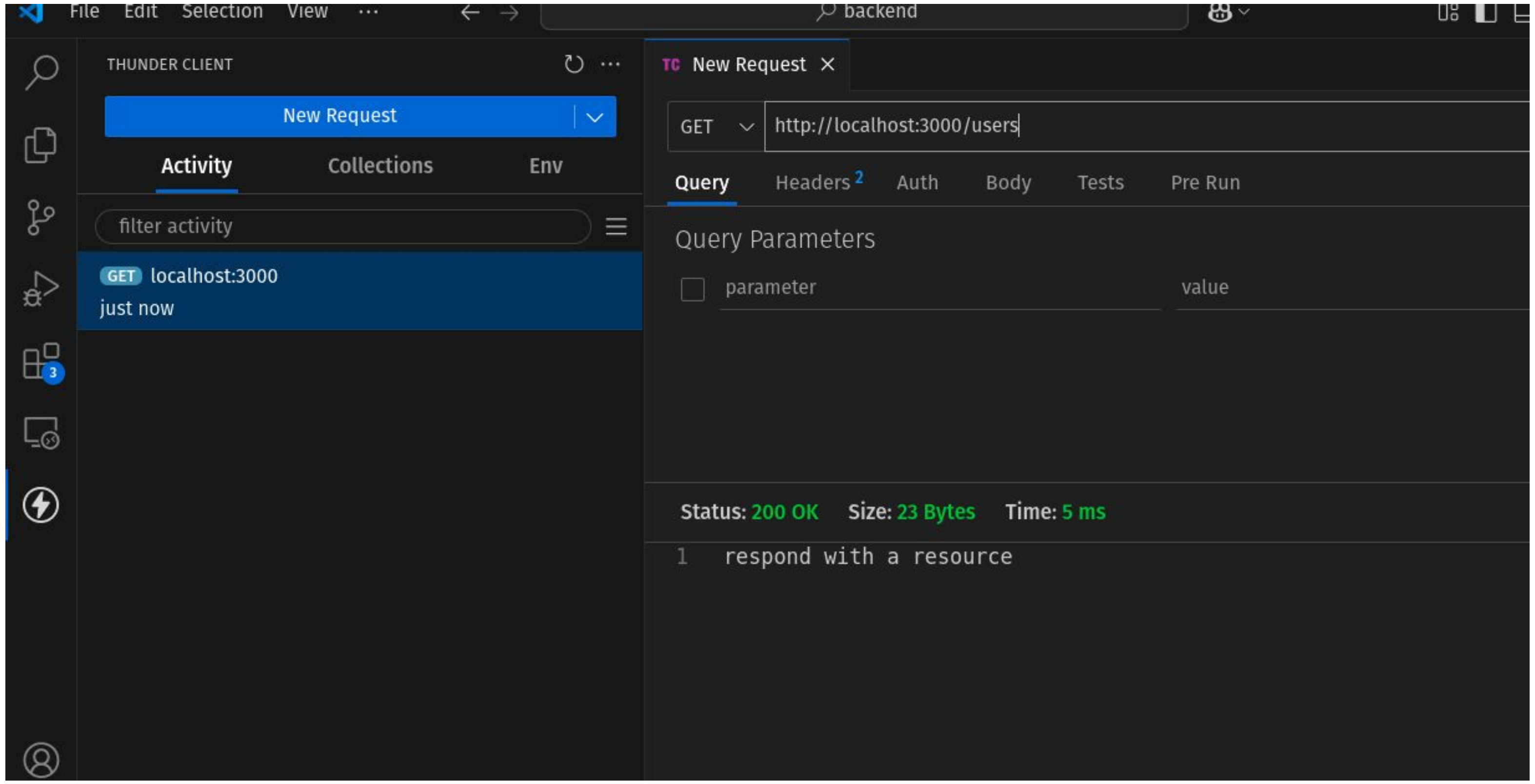
AI needs context. APIs deliver it.

Postman is the platform where teams build those APIs together. With built-in support for the Model Context Protocol (MCP), Postman helps you design, test, and manage APIs that power both human workflows and intelligent agents.

Download the desktop app for

Testing pre-existing users API

Go to Thunder client/postman and check for a GET request to http://localhost:3000



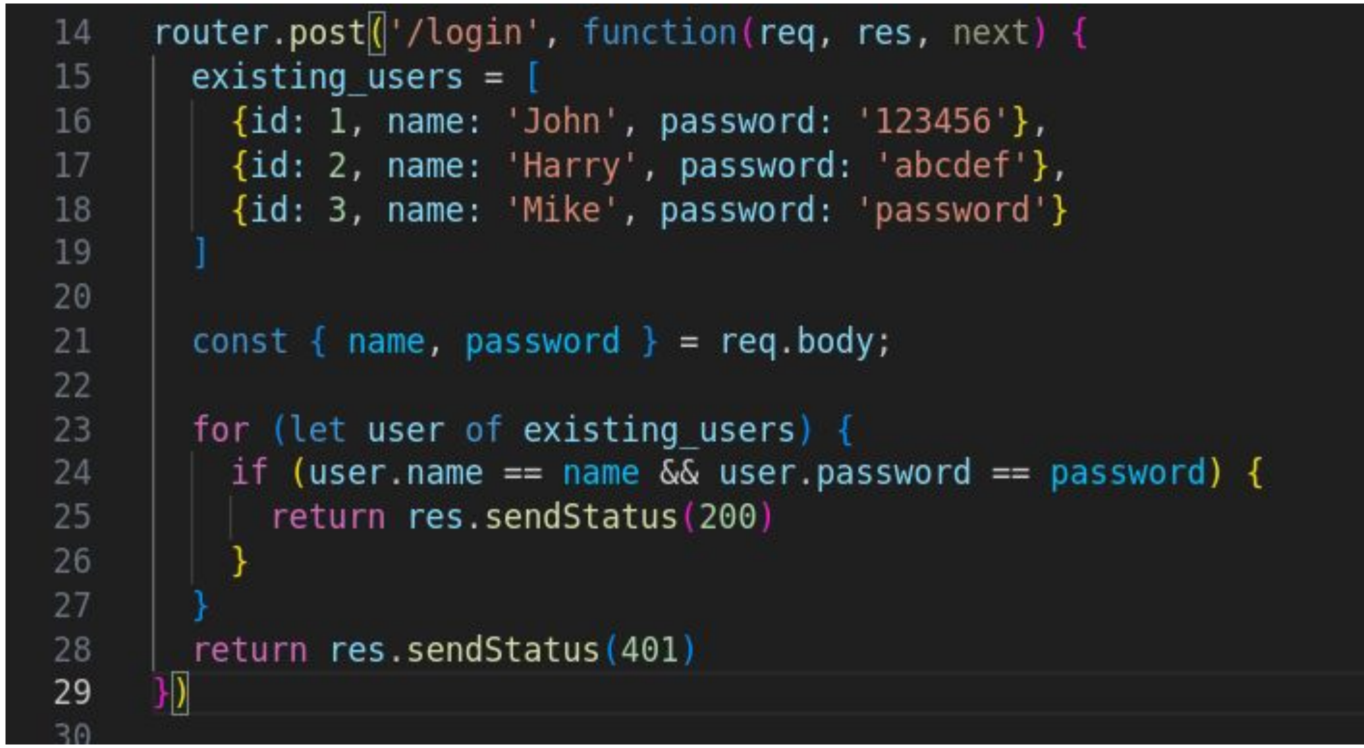
GET http://localhost:3000/users

Status: 200 OK Size: 23 Bytes Time: 5 ms

```
respond with a resource
```

Login Simulation

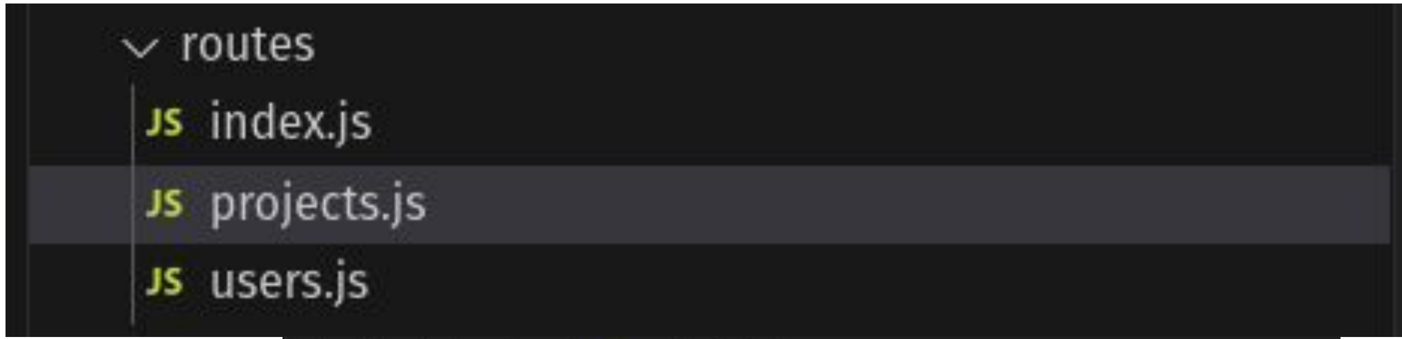
- Add a new api /login (router.post) so that we can send the login information.
- All our apis in user.js is accessible through /users/..
- This api goes through all available users and sends OK if user is found in db.
- Learn about HTTP Status Code



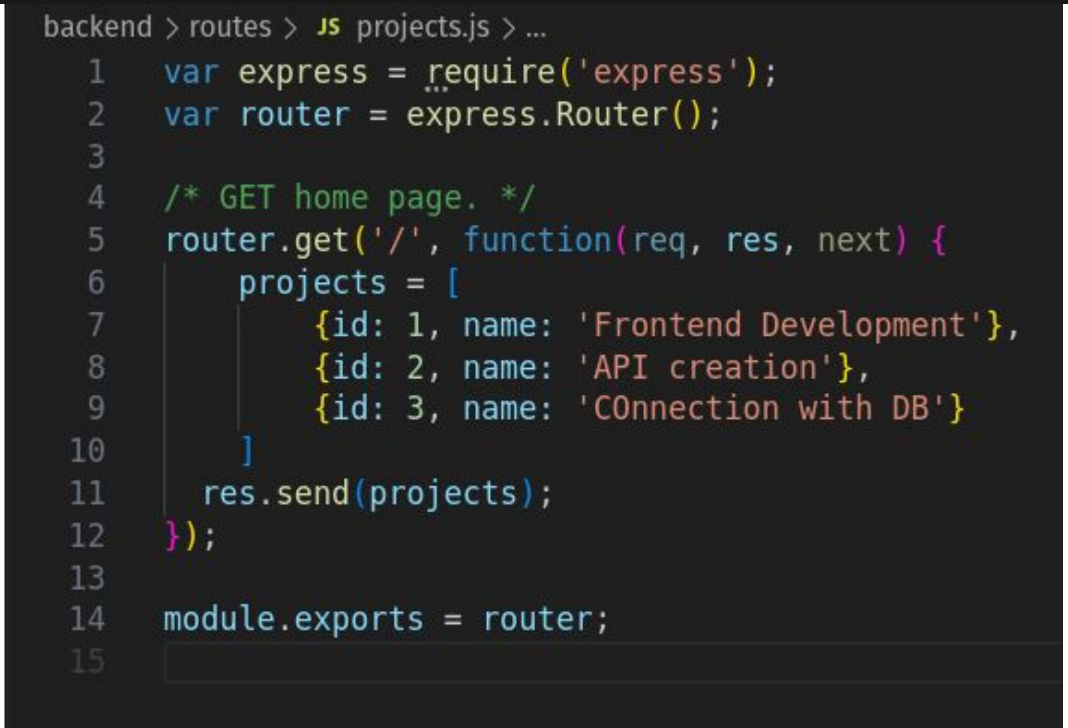
```
router.post('/login', function(req, res, next) {
  existing_users = [
    {id: 1, name: 'John', password: '123456'},
    {id: 2, name: 'Harry', password: 'abcdef'},
    {id: 3, name: 'Mike', password: 'password'}
  ]

  const { name, password } = req.body;

  for (let user of existing_users) {
    if (user.name == name && user.password == password) {
      return res.sendStatus(200)
    }
  }
  return res.sendStatus(401)
})
```

Creating Project APIs

- Inside routes/, make a new file project.js
- Inside the / route, make a simple api that returns the list of projects.
- Test the api http://localhost:3000/projects.
- Add another api and access them using /projects/<name>



```
routes
  index.js
  projects.js
  users.js
```



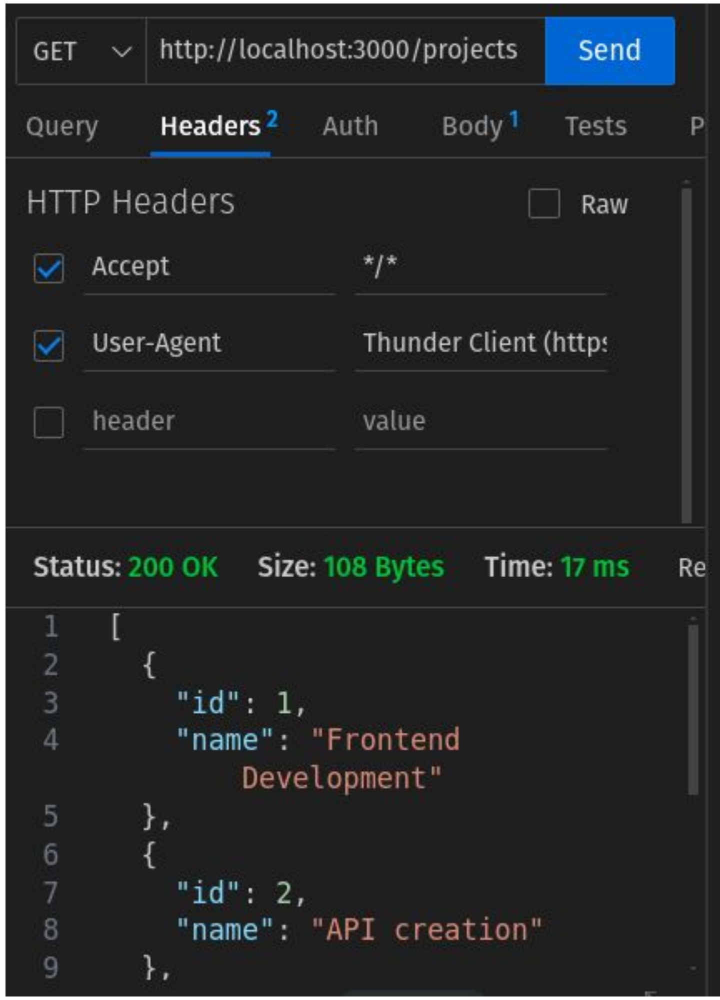
backend > routes > projects.js > ...

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
    projects = [
        {id: 1, name: 'Frontend Development'},
        {id: 2, name: 'API creation'},
        {id: 3, name: 'COnnection with DB'}
    ]
   res.send(projects);
});

module.exports = router;
```

Creating Project APIs

DIY

- Add new methods to add/delete the items from the project list.



http://localhost:3000/projects

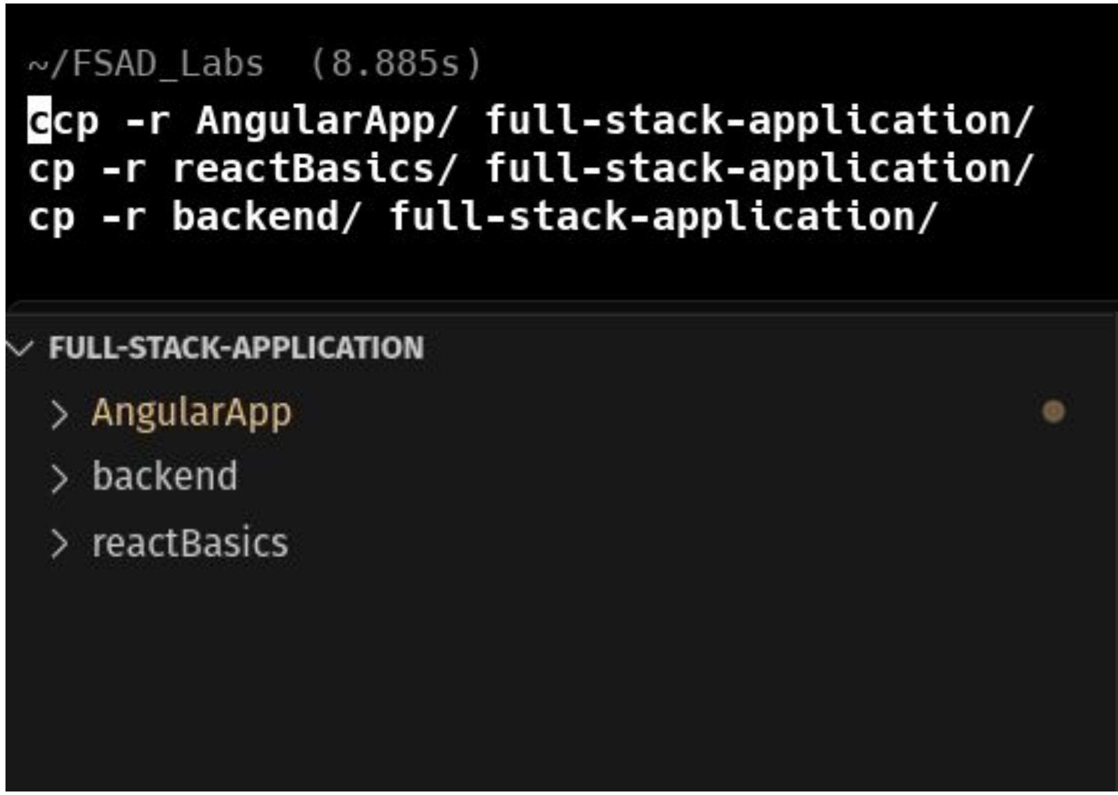
✓	Accept	*/*
✓	User-Agent	Thunder Client (https
	header	value

Status: 200 OK Size: 108 Bytes Time: 17 ms

```
[
  {
    "id": 1,
    "name": "Frontend
        Development"
  },
  {
    "id": 2,
    "name": "API creation"
  },
```

Integration with frontend/

- Make a new folder full-stack-application
- Copy your angular/react basics project and the backend folder to full-stack-application.
- Open it in vs code.

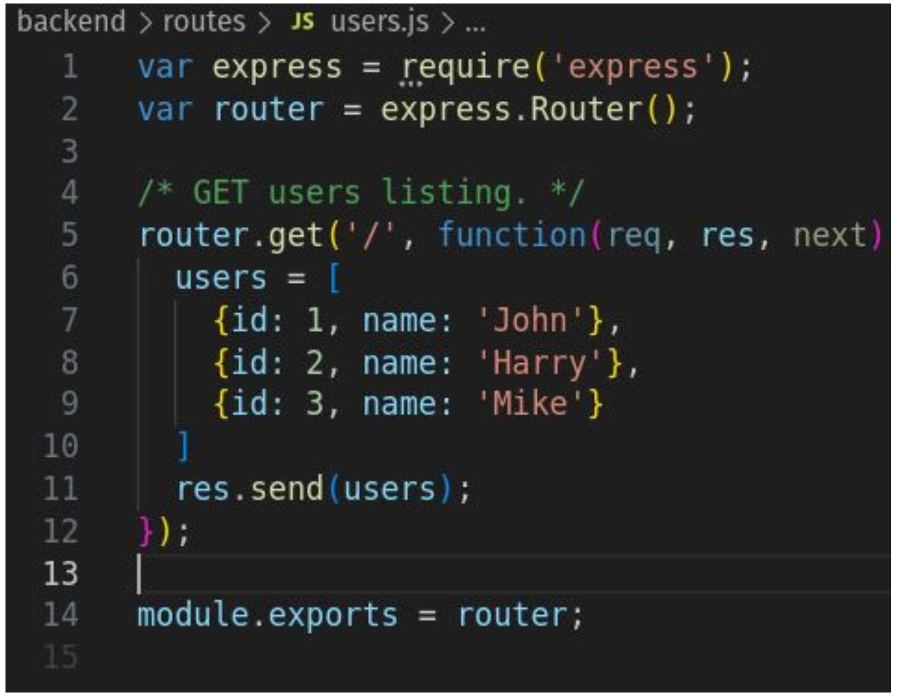


```
~/FSAD_Labs  (8.885s)
ccp -r AngularApp/ full-stack-application/
cp -r reactBasics/ full-stack-application/
cp -r backend/ full-stack-application/
```

```
FULL-STACK-APPLICATION
  > AngularApp
  > backend
  > reactBasics
```

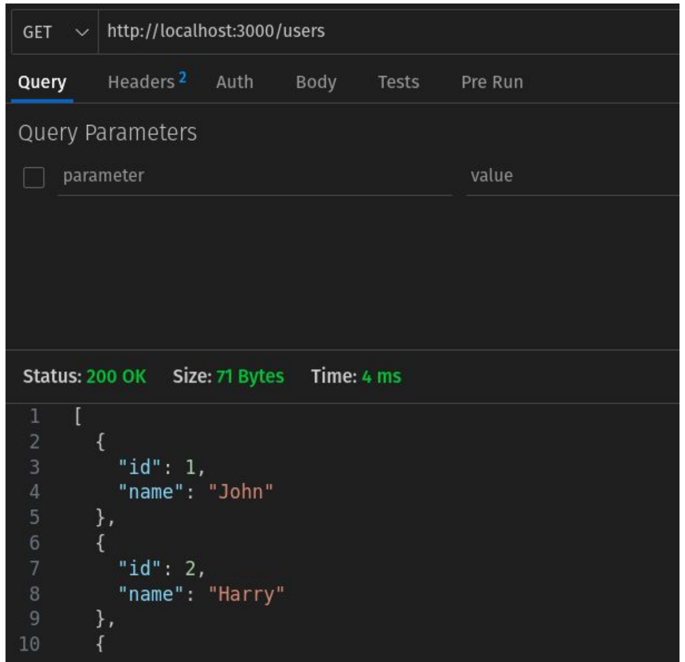
Users API

- Modify your backend/routes/users, to return an array of users instead of a message.
- Test the users api first before calling it from the frontend.



backend > routes > JS users.js > ...

```javascript
var express = require('express');
var router = express.Router();

/* GET users listing. */
router.get('/', function(req, res, next)
  users = [
    {id: 1, name: 'John'},
    {id: 2, name: 'Harry'},
    {id: 3, name: 'Mike'}
  ]
  res.send(users);
});

module.exports = router;
```



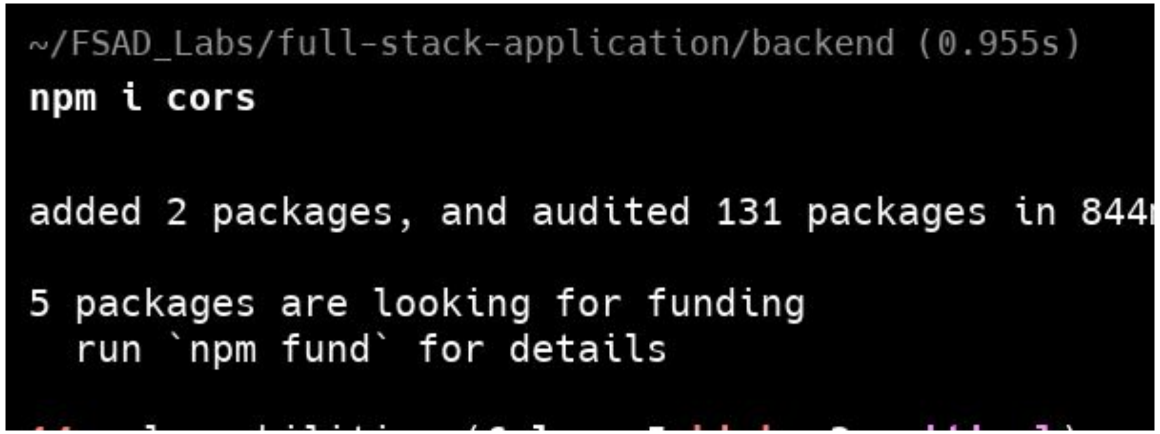
GET http://localhost:3000/users

Status: 200 OK Size: 71 Bytes Time: 4 ms

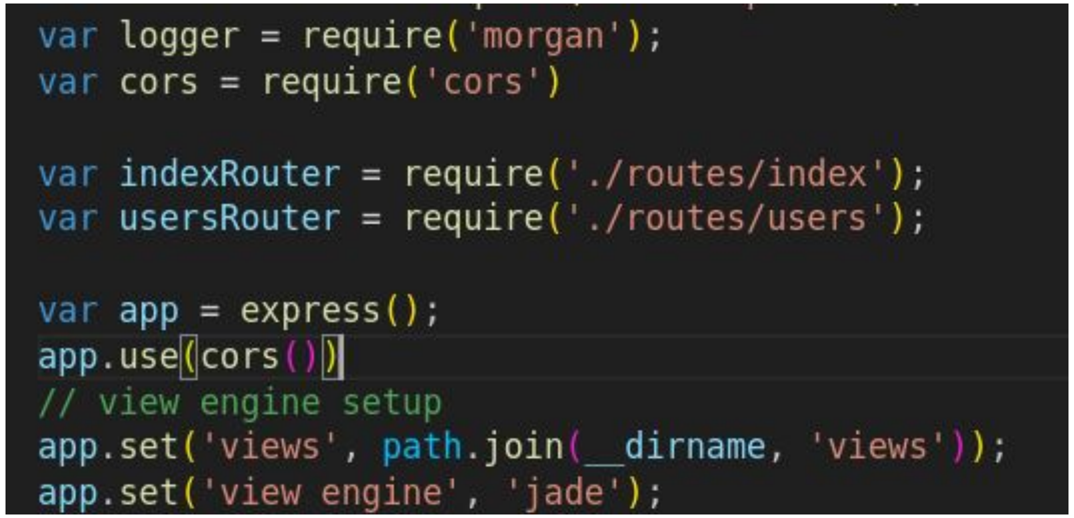
```
[
  {
    "id": 1,
    "name": "John"
  },
  {
    "id": 2,
    "name": "Harry"
  },
  {
```

Calling the users API

- Install cors package in your backend using
  npm i cors
- Modify your app.js to handle cors error.
- Add var cors = require('cors')
- Add app.use(cors()) after you define var app = express()



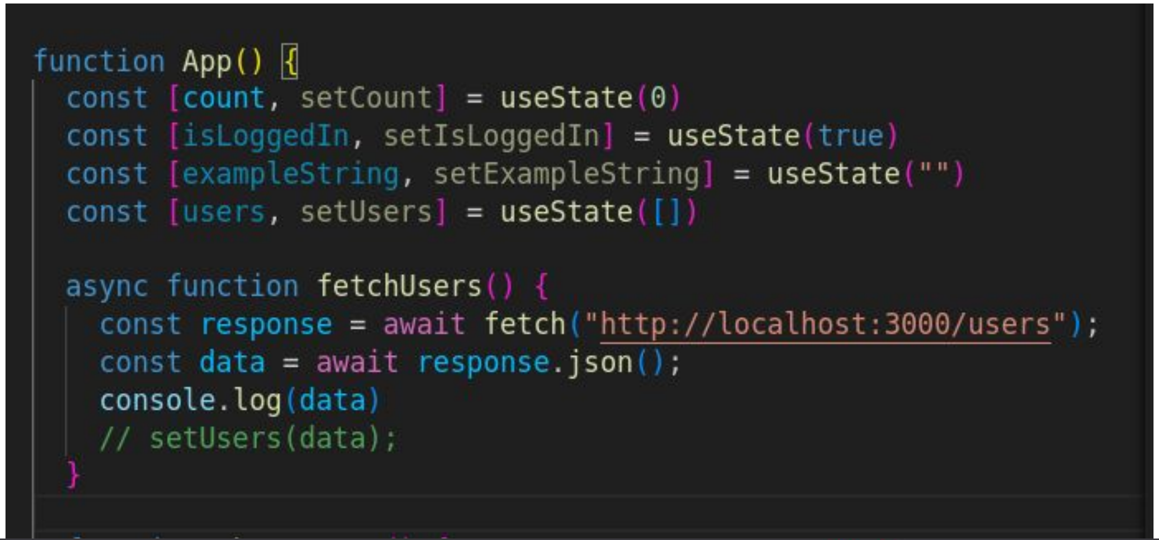
```
~/FSAD_Labs/full-stack-application/backend (0.955s)
npm i cors

added 2 packages, and audited 131 packages in 844

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



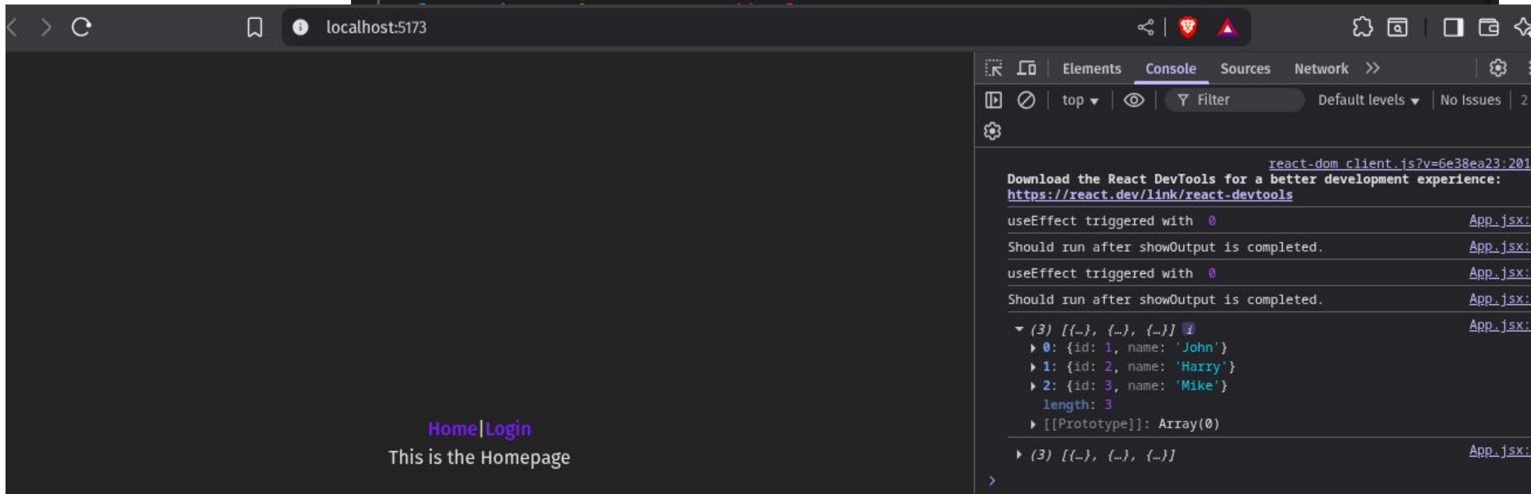
```
var logger = require('morgan');
var cors = require('cors')

var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');

var app = express();
app.use(cors())
// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');
```

Calling the users API

- Inside your reactapp, change the fetchUsers(), to fetch users from http://localhost:3000/users
- Launch you react application.
- Check you console, you will see the array of users that we sent as a response in our express app.
- *If the users [] actually came from the database, this would be a complete fsad process.*



```
function App() {
  const [count, setCount] = useState(0)
  const [isLoggedIn, setIsLoggedIn] = useState(true)
  const [exampleString, setExampleString] = useState("")
  const [users, setUsers] = useState([])

  async function fetchUsers() {
    const response = await fetch("http://localhost:3000/users");
    const data = await response.json();
    console.log(data)
    // setUsers(data);
  }
```

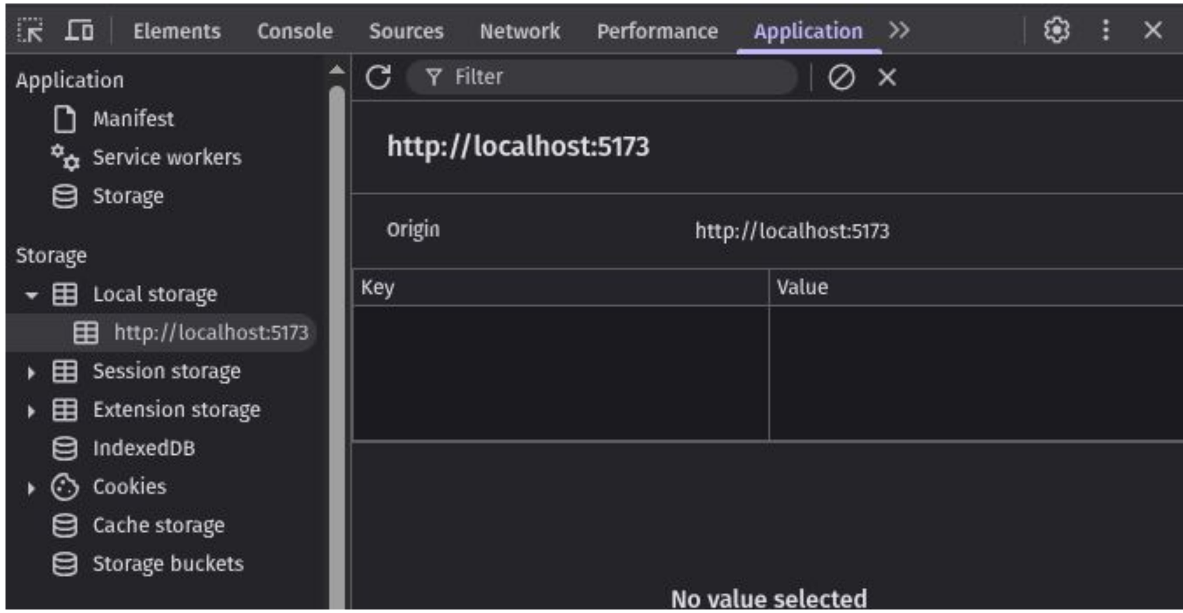


This is the Homepage

```
react-dom client.js?v=6e38ea23:201
Download the React DevTools for a better development experience:
https://react.dev/link/react-devtools
useEffect triggered with  0                              App.jsx:
Should run after showOutput is completed.                App.jsx:
useEffect triggered with  0                              App.jsx:
Should run after showOutput is completed.                App.jsx:
▾ (3) [{…}, {…}, {…}]                                    App.jsx:
  ▸ 0: {id: 1, name: 'John'}
  ▸ 1: {id: 2, name: 'Harry'}
  ▸ 2: {id: 3, name: 'Mike'}
    length: 3
  ▸ [[Prototype]]: Array(0)
▸ (3) [{…}, {…}, {…}]                                    App.jsx:
```

LocalStorage

- localStorage is a **browser storage API**.
- Stores data **as key-value pairs**.
- Data persists even if the browser is closed and reopened.



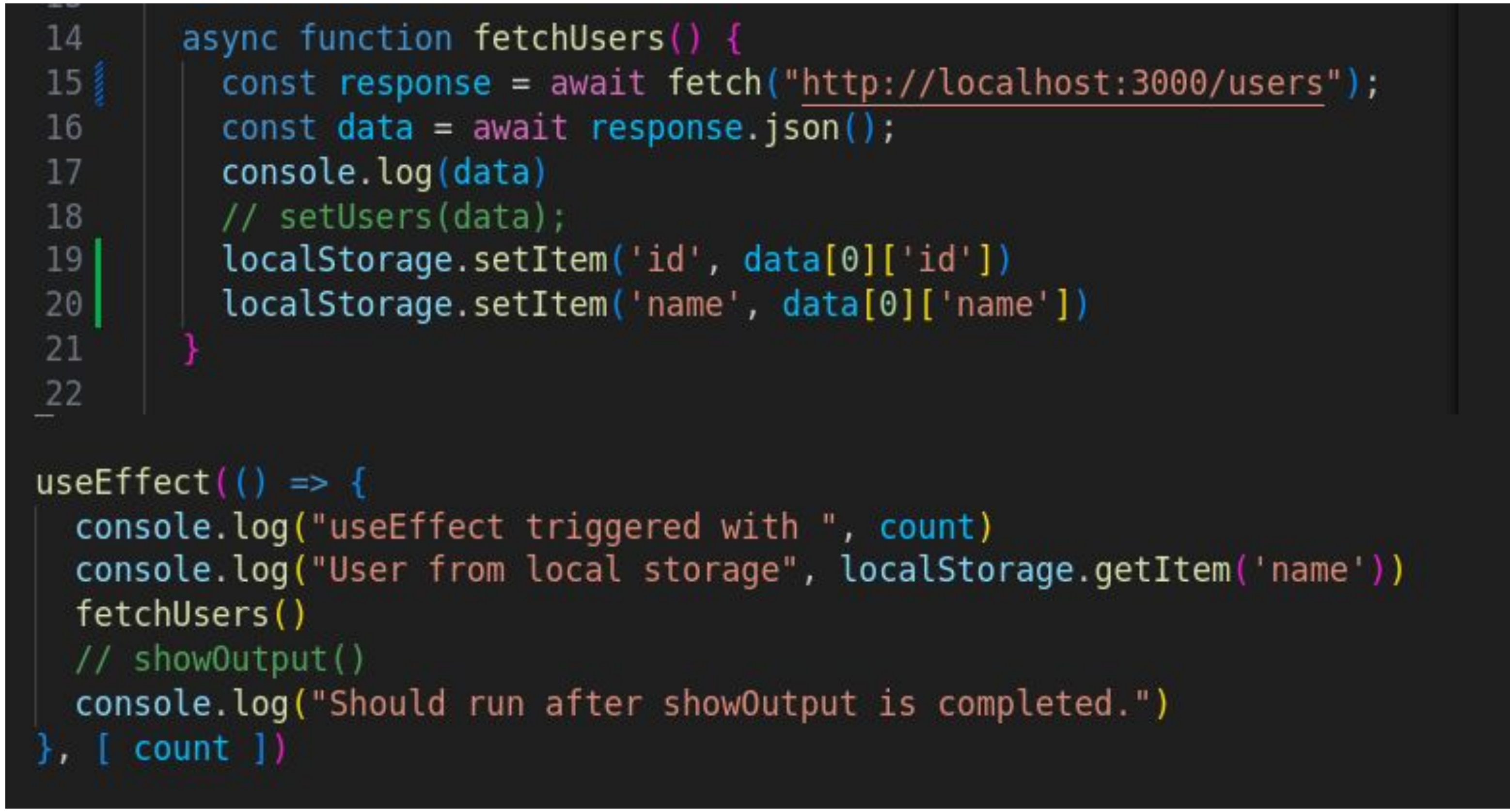
WHY?

- Save **user login status** (like “remember me”)
- Store **user preferences or theme**
- Temporary caching of **API responses**

Not secure for sensitive data like passwords — use tokens for auth.

LocalStorage

- Learn more about Localstorage
- Methods:
  - setItem(key, value)
  - getItem(key)
  - removeItem(key)
  - clear()



```
async function fetchUsers() {
  const response = await fetch("http://localhost:3000/users");
  const data = await response.json();
  console.log(data)
  // setUsers(data);
  localStorage.setItem('id', data[0]['id'])
  localStorage.setItem('name', data[0]['name'])
}

useEffect(() => {
  console.log("useEffect triggered with ", count)
  console.log("User from local storage", localStorage.getItem('name'))
  fetchUsers()
  // showOutput()
  console.log("Should run after showOutput is completed.")
}, [ count ])
```

Login.jsx

```
import React, { useState } from 'react'
import { useNavigate } from 'react-router-dom'

function Login(props) {
 const [name, setName] = useState('')
 const [password, setPassword] = useState('')
 const [error, setError] = useState('')
 const navigator = useNavigate()

 const handleSubmit = async() => {
   if (!name || !password){
     setError('All values are required')
     return;
   }

   const res = await fetch('http://localhost:3000/users/login', {
     method: 'POST',
     headers: { 'Content-Type': 'application/json' },
     body: JSON.stringify({ name, password }),
   });

   if (res.status == 200) {
     localStorage.setItem("user", 'LoggedInUser')
```

App.jsx

```
import { useEffect, useState } from 'react'
import './App.css'
import Homepage from './components/Homepage'
import Login from './components/Login'
import { Route, Routes, Link } from 'react-router-dom'

function App() {
 const [count, setCount] = useState(0)
 const [isLoggedIn, setIsLoggedIn] = useState(true)
 const [exampleString, setExampleString] = useState("")
 const [users, setUsers] = useState([])

 async function fetchUsers() {
   const response = await fetch("http://localhost:3000/users");
   const data = await response.json();
 }

 function showOutput() {
   console.log("SHowing nth")
 }

 useEffect(() => {
   console.log("useEffect triggered with ", count)
```