FAKE SERVER:

It is mirage server. Server provides backend. Whereas frontend is the client side or UI.

Mirage is used to develop fake server(backend) on the client side. The benefit is we can provide real time deployment for testing.

INSTALLATION:

*# Using npm*

npm install --save-dev miragejs

*# Using Yarn*

yarn add --dev miragejs

CREATING SERVER:

create mirage folder and in it, create index.js and here we will make fake server

Reference tutorial: [Tutorial: Part 1 – Static GET Handler • Mirage JS](https://miragejs.com/tutorial/part-1/)

import { createServer } from "miragejs"

export default function () {

createServer({

routes() {

this.get("/api/reminders", () => ({

reminders: [],

}))

},

})

}

this.get takes two arguments, one is api which we wanna use for server and other one is the function which can be some function, object, array etc.

This is the code for our server

import { createServer } from "miragejs"

export default function () {

    let arr = [{id:"1", text:"hello"}]

    let users = [{name:"Muhammad Muneeb", age: 20}]

  createServer({

    routes() {

            this.namespace="/fakeapi"

        this.get("/getTodos", {todos:arr})//(api, data)//for getting or fetching data

        //another api

        this.get("/getUsers", {todos:users})

        // this.post()//for adding the data

        // this.put()//for updating the data

        // this.delete()//for deleting the data

      },

    }

  )

}

        // this.post()//for adding the data

        // this.put()//for updating the data

        // this.delete()//for deleting the data

      },

    }

  )

}

There is nothing much, we are just passing and api to server, and in that api, we want arr to show

We have two apis or two end points of one website. One is getTodos and the other one is getUsers and we are getting data from these two apis.

Namespace is used as its value will be stick with all the apis like

/fakeapi/getTodos

Actually what we are doing is we are creating a fake server, for that we are creating a fake api and give it any name, the purpose of fake server is to show the data on UI or on client side. Now we created fake server consisting of the data. Now in order to fetch the data on UI or client side, we need to fetch the data from the fake server

CLIENT SIDE OR UI OR FETCHING DATA FROM THE FAKE SERVER:

Now its time to call this api or server from the client side.

Create components folder, in that create todo.js and in that, we are fetching data from the server to show it on UI or on client side or on the app.

import React, { useEffect } from 'react';

let baseUrl = "fakeapi"

export const Todos = () => {

    useEffect (()=>{

        //fetching data from getTodos api

        fetch(`/${baseUrl}/getTodos`)

            .then(res => res.json())

            .then(data => console.log('data', data))

        //fetching data from getUsers api

        fetch(`/${baseUrl}/getUsers`)

            .then(res => res.json())

            .then(data => console.log('data', data))

    }

)

    return (

        <div>

            Hello from Todos

        </div>

    )

}

We are using useEffect to perform async and use fetch to fetch the data from the apsi which we created on the server, then use promise and converts that data to json and then again use promise and display data on console. What does we get on console is the {todos:arr} first and then {todos:users}

Also update App.js

import React from 'react';

import './App.css';

import { Todos } from './components/Todo';

import MirageServer from './mirage';

MirageServer()

function App() {

  return (

    <div className="App">

    <Todos />

    </div>

  );

}

export default App;

Similarly, we can also post data from client side to server.

        // this.post()//for adding the data

        this.post("/addTodos",(\_,req)=>{

            console.log('req', req)

            arr.push(req.requestBody)

        } )

requestBody is used to show body of the request which we are sending on the server. And in the body, we pass the data, so req.requestBody will show data which we are passing in the body of the request.

Now again in todos.js

    const addToDoHandler = () => {

        fetch(`/${baseUrl}/addTodos`,{

            method:"POST",

            body:{id:1234, text:"text"}

        }).then(res=>{

            console.log("success", res)

        }).catch(error =>{

            console.log("Error addTodo", error)

        })

    }

    return (

        <div>

            Hello from Todos

            <button onClick={addToDoHandler}>add</button>

        </div>

    )

}

We made addToDoHandler outside useEffect, as it is not the async part. In that, we fetch api, in that api we pass method POST, as we are sending data from client side to server, in the body part, we pass the data which we want to send on the server, then we run promise in case of the success and the error. Now upon clicking on the button, this data will be pushed to the arr

**Uptill now we are visualizing data on the console. Now we want our data to be visualized on the UI**

After addToDoHandler in Todo.js, we are making getDataHandler to get or show data on the UI

  import React, { useEffect, useState } from 'react';

let baseUrl = "fakeapi"

export const Todos = () => {

    let [data, setData] = useState([]);

    useEffect (()=>{

        //fetching data from getTodos api

        fetch(`/${baseUrl}/getTodos`)

            .then(res => res.json())

            .then(data => console.log('data', data))

        //fetching data from getUsers api

        fetch(`/${baseUrl}/getUsers`)

            .then(res => res.json())

            .then(data => console.log('data', data))

    }

)

    const addToDoHandler = () => {

        fetch(`/${baseUrl}/addTodos`,{

            method:"POST",

            body:{id:1234, text:"text"}

        }).then(res=>{

            console.log("success", res)

        }).catch(error =>{

            console.log("Error addTodo", error)

        })

    }

    const getDataHandler = () => {

        fetch(`/${baseUrl}/getTodos`)

        .then(res => res.json())

        .then(data => {

            setData(data.todos)

            console.log('data', data)})

    }

    return (

        <div>

            <div>

                Hello from Todos

                <br />

                <button onClick={addToDoHandler}>add</button>

            </div>

            <div>

                <button onClick={getDataHandler}>get Todos</button>

                    {

                        JSON.stringify(data)

                    }

            </div>

        </div>

    )

}

We use useState for adding or updating the data everytime as we click on add which use addToDoHandler to post data on the arr.

    let [data, setData] = useState([]);

Then

    const getDataHandler = () => {

        fetch(`/${baseUrl}/getTodos`)

        .then(res => res.json())

        .then(data => {

            setData(data.todos)

            console.log('data', data)})

    }

Here again we are fetching getTodos api, getting data in json, and updating data using setData which uses useState.

                <button onClick={getDataHandler}>get Todos</button>

                    {

                        JSON.stringify(data)

                    }

Now by clicking on the getTodos button, getDataHandler will run which get data and we can visualize that data on UI using JSON.stringify which show that in string format. Everytime we click on add button, new data will be pushed to arr as it uses arr.push, and that data can be get or visualized on UI using getTodos button which uses getDataHandler.