**REACT.JS**

Table of Contents:

    Reacts Fragments..................................................................................1

    React Portals......................................................................................... 1

    React Refs (References).........................................................................2

    What is an Effect?..................................................................................3

    React use Reducer () .............................................................................3

    React use State () ..................................................................................4

Project ………………………………………………………………………………………………5

Outputs……………………………………………………………………………………………...18

1.React Fragments:

* We can use <> and </> instead of <div> and </div>. Because, it is an empty wrapper component. It doesn’t render any real HTML element to the DOM (Document Object Model). But it fulfills React’s/JSX requirement.
* The Code Execution is faster with use of React Fragments as Compared to the div tag.
* Less utilization of memory with the use of react Fragments instead of div tag.
* You can declare Fragments in two types: -
* <> and </>
* <Fragment> and </Fragment>

2. React Portals:

* React Portal is a first-class way to render child components into a DOM node outside of the parent DOM hierarchy defined by the component hierarchy.

3. React Refs (References):

* Refs are a function provided by react to access the DOM element and react element that you might have created on your own.
* They are used in cases where we want to change the value of the child component, without using props and all.
* They also provide us with good functionality as we can use callbacks with them.
* Depending on the type of the node, the value of the ref can defer.
* The ref attribute doesn’t have instances, and can’t be used on Functional Components.

Example: Adding Refs to DOM elements.

**import** React, { Component } from 'react';     
import { render } from 'react-dom';     
class App **extends** React.Component {     
constructor(props) {     
super(props);     
this.callRef = React.createRef();     
this.addingRefInput = **this**.addingRefInput.bind(**this**);     
}     
addingRefInput() {     
this.callRef.current.focus();     
}     
render() {     
return (     
<div>     
<h1>Hello World</h1>     
<input     
type="text"     
ref={**this**.callRef} />     
<input     
type="button"     
value="Enter text"     
onClick={**this**.addingRefInput}     
/>     
</div>    
);     
}     
}     
export **default** App;

* **Output:**   
  

4. React use Effect ():

* The use Effect () hook allows you to performs side effects in your components.
* Some examples of side components are: fetching data, directly updating the DOM and timers.

5. React use Reducer ():

* The use Reducer () can be used as a replacement for use State (), if you need “more powerful state management”.
* It allows custom state logic.
* The use Reducer () hook accepts two arguments.

* How is use Reducer () is used?

Const [state, dispatch FN] = use Reducer (reducer FN, initial State, init FN);

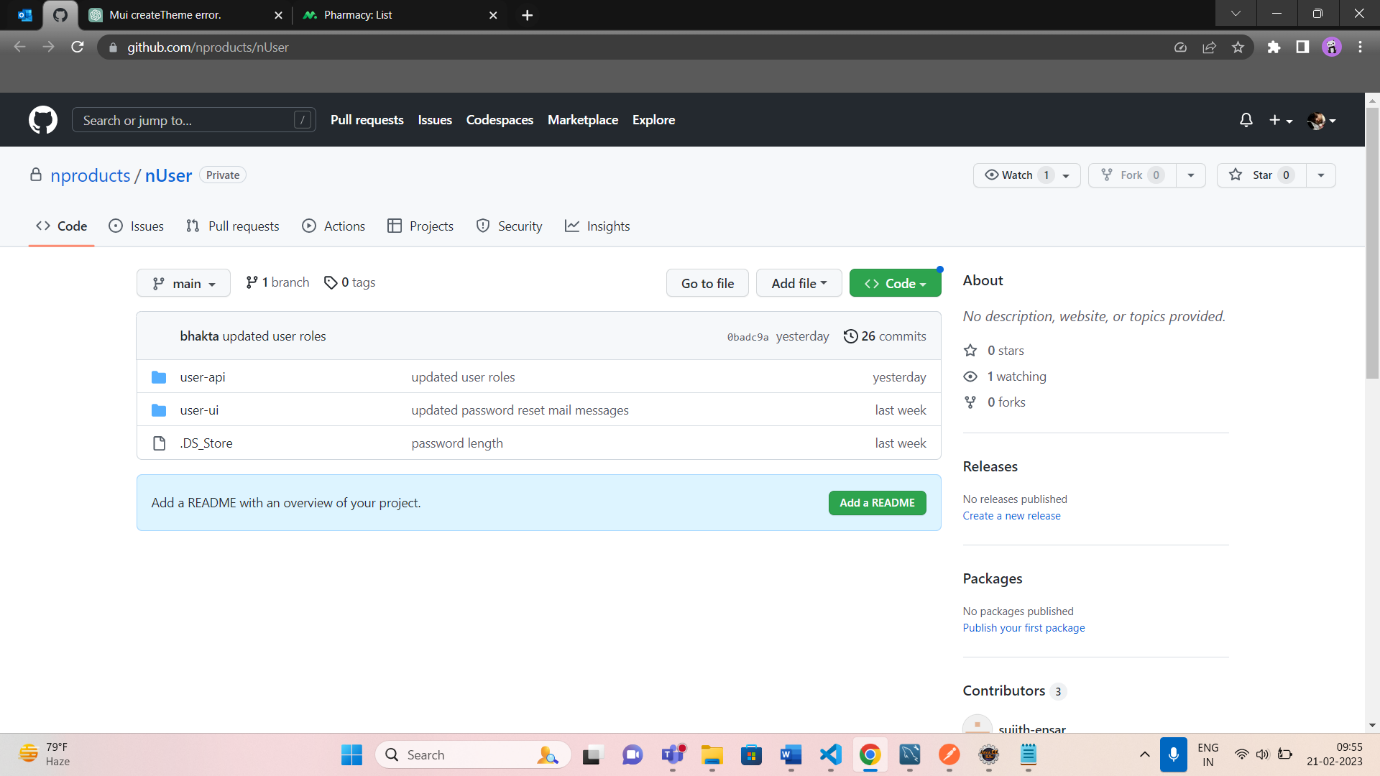
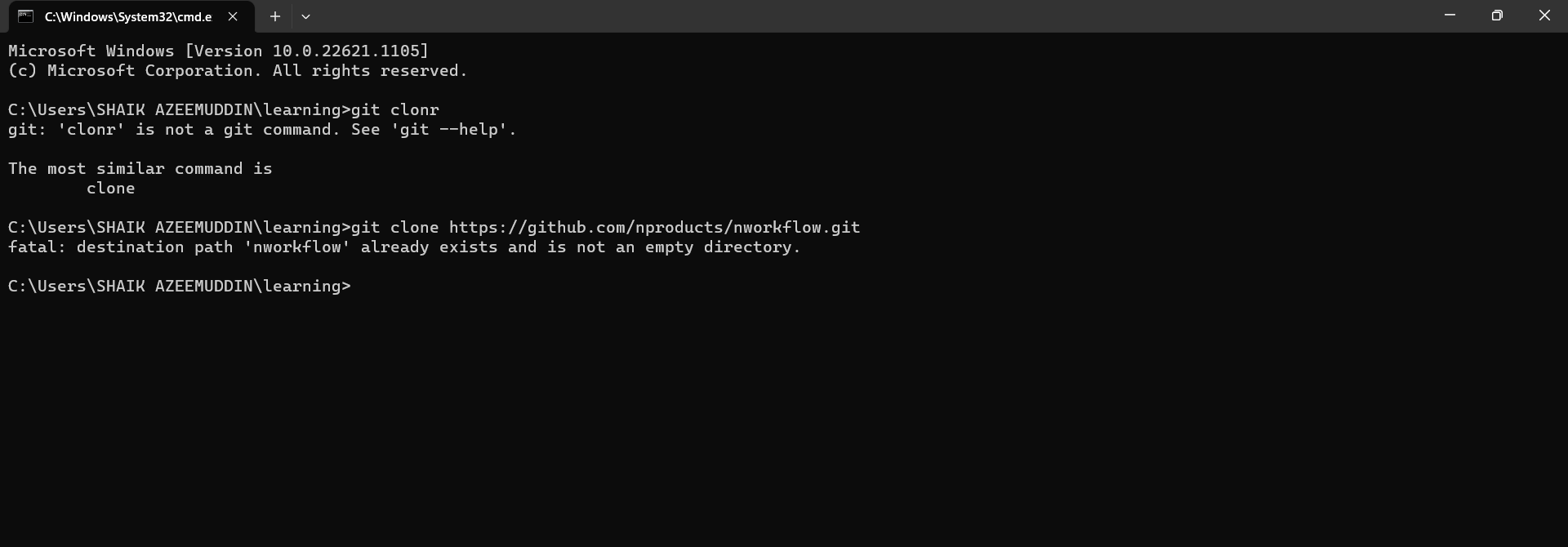
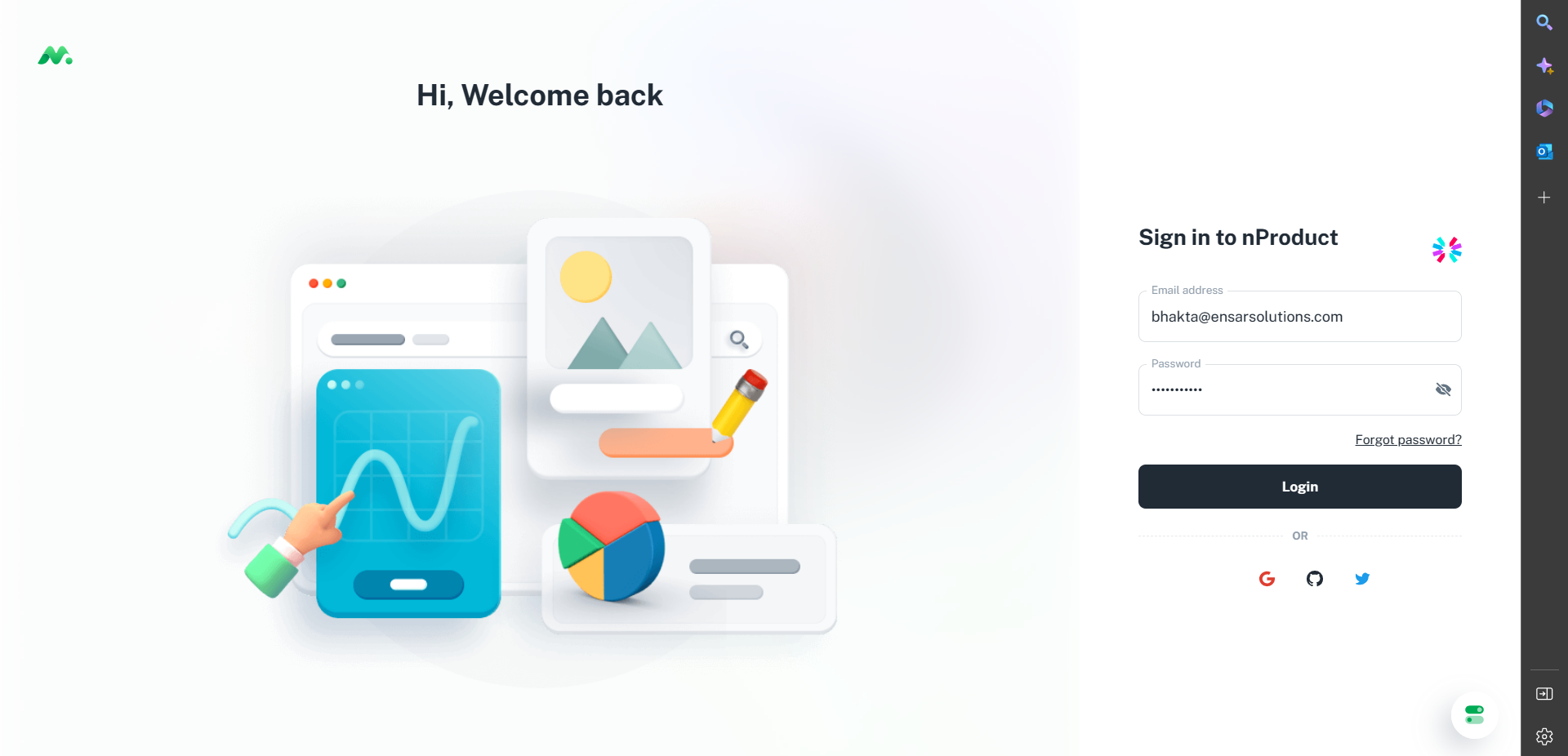
* State= The state snapshot used in the component render
* Dispatch FN= A function that is used to dispatch a new action
* Reducer FN= A function that is triggered automatically once an action is dispatched.
* Intial State= Initial state
* Init FN= A function to set the initial state programmatically.

6. React use State ():

* The main state management tool
* Great for independent use of state.
* Great if state updates are easy and limited to a few kinds of updates.
* At the top of your component, import the use State () Hook:

import {use State} from "react";

Started working on Task

* First of all cloned the existing template from Git.
* 
* Copied that link and selects the path in file manager and opens the CMD.
* 
* First, I Successfully Run the Frontend and backend
* This is the Home Page of the Login Page
* 
* First, I Cloned nUser UI And nUser from N products Organization.
* I Connected nUser UI in Visual Studio Code and Installed Node Modules in it.
* Next, I Opened NUser API Eclipse and run it as Gradle tasks.
* Next, I Connected SQL Workbench by giving SQL Password

JWT\_SECRET\_KEY=abhra\_ka\_dhabra

JWT\_TTL\_MINS=180

DB\_SERVER=localhost

DB\_PORT=3306

DB\_SCHEMA=hr

DB\_USE\_SSL=false

DB\_REQUIRE\_SSL=false

DB\_USER=root

DB\_PASSWORD=Badulla@786

EMAIL\_FROM=bhaktaensar@gmail.com

EMAIL\_USER\_NAME=bhaktaensar@gmail.com

EMAIL\_USER\_PWD=vtsxyxinqthocdnf

EMAIL\_HOST=smtp.gmail.com

EMAIL\_PORT=587

AWS\_ACCESS\_KEY=<access\_key>

AWS\_SECRET\_KEY=<access\_secret>

AWS\_ACCOUNT\_ID=<aws\_act>

AWS\_USER\_ARN=<arn\_url>

spring.redis.host=localhost

spring.redis.port=6379

spring.redis.database=0

kafka.bootstrapAddress=localhost:29092

management.tracing.sampling.probability=1.0

api.version=1.0-SNAPSHOT

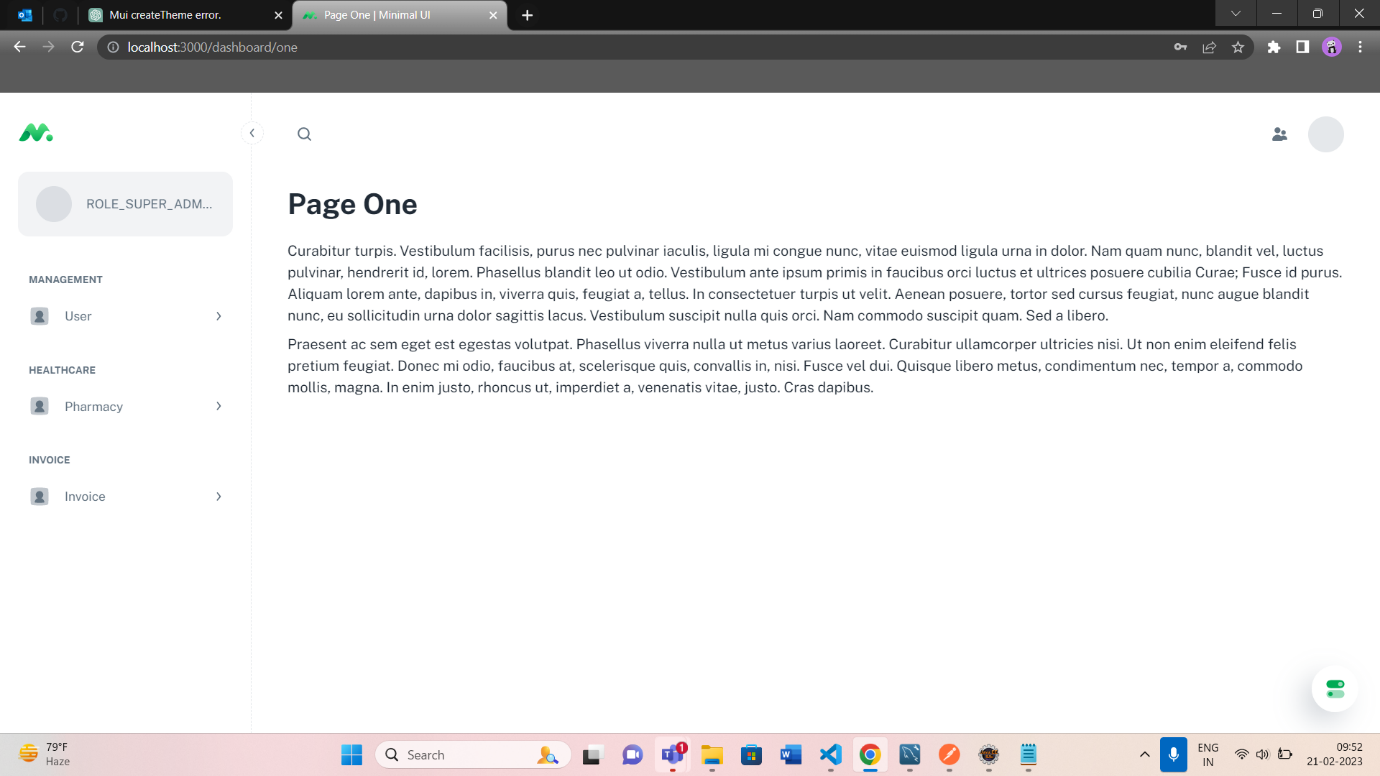
tos.uri=terms-of-service

api.server.url=https://www.ensar.com

api.description=The Product API is used to create, update, and delete users. Users can be created with or without an associated account. If an account is created, the user will be granted the <strong>ROLE\_USER</strong> role. If an account is not created, the user will be granted the <b>ROLE\_USER</b> role.

springdoc.swagger-ui.operationsSorter=alpha

springdoc.swagger-ui.tagsSorter=alpha

* 
* After cloning frontend backend working on my task.

Code:

Config-navigation.ts

// routes

import { PATH\_DASHBOARD } from '../../../routes/paths';

// components

import SvgColor from '../../../components/svg-color';

// ----------------------------------------------------------------------

const icon = (name: string) => (

  <SvgColor src={`/assets/icons/navbar/${name}.svg`} sx={{ width: 1, height: 1 }} />

);

const ICONS = {

  user: icon('ic\_user'),

  ecommerce: icon('ic\_ecommerce'),

  analytics: icon('ic\_analytics'),

  dashboard: icon('ic\_dashboard'),

};

const addingZero = (val: number) => (val < 10 ? `0${val}` : val);

const today = new Date();

const formatDate = `${today.getFullYear()}-${addingZero(today.getMonth() + 1)}-${today.getDate()}`;

const navConfig = [

  // GENERAL

  // ----------------------------------------------------------------------

  // {

  //   subheader: 'dashboard',

  //   items: [

  //     { title: 'One', path: PATH\_DASHBOARD.one, icon: ICONS.dashboard },

  //     { title: 'Two', path: PATH\_DASHBOARD.two, icon: ICONS.ecommerce },

  //     { title: 'Three', path: PATH\_DASHBOARD.three, icon: ICONS.analytics },

  //   ],

  // },

  // MANAGEMENT

  // ----------------------------------------------------------------------

  {

    subheader: 'management',

    items: [

      {

        title: 'user',

        path: PATH\_DASHBOARD.user.root,

        icon: ICONS.user,

        children: [

          { title: 'list', path: PATH\_DASHBOARD.user.list },

          { title: 'create', path: PATH\_DASHBOARD.user.new },

          { title: 'role', path: PATH\_DASHBOARD.user.role },

          { title: 'ecommerce', path: PATH\_DASHBOARD.user.new },

        ],

      },

    ],

  },

  {

    subheader: 'Demo',

    items: [

      {

        title: 'All',

        path: PATH\_DASHBOARD.demo.root,

        icon: ICONS.user,

        children: [

          { title: 'list', path: PATH\_DASHBOARD.demo.list },

          { title: 'create', path: PATH\_DASHBOARD.demo.new, roles: ['ROLE\_SUPER\_ADMIN'] },

        ],

      },

    ],

  },

  // MANAGEMENT

  // ----------------------------------------------------------------------

  {

    subheader: 'invoice',

    items: [

      {

        title: 'Invoice',

        path: PATH\_DASHBOARD.invoice.root,

        icon: ICONS.user,

        children: [

          { title: 'list', path: PATH\_DASHBOARD.invoice.list },

          { title: 'details', path: PATH\_DASHBOARD.invoice.demoView },

          { title: 'create', path: PATH\_DASHBOARD.invoice.new },

          { title: 'edit', path: PATH\_DASHBOARD.invoice.demoEdit },

        ],

      },

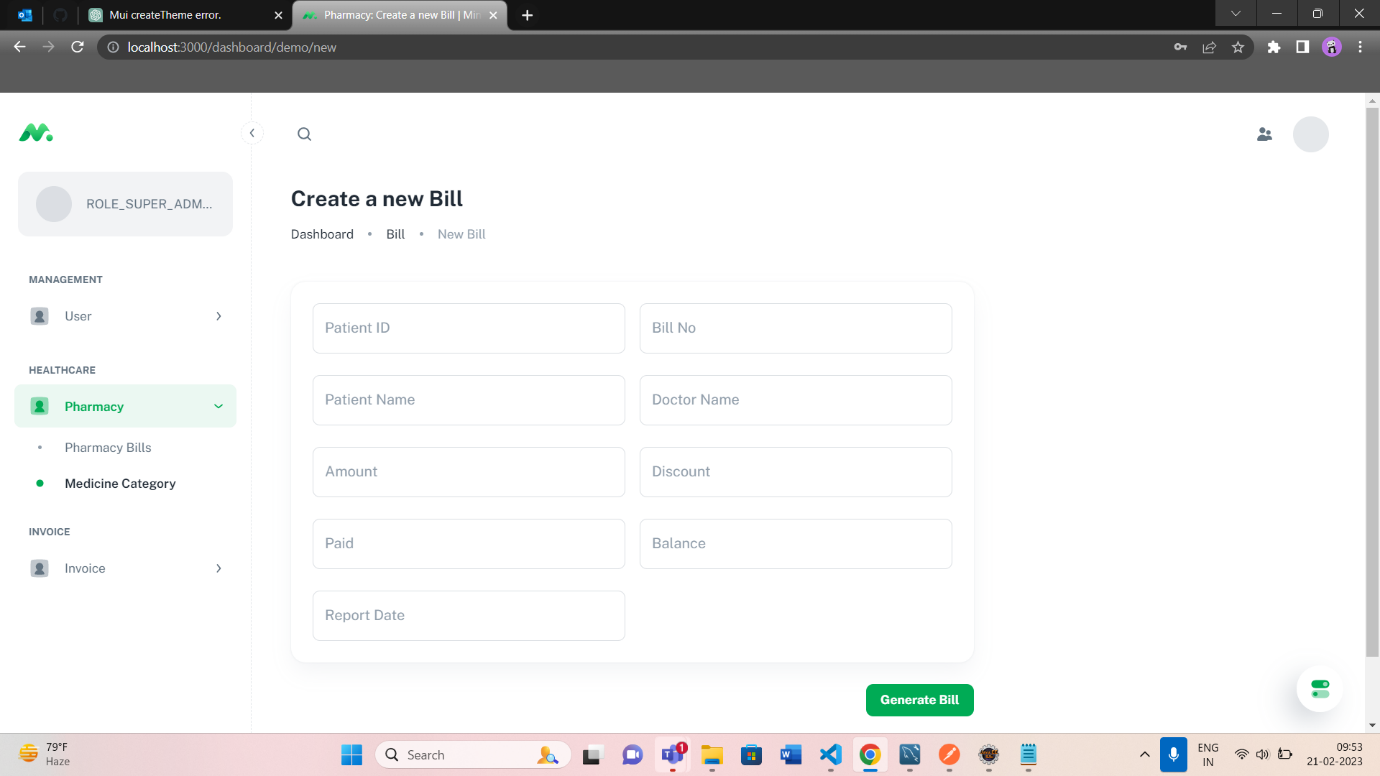
    ],

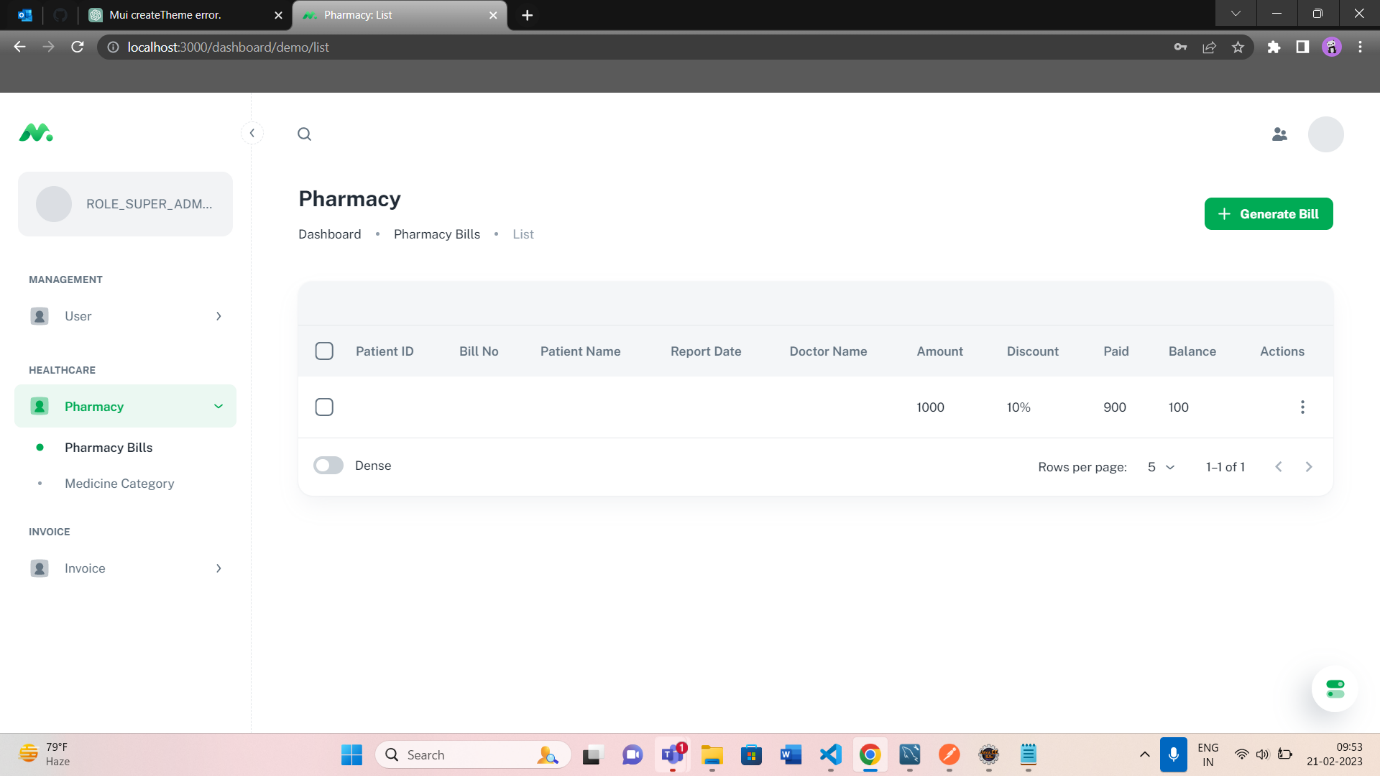
  },

];

export default navConfig;

OutPut:





Thank You,