Frontend

To make simple react frontend

git clone --single-branch -b react-mini <https://github.com/safak/youtube.git>

**LIBRARIES:**

Yarn add styled-components

Yarn add @material-ui/core

Yarn add react-stripe-checkout

Yarn add axios

yarn add @reduxjs/toolkit react-redux

**Yarn add redux-persist**

**Using Urbanist Font family:**

Fonts.google.com

Urbanist

Select all styles

Copy link from right side and also font family

Update index.html in public folder as;

<link href="https://fonts.googleapis.com/css2?family=Urbanist:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&display=swap" rel="stylesheet">

  <style>

    \* {

      margin: 0;

      font-family: 'Urbanist', sans-serif;

    }

  </style>

**History in react router 6 or Navigate or useNavigate and useLocation**

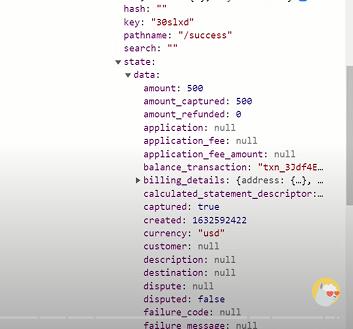
const history = useNavigate();

history('/success', {data: payment.data})

**The data we have passed in history can be accessed using useLocation**

  const location = useLocation();

  console.log(location)



**Persist login or login or keep user login or store with user login persist**

1. **Earlier store**

**Store:**

import {configureStore} from '@reduxjs/toolkit';

import cartReducer from './cartRedux'

import userReducer from './userRedux'

export default configureStore({

    reducer: {

        cart: cartReducer,

        user: userReducer,

    }

})

1. **With persist login**

**Yarn add redux-persist**

import { configureStore, combineReducers } from "@reduxjs/toolkit";

import cartReducer from "./cartRedux";

import userReducer from "./userRedux";

import {

  persistStore,

  persistReducer,

  FLUSH,

  REHYDRATE,

  PAUSE,

  PERSIST,

  PURGE,

  REGISTER,

} from "redux-persist";

import storage from "redux-persist/lib/storage";

const persistConfig = {

  key: "root",

  version: 1,

  storage,

};

    //if any reducer without persist on reloading and any with persist then

    // const persistedReducer = persistReducer(persistConfig, userReducer);

    // export const store = configureStore({

    //   reducer: {

    //         user: persistedReducer,

    //         cart:cartReducer         //for example: not persist on reloading

    // },

    //   //if any reducer with persist

    //   reducer:{

    //       user: persist

    //   }

    //   middleware: (getDefaultMiddleware) =>

    //     getDefaultMiddleware({

    //       serializableCheck: {

    //         ignoredActions: [FLUSH, REHYDRATE, PAUSE, PERSIST, PURGE, REGISTER],

    //       },

    //     }),

    // });

    //with all reducers persist on reloading

    const rootReducer = combineReducers({ user: userReducer, cart: cartReducer });

    const persistedReducer = persistReducer(persistConfig, rootReducer);

    export const store = configureStore({

      reducer: persistedReducer,

      middleware: (getDefaultMiddleware) =>

        getDefaultMiddleware({

          serializableCheck: {

            ignoredActions: [FLUSH, REHYDRATE, PAUSE, PERSIST, PURGE, REGISTER],

          },

        }),

    });

    export let persistor = persistStore(store);

**Updated Index.js**

import React from 'react';

import ReactDOM from 'react-dom';

import App from './App';

import {Provider} from 'react-redux';

import {store,persistor} from './redux/store'

import { PersistGate } from 'redux-persist/integration/react'

ReactDOM.render(

  <Provider store={store}>

    <PersistGate loading={null} persistor={persistor}>

      <App />

    </PersistGate>

  </Provider>,

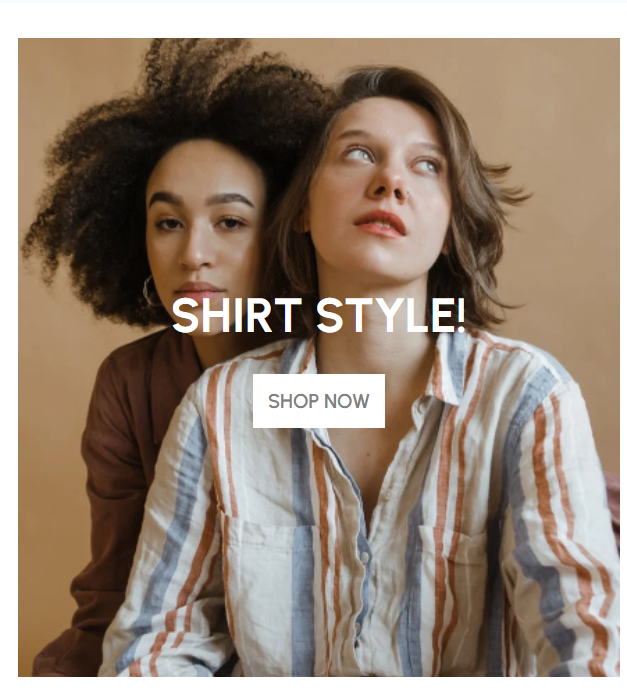
  document.getElementById('root')

);

**\*\*\*\*\*\*\*\*\*\*\*REDUX\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**yarn add @reduxjs/toolkit react-redux**

**Text on Image:**



For text and button section, make it in a div

Div should be absolute and

const Container = styled.div`

    flex: 1;

    margin: 3px;

    height: 70vh;

    position: relative;

`

const Info = styled.div`

    position: absolute;

    width: 100%;

    height: 100%;

    top: 0;

    left: 0;

    display: flex;

    align-items: center;

    justify-content: center;

    flex-direction: column;

`

<Container>

        <Image src={item.img}/>

        <Info>

            <Title>{item.title}</Title>

            <Button>SHOP NOW</Button>

        </Info>

    </Container>

It’s complete code is

import React from 'react'

import styled from 'styled-components'

const Container = styled.div`

    flex: 1;

    margin: 3px;

    height: 70vh;

    position: relative;

`

const Image = styled.img`

    width: 100%;

    height: 100%;

    object-fit: cover;

`

const Info = styled.div`

    position: absolute;

    width: 100%;

    height: 100%;

    top: 0;

    left: 0;

    display: flex;

    align-items: center;

    justify-content: center;

    flex-direction: column;

`

const Title = styled.h1`

    color: white;

    margin-bottom: 20px;

`

const Button = styled.button`

    border: none;

    padding: 10px;

    background-color: white;

    color: gray;

    cursor: pointer;

    font-weight: 600;

`

function CategoryItem({item}) {

  return (

    <Container>

        <Image src={item.img}/>

        <Info>

            <Title>{item.title}</Title>

            <Button>SHOP NOW</Button>

        </Info>

    </Container>

  )

}

export default CategoryItem

**USING STYLED COMPONENTS:**

Yarn add styled-components

import React from 'react'

import styled from 'styled-components'

const Container = styled.div`

    height: 60px;

    background-color: black;

`

function Navbar() {

  return (

    <Container>

        Navbar

    </Container>

  )

}

export default Navbar

We create container styled component targeting div, and instead of div in html, we used container

**To divide 3 columns or divs or div in equal spaces:**

const Left = styled.div`

    flex: 1;

`

const Center = styled.div`

    flex: 1;

`

const Right = styled.div`

    flex: 1;

`

Use of Props in Styled components:

const Arrow = styled.div`

    width: 50px;

    height: 50px;

    background-color: #fff7f7;

    border-radius: 50%;

    display: flex;

    align-items: center;

    justify-content: center;

    position: absolute;

    top: 0;

    bottom: 0;

    left: ${props => props.direction === "left" && "10px"};

    right: ${props => props.direction === "right" && "10px"};

    margin: auto;

`

function Slider() {

  return (

    <Container>

        <Arrow direction="left">

            <ArrowLeftOutlined/>

        </Arrow>

        <Arrow direction="right">

            <ArrowRightOutlined/>

        </Arrow>

HOVER IN STYLED COMPONENTS:

const Icon = styled.div`

    width: 40px;

    height: 40px;

    border-radius: 50%;

    background-color: white;

    display: flex;

    align-items: center;

    justify-content: center;

    margin: 10px;

    cursor: pointer;

    transition: all 0.5s ease;

    &:hover{

        background-color: #e9f5f5;

        transform: scale(1.1);

    }

IMPACT OTHER CLASS ON HOVER:

Change info class when hover on container:

const Info = styled.div`

    opacity: 0;

    width: 100%;

    height: 100%;

    position: absolute;

    top: 0;

    left: 0;

    background-color: rgba(0,0,0,0.2);

    z-index: 3;

    display: flex;

    align-items: center;

    justify-content: center;

    transition: all 0.5s ease;

    cursor: pointer;

`

const Container = styled.div`

    flex: 1;

    margin: 5px;

    min-width: 280px;

    height: 350px;

    display: flex;

    align-items: center;

    justify-content: center;

    background-color: #f5fbfd;

    position: relative;

    &:hover ${Info}{

        opacity: 1;

    }

`

MOBILE RESPONSIVENESS IN STYLED COMPONENTS

First Method:

const Container = styled.div`

    height: 60px;

@media only screen and (max-width:480px){

Height: 40px;

}

   `

Second Method:

Create JS function in responsive.js file at src destination:

import {css} from 'styled-components'

export const mobile  = (props) => {

    return css`

        @media only screen and (max-width: 480px) {

            ${props}

        }

    `

};

Import where media query required

import {mobile} from '../responsive'

const Container = styled.div`

    height: 60px;

    ${mobile( {backgroundColor: "red"})}

`

Backend

Index.js

Npm init

Install libraries

Node index.

Yarn add mongoose nodemon dotenv express

Yarn add crypto-js (password hashing)

yarn add jsonwebtoken (for verify whether order etc belongs to corresponding user)

yarn add stripe

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* STRIPE PAYMENT METHOD \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*:**

[added stripe · MUHAMMAD-MUNEEB-WASEEM/React-Ecommerce@8bc8879 (github.com)](https://github.com/MUHAMMAD-MUNEEB-WASEEM/React-Ecommerce/commit/8bc8879ac1b1b62ac744689090c69eef557dcdab)

Backend:

Yarn add stripe

const router = require("express").Router();

// const stripe = require("stripe")(process.env.STRIPE\_KEY);

const dotenv = require('dotenv')

dotenv.config();

const KEY = process.env.STRIPE\_KEY

const stripe = require("stripe")(KEY);

router.post("/payment", (req, res) => {

  stripe.charges.create(

    {

      source: req.body.tokenId,

      amount: req.body.amount,

      currency: "usd",

    },

    (stripeErr, stripeRes) => {

      if (stripeErr) {

        res.status(500).json(stripeErr);

      } else {

        res.status(200).json(stripeRes);

      }

    }

  );

});

module.exports = router;

And add stripe secret key (stripe sign in -> developers -> api keys -> copy secret key)in .env file

.env file:

MONGO\_URL = mongodb+srv://admin:DscVShSuT1ywR5gz@cluster0.vk4fa.mongodb.net/?retryWrites=true&w=majority

PORT = 5000

PASS\_SEC = muneeb

JWT\_SEC = muneeb

STRIPE\_KEY = sk\_test\_51JhERSGkYc9g1BvdKp9E0XKpPfoSNDOjmj1SfTjSEhCoARvnO40vktSAlo8xH2UlwYzl76LkcZPaA8e9mqn89Q7u00Ix1u8hFw

NOW CONNECTE API TO CLIENT SIDE:

Yarn add react-stripe-checkout

1. Create Pay page or page where pay now button exist that wrap with stripe checkout to open stripe checkout
2. This stripecheckout component can have multiple properties like name of shop, logo url, billing and shipping address etc.
3. Now post data (like amout, token etc) to stripe api using useeffect
4. PAY PAGE

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

import StripeCheckout from 'react-stripe-checkout'

import axios from 'axios'

function Pay() {

    const KEY = "pk\_test\_51JhERSGkYc9g1BvdAdto0rO94IZQ6BAqLaShXV1pie6mHqnY0dTObrEPGgBsZRffjvKBvB2SBviBv1FIPQ1Mlpit00Yw4DxR4C"

    const [stripeToken,setStripeToken] = useState(null);

    const onToken = (token) => {

        setStripeToken(token)

    }

    useEffect(()=>{

        stripeToken && (

        axios.post('http://localhost:5000/api/checkout/payment', {

            tokenId: stripeToken.id,

            amount: 2000,

        },{

        Authorization: "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYyODM2ZDVmYmU4ODk3NjUwYTVmNzYyMiIsImlzQWRtaW4iOnRydWUsImlhdCI6MTY1Mjc4MDQzNSwiZXhwIjoxNjUzMDM5NjM1fQ.UlNIUUc2kfeVtCFNVW2bIYTLijpTzYz5-u1-PeSfpUY"

        })

        .then(payment=>{

            console.log(payment)

        })

        .catch(err=>{

            console.log(err.response)

        })

        )

    }, [stripeToken])

  return (

    <div

        style={{

            display:"flex",

            alignItems:"center",

            justifyContent:"center",

            margin: "auto",

            height: "100vh"

    }}>

        <StripeCheckout

            name='MUNEEB SHOP'

            image='https://avatars.githubusercontent.com/u/50763652?v=4'

            billingAddress

            shippingAddress

            description='Your total is $20'

            amount={2000}

            token={onToken}

            stripeKey={KEY}

            >

            <button>PAY NOW</button>

        </StripeCheckout>

    </div>

  )

1. }
2. export default Pay

2. The above pay page has paynow button wrapped under stripecheckout

<StripeCheckout

            name='MUNEEB SHOP'

            image='https://avatars.githubusercontent.com/u/50763652?v=4'

            billingAddress

            shippingAddress

            description='Your total is $20'

            amount={2000}

            token={onToken}

            stripeKey={KEY}

            >

            <button>PAY NOW</button>

        </StripeCheckout>

1. Similarly part 1 has useeffect posting required data to stripe api at backend

    const KEY = "pk\_test\_51JhERSGkYc9g1BvdAdto0rO94IZQ6BAqLaShXV1pie6mHqnY0dTObrEPGgBsZRffjvKBvB2SBviBv1FIPQ1Mlpit00Yw4DxR4C"

    const [stripeToken,setStripeToken] = useState(null);

    const onToken = (token) => {

        setStripeToken(token)

    }

    useEffect(()=>{

        stripeToken && (

        axios.post('http://localhost:5000/api/checkout/payment', {

            tokenId: stripeToken.id,

            amount: 2000,

        },{

        Authorization: "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYyODM2ZDVmYmU4ODk3NjUwYTVmNzYyMiIsImlzQWRtaW4iOnRydWUsImlhdCI6MTY1Mjc4MDQzNSwiZXhwIjoxNjUzMDM5NjM1fQ.UlNIUUc2kfeVtCFNVW2bIYTLijpTzYz5-u1-PeSfpUY"

        })

        .then(payment=>{

            console.log(payment)

        })

        .catch(err=>{

            console.log(err.response)

        })

        )

    }, [stripeToken])

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* STRIPE PAYMENT METHOD \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*:**

HASHING PASSWORD:

router.post('/register', (req,res)=>{

    const newUser = new User({

        username: req.body.username,

        email: req.body.email,

        password: CryptoJS.AES.encrypt(req.body.password, process.env.PASS\_SEC).toString(),

    })

    newUser

    .save()

    .then(result=>{

        res.status(201).json(result)

    })

    .catch(err=>{

        res.status(500).json(err)

    })

})

PASS\_SEC passed in .env file

Update state based on onChange:

 <Select name="color" onChange={handleFilters}>

                    <Option disabled selected>Color</Option>

                    <Option>white</Option>

                    <Option>black</Option>

                    <Option>red</Option>

                    <Option>blue</Option>

                    <Option>yellow</Option>

                    <Option>green</Option>

                </Select>

                <Select name="size" onChange={handleFilters}>

                    <Option disabled>Size</Option>

                    <Option>XS</Option>

                    <Option>S</Option>

                    <Option>M</Option>

                    <Option>L</Option>

                    <Option>XL</Option>

                </Select>

const [filter, setFilters] = useState({});

 const handleFilters = (e) => {

    const value = e.target.value;

    setFilters({

        ...filter,

        [e.target.name]:value,

    })

 }

Another Method:

                <Select onChange={e => setSort(e.target.value)}>

                    <Option value="newest">Newest</Option>

                    <Option value="asc">Price (asc)</Option>

                    <Option value="desc">Price (desc)</Option>

                </Select>

Admin