RestAPI Of My project

```
import axios from 'axios';

const API = axios.create({ baseURL:
'http://localhost:5000' });

API.interceptors.request.use((req) => {
  if (localStorage.getItem('profile')) {
    req.headers.Authorization = `Bearer
${JSON.parse(localStorage.getItem('profile')).token}`;
  }

  return req;
});

export const fetchPost = (id) =>
API.get(`/posts/${id}`);
export const fetchPosts = (page) =>
API.get(`/posts?page=${page}`);
export const fetchPostsBySearch = (searchQuery) =>
API.get(`/posts/search?searchQuery=
              ${searchQuery.search ||
'none'}&tags=${searchQuery.tags}`);
export const createPost = (newPost) =>
API.post('/posts', newPost);
export const likePost = (id) =>
API.patch(`/posts/${id}/likePost`);
export const updatePost = (id, updatedPost) =>
API.patch(`/posts/${id}`, updatedPost);
export const deletePost = (id) =>
API.delete(`/posts/${id}`);
```

ACTIONS OF REDUX
//What to do? like if we want to do increment or decrement

```
// import { FETCH_ALL, CREATE, UPDATE, DELETE,
LIKE } from '../constants/typeofaction';
import * as api from '../api/index.js';
export const getPost = (id) => async (dispatch) => {
  try {
    dispatch({ type: 'START_LOADING' });

    const { data } = await api.fetchPost(id);

    dispatch({ type: 'FETCH_POST', payload: { post: data
} });
```

RestFUL API (An API in server which implements the API

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

// import express from 'express';
const express=require('express')
const
PostMessage=require('../models/postMessage.js');

// import { getPosts, getPostsBySearch, getPost,
createPost, updatePost, likePost, deletePost }
from '../controller/posts.js';
const jwt = require('jsonwebtoken');
// const getPosts=require('../controller/posts.js');
// const
getPostsBySearch=require('../controller/posts.js');
// const getPost=require('../controller/posts.js');
// const
createPost=require('../controller/posts.js');
// const
updatePost=require('../controller/posts.js');
// const likePost=require('../controller/posts.js');
// const
deletePost=require('../controller/posts.js');



const bcrypt = require('bcryptjs');
const Authenticate =
require("../middleware/Authenticate");
const User = require('../models/user');

console.log("Router Came");

const router = express.Router();
// import auth from "../middleware/auth.js";
router.get('/posts/search', async(req,res)=>{
   console.log("Search");
   const { searchQuery, tags } = req.query;

   try {
      const title = new RegExp(searchQuery, "i");

      const posts = await PostMessage.find({ $or: [ {
title }, { tags: { $in: tags.split(',') } } ]});
```

```
  } catch (error) {
    console.log(error);
  }
};

export const getPosts = (page) => async (dispatch) =>
{
  try {
    dispatch({ type: 'START_LOADING' });
    const { data: { data, currentPage, numberOfPages }
} = await api.fetchPosts(page);

    dispatch({ type: 'FETCH_ALL', payload: { data,
currentPage, numberOfPages } });
    dispatch({ type: 'END_LOADING' });
  } catch (error) {
    console.log(error);
  }
};

export const getPostsBySearch = (searchQuery) =>
async (dispatch) => {
  try {
    dispatch({ type: 'START_LOADING' });
    const { data: { data } } = await
api.fetchPostsBySearch(searchQuery);

    dispatch({ type: 'FETCH_BY_SEARCH', payload: {
data } });
    dispatch({ type: 'END_LOADING' });
  } catch (error) {
    console.log(error);
  }
};

export const createPost = (post, history) => async
(dispatch) => {
  try {
    dispatch({ type: 'START_LOADING' });
    const { data } = await api.createPost(post);

    dispatch({ type: 'CREATE', payload: data });

    history.push(`/posts/${data._id}`);
  } catch (error) {
    console.log(error);
  }
};
```

```
      res.json({ data: posts });
    } catch (error) {
      res.status(404).json({ message: error.message
});
    }
});
router.get('/posts/', async(req,res)=>{
    console.log("Get");
    const { page } = req.query;

    try {
      const LIMIT = 8;
      const startIndex = (Number(page) - 1) * LIMIT;
// get the starting index of every page

      const total = await
PostMessage.countDocuments({});
      const posts = await PostMessage.find().sort({
_id: -1 }).limit(LIMIT).skip(startIndex);

      res.json({ data: posts, currentPage:
Number(page), numberOfPages: Math.ceil(total /
LIMIT)});
    } catch (error) {
      res.status(404).json({ message: error.message
});
    }
});
router.get('/posts/:id', async(req,res)=>{
    console.log("Get post");
    const { id } = req.params;

    try {
      const post = await PostMessage.findById(id);

      res.status(200).json(post);
    } catch (error) {
      res.status(404).json({ message: error.message
});
    }
});

router.post('/posts/', async(req,res)=>{
    console.log("Newpost");
    const { title, message, selectedFile, creator, tags
} = req.body;
```

```
export const updatePost = (id, post) => async
(dispatch) => {
  try {
    const { data } = await api.updatePost(id, post);

    dispatch({ type: 'UPDATE', payload: data });
  } catch (error) {
    console.log(error);
  }
};

export const likePost = (id) => async (dispatch) => {
  const user =
JSON.parse(localStorage.getItem('profile'));

  try {
    const { data } = await api.likePost(id, user?.token);

    dispatch({ type: 'LIKE', payload: data });
  } catch (error) {
    console.log(error);
  }
};

export const deletePost = (id) => async (dispatch) => {
  try {
    await await api.deletePost(id);

    dispatch({ type: 'DELETE', payload: id });
  } catch (error) {
    console.log(error);
  }
};
```

REDUCER OF REDUX

```
//Reducers tell how to do? It contains state and the
action
//Functionality is in the reducer


// import { FETCH_ALL, CREATE, UPDATE, DELETE,
LIKE } from '../constants/actionTypes';
// import { FETCH_ALL} from
'../constants/actionTypes';
```

```
    const newPostMessage = new PostMessage({
title, message, selectedFile, creator, tags })

    try {
      await newPostMessage.save();

      res.status(201).json(newPostMessage );
    } catch (error) {
      res.status(409).json({ message: error.message
});
    }
});
router.patch('/posts/:id' , async(req,res)=>{
    console.log("UpdatePost");
    const { id } = req.params;
    const { title, message, creator, selectedFile, tags
} = req.body;

    if (!mongoose.Types.ObjectId.isValid(id)) return
res.status(404).send(`No post with id: ${id}`);

    const updatedPost = { creator, title, message,
tags, selectedFile, _id: id };

    await PostMessage.findByIdAndUpdate(id,
updatedPost, { new: true });

    res.json(updatedPost);
});
router.delete('/posts/:id',async(req,res)=>{
    console.log("DeletPost");
    const { id } = req.params;

    if (!mongoose.Types.ObjectId.isValid(id)) return
res.status(404).send(`No post with id: ${id}`);

    await PostMessage.findByIdAndRemove(id);

    res.json({ message: "Post deleted successfully."
});
});
router.patch('/posts/:id/likePost',
async(req,res)=>{
    console.log("LikePost");
    const { id } = req.params;

    console.log(id);
```

```
// eslint-disable-next-line import/no-anonymous-default-export
export default (state = { isLoading: true, posts: [] }, action) => {
  switch (action.type) {
    case 'START_LOADING':
      return { ...state, isLoading: true };
    case 'END_LOADING':
      return { ...state, isLoading: false };
    case 'FETCH_ALL':
      return {
        ...state,
        posts: action.payload.data,
        currentPage: action.payload.currentPage,
        numberOfPages: action.payload.numberOfPages,
      };
    case 'FETCH_BY_SEARCH':
      return { ...state, posts: action.payload.data };
    case 'FETCH_POST':
      return { ...state, post: action.payload.post };
    case 'LIKE':
      return { ...state, posts: state.posts.map((post) => (post._id === action.payload._id ? action.payload : post)) };
    case 'CREATE':
      return { ...state, posts: [...state.posts, action.payload] };
    case 'UPDATE':
      return { ...state, posts: state.posts.map((post) => (post._id === action.payload._id ? action.payload : post)) };
    case 'DELETE':
      return { ...state, posts: state.posts.filter((post) => post._id !== action.payload) };
    default:
      return state;
  }
};
```

REDUX STORE

```
const store = createStore(reducers, compose(applyMiddleware(thunk)));

ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById('root'),
);
```

```
  if (!mongoose.Types.ObjectId.isValid(id)) return res.status(404).send(`No post with id: ${id}`);

  const post = await PostMessage.findById(id);

  const updatedPost = await PostMessage.findByIdAndUpdate(id, { likeCount: post.likeCount + 1 }, { new: true });

  res.json(updatedPost);
});

router.post('/register', async (req,res) => {
  console.log("signup");
  const { name, email, phone, work, password, Confpassword} = req.body;

  if(!name  || !email || !phone || !work || !password || !Confpassword){
     return res.status(422).json({error: "Please fill all fields."});              //422 some filled is missing
  }

  try {

     const userExist = await User.findOne({email: email});            //left email is from database and right is that user is filling on resgistration form
     if (userExist) {                                                  //if user already exists
        return res.status(422).json({error: "Email already exits"});
     }else if(password != Confpassword){
        return res.status(422).json({error: "Password and confirm password doesn't match."});
     }

     const user = new User({ name, email, phone, work, password, Confpassword });

     //befor saving the data password will be encyrpted in user file than it will be save
     await user.save();
     res.status(201).json({message: "Registration successfull."});
```

ROUTING IN ASP.NET it's in webapniconfig.cs

```
routes.MapHttpRoute(
   name: "ActionApi",
   routeTemplate: "api/{controller}/{action}/{id}",
   defaults: new { id = RouteParameter.Optional }
);
```

```
   }catch(err) {
      console.log(err);
   }

});


router.post('/signin', async (req,res) =>{
   console.log("Login");
   // res.json({ message: "hello"})
   try{
      let token;
      const { email,password} = req.body;
      if (!email || !password){
         return res.status(400).json({error:"Please
enter email and password"})
      }

      const userLogin = await
User.findOne({email:email});
      //console.log(userLogin);
      if(userLogin){
         const isMatch = await
bcrypt.compare(password,userLogin.password);
//checking the password is match with the
username that is put

         token = await
userLogin.genrateAuthToken();
         console.log(token);
         res.cookie("jwtoken",token, {
            expires:new Date(Date.now() +
25892000000),              //cookies expires after
30 days
            httpOnly:true
         });
         if(!isMatch){                              //if
data is present
            res.status(400).json({error: "Password
doesn't match with email."});
         }else{
            res.json({message: "Sigin successfully."});
         }
      }else{
         res.status(400).json({error: "Please enter
valid email."});
```

```
    }

  }catch(err){
    console.log(err);
  }
})

// About Page


router.get('/profile', (req,res) => {
  console.log('Hello from about');
  res.send('hello wolrd message from About
server.');
});


router.get('/Index', (req,res) => {
  console.log('Index');
  res.send('Index');
});

router.get('/logout', (req, res) => {
  console.log('Hello from logout page');
  res.clearCookie('jwtoken', {path:'/'});
  res.status(200).send('User logout');
});

// export default router;
module.exports=router
```

INDEX.JS BEFORE REDUCER

```
import { combineReducers } from 'redux';

import posts from './posts';

export const reducers = combineReducers({ posts
});
```