**A MINI-PROJECT REPORT ON**

**“Blogging Website”**

SUBMITTED TO SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE

**Bachelor of Engineering**

In

**Information Technology**

##### **Class: T.E**

##### **BY**

**Abhishek Khairnar(T-02)**

**Suyog Patil(T-75)**

**Pranay Mule(T-45)**

**Harshal Nahire(T-46)**

**Under the guidance of**

Prof. Suhas Chavan



## DEPARTMENT OF INFORMATION TECHNOLOGY

## RMD SINHGAD SCHOOL OF ENGINEERING

WARJE, PUNE-411058

## A.Y: 2021 - 22



## DEPARTMENT OF INFORMATION TECHNOLOGY

## RMD SINHGAD SCHOOL OF ENGINEERING

WARJE, PUNE-411058

**CERTIFICATE**

This is to certify that the Mini-Project Report entitled

**“Blogging Website”**

Submitted by

Name: Harshal Nahire Roll No. /Exam Seat No. : T-046

is a bonafide work carried out by him/her under the supervision of Prof. Suhas Chavan sir and it is submitted towards the partial fulfillment of the requirement for T.E (Information Technology) – 2019 course of Savitribai Phule Pune University, Pune in the academic year 2021-2022

**(Mrs.** **Suhas Chavan)**  **(Prof. Sweta Kale)**

Guide Head,

Department of Information Technology Department of Information Technology

**(Dr. V.V. Dixit)**

Principal,

Place: RMD Sinhgad School of Engineering Pune – 58

Date:

**ACKNOWLEDGEMENT**

The satisfaction that accompanies that the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success.

We are grateful to our project guide **Mr. Suhas Chavan Sir**. for the guidance, inspiration and constructive suggestions that helpful us in the preparation of this project.

We also thank our colleagues who have helped in successful completion of the project.

**NAME OF STUDENTS**

**Suyog Patil**

**Parnay Mule**

**Abhishek Khairnar**

**Harshal Nahire**

**CONTENTS**

1. **Introduction** 
   1. Objectives
   2. Problem Statement
2. **Scope and Limitations**
3. **Requirement Analysis** 
   1. Requirement gathering and Analysis
   2. Functional Requirements
   3. Non- Functional Requirements
4. **System requirements** 
   1. Hardware Requirements
   2. Software Requirements
5. **Application Design**
6. **Technology**
7. **Project Implementation**
8. **Testing**
9. **Screenshots**
10. **Conclusion**
11. **References**

### Introduction

#### A blog is a type of website where the content is presented in reverse chronological order (newer content appear first). Blog content is often referred to as entries or “blog posts”. Blogs are typically run by an individual or a small group of people to present information in a conversational style.

#### 1.1 Objectives

The “Online Blogging System ” has been developed to override the problems prevailing in the practicing manual system.The software is supposed to eliminate and in some cases reduce the hardships faced by this existing system. Morever this system is designed for the particular need of the company to carry out operations in a smooth and effective manner. The main objective of the system is to connect people socially with each other and share idea among them. User can share the photo, video and content of there choice.

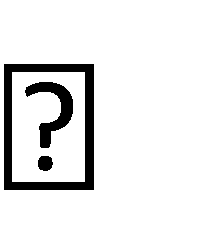
#### 1.2 Problem Statement

The main problem that is faced by the bloggers during writing a blog is not having a clear topic and niche for the blog. There is lack of quality content and writing style. While writing manually a blogger is not able to stick to a topic and has less people to access that.

**1.3 Purpose :**

The purpose of developing Blogging Website is to manage the details of Blogs Comment New Category, New Blog Technology Blog It manages all the information about Blogs, Comment Technology Blog Blogs. The project is totally but at administrative end and thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work for managing the Blogs Comment Comment, New Category It tracks all the details about the New Category. New Blog, Technology Blog.

**Scope & Limitations :**



The scope of the project is that may help collecting perfect management in details. In a very short time, the collection will be obvious, simple and sensible. It will help a person to know the management of passed year perfectly and vividly. It also helps in current all works relative to Online Blogging System. It will be also reduced the cost of collecting the management & collection procedure will go on smoothly.

The intention of developing Blogging Website is to socialize people of different thoughts, different views with each other and to make a blog more effective and productive. The purpose

of a blog is to provide content on your website that answers your prospective customers' questions and helps them learn about your product or service.

1. **System requirement** 
   1. **Software Requirements:**

* + - VS Studio Code
    - APACHE HTTP Server
    - NodeJs ,ExpressJs , MongoDB, ReactJs
    - Microsoft Windows or Linux
  1. **Hardware Requirements:**

1.Ryzen 3, 5 processor or equivalent or higher.

2. 4 gb ram or Higher.

3. 20GB HDD or Higher.

4.Good Network Connectivity.

5. Windows 10 or higher.

**Characteristics :**

* **User Friendly**:- The proposed system is user friendly because the retrieval and storing of data is fast and data is maintained efficiently. Moreover the graphical user interface is provided in the proposed system, which provides user to deal with the system very easily.

* **Blogs are uploaded easily:** You can easily upload your blog with no system errors.

* **Very less paper work**: The proposed system requires very less paper work. All the data is feted into the computer immediately and blogs can be generated through computers. Moreover work become very easy because there is no need to keep data on papers.

**Platform :** This provides a big and vast platform for blogger to show his/her skills to people.

**6. Technologies Used:**

**Front end**: HTML, CSS, JavaScript

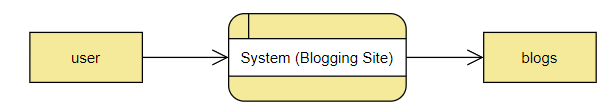
1. HTML: HTML is used to create and save web document. E.g. Notepad/Notepad++
2. CSS : (Cascading Style Sheets) Create attractive Layout
3. JavaScript: it is a programming language, commonly use with web browsers.
4. Bootstrap : for responsive and mobile friendly design.
5. React

**Back end**: Nodejs, Express, Mongo dB

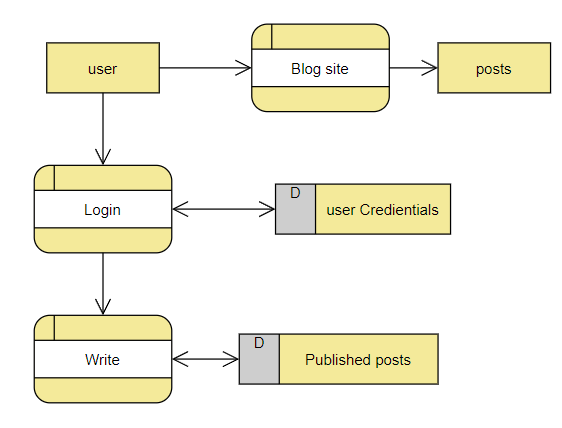
* **Nodejs-** **Node.js** is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser.
* **Express-** The primary use of Express is to provide server-side logic for web and mobile applications, and as such it's used all over the place.
* **Mongo dB-** MongoDB is an open-source NoSQL database management program. NoSQL is used as an alternative to traditional relational databases. NoSQL databases are quite useful for working with large sets of distributed data. MongoDB is a tool that can manage document-oriented information, store or retrieve information.

**Data Flow Diagrams:**

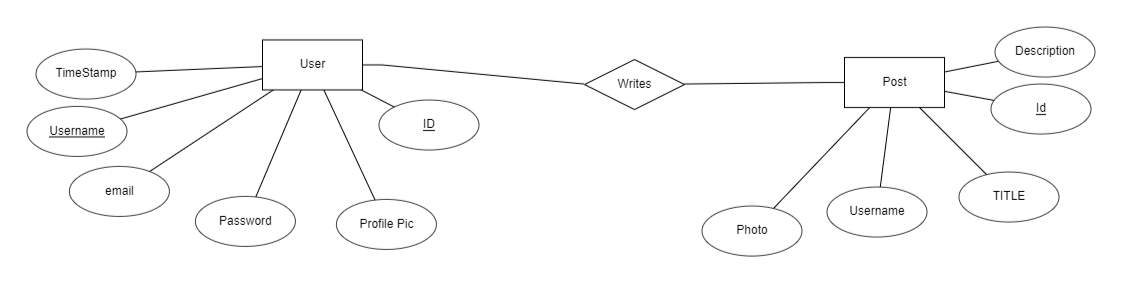
**0-level Flow Diagram:-**

****

**1-level Flow Diagram:-**

****

1. **ER Diagram :-**



1. **Project Implementation (**Codeing ,Testing & Screenshot**):**

**Code :- Backend (Express and NodeJs)**

**Index.js**

**const express = require("express");**

**const app = express();**

**const dotenv = require("dotenv");**

**const mongoose = require("mongoose");**

**const authRoute = require("./routes/auth");**

**const userRoute = require("./routes/users");**

**const postRoute = require("./routes/posts");**

**const multer = require("multer");**

**const path = require("path");**

**var mysql = require('mysql');**

**dotenv.config();**

**app.use(express.json());**

**app.use("/images", express.static(path.join(\_\_dirname, "/images")));**

**mongoose**

**.connect(process.env.MONGO\_URL, {**

**useNewUrlParser: true,**

**useUnifiedTopology: true,**

**useCreateIndex: true,**

**useFindAndModify:true**

**})**

**.then(console.log("Connected to MongoDB"))**

**.catch((err) => console.log(err));**

**const storage = multer.diskStorage({**

**destination: (req, file, cb) => {**

**cb(null, "images");**

**},**

**filename: (req, file, cb) => {**

**cb(null, req.body.name);**

**},**

**});**

**const upload = multer({ storage: storage });**

**app.post("/api/upload", upload.single("file"), (req, res) => {**

**res.status(200).json("File has been uploaded");**

**});**

**app.use("/api/auth", authRoute);**

**app.use("/api/users", userRoute);**

**app.use("/api/posts", postRoute);**

**app.listen(process.env.PORT || 5000, () => {**

**console.log("Backend is running.");**

**});**

**Routes/ AuthJs :-**

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

**const User = require("../models/User");**

**const bcrypt = require("bcrypt");**

**//REGISTER**

**router.post("/register", async (req, res) => {**

**try {**

**const salt = await bcrypt.genSalt(10);**

**const hashedPass = await bcrypt.hash(req.body.password, salt);**

**const newUser = new User({**

**username: req.body.username,**

**email: req.body.email,**

**password: hashedPass,**

**});**

**const user = await newUser.save();**

**res.status(200).json(user);**

**} catch (err) {**

**res.status(500).json(err);**

**}**

**});**

**//LOGIN**

**router.post("/login", async (req, res) => {**

**try {**

**const user = await User.findOne({ username: req.body.username });**

**!user && res.status(400).json("Wrong credentials!");**

**const validated = await bcrypt.compare(req.body.password, user.password);**

**!validated && res.status(400).json("Wrong credentials!");**

**const { password, ...others } = user.\_doc;**

**res.status(200).json(others);**

**} catch (err) {**

**res.status(500).json(err);**

**}**

**});**

**module.exports = router;**

**Routes/Ports.Js:-**

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

**const User = require("../models/User");**

**const Post = require("../models/Post");**

**//CREATE POST**

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

**const newPost = new Post(req.body);**

**try {**

**const savedPost = await newPost.save();**

**res.status(200).json(savedPost);**

**} catch (err) {**

**res.status(500).json(err);**

**}**

**});**

**//UPDATE POST**

**router.put("/:id", async (req, res) => {**

**try {**

**const post = await Post.findById(req.params.id);**

**if (post.username === req.body.username) {**

**try {**

**const updatedPost = await Post.findByIdAndUpdate(**

**req.params.id,**

**{**

**$set: req.body,**

**},**

**{ new: true }**

**);**

**res.status(200).json(updatedPost);**

**} catch (err) {**

**res.status(500).json(err);**

**}**

**} else {**

**res.status(401).json("You can update only your post!");**

**}**

**} catch (err) {**

**res.status(500).json(err);**

**}**

**});**

**//DELETE POST**

**router.delete("/:id", async (req, res) => {**

**try {**

**const post = await Post.findById(req.params.id);**

**if (post.username === req.body.username) {**

**try {**

**await post.delete();**

**res.status(200).json("Post has been deleted...");**

**} catch (err) {**

**res.status(500).json(err);**

**}**

**} else {**

**res.status(401).json("You can delete only your post!");**

**}**

**} catch (err) {**

**res.status(500).json(err);**

**}**

**});**

**//GET POST**

**router.get("/:id", async (req, res) => {**

**try {**

**const post = await Post.findById(req.params.id);**

**res.status(200).json(post);**

**} catch (err) {**

**res.status(500).json(err);**

**}**

**});**

**//GET ALL POSTS**

**router.get("/", async (req, res) => {**

**const username = req.query.user;**

**const catName = req.query.cat;**

**try {**

**let posts;**

**if (username) {**

**posts = await Post.find({ username });**

**} else if (catName) {**

**posts = await Post.find({**

**categories: {**

**$in: [catName],**

**},**

**});**

**} else {**

**posts = await Post.find();**

**}**

**res.status(200).json(posts);**

**} catch (err) {**

**res.status(500).json(err);**

**}**

**});**

**module.exports = router;**

**Router/User.Js:-**

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

const User = require("../models/User");

const Post = require("../models/Post");

const bcrypt = require("bcrypt");

//UPDATE

router.put("/:id", async (req, res) => {

if (req.body.userId === req.params.id) {

if (req.body.password) {

const salt = await bcrypt.genSalt(10);

req.body.password = await bcrypt.hash(req.body.password, salt);

}

try {

const updatedUser = await User.findByIdAndUpdate(

req.params.id,

{

$set: req.body,

},

{ new: true }

);

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

} catch (err) {

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

}

} else {

res.status(401).json("You can update only your account!");

}

});

//DELETE

router.delete("/:id", async (req, res) => {

if (req.body.userId === req.params.id) {

try {

const user = await User.findById(req.params.id);

try {

await Post.deleteMany({ username: user.username });

await User.findByIdAndDelete(req.params.id);

res.status(200).json("User has been deleted...");

} catch (err) {

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

}

} catch (err) {

res.status(404).json("User not found!");

}

} else {

res.status(401).json("You can delete only your account!");

}

});

//GET USER

router.get("/:id", async (req, res) => {

try {

const user = await User.findById(req.params.id);

const { password, ...others } = user.\_doc;

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

} catch (err) {

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

}

});

module.exports = router;

**Modes/Post.Js:-**

const mongoose = require("mongoose");

const PostSchema = new mongoose.Schema(

{

title: {

type: String,

required: true,

unique: true,

},

desc: {

type: String,

required: true,

},

photo: {

type: String,

required: false,

},

username: {

type: String,

required: true,

},

categories: {

type: Array,

required: false,

},

},

{ timestamps: true }

);

module.exports = mongoose.model("Post", PostSchema);

**Modes/User.Js:-**

const mongoose = require("mongoose");

const UserSchema = new mongoose.Schema(

{

username: {

type: String,

required: true,

unique: true,

},

email: {

type: String,

required: true,

unique: true,

},

password: {

type: String,

required: true,

},

profilePic: {

type: String,

default: "",

},

},

{ timestamps: true }

);

**module.exports = mongoose.model("User", UserSchema);**

**Front End (React.js):-**

**Src/components/ header /header.js**

import "./header.css";

import Carousel from "react-bootstrap/Carousel";

export default function Header() {

return (

<div>

<Carousel fade>

<Carousel.Item>

<img

className="d-block w-100"

src="https://source.unsplash.com/1400x400/?creative"

alt="First slide"

/>

<Carousel.Caption className="bg-dark opacity-75">

<h3>Create Your Blogs</h3>

<p>There's a way to do it better - find it.</p>

</Carousel.Caption>

</Carousel.Item>

<Carousel.Item>

<img

className="d-block w-100"

src="https://source.unsplash.com/1400x400/?creative,explore"

alt="Second slide"

/>

<Carousel.Caption className="bg-dark opacity-75">

<h3>Learn with Others</h3>

<p>Never stop learning, because Life never stops teaching.</p>

</Carousel.Caption>

</Carousel.Item>

<Carousel.Item>

<img

className="d-block w-100"

src="https://source.unsplash.com/1400x400/?creative,idea"

alt="Third slide"

/>

<Carousel.Caption className="bg-dark opacity-75">

<h3>Discover the Worlds Through Blogs</h3>

<p>Get Closer to your Goals</p>

</Carousel.Caption>

</Carousel.Item>

</Carousel>

</div>

);

}

**Src/components/ header /header.css**

.header {

  margin-top: 60px;

}

.headerTitles {

  display: flex;

  flex-direction: column;

  align-items: center;

  font-family: "Lora", serif;

  color: #444;

}

.headerTitleSm {

  position: absolute;

  top: 18%;

  font-size: 20px;

}

.headerTitleLg {

  position: absolute;

  top: 20%;

  font-size: 100px;

}

.headerImg{

    /\* width: 100%;

    height: 450px;

    margin-top: 80px;

    object-fit: cover; \*/

max-height: 90vh ;

}

.customHeight {

  height: fit-content;

}

**src/components/ post /post.js**

import "./post.css";

import { Link } from "react-router-dom";

export default function Post({ post }) {

  const PF = "http://localhost:5000/images/";

  return (

    <div className="post">

      {post.photo && <img className="postImg" src={PF + post.photo} alt="" />}

      <div className="postInfo">

        <div className="postCats">

          {post.categories.map((c) => (

            <span className="postCat">{c.name}</span>

          ))}

        </div>

        <Link to={`/post/${post.\_id}`} className="link">

          <span className="postTitle">{post.title}</span>

        </Link>

        <hr />

        <span className="postDate">

          {new Date(post.createdAt).toDateString()}

        </span>

      </div>

      <p className="postDesc">{post.desc}</p>

    </div>

  );

}

**src/components/ post /post.css**

.post {

  width: 385px;

  margin: 0px 25px 40px 25px;

}

.postImg {

  width: 100%;

  height: 280px;

  object-fit: cover;

  border-radius: 7px;

}

.postInfo {

  display: flex;

  flex-direction: column;

  align-items: center;

}

.postCat {

  font-family: "Varela Round", sans-serif;

  font-size: 11px;

  color: #be9656;

  line-height: 20px;

  margin-top: 15px;

  margin-right: 10px;

  cursor: pointer;

}

.postTitle {

  font-family: "Josefin Sans", sans-serif;

  font-size: 24px;

  font-weight: 700;

  margin-top: 15px;

  cursor: pointer;

}

.postDate {

  font-family: "Lora", serif;

  font-style: italic;

  font-size: 13px;

  color: #999;

  margin-top: 15px;

}

.postDesc {

  font-family: "Varela Round", sans-serif;

  font-size: 14px;

  color: #444;

  line-height: 24px;

  margin-top: 15px;

  overflow: hidden;

  text-overflow: ellipsis;

  display: -webkit-box;

  -webkit-line-clamp: 4;

  -webkit-box-orient: vertical;

**}**

**src/components/ posts /posts.js**

import Post from "../post/Post";

import "./posts.css";

export default function Posts({ posts }) {

return (

<div className="posts">

{posts.map((p) => (

<Post post={p} />

))}

</div>

);

}

**src/components/ posts /posts.css**

.posts{

    flex: 9;

    display: flex;

    flex-wrap: wrap;

    margin: 20px;

}

**src/components/ SinglePost /SinglePost.js**

import axios from "axios";

import { useContext, useEffect, useState } from "react";

import { useLocation } from "react-router";

import { Link } from "react-router-dom";

import { Context } from "../../context/Context";

import "./singlePost.css";

export default function SinglePost() {

  const location = useLocation();

  const path = location.pathname.split("/")[2];

  const [post, setPost] = useState({});

  const PF = "http://localhost:5000/images/";

  const { user } = useContext(Context);

  const [title, setTitle] = useState("");

  const [desc, setDesc] = useState("");

  const [updateMode, setUpdateMode] = useState(false);

  useEffect(() => {

    const getPost = async () => {

      const res = await axios.get("/posts/" + path);

      setPost(res.data);

      setTitle(res.data.title);

      setDesc(res.data.desc);

    };

    getPost();

  }, [path]);

  const handleDelete = async () => {

    try {

      await axios.delete(`/posts/${post.\_id}`, {

        data: { username: user.username },

      });

      window.location.replace("/");

    } catch (err) {}

  };

  const handleUpdate = async () => {

    try {

      await axios.put(`/posts/${post.\_id}`, {

        username: user.username,

        title,

        desc,

      });

      setUpdateMode(false);

    } catch (err) {}

  };

  return (

    <div className="singlePost">

      <div className="singlePostWrapper">

        {post.photo && (

          <img src={PF + post.photo} alt="" className="singlePostImg" />

        )}

        {updateMode ? (

          <input

            type="text"

            value={title}

            className="singlePostTitleInput"

            autoFocus

            onChange={(e) => setTitle(e.target.value)}

          />

        ) : (

          <h1 className="singlePostTitle">

            {title}

            {post.username === user?.username && (

              <div className="singlePostEdit">

                <i

                  className="singlePostIcon far fa-edit"

                  onClick={() => setUpdateMode(true)}

                ></i>

                <i

                  className="singlePostIcon far fa-trash-alt"

                  onClick={handleDelete}

                ></i>

              </div>

            )}

          </h1>

        )}

        <div className="singlePostInfo">

          <span className="singlePostAuthor">

            Author:

            <Link to={`/?user=${post.username}`} className="link">

              <b> {post.username}</b>

            </Link>

          </span>

          <span className="singlePostDate">

            {new Date(post.createdAt).toDateString()}

          </span>

        </div>

        {updateMode ? (

          <textarea

            className="singlePostDescInput"

            value={desc}

            onChange={(e) => setDesc(e.target.value)}

          />

        ) : (

          <p className="singlePostDesc">{desc}</p>

        )}

        {updateMode && (

          <button className="singlePostButton" onClick={handleUpdate}>

            Update

          </button>

        )}

      </div>

    </div>

  );

}

**src/components/ SinglePost /SinglePost.css**

import Post from "../post/Post";

import "./posts.css";

export default function Posts({ posts }) {

  return (

    <div className="posts">

      {posts.map((p) => (

        <Post post={p} />

      ))}

    </div>

  );

}

**src/components/ topbar /topbar.js**

import React from "react";

import { useContext } from "react";

import { Link } from "react-router-dom";

import { Context } from "../../context/Context";

import "./topbar.css";

export default function TopBar() {

  const { user, dispatch } = useContext(Context);

  const PF = "http://localhost:5000/images/";

  const handleLogout = () => {

    dispatch({ type: "LOGOUT" });

  };

  return (

    <div className="top">

      <div className="topLeft">

        <h1>BLOGS</h1>

      </div>

      <div className="topCenter">

        <ul className="topList">

          <li className="topListItem">

            <Link className="link" to="/">

              HOME

            </Link>

          </li>

          <li className="topListItem">

            <Link className="link" to="/about">

              ABOUT

            </Link>

          </li>

          <li className="topListItem">

            <Link className="link" to="/write">

              WRITE

            </Link>

          </li>

          <li className="topListItem" onClick={handleLogout}>

            {user && "LOGOUT"}

          </li>

        </ul>

      </div>

      <div className="topRight">

        {user ? (

          <Link to="/settings">

            <img

              className="topImg"

              src={

                user.profilePic

                  ? PF + user.profilePic

                  : "https://cdn.pixabay.com/photo/2015/10/05/22/37/blank-profile-picture-973460\_960\_720.png"

              }

              alt=""

            />

          </Link>

        ) : (

          <ul className="topList">

            <li className="topListItem">

              <Link className="link" to="/login">

                LOGIN

              </Link>

            </li>

            <li className="topListItem">

              <Link className="link" to="/register">

                REGISTER

              </Link>

            </li>

          </ul>

        )}

      </div>

    </div>

  );

}

**src/ context /Actions.js**

export const LoginStart = (userCredentials) => ({

  type: "LOGIN\_START",

});

export const LoginSuccess = (user) => ({

  type: "LOGIN\_SUCCESS",

  payload: user,

});

export const LoginFailure = () => ({

  type: "LOGIN\_FAILURE",

});

export const Logout = () => ({

  type: "LOGOUT",

});

export const UpdateStart = (userCredentials) => ({

  type: "UPDATE\_START",

});

export const UpdateSuccess = (user) => ({

  type: "UPDATE\_SUCCESS",

  payload: user,

});

export const UpdateFailure = () => ({

  type: "UPDATE\_FAILURE",

});

**src/ context /Context.js**

import { createContext, useEffect, useReducer } from "react";

import Reducer from "./Reducer";

const INITIAL\_STATE = {

  user: JSON.parse(localStorage.getItem("user")) || null,

  isFetching: false,

  error: false,

};

export const Context = createContext(INITIAL\_STATE);

export const ContextProvider = ({ children }) => {

  const [state, dispatch] = useReducer(Reducer, INITIAL\_STATE);

  useEffect(() => {

    localStorage.setItem("user", JSON.stringify(state.user));

  }, [state.user]);

  return (

    <Context.Provider

      value={{

        user: state.user,

        isFetching: state.isFetching,

        error: state.error,

        dispatch,

      }}

    >

      {children}

    </Context.Provider>

  );

};

**src/context /Reducer.js**

const Reducer = (state, action) => {

  switch (action.type) {

    case "LOGIN\_START":

      return {

        user: null,

        isFetching: true,

        error: false,

      };

    case "LOGIN\_SUCCESS":

      return {

        user: action.payload,

        isFetching: false,

        error: false,

      };

    case "LOGIN\_FAILURE":

      return {

        user: null,

        isFetching: false,

        error: true,

      };

      case "UPDATE\_START":

        return {

          ...state,

          isFetching:true

        };

      case "UPDATE\_SUCCESS":

        return {

          user: action.payload,

          isFetching: false,

          error: false,

        };

      case "UPDATE\_FAILURE":

        return {

          user: state.user,

          isFetching: false,

          error: true,

        };

    case "LOGOUT":

      return {

        user: null,

        isFetching: false,

        error: false,

      };

    default:

      return state;

  }

};

export default Reducer;

**src/pages/about/about.js**

import "./about.css";

export default function About() {

  return (

    <>

      <div className="d-flex flex-row bd-highlight">

        <div className="p-2 flex-grow-1 bd-highlight">

          <div className="container m-4">

            <h2>Blogging Site</h2>

            <h6>

              Features of This Site.

              <ul>

                <li>Blog creaters can create there account on this site.</li>

                <li>

                  User can Share the Ideas, Knowledge, Experience with whole

                  world.

                </li>

                <li>All other users can freely access to the blogs.</li>

                <li>Connect with other User.</li>

                <li>Discover all the blogs on the Home Page.</li>

              </ul>

              <b>

                Created by:

                <ul>

                  <li> Suyog Patil</li>

                  <li> Pranay Mule</li>

                  <li> Abhishek Khairnar</li>

                  <li> Harshal Nahire</li>

                </ul>

              </b>

            </h6>

          </div>

        </div>

        <div className="p-2 pl-2">

          <div className="headerTitles bd-highlight">

            <span className="text-light headerTitleSm">React & Node</span>

            <span className="text-light headerTitleLg">About Us</span>

          </div>

          <img

            className="headerImg"

            src="https://images.unsplash.com/photo-1623186786193-1294c5348b12?crop=entropy&cs=tinysrgb&fit=crop&fm=jpg&h=720&ixid=MnwxfDB8MXxyYW5kb218MHx8YWJvdXR8fHx8fHwxNjUxOTMwNzIw&ixlib=rb-1.2.1&q=80&utm\_campaign=api-credit&utm\_medium=referral&utm\_source=unsplash\_source&w=600"

            alt="About"

          />

        </div>

      </div>

    </>

  );

}

**Src/page/home/homr.js**

import { useEffect, useState } from "react";

import Header from "../../components/header/Header";

import Posts from "../../components/posts/Posts";

import "./home.css";

import axios from "axios";

import { useLocation } from "react-router";

export default function Home() {

  const [posts, setPosts] = useState([]);

  const { search } = useLocation();

  useEffect(() => {

    const fetchPosts = async () => {

      const res = await axios.get("/posts" + search);

      setPosts(res.data);

    };

    fetchPosts();

  }, [search]);

  return (

    <>

      <Header />

      <div className="home">

        <Posts posts={posts} />

      </div>

    </>

  );

}

**Src/page/login/login.js**

import axios from "axios";

import { useContext, useRef } from "react";

import { Link } from "react-router-dom";

import { Context } from "../../context/Context";

import "./login.css";

export default function Login() {

  const userRef = useRef();

  const passwordRef = useRef();

  const { dispatch, isFetching } = useContext(Context);

  const handleSubmit = async (e) => {

    e.preventDefault();

    dispatch({ type: "LOGIN\_START" });

    try {

      const res = await axios.post("/auth/login", {

        username: userRef.current.value,

        password: passwordRef.current.value,

      });

      dispatch({ type: "LOGIN\_SUCCESS", payload: res.data });

    } catch (err) {

      dispatch({ type: "LOGIN\_FAILURE" });

    }

  };

  return (

    <div className="login">

      <span className="loginTitle">Login</span>

      <form className="loginForm" onSubmit={handleSubmit}>

        <label>Username</label>

        <input

          type="text"

          className="loginInput"

          placeholder="Enter your username..."

          ref={userRef}

        />

        <label>Password</label>

        <input

          type="password"

          className="loginInput"

          placeholder="Enter your password..."

          ref={passwordRef}

        />

        <button className="loginButton" type="submit" disabled={isFetching}>

          Login

        </button>

      </form>

      <button className="loginRegisterButton">

        <Link className="link" to="/register">

          Register

        </Link>

      </button>

    </div>

  );

}

**Src/page/register/register.js**

import axios from "axios";

import { useContext, useRef } from "react";

import { Link } from "react-router-dom";

import { Context } from "../../context/Context";

import "./login.css";

export default function Login() {

  const userRef = useRef();

  const passwordRef = useRef();

  const { dispatch, isFetching } = useContext(Context);

  const handleSubmit = async (e) => {

    e.preventDefault();

    dispatch({ type: "LOGIN\_START" });

    try {

      const res = await axios.post("/auth/login", {

        username: userRef.current.value,

        password: passwordRef.current.value,

      });

      dispatch({ type: "LOGIN\_SUCCESS", payload: res.data });

    } catch (err) {

      dispatch({ type: "LOGIN\_FAILURE" });

    }

  };

  return (

    <div className="login">

      <span className="loginTitle">Login</span>

      <form className="loginForm" onSubmit={handleSubmit}>

        <label>Username</label>

        <input

          type="text"

          className="loginInput"

          placeholder="Enter your username..."

          ref={userRef}

        />

        <label>Password</label>

        <input

          type="password"

          className="loginInput"

          placeholder="Enter your password..."

          ref={passwordRef}

        />

        <button className="loginButton" type="submit" disabled={isFetching}>

          Login

        </button>

      </form>

      <button className="loginRegisterButton">

        <Link className="link" to="/register">

          Register

        </Link>

      </button>

    </div>

  );

}

**Src/page/settings/settings.jsx**

import "./settings.css";

import { useContext, useState } from "react";

import { Context } from "../../context/Context";

import axios from "axios";

export default function Settings() {

  const [file, setFile] = useState(null);

  const [username, setUsername] = useState("");

  const [email, setEmail] = useState("");

  const [password, setPassword] = useState("");

  const [success, setSuccess] = useState(false);

  const { user, dispatch } = useContext(Context);

  const PF = "http://localhost:5000/images/";

  const handleSubmit = async (e) => {

    e.preventDefault();

    dispatch({ type: "UPDATE\_START" });

    const updatedUser = {

      userId: user.\_id,

      username,

      email,

      password,

    };

    if (file) {

      const data = new FormData();

      const filename = Date.now() + file.name;

      data.append("name", filename);

      data.append("file", file);

      updatedUser.profilePic = filename;

      try {

        await axios.post("/upload", data);

      } catch (err) {}

    }

    try {

      const res = await axios.put("/users/" + user.\_id, updatedUser);

      setSuccess(true);

      dispatch({ type: "UPDATE\_SUCCESS", payload: res.data });

    } catch (err) {

      dispatch({ type: "UPDATE\_FAILURE" });

    }

  };

  return (

    <div className="settings">

      <div className="settingsWrapper">

        <div className="settingsTitle">

          <span className="settingsUpdateTitle">Update Your Account</span>

          <span className="settingsDeleteTitle">Delete Account</span>

        </div>

        <form className="settingsForm" onSubmit={handleSubmit}>

          <label>Profile Picture</label>

          <div className="settingsPP">

            <img

              src={

                file

                  ? URL.createObjectURL(file)

                  : user.profilePic

                  ? PF + user.profilePic

                  : "https://cdn.pixabay.com/photo/2015/10/05/22/37/blank-profile-picture-973460\_960\_720.png"

              }

              alt=""

            />

            <label htmlFor="fileInput">

              <i className="settingsPPIcon far fa-user-circle"></i>

            </label>

            <input

              type="file"

              id="fileInput"

              style={{ display: "none" }}

              onChange={(e) => setFile(e.target.files[0])}

            />

          </div>

          <label>Username</label>

          <input

            type="text"

            placeholder={user.username}

            onChange={(e) => setUsername(e.target.value)}

          />

          <label>Email</label>

          <input

            type="email"

            placeholder={user.email}

            onChange={(e) => setEmail(e.target.value)}

          />

          <label>Password</label>

          <input

            type="password"

            onChange={(e) => setPassword(e.target.value)}

          />

          <button className="settingsSubmit" type="submit">

            Update

          </button>

          {success && (

            <span

              style={{ color: "green", textAlign: "center", marginTop: "20px" }}

            >

              Profile has been updated...

            </span>

          )}

        </form>

      </div>

    </div>

  );

}

Src/page/settings/single.jsx

import "./single.css";

import SinglePost from "../../components/singlePost/SinglePost";

export default function Single() {

  return (

    <div className="single">

      <SinglePost/>

    </div>

  );

}

**Src/page/write.jsx**

import { useContext, useState } from "react";

import "./write.css";

import axios from "axios";

import { Context } from "../../context/Context";

export default function Write() {

  const [title, setTitle] = useState("");

  const [desc, setDesc] = useState("");

  const [file, setFile] = useState(null);

  const { user } = useContext(Context);

  const handleSubmit = async (e) => {

    e.preventDefault();

    const newPost = {

      username: user.username,

      title,

      desc,

    };

    if (file) {

      const data =new FormData();

      const filename = Date.now() + file.name;

      data.append("name", filename);

      data.append("file", file);

      newPost.photo = filename;

      try {

        await axios.post("/upload", data);

      } catch (err) {}

    }

    try {

      const res = await axios.post("/posts", newPost);

      window.location.replace("/post/" + res.data.\_id);

    } catch (err) {}

  };

  return (

    <div className="write">

      {file && (

        <img className="writeImg" src={URL.createObjectURL(file)} alt="" />

      )}

      <form className="writeForm" onSubmit={handleSubmit}>

        <div className="writeFormGroup">

          <label htmlFor="fileInput">

            <i className="writeIcon fas fa-plus"></i>

          </label>

          <input

            type="file"

            id="fileInput"

            style={{ display: "none" }}

            onChange={(e) => setFile(e.target.files[0])}

          />

          <input

            type="text"

            placeholder="Title"

            className="writeInput"

            autoFocus={true}

            onChange={e=>setTitle(e.target.value)}

          />

        </div>

        <div className="writeFormGroup">

          <textarea

            placeholder="Tell your story..."

            type="text"

            className="writeInput writeText"

            onChange={e=>setDesc(e.target.value)}

          ></textarea>

        </div>

        <button className="writeSubmit" type="submit">

          Publish

        </button>

      </form>

    </div>

  );

}

**Src/app.js**

import Home from "./pages/home/Home";

import TopBar from "./components/topbar/TopBar";

import Single from "./pages/single/Single";

import Write from "./pages/write/Write";

import Settings from "./pages/settings/Settings";

import Login from "./pages/login/Login";

import Register from "./pages/register/Register";

import About from "./pages/about/About";

import { BrowserRouter as Router, Switch, Route} from "react-router-dom";

import { useContext } from "react";

import { Context } from "./context/Context";

function App() {

  const { user } = useContext(Context);

  return (

    <Router>

      <TopBar />

      <Switch>

        <Route exact path="/">

          <Home />

        </Route>

        <Route path="/register">{user ? <Home /> : <Register />}</Route>

        <Route path="/login">{user ? <Home /> : <Login />}</Route>

        <Route path="/write">{user ? <Write /> : <Register />}</Route>

        <Route path="/settings">{user ? <Settings /> : <Register />}</Route>

        <Route path="/post/:postId">

          <Single />

        </Route>

        <Route path="/about">{<About />}</Route>

      </Switch>

    </Router>

  );

}

export default App;

**Src/index.js**

import React from "react";

import ReactDOM from "react-dom";

import App from "./App";

import { ContextProvider } from "./context/Context";

ReactDOM.render(

  <React.StrictMode>

    <ContextProvider>

      <App />

    </ContextProvider>

  </React.StrictMode>,

  document.getElementById("root")

);

**Src/public/index.html**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="utf-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <meta name="theme-color" content="#000000" />

  <meta name="description" content="Web site created using create-react-app" />

  <link rel="preconnect" href="https://fonts.googleapis.com">

  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

  <link href="https://fonts.googleapis.com/css2?family=PT+Serif:wght@400;700&family=Source+Serif+Pro&display=swap" rel="stylesheet">

  <!-- CSS only -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

  <style>

    \* {

      margin: 0;

    }

  </style>

  <style>

    .link{

      text-decoration: none;

      color: inherit;

    }

  </style>

  <title>Blog React App</title>

</head>

<body>

  <noscript>You need to enable JavaScript to run this app.</noscript>

  <div id="root"></div>

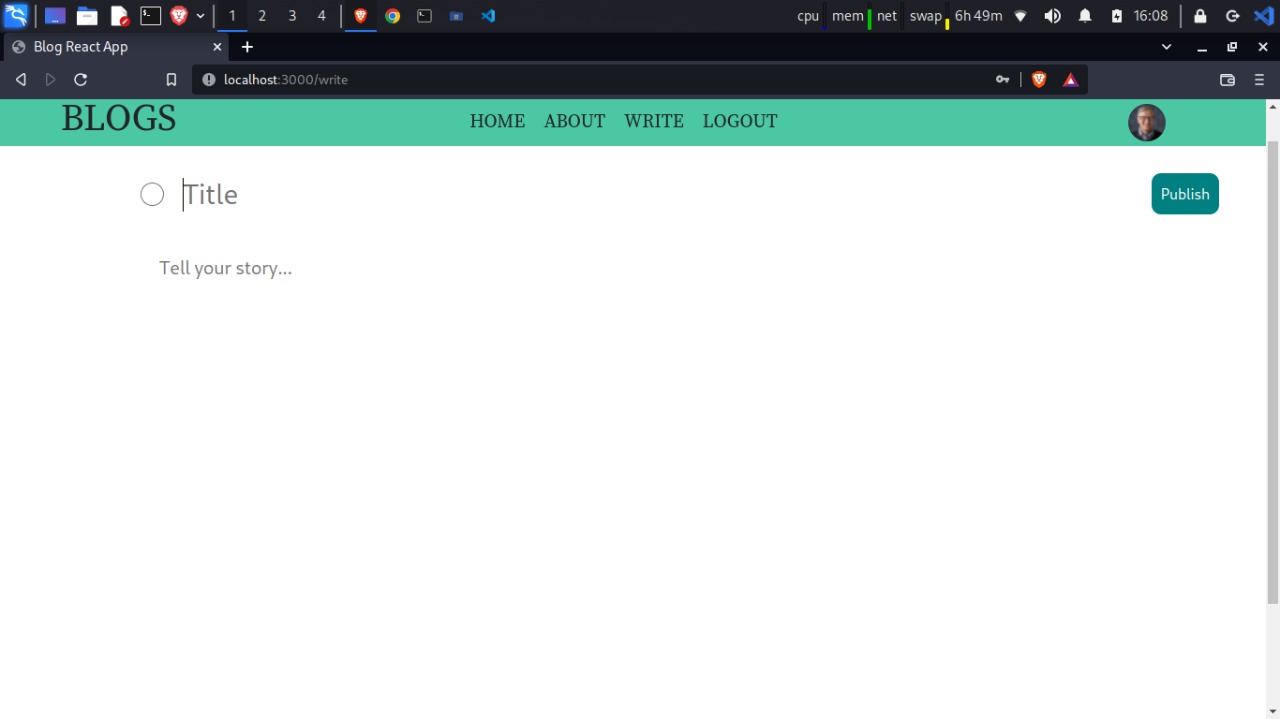
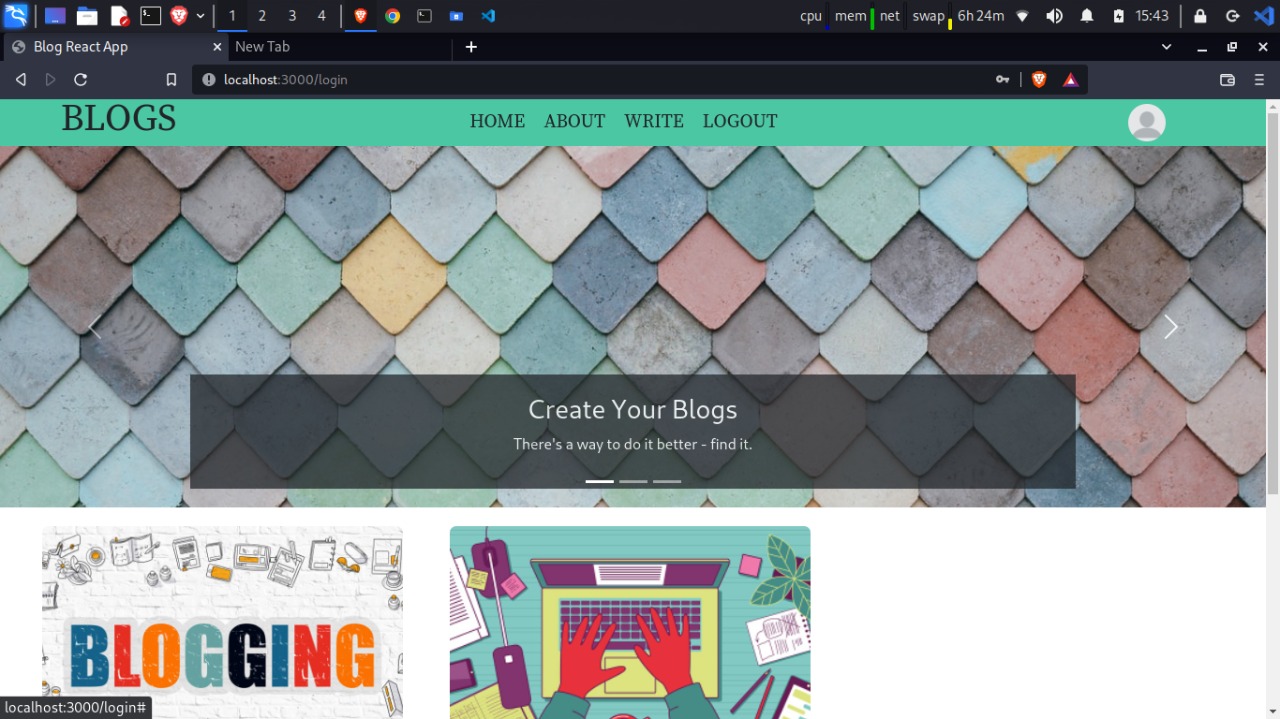
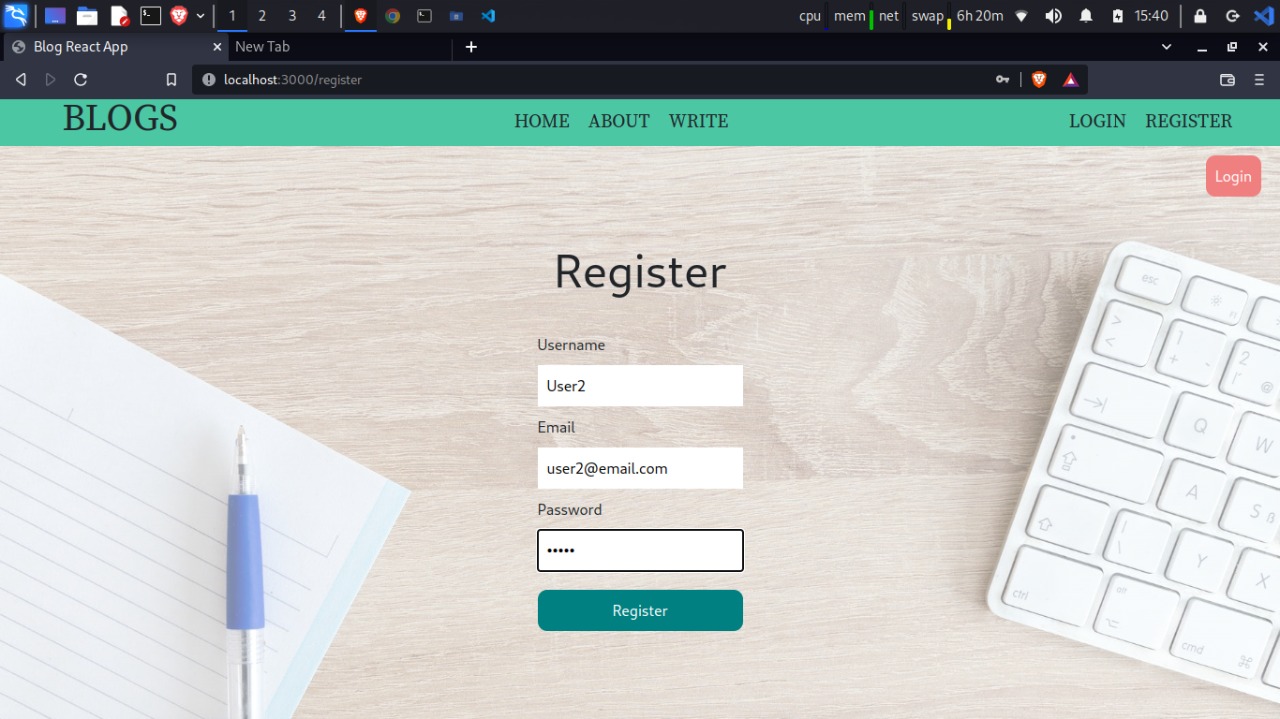
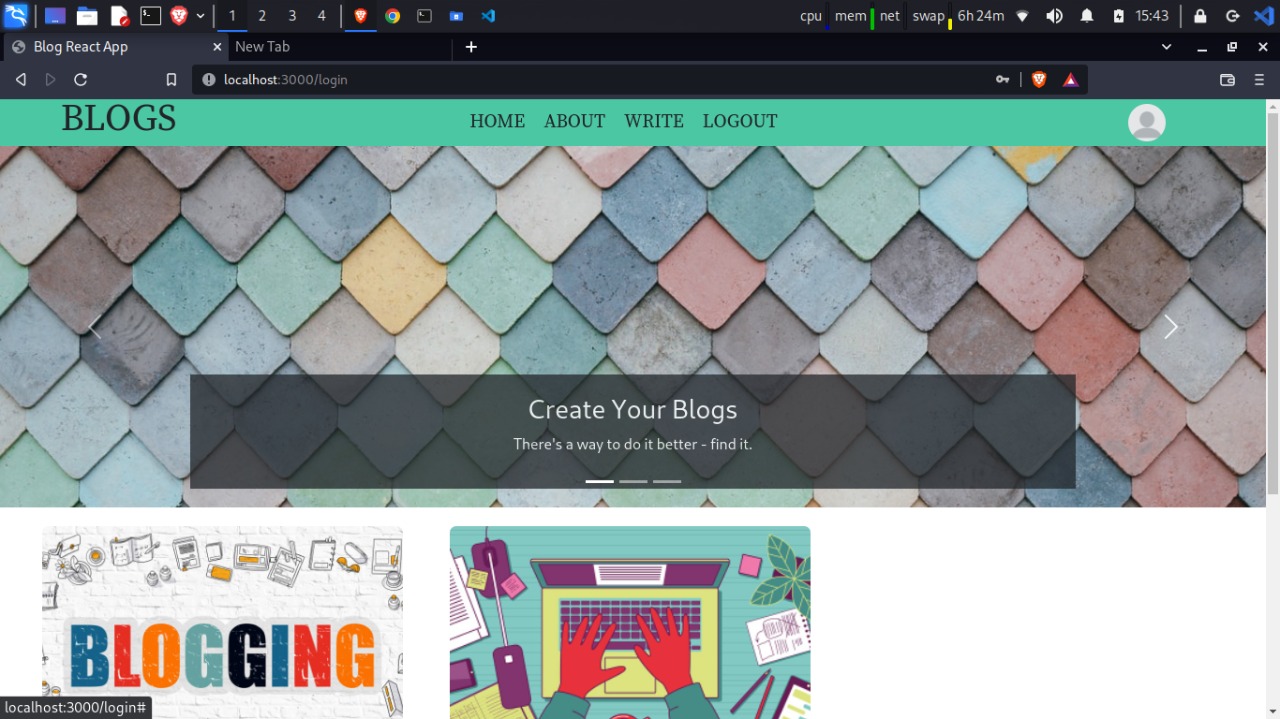
  <!-- JavaScript Bundle with Popper -->

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>

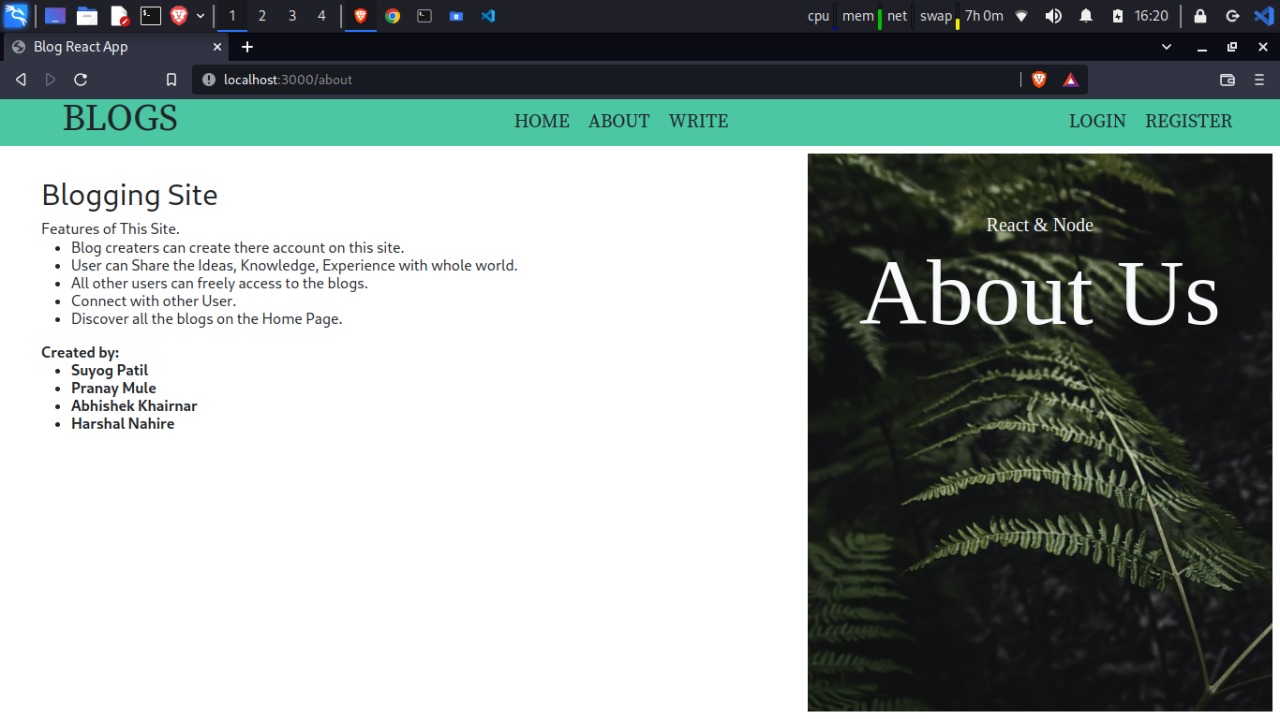
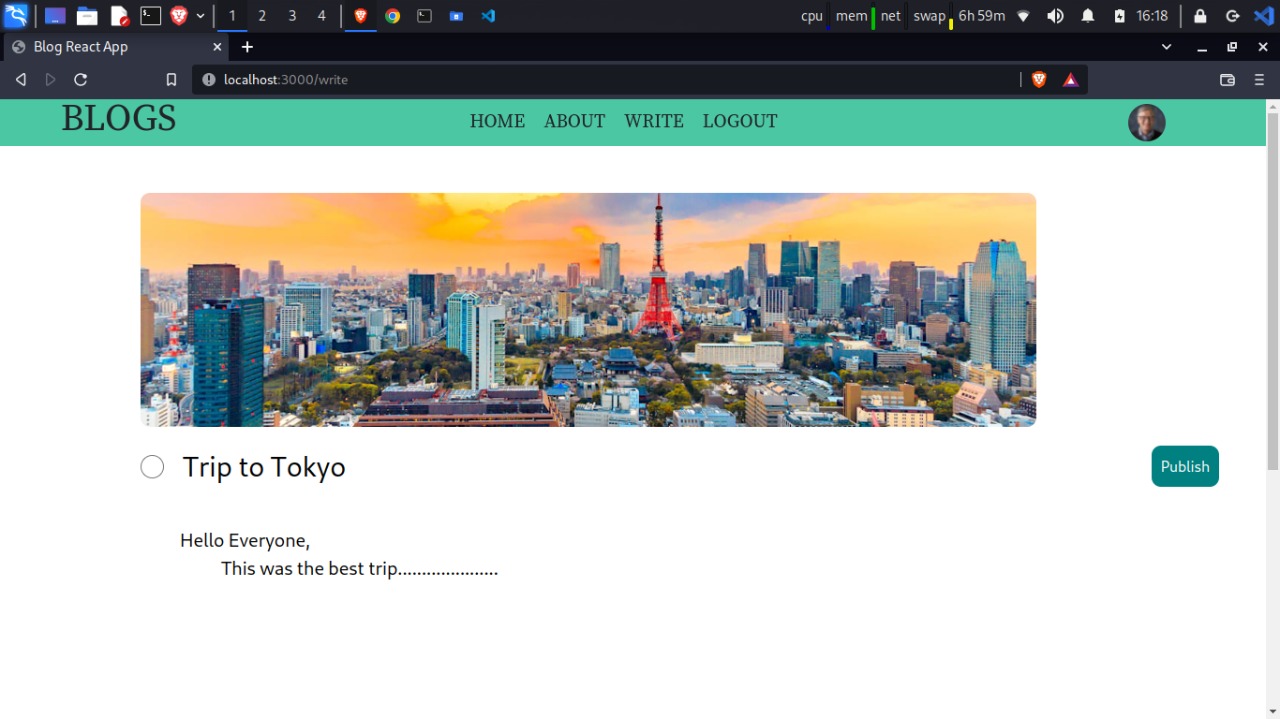
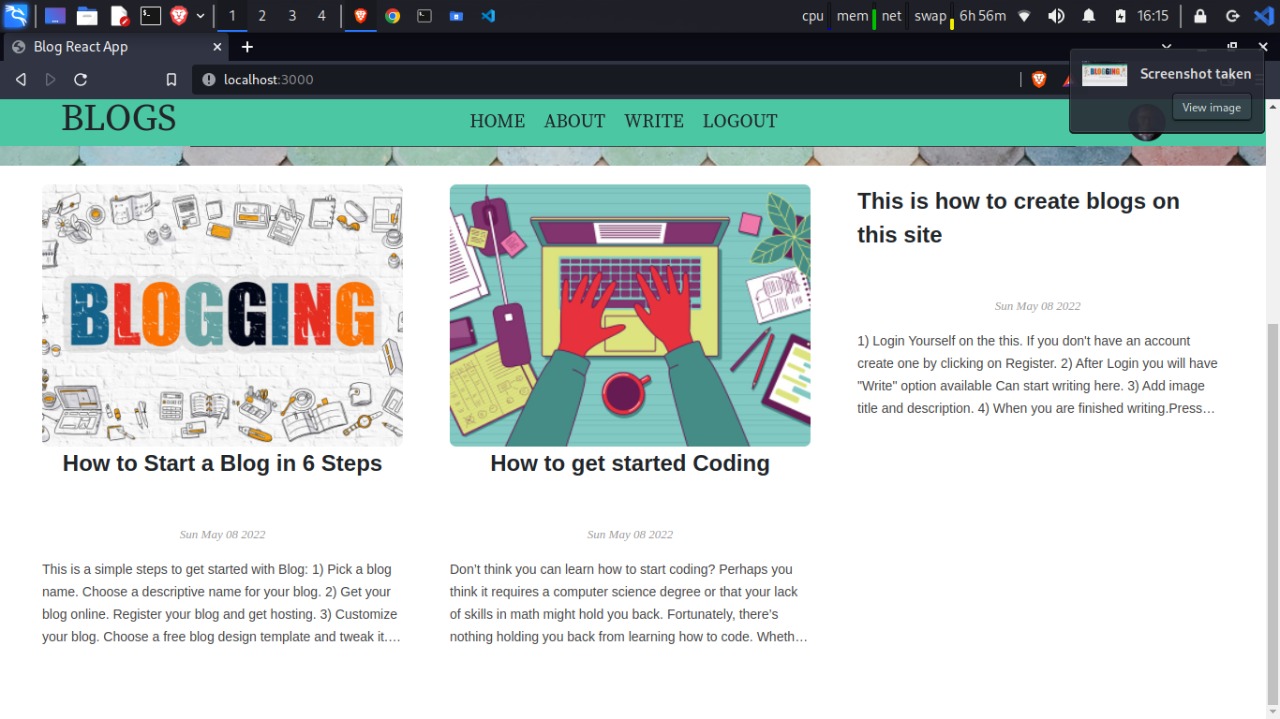
</body>

</html>

OUTPUT



****



1. **Future work :**

We will make some future improvement in our project by making this website more responsive and will add a feature of selling premium blogs which will help people to access premium blogs and through which even the blogger can earn.

We will be adding a feature through which the blogger can get feedback from the user.

We are even planning to add blog categories which will help the user to figure out perfect blogs for them and will be adding a feature which will allow the user to follow the specific blogger they like through this the blogger will get more attention.

1. **Conclusion**

Here we conclude that we have created a application using MERN full stack stack used to manage and publish content on the internet in the form of a blog**.**

**11. References:**

* <https://nodejs.org/docs/latest-v17.x/api/documentation.html>
* <https://www.npmjs.com/>
* <https://mongoosejs.com/docs/>
* <https://expressjs.com/en/5x/api.html>
* <https://reactnative.dev/docs/tutorial>