Buffer Overflow

A PROJECT REPORT
for
Mini Project (KCA353)
Session (2023-24)

Submitted by

Himanshu Jaiswal
2200290140071
Himanshu Kumar
2200290140072

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Dr. Vipin Kumar
Associate Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206
(Feburary-2024)

DECLARATION

I hereby declare that the work presented in report entitled “Buffer Overflow” was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University of Institute. I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, that are not my original contribution. I have used quotation marks to identify verbatim sentences and give credit to the original authors/sources. I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name: Himanshu Jaiswal

Roll No.: 2200290140071

Name: Himanshu Kumar

Roll No.: 2200290140072

CERTIFICATE

Certified that **Himanshu Jaiswal 2200290140071, Himanshu Kumar 2200290140072** have carried out the project work having "**Buffer Overflow**" (**Mini Project-KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the students themselves and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Himanshu Jaiswal 2200290140071
Himanshu Kumar2200290140072

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Vipin Kumar
Associate Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Arun Tripathi
Head
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Buffer Overflow

Himanshu Jaiswal
Himanshu Kumar

ABSTRACT

Buffer Overflow is a web platform designed for our college's coding community, drawing inspiration from successful programming platforms. It fosters collaborative learning, providing a localized space for students to share knowledge, engage in coding challenges, and build a vibrant coding culture. The project aims to enhance the student experience through tailored features, including mentorship programs, localized challenges, and a job board. "Buffer Overflow" envisions becoming the go-to platform for coding enthusiasts within our campus, promoting collaboration, skill development, and community building. Join us in shaping a dynamic coding culture with Buffer Overflow. The platform facilitates seamless knowledge exchange through features like user-friendly question and answer posting, collaborative coding challenges, and a reputation system that motivates active participation. Emphasizing inclusivity and community building, "Buffer Overflow" integrates with campus events, courses, and coding clubs to provide a holistic learning experience.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my thesis supervisor, Dr. Vipin Kumar (Associate Professor) for his guidance, help and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions have guided me a lot in completing this project successfully.

Words are not enough to express my gratitude to Dr. Arun Kumar Tripathi, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions. Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep mylife filled with enjoyment and happiness.

Himanshu Jaiswal (2200290140071)

Himanshu Kumar (2200290140072)

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

Welcome to, your gateway to a vibrant coding community! In the dynamic landscape of coding, honing skills is paramount, and we recognize the need to provide aspiring developers with a unique space to collaborate, learn, and excel in their programming journey. It is an innovative web platform designed to redefine the coding community experience.

Buffer Overflow invites coders to join this transformative journey, contributing to the establishment of a dynamic and thriving coding culture. Positioned as the central hub for coding enthusiasts, the platform encourages collaboration, learning, and the collective shaping of the future of coding excellence. Join us in constructing a dynamic and thriving coding culture with Buffer Overflow.

Buffer Overflow stands as a dynamic web space, inspired by successful programming communities, offering a localized hub where coding enthusiasts can actively engage, share knowledge, and collaborate. The platform facilitates seamless knowledge exchange through a user-friendly interface, enabling users to post and answer coding questions. A distinctive reputation system adds a motivational aspect, acknowledging and rewarding valuable contributions.

1.2 Problem Statement

In the realm of coding and skill development, aspiring programmers often face challenges in finding a centralized and inclusive platform tailored to their learning needs.

Furthermore, the absence of a dedicated platform within a campus environment leaves students without a unified space to actively collaborate, share knowledge, and participate in coding challenges. Existing coding communities may lack tailored features such as mentorship programs, campus-specific challenges, and localized job boards, hindering students' ability to seamlessly integrate coding practice into their educational journey.

1.3 Objectives

The primary objectives of Buffer Overflow are to establish a centralized and localized web platform within the campus community. The platform aims to foster seamless knowledge exchange by providing a user-friendly interface for posting and answering coding questions. Buffer Overflow seeks to motivate users through a distinctive reputation system, recognizing and rewarding valuable contributions to encourage active participation. The platform aims to promote real-time collaboration among users through interactive discussion forums, creating a dynamic and engaging coding culture. Additionally, Buffer Overflow will provide customizable testing and learning paths, allowing users to focus on specific coding areas based on their strengths and weaknesses.

1.4 Scope

- Develop Buffer Overflow as a centralized web platform, providing a dedicated space for students and coding enthusiasts within the campus community.
- Implement features that facilitate seamless knowledge exchange, allowing users to post and answer coding questions in a collaborative environment.
- Integrate a distinctive reputation system into the platform, acknowledging and rewarding users for valuable contributions to encourage active participation.
- Integrate Buffer Overflow with campus events, coding clubs, and courses to enhance inclusivity and provide a holistic learning experience for users.
- Develop and implement features such as mentorship programs, campus-specific coding challenges, and a dedicated job board to cater specifically to the campus coding community.
- Provide tools for real-time collaboration among users, including interactive discussion forums and features that encourage engagement within the coding culture.
- Include discussion forums to foster a collaborative learning community among students.
- Create a supportive environment within Buffer Overflow that boosts users' confidence, reduces coding apprehension, and serves as a dynamic space for successful skill development.
- Regularly update the platform to align with industry trends, coding challenges, and exam patterns, ensuring users have access to the latest resources for skill development.

1.5 Feature

- Dedicated platform for posting and answering coding questions within a collaborative environment.
- A unique system that acknowledges and rewards users for valuable coding contributions.
- Integration with campus events, coding clubs, and courses to enhance inclusivity and provide a holistic learning experience.

- Features such as mentorship programs, campus-specific coding challenges, and a dedicated job board for the campus coding community.
- Tools for real-time collaboration among users, including interactive discussion forums and features that encourage engagement within the coding culture.
- A user-friendly interface designed for seamless navigation, ensuring a positive and intuitive experience.
- Regular updates to align the platform with industry trends, coding challenges, and exam patterns, providing users with the latest resources for skill development.

1.6 Hardware / Software Requirement

S. N.	Description
1	Hard disk.
2	RAM.
3	i3 core or above processor.

Table 1.1 Hardware Requirements

1. PC with 5 GB or more Hard disk:

This specifies the storage requirement for the PC. It should have a hard disk with a capacity of 5 gigabytes (GB) or more. This is where you store your operating system, software applications, and data.

2. PC with 2 GB RAM:

This sets the minimum random-access memory (RAM) requirement for the PC. It should have at least 2 gigabytes of RAM. RAM is essential for running applications and the operating system efficiently**.**

3. PC with core i3 or above processor:

This specifies the processor requirement for the PC. It should have an Intel Core i3 processor or a more powerful one. The processor is a crucial component that determines the computer's overall speed and performance.

S. N.	Description	Type
1	Operating System	Windows 10 or 11 or Ubuntu 18.04 or above
2	Front End	HTML5, CSS3, JavaScript, React JS v18
3	Back End	Node JS v20, MongoDB
4	IDE	VS Code
5	Browser	Chrome, Firefox, Edge

Table 1.2 Software Requirements

Operating System:

Windows 10 or 11
Ubuntu 18.04 or above
These are the supported operating systems for the development environment. You can use either Windows 10 or 11, or Ubuntu 18.04 or a newer version.

Front End:

HTML5
CSS3
JavaScript
React JS v18
These are the technologies for the front end of a web application. HTML (Hypertext Markup Language) is used for structuring content, and CSS (Cascading Style Sheets) is used for styling and layout. JavaScript provides the foundation for dynamic behaviour, while React JS simplifies the creation of complex user interfaces through its component-based structure and efficient rendering mechanisms.

Back End:

Node JS v20
MongoDB
Node.js v20 and MongoDB form a powerful backend technology stack. Node.js facilitates server-side JavaScript execution, while MongoDB offers a scalable and flexible data storage solution.

IDE (Integrated Development Environment):

VS Code
Visual Studio Code (VS Code) is the preferred integrated development environment for coding. It provides features like code highlighting, debugging, and version control integration.

Browser:

Chrome
Firefox
Edge

CHAPTER 2

FEASIBILITY STUDY

After doing the project Buffer Overflow, study and analysing all the existing or required functionalities of the system, the next task is to do the feasibility study for the project. All projects are feasible - given unlimited resources and infinite time. Feasibility study includes consideration of all the possible ways to provide a solution to the given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirements.

2.1 Technical Feasibility

- **Infrastructure**: Ensure technical requirements (servers, databases, web development) align with available resources.
- **Technology Stack**: The chosen technology stack should be appropriate for developing a scalable and user-friendly platform.

2.2 Operational Feasibility

- **Team Competency**: Ensure that the development team possesses the necessary skills and expertise.
- **User Training**: Assess the ease with which users can navigate the platform and understand its features**.**

2.3 Economical Feasibility

- **Budget**: The project budget should cover development, maintenance, marketing, and potential scalability costs.
- **Revenue Model**: A clear revenue model, such as subscription plans, advertisements, or partnerships, should be in place to sustain the project.

2.4 Behavioural Feasibility

- Assess programmers' openness to adopting Buffer Overflow, considering potential adjustments in coding habits.
- Implement engagement strategies to foster a sense of community, collaboration, and inclusivity.
- Ensure Buffer Overflow caters to a diverse user base by implementing features that resonate with different programming backgrounds.
- Identify and address potential obstacles to adoption, ensuring a seamless and welcoming experience for all users.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 Functional requirements

- **User Registration and Authentication:**
  Users should be able to register accounts securely and log in using authentication methods.
- **Profile Management:**
  Users should have the ability to create, edit, and manage their profiles, including personal details and preferences.
- **Question Posting and Answering:**
  Facilitate the posting of coding questions and providing answers within the community.
- **User Reputation System:**
  Recognize and reward users based on their contributions to the community.
- **Collaborative Coding Tools:**
  Enhance user experience by providing tools for collaborative coding efforts.
- **Community Forums:**
  Foster a sense of community through dedicated discussion forums.
- **Commenting and Feedback:**
  Facilitate user interactions through commenting and provide mechanisms for constructive feedback.
- **User Activity Feed:**

  Keep users informed about recent activities and engagements within the community.

3.2 Non-functional requirements

- **Performance:**

  The platform should provide a responsive and efficient user experience, with minimal loading times for quizzes, tests, and other features, even during peak usage.

- **Scalability:**

  The system should be scalable to handle an increasing number of users, quizzes, and content without compromising performance.

- **Reliability:**

  The platform should be reliable, with minimal downtime and a robust backup and recovery system in place**.**

- **Security:**

  User data should be securely stored and transmitted. The platform should implement authentication mechanisms, encryption, and protection against common security threats.

- **Compatibility:**

  The platform should be compatible with various web browsers, operating systems, and devices to ensure a broad user reach.

- **Usability:**

  The user interface should be intuitive and user-friendly, catering to users with varying levels of technological proficiency.

- **Availability:**

  The platform should be available 24/7, with scheduled maintenance communicated to users in advance.

- **Accessibility:**

  The platform should be accessible to users with disabilities, complying with accessibility standards to ensure inclusivity.

- **Load Balancing:**

  Load balancing mechanisms should be in place to distribute user traffic efficiently across servers, preventing performance bottlenecks.

- **Data Backup and Recovery:**

  Regular backups of user data should be performed, and a reliable recovery plan should be in place in case of data loss or system failures**.**

- **Compliance:**

  The platform should comply with relevant data protection regulations and other legal requirements in the jurisdictions it operates.

- **Interoperability:**

  The system should be designed to seamlessly integrate with other educational platforms or tools, if necessary**.**

- **Response Time:**

  The platform should have acceptable response times for user interactions, ensuring a smooth and interactive experience.

- **Capacity Planning:**

  Regular capacity planning assessments should be conducted to anticipate and address potential issues related to system load and resource usage.

- **Feedback and Error Handling:**

  The platform should provide clear feedback to users during interactions and effective error handling to guide users in case of mistakes or system issues.

CHAPTER 4

Testing

4.1 Unit Testing:

- Verify that individual functions/methods responsible for user registration validate input data correctly.
- Test the login functionality in isolation to ensure proper handling of authentication.
- Check the collaborative code editing module to ensure seamless integration and proper version control.

4.2 Integration Testing:

- Test the integration between the user registration module and the database to ensure user data is correctly stored.
- Verify the integration between the login module and the authentication system.
- Verify that user feedback, comments, and ratings are correctly associated with specific questions and answers.

4.3 System Testing:

- Test the entire user registration process, including form validation, database interaction, and confirmation email generation.
- Ensure that user profiles are created accurately, and email confirmations are sent promptly.
- Verify the entire collaborative coding process, from initiating a collaborative coding session to saving and sharing the code.
- Check for version control functionality to track changes made during collaborative coding.
- Check for proper access controls to ensure users can only view their own quiz results.
- Verify that the admin section is accessible only to authorized administrators.
- Verify the website's scalability by gradually increasing the number of registered users.

CHAPTER 5

Design

5.1 Flowchart

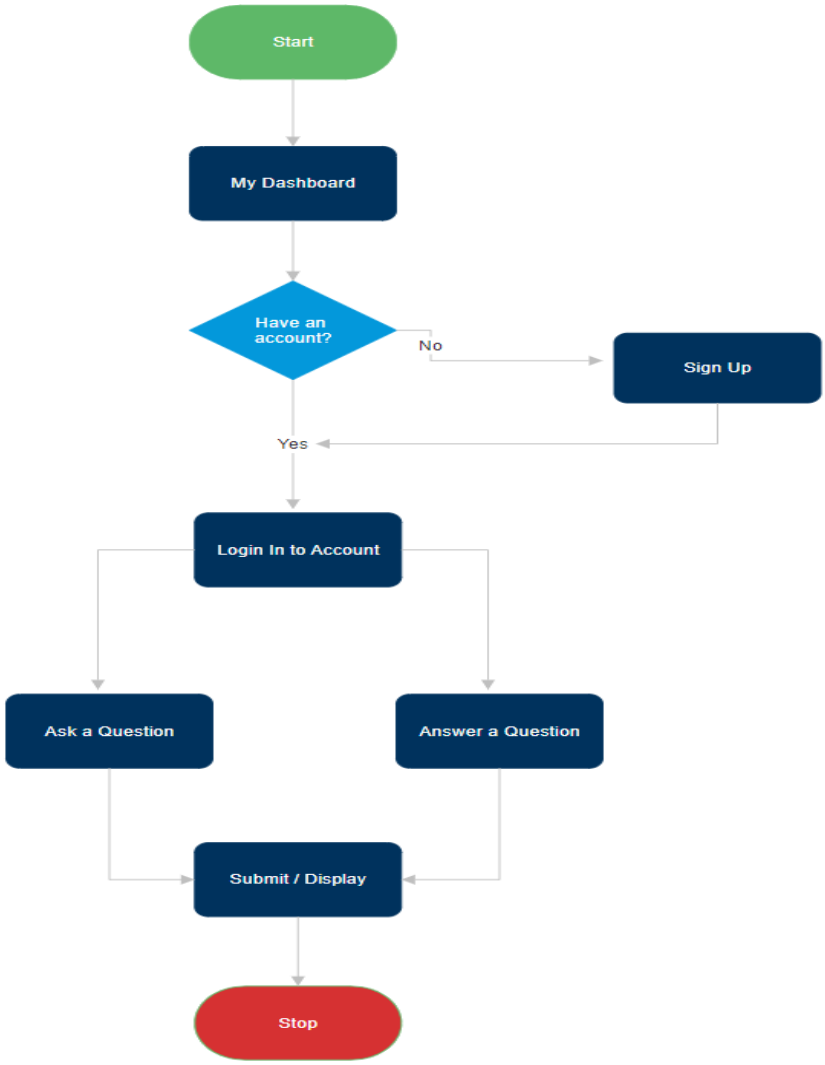

Fig. 5.1. Flowchart

5.2 DFD

0 Level Diagram

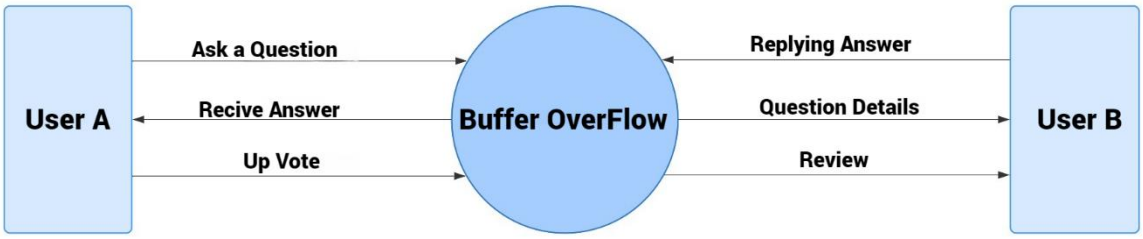

Fig. 5.2 0 Level Diagram

1.Level Diagram

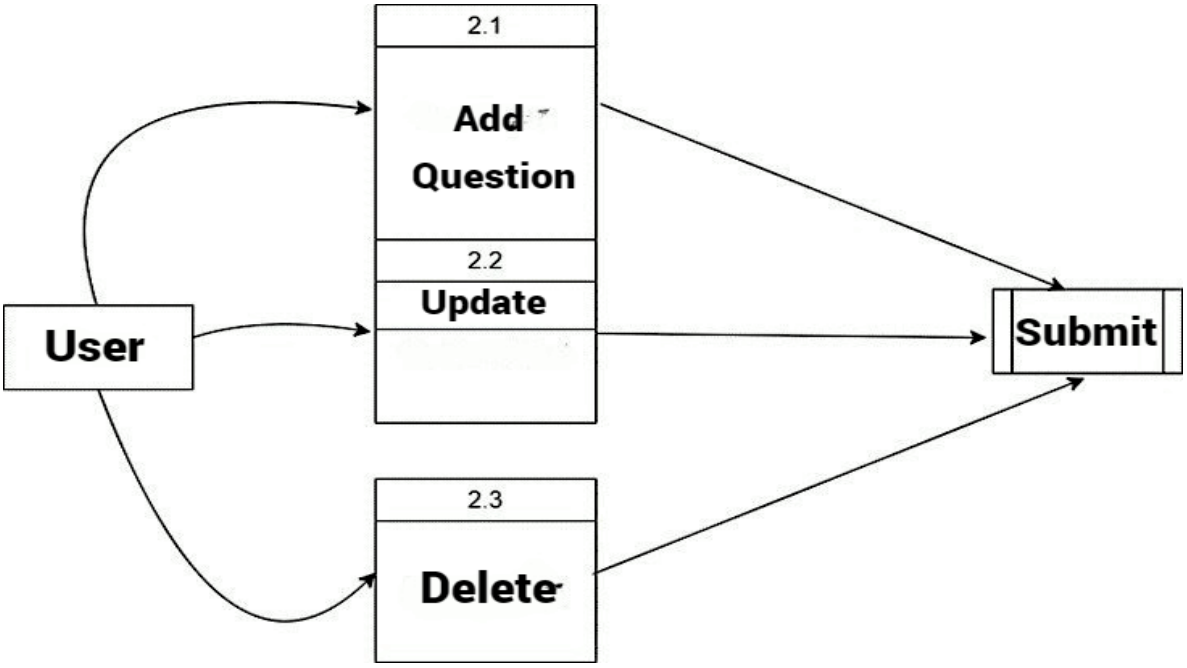

Fig. 5.2 1 Level Diagram

5.3 ER Diagram

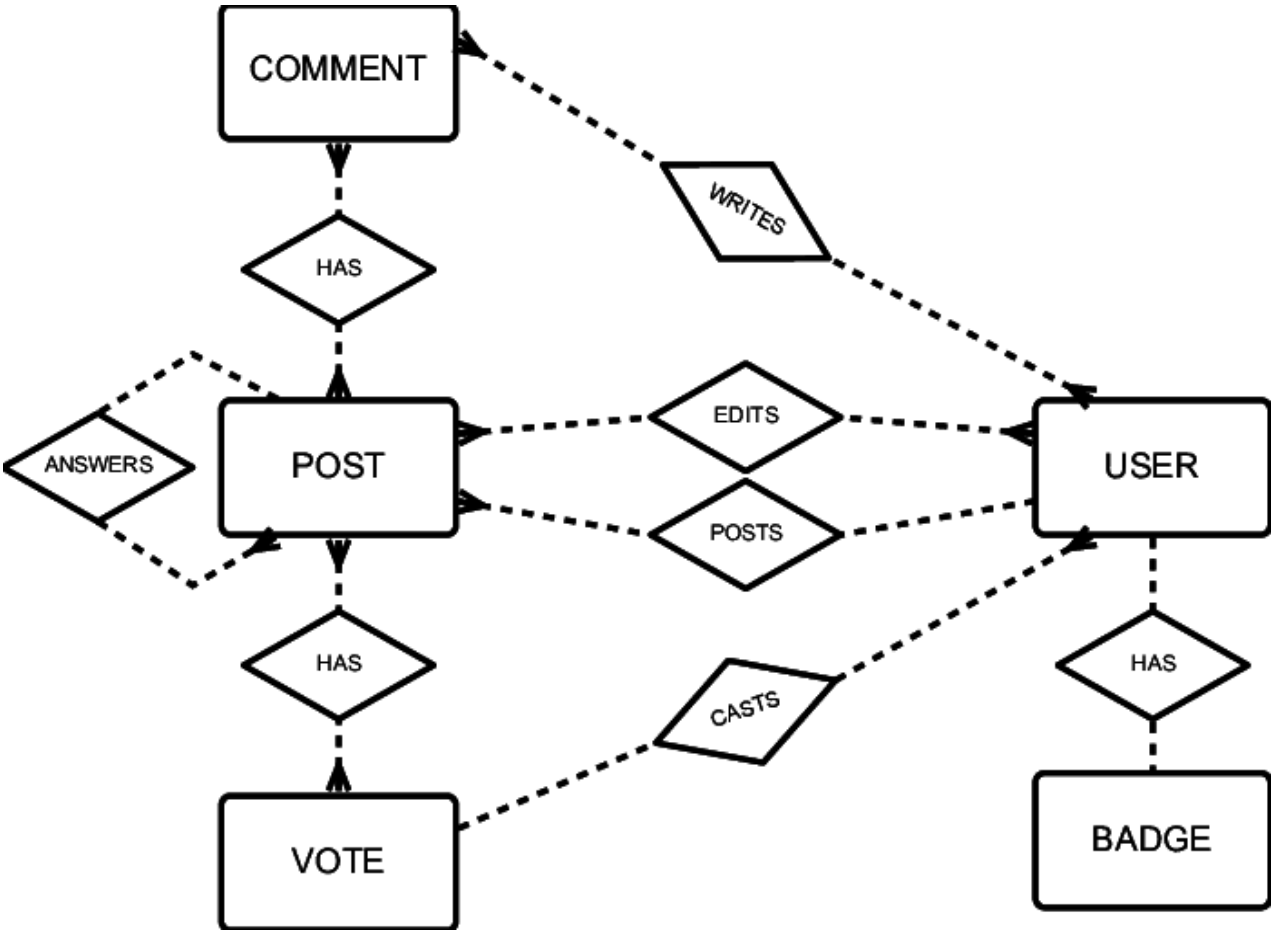

Fig.5.3 ER Diagram

5.4 Use Case Diagram

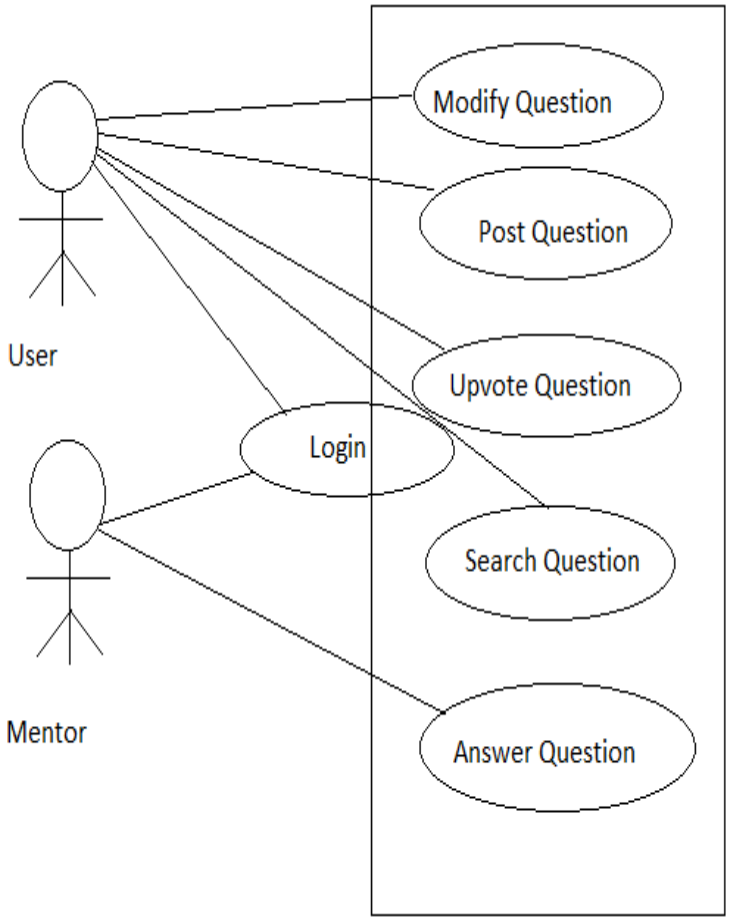

Fig.5.4. Use Case Diagram

CHAPTER 6

PROJECT SCREENSHOT

6.1 HOME PAGE

The home page serves as the entry point to the platform, presenting users with three key modules: Login, Register, and Admin. Its clean and intuitive design ensures ease of navigation for users of all levels.

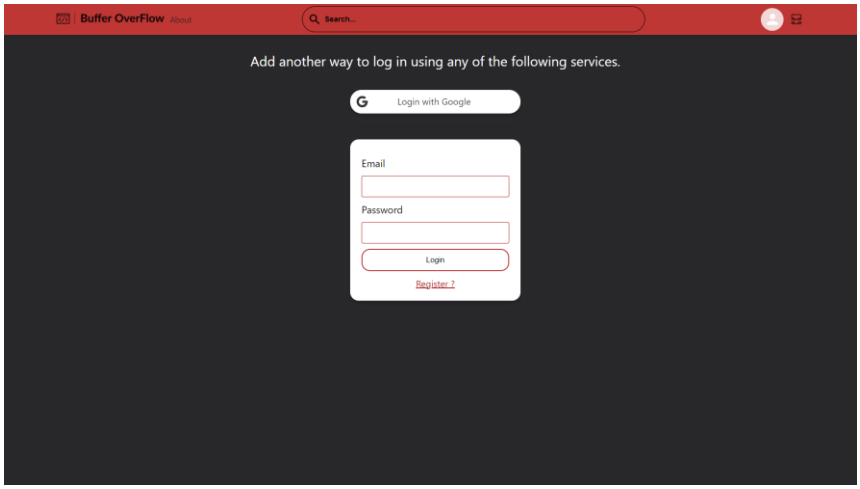

Fig.6.1 Home Page

6.2 USER SIGN-UP PAGE

The User Sign-Up Page is dedicated to new users aiming to join the platform. It prompts users to enter essential information such as their name, email, college name, and password, streamlining the registration process.

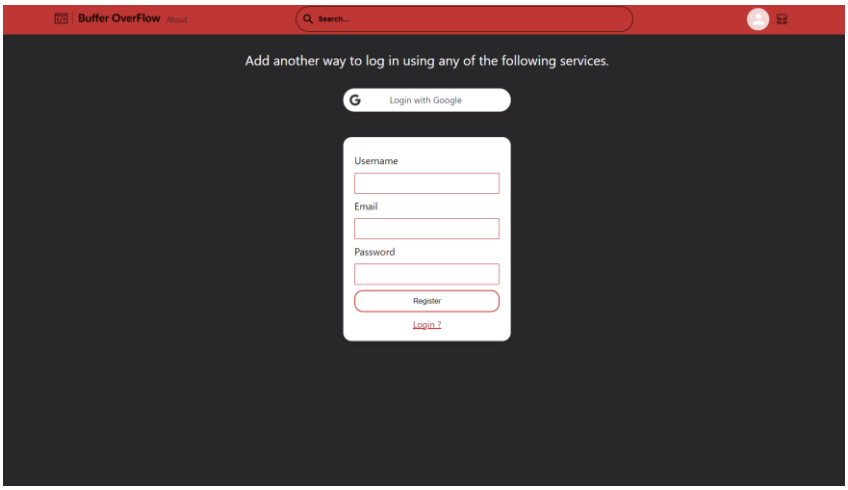

Fig.6.2 User Sign Up Page

6.3 USER LOGIN & REGISTER WITH GOOGLE

The User Login Page provides a secure gateway for existing users to access their accounts. Users are required to input their email and password to log in successfully.

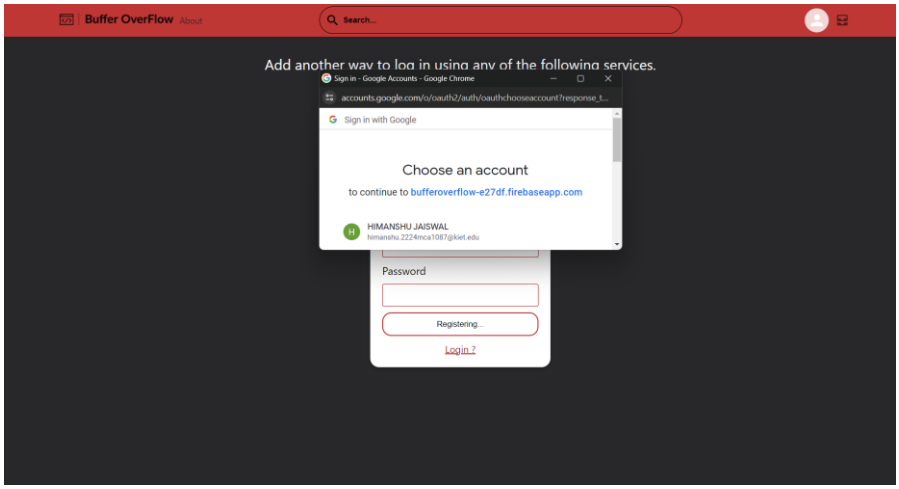

Fig.6.3 User Login & Register Page

6.4 USER DASHBOARD

Upon successful login, users are directed to the User Dashboard. This centralized hub serves as the starting point for test attempts, providing a user-friendly interface for navigating through various features.

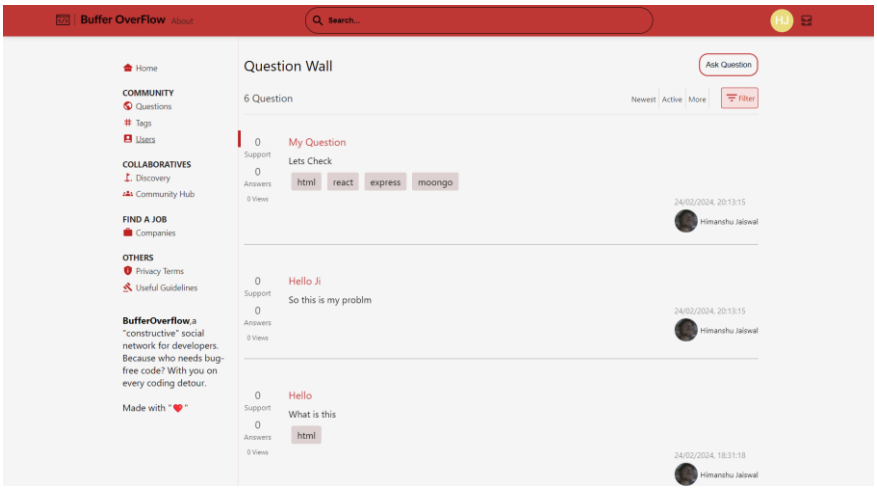

Fig.6.4 User Dashboard

6.5 ASK QUESTION PAGE

The Ask Question Page is where users can ask questions. Organized by Title, Body and Tags users can choose and ask based on their preferences.

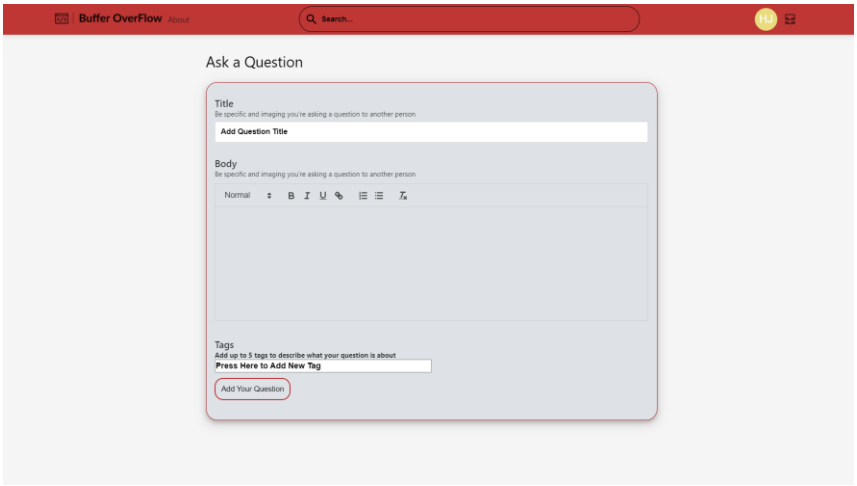

Fig.6.5 Ask Question Page

6.6 USER ANSWER PAGE

The User Answer Page serves as a platform for users to engage with and provide solutions to questions posed by the community. It facilitates a seamless interaction between users seeking answers and those offering solutions.

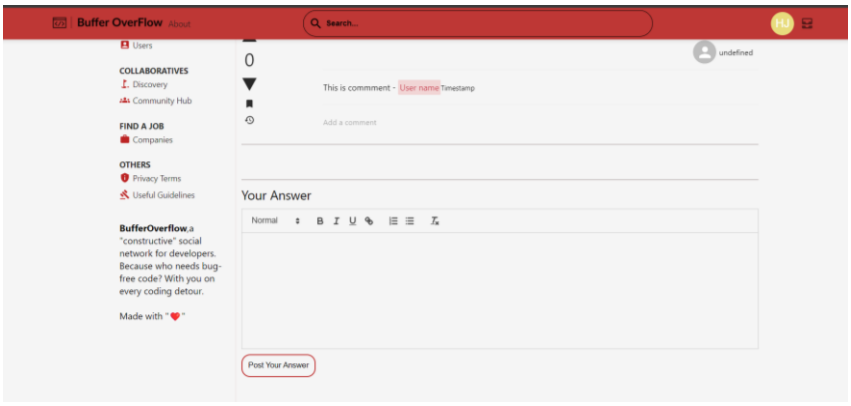

Fig.6.6 User Answer Page

6.7 ABOUT PAGE

The About page provides users with an overview of our app's mission and features. Learn about how our platform empowers users to connect, learn, and grow together by tracking questions, exploring solutions, and engaging in discussions. We are dedicated to continuous improvement and value your feedback to enhance your experience. Join the Buffer Overflow community today and be part of a vibrant exchange of knowledge and ideas.

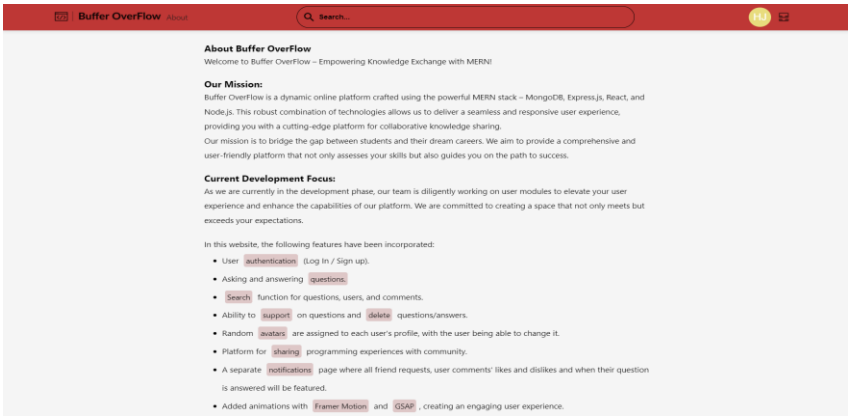

Fig.6.7 About Page

CHAPTER 8

CODING

Frontend

Sidebar

```
import React from 'react'
import {Public,  Work } from '@mui/icons-material'
import GroupsIcon from '@mui/icons-material/Groups';
import OtherHousesIcon from '@mui/icons-material/OtherHouses';
import TagIcon from '@mui/icons-material/Tag';
import AccountBoxIcon from '@mui/icons-material/AccountBox';
import GolfCourseIcon from '@mui/icons-material/GolfCourse';
import PrivacyTipIcon from '@mui/icons-material/PrivacyTip';
import GavelIcon from '@mui/icons-material/Gavel';



import { Link } from 'react-router-dom';
import './css/Sidebar.css'

function Sidebar() {
  return (
```

```
<div className='sidebar'>
   <div className='sidebar-container'>
      <div className='sidebar-options'>
         <div className='sidebar-option'>
         <div className='link'>
         <div className='link-tag'>
            <OtherHousesIcon />
            <Link>Home</Link>
            </div>
            </div>
         </div>
         <div className='sidebar-option'>
            <p><b>COMMUNITY</b></p>
            <div className='link'>
               <div className='link-tag'>
                     <Public />
                     <Link to ="/">Questions</Link>

               </div>
               <div className='link-tag'>
                     <TagIcon />
                     <Link>Tags</Link>
```

```
        </div>
        <div className='link-tag'>
              <AccountBoxIcon />
              <Link>Users</Link>

        </div>

    </div>
</div>
    <div className='sidebar-option'>
        <p><b>COLLABORATIVES</b></p>
     <div className='link'>
        <div className='link-tag'>
              <GolfCourseIcon />
              <Link>Discovery</Link>

        </div>

        <div className='link-tag'>
              <GroupsIcon />
              <Link>Community Hub</Link>

        </div>
```

```
</div>
</div>

<div className='sidebar-option'>
   <p><b>FIND A JOB</b></p>
<div className='link'>
   <div className='link-tag'>
         <Work />
          <Link>Companies</Link>

   </div>

</div>
</div>

<div className='sidebar-option'>
   <p><b>OTHERS</b></p>
   <div className='link'>
   <div className='link-tag'>
         <PrivacyTipIcon  />
         <Link>Privacy Terms</Link>

   </div>
```

```
                <div className='link-tag'>
                    <GavelIcon />
                    <Link>Useful Guidelines</Link>

                </div>

                </div>

            </div>
            <br/>
            <div className='option'>
                <p><b>BufferOverflow</b>,a "constructive" social network for developers. Because who needs bug-free code? With you on every coding detour.<br/><br/> Made with "❤️"</p>
            </div>
        </div>
    </div>
  )
}

export default Sidebar
```

Add Question

```
import React, { useState} from 'react'
import { useSelector } from "react-redux";
```

```
import { selectUser } from "../../features/userSlice";
import ReactQuill from 'react-quill'
import 'react-quill/dist/quill.snow.css'
import { TagsInput } from 'react-tag-input-component'
import './index.css'
//import ChipsArray from "./TagsInput";
import { useNavigate } from 'react-router-dom'
import axios from "axios";
//import {WithContext as TagsInput} from  'react-tag-input'

function Question() {
  const user = useSelector(selectUser);
  const [title, setTitle] = useState("");
  const [body, setBody] = useState("");
  const [tags, setTag] = useState([]);
  const navigate = useNavigate();
  const [loading, setLoading] = useState(false)


  const handleQuill = (value) => {
    setBody(value);
  };

  const handleSubmit = async (e) => {
```

```
  e.preventDefault();

  if (title !== "" && body !== "") {
    setLoading(true);
    const bodyJSON = {
      title: title,
      body: body,
      tag: JSON.stringify(tags),
      user: user,
    };
    await axios
      .post("/api/question", bodyJSON)
      .then((res) => {
        // console.log(res.data);
        alert("Question added successfully");
        setLoading(false);
        navigate("/");
      })
      .catch((err) => {
        console.log(err);
        setLoading(false);
      });
  }
};
```

```
return (
  <div className='add-question'>
    <div className='add-question-container'>
      <div className='head-title'>
        <h1><center>Ask a Question</center></h1>
      </div>
      <div className='question-container'>
       <div className='question-options'>
          <div className='question-option'>
            <div className='title'>
              <h3>Title</h3>
              <small>
              Be specific and imaging you're asking a question to another person
              </small>
              <input onChange={(e) => setTitle(e.target.value)} type='text' placeholder='Add Question Title' />
            </div>
          </div>
        <div className='question-option'>
        <div className='title'>
              <h3>Body</h3>
              <small>
              Be specific and imaging you're asking a question to another person
```

```
                </small>
                <ReactQuill onChange={handleQuill} value={body}
              className='react-quill' theme='snow'/>
                </div>
            </div>
            <div className='question-option'>
                <div className='title'>
                <h3>Tags</h3>
                <small> <b>
                Add up to 5 tags to describe what your question is about</b>
                </small>

                <TagsInput value={tags} onChange={setTag} name='tags' placeHolder='Press Here to Add New Tag'  style={{ border: 'solid black' }} />
                {/* <ChipsArray /> */}
                <button disabled={loading} type='submit' onClick={handleSubmit} className='button'>{ loading ? "Adding Question..." : "Add Your Question"}</button>
                </div>
            </div>
          </div>
        </div>

    </div>

  </div>
```

```
  )
}

export default Question
```

About

```
import React from 'react'
import "./about.css"

function about() {
  return (
    <div className="home-container-1 about">

      <div className="home-container-2">
        <div className="main-bar" >
          <h3 className="about-h3">
            About Buffer OverFlow
          </h3>

          <p className="about-p">
           Welcome to Buffer OverFlow –  Empowering Knowledge Exchange with MERN!<br /></p><br/>
           <h3 className="about-h3">
           Our Mission:
```

</h3>

Buffer OverFlow is a dynamic online platform crafted using the powerful MERN stack – MongoDB, Express.js, React, and Node.js. This robust combination of technologies allows us to deliver a seamless and responsive user experience, providing you with a cutting-edge platform for collaborative knowledge sharing.

Our mission is to bridge the gap between students and their dream careers. We aim to provide a comprehensive and user-friendly platform that not only assesses your skills but also guides you on the path to success.

<h3 className="about-h3">

Current Development Focus:

</h3>

As we are currently in the development phase, our team is diligently working on user modules to elevate your user experience and enhance the capabilities of our platform. We are committed to creating a space that not only meets but exceeds your expectations.

In this website, the following features have been incorporated:

<li className="about-p">

User authentication (Log In /

```
  Sign up).
</li>
<li className="about-p">
  Asking and answering <span className="keyword">questions.</span>
</li>
<li className="about-p">
  <span className="keyword">Search</span> function for questions,
  users, and comments.
</li>
<li className="about-p">
  Ability to <span className="keyword">support</span> on questions and{" "}
  <span className="keyword">delete</span> questions/answers.
</li>

<li className="about-p">
  Random <span className="keyword">avatars</span> are assigned to
  each user's profile, with the user being able to change it.
</li>

<li className="about-p">
  Platform for <span className="keyword">sharing</span> programming
  experiences with community.
</li>
<li className="about-p">
```

```
          A separate <span className="keyword">notifications</span> page
          where all friend requests, user comments' likes and dislikes and
          when their question is answered will be featured.
        </li>
        <li className="about-p">
          Added animations with{" "}
          <span className="keyword">Framer Motion</span> and{" "}
          <span className="keyword">GSAP</span>, creating an engaging user
          experience.
        </li>

      </ul>
      </div>
      <br />
      <hr />
      <br />
      <h3 className="about-h3">
        Teams
      </h3>

      <h3>Himanshu Jaiswal <a href="https://www.linkedin.com/in/itshimanshu010/" target="_blank" rel="noopener noreferrer">Linkedin</a></h3>

      <h3>Himanshu Kumar <a href="https://www.linkedin.com/in/himanshu-kumar-81a050233/" target="_blank" rel="noopener noreferrer">Linkedin</a></h3>
```

```
        </div>
      </div>

    </div>

 )
}

export default about
```

Authentication

```
import {
  createUserWithEmailAndPassword,
  signInWithEmailAndPassword,
  signInWithPopup,
} from "firebase/auth";
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";
import { auth, provider } from "../../firebase";
import "./index.css";
import { Google } from "@mui/icons-material";

function Index() {
  const navigate = useNavigate();
```

```
const [register, setRegister] = useState(false);
const [email, setEmail] = useState("");
const [password, setPassword] = useState("");
const [username, setUsername] = useState("");
const [error, setError] = useState("");
const [loading, setLoading] = useState(false);



function validateEmail(email) {
  const reg = /^([A-Za-z0-9_\-.])+@([A-Za-z0-9_\-.])+\.([A-Za-z]{2,4})$/;
  if (reg.test(email) === false) {
    return false;

  } else return true;
}

const handleGoogleSignIN = () => {
  setLoading(true);
  signInWithPopup(auth, provider)
    .then((res) => {
      setLoading(false);
      // console.log(res);
      navigate("/");
```

```
    })
    .catch((error) => {
      setLoading(false);
      console.log(error);
    });
};

const handleSignIn = (e) => {
  e.preventDefault();
  setError("");
  setLoading(true);

  if (email === "" || password === "") {
    setError("Required field is missing");
    setLoading(false);
  }
  else if (!validateEmail(email)) {
    setError("Email is malformed");
    setLoading(false);
  }
  else {
    signInWithEmailAndPassword(auth, email, password)
      .then((res) => {
        console.log(res);
```

```
        navigate("/");
        setLoading(false);
      })
      .catch((error) => {
        console.log(error.code);
        setError(error.message);
        setLoading(false);
      });
  }
};

const handleRegister = (e) => {
  setError("");
  e.preventDefault();
  setLoading(true);

  if (email === "" || password === "" || username === "") {
    setError("Required field is missing.");
    setLoading(false);
  }
  else if (!validateEmail(email)) {
    setError("Email is malformed");
    setLoading(false);
  }
```

```
  else {
    createUserWithEmailAndPassword(auth, email, password)
      .then((res) => {
        console.log(res);
        navigate("/");
        setLoading(false);
      })
      .catch((error) => {
        console.log(error);
        setError(error.message);
        setLoading(false);
      });
  }
};
return (
  <div className="auth">
    <div className="auth-container">
      <p>Add another way to log in using any of the following services. </p>
      <div className="sign-options">
        <div onClick={handleGoogleSignIN} className="single-option">
          <Google/>
          <p>{loading ? "Signing in..." : "Login with Google"}</p>
        </div>
      </div>
```

```
<div className="auth-login">
  <div className="auth-login-container">
    {register ? (
      <>
        {" "}
        <div className="input-field">
          <p>Username</p>
          <input
            value={username}
            onChange={(e) => setUsername(e.target.value)}
            type="text"
          />
        </div>
        <div className="input-field">
          <p>Email</p>
          <input
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            type="email"
          />
        </div>
        <div className="input-field">
          <p>Password</p>
          <input
```

```
        value={password}
        onChange={(e) => setPassword(e.target.value)}
        type="password"
      />
    </div>
    <button
      onClick={handleRegister}
      disabled={loading}
      style={{
        marginTop: "10px",
      }}
    >
      {loading ? "Registering..." : "Register"}
    </button>
  </>
) : (
  <>
    <div className="input-field">
      <p>Email</p>
      <input value={email}
        onChange={(e) => setEmail(e.target.value)}
        type="email" />
    </div>
    <div className="input-field">
```

```
      <p>Password</p>
      <input value={password}
        onChange={(e) => setPassword(e.target.value)}
        type="password" />
    </div>
    <button
      onClick={handleSignIn}
      disabled={loading}
      style={{
        marginTop: "10px",
      }}
    >
      {loading ? "Logging in..." : "Login"}
    </button>
  </>
)}

<p
  onClick={() => setRegister(!register)}
  style={{
    marginTop: "10px",
    textAlign: "center",
    color: "#aa1313",
    textDecoration: "underline",
```

```
              cursor: "pointer",
            }}
          >
            {register ? "Login" : "Register"} ?
          </p>
        </div>
      </div>
      {error !== "" && (
        <p
          style={{
            color: "red",
            fontSize: "14px",
          }}
        >
          {error}
        </p>
      )}
    </div>
  </div>
 );
}

export default Index;
```

Main Page

```
import React from 'react'
import { Link } from 'react-router-dom'
import QuestionWall from './QuestionWall'
import FilterList from '@mui/icons-material/FilterList';
import './css/Main.css'

function Main({questions}) {
  return (
    <div className="main">
    <div className="main-container">
      <div className="main-top">
        <h2>Question Wall</h2>
        <Link to ="/add-question">
          <button>Ask Question</button>
        </Link>
      </div>
      <div className="main-desc">
        <p>{questions.length} Question</p>
        <div className="main-filter">
          <div className="main-tabs">
            <div className="main-tab">
              <Link>Newest</Link>
            </div>
```

```
      <div className="main-tab">
        <Link>Active</Link>
      </div>

      <div className="main-tab">
        <Link>More</Link>
      </div>
    </div>
    <div className="main-filter-item">
      <FilterList />
      <p>Filter</p>
    </div>
  </div>
</div>
<div className="questions">
    {questions?.map((_q,index) => (
      <>
      <div key={index} className="question">
        <QuestionWall question={_q} />
      </div>
      </>
    ))}
```

```
      </div>

      </div>
    </div>

  )
}

export default Main
```

Question Wall

```
import React from 'react'
import { Avatar } from "@mui/material";
import { Link } from 'react-router-dom';
import './css/QuestionWall.css'
import ReactHtmlParser from 'react-html-parser';

function QuestionWall({ question }) {
  function truncate(str, n) {
    return str?.length > n ? str.substr(0, n - 1) + "..." : str;
  }
  let tags = [0];
```

```
try {
  if (question.tags[0]) {
    tags = JSON.parse(question.tags[0]);
  }
} catch (error) {
  console.error("Error parsing tags:", error);
}
return (
  <div className="question-wall">
    <div className="question-wall-container">
      <div className="question-wall-left">
        <div className="all-options">
          <div className="all-option">
            <p>0</p>
            <span>Support</span>
          </div>
          <div className="all-option">
            <p>{question?.answerDetails?.length}</p>
            <span>Answers</span>
          </div>
          <div className="all-option">
            <small>0 Views</small>
          </div>
        </div>
```

```
</div>
<div className="question-answer">
  <Link to={`/question?q=${question?._id}`}>{question?.title}</Link>

  <div
    style={{
      maxWidth: "90%",
    }}
  >
    <div className='answer-color'>
    {ReactHtmlParser(truncate(question.body, 200))}
      </div>
    </div>



    <div
      style={{
        display: "flex",
      }}
    >
      {tags.map((_tag) => (
    <>
     <span className='question-tags'>{_tag}</span>
```

```
          </>
          ))}
          </div>

        <div className="author">
          <small>{new Date(question?.created_at).toLocaleString()}</small>
          <div className="author-details">
            <Avatar src = {question?.user?.photo}/>
            <p>
              {question?.user?.displayName ? question?.user?.displayName : String(question?.user?.email).split('@')[0]}
            </p>
          </div>
        </div>
      </div>
    </div>
  </div>
 )
}

export default QuestionWall
```

Header

```
import React from 'react'
 import './css/Header.css'
 import backimg from "../../assets/logo.png"
 import SearchIcon from "@mui/icons-material/Search";

 import { Link} from 'react-router-dom'
 import AllInboxIcon from '@mui/icons-material/AllInbox';
 import { Avatar } from "@mui/material";
import { useSelector } from 'react-redux';
import { selectUser } from '../../features/userSlice';
import { auth } from '../../firebase';


 function Header() {
  const user = useSelector(selectUser);
  // console.log(user);
  function stringToColor(string) {
    let hash = 0;
    let i;

    /* eslint-disable no-bitwise */
    for (i = 0; i < string.length; i += 1) {
      hash = string.charCodeAt(i) + ((hash << 5) - hash);
    }
```

```
  let color = "#";

  for (i = 0; i < 3; i += 1) {
    const value = (hash >> (i * 8)) & 0xff;
    color += `00${value.toString(16)}`.substr(-2);
  }
  /* eslint-enable no-bitwise */

  return color;
}

function stringAvatar(name) {
  return {
    sx: {
      bgcolor: name ? stringToColor(name) : "rgba(255,255,255,0.8)",
    },
    children: name && `${name.split(" ")[0][0]}${name.split(" ")[1][0]}`,
  };
}

 return (
   <header>
    <div className="header-container">
```

```
<div className="header-left">
    <Link to ="/"><img
      src={backimg}
      alt="logo" className='immg'
    /></Link>


  <Link to="/about"><h3 className='hel'>About</h3></Link>


</div>
<div className="header-middle">
  <div className="header-search-container">
    <SearchIcon />
    <input type="text" placeholder="Search..." />
  </div>
</div>
<div className="header-right">
  <div className="header-right-container">
   <Avatar src='.user?.photo' style={{
       cursor: "pointer",
      }}
      {...stringAvatar(user && user.displayName)}
      onClick={() =>
```

```
            auth.signOut()} />
          <AllInboxIcon />
        </div>
      </div>
    </div>
   </header>
  )
}

export default Header
```

UserSlice

```
import { createSlice } from "@reduxjs/toolkit";

export const userSlice = createSlice({
  name: "user",
  initialState: {
    value: null,
  },
  reducers: {
    login: (state, action) => {
      state.user = action.payload;
    },
```

```
    logout: (state) => {
      state.user = null;
    },
  },
});

export const { login, logout } = userSlice.actions;

export const selectUser = (state) => state.user.user;

export default userSlice.reducer;
```

App.js

```
import React, { useEffect } from 'react';
import './App.css';
import Header from './components/Header/Header';
import {
  BrowserRouter as Router,Routes, Route, Navigate,Outlet
} from 'react-router-dom'
import Question from './components/Add-Question';
import ViewQuestion from "./components/ViewQuestion"
import BufferOverflow from './components/BufferOverFlow';
import About from './components/About';
import Auth from './components/Auth'
```

```
import { login, logout, selectUser } from './features/userSlice';
import { useDispatch, useSelector } from 'react-redux';
import { auth } from './firebase';

function App() {
  const user = useSelector(selectUser);
  const dispatch = useDispatch();

  useEffect(() => {
    auth.onAuthStateChanged((authUser) => {
      if (authUser) {
        dispatch(
          login({
            uid: authUser.uid,
            photo: authUser.photoURL,
            displayName: authUser.displayName,
            email: authUser.email,
          })
        );
      } else {
        dispatch(logout());
      }
      // console.log(authUser);
    });
```

```
}, [dispatch]);


const PrivateRoute = ({ component: Component, ...rest }) => {
  //console.log(user)
  return user ? <Outlet/>:<Navigate to="/auth"/>
  // <Route
  //   {...rest}
  //   render={(props) =>
  //     user ? (
  //       <Component {...props} />
  //     ) : (
  //       <Navigate
  //         replace to={{
  //           pathname: "/auth",
  //           state: {
  //             from: props.location,
  //           },
  //         }}
  //       />
  //     )
  //   }
  // />
};
```

```
 return (
   <div className="App">
     <Router>
     <Header />
       <Routes>
      <Route exact path={user ? "/" : "/auth"} element={user ? <BufferOverflow/> : <Auth/>}/>

       <Route exact path="/" element={<PrivateRoute/>}>
         <Route exact path="/" element={<BufferOverflow/>}/>
       </Route>
       <Route exact path="/add-question" element={<PrivateRoute/>}>
       <Route exact path="/add-question" element={<Question/>}/>
       </Route>

       <Route exact path="/question" element={<PrivateRoute/>}>
       <Route exact path="/question" element={<ViewQuestion/>}/>
       </Route>

       <Route exact path="/about" Component={About}></Route>
       </Routes>
     </Router>
```

```
    </div>
  );
}

export default App;
```

Firebase

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import {getAuth,GoogleAuthProvider} from 'firebase/auth';
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "",
  authDomain: "bufferoverflow-e27df.firebaseapp.com",
  projectId: "bufferoverflow-e27df",
  storageBucket: "bufferoverflow-e27df.appspot.com",
  messagingSenderId: "231207074489",
  appId: "",
  measurementId: ""
```

```
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
export const auth = getAuth();
export const provider = new GoogleAuthProvider();
```

Backend

Server

```
if (process.env.NODE_ENV !== "production") {
  require("dotenv").config();
}

const express = require("express");
const cors = require("cors");
const path = require("path");
const app = express();
const router = require("./routers");
const bodyParser = require("body-parser");
const PORT = process.env.PORT || 80;


//db connection
const db = require("./db");
```

```
db.connect();

//middlwware
app.use(bodyParser.json({ limit: "500mb" }));
app.use(bodyParser.urlencoded({ extended: true, limit: "500mb" }));

app.use(express.json());

//headers
app.use((req, res, next) => {
  res.header("Access-Control-Allow-Origin", "*");
  res.header("Access-Control-Allow-Headers", "*");
  next();
});

//api
app.use("/api", router);

//static resources for running frontend and backend at same time
app.use("/uploads", express.static(path.join(__dirname, "/../uploads")));
app.use(express.static(path.join(__dirname, "/../frontend/build")));


app.get("*", (req, res) => {
```

```
  try {
    res.sendFile(path.join(`${__dirname}/../frontend/build/index.html`));
  } catch (e) {
    res.send("Welcome to Buffer Overflow");
  }
});



//cors
app.use(cors());



//server listen
app.listen(PORT, () => {
  console.log(`Buffer Overflow API is running on PORT No- ${PORT}`);
});
```

Router Answer

```
const express = require("express");
const mongoose = require("mongoose");
const router = express.Router();
// const mongoose = require('mongoose')
const QuestionDB = require("../models/Question");
```

```
router.post("/", async (req, res) => {
  const questionData = new QuestionDB({
    title: req.body.title,
    body: req.body.body,
    tags: req.body.tag,
    user: req.body.user,
  });

  await questionData
    .save()
    .then((doc) => {
      res.status(201).send(doc);
    })
    .catch((err) => {
      res.status(400).send({
        message: "Question not added successfully",
      });
    });
});

// router.get("/", async (req, res) => {
//   const questions = await QuestionDB.find({});

//   try {
```

```
//     if (questions) {
//       res.status(200).send({ questions });
//     } else {
//       res.status(400).send({
//         message: "question not found",
//       });
//     }
//   } catch (e) {
//     res.status(400).send({
//       message: "Error in getting question",
//     });
//   }
// });

router.get("/:id", async (req, res) => {
  try {
    // const question = await QuestionDB.findOne({ _id: req.params.id });
    // res.status(200).send(question);
    QuestionDB.aggregate([
      {
        $match: { _id: mongoose.Types.ObjectId(req.params.id) },
      },
      {
        $lookup: {
```

```
    from: "answers",
    let: { question_id: "$_id" },
    pipeline: [
      {
        $match: {
          $expr: {
            $eq: ["$question_id", "$$question_id"],
          },
        },
      },
      {
        $project: {
          _id: 1,
          user: 1,
          answer: 1,
          // created_at: 1,
          question_id: 1,
          created_at: 1,
        },
      },
    ],
    as: "answerDetails",
  },
},
```

```
{
  $lookup: {
    from: "comments",
    let: { question_id: "$_id" },
    pipeline: [
      {
        $match: {
          $expr: {
            $eq: ["$question_id", "$$question_id"],
          },
        },
      },
      {
        $project: {
          _id: 1,
          question_id: 1,
          user: 1,
          comment: 1,
          // created_at: 1,
          // question_id: 1,
          created_at: 1,
        },
      },
    ],
```

```
      as: "comments",
    },
  },
  // {
  //   $unwind: {
  //     path: "$answerDetails",
  //     preserveNullAndEmptyArrays: true,
  //   },
  // },
  {
    $project: {
      __v: 0,
      // _id: "$_id",
      // answerDetails: { $first: "$answerDetails" },
    },
  },
])
  .exec()
  .then((questionDetails) => {
    res.status(200).send(questionDetails);
  })
  .catch((e) => {
    console.log("Error: ", e);
    res.status(400).send(e);
```

```
    });
  } catch (err) {
    res.status(400).send({
      message: "Question not found",
    });
  }
});

router.get("/", async (req, res) => {
  const error = {
    message: "Error in retrieving questions",
    error: "Bad request",
  };

  QuestionDB.aggregate([
    {
      $lookup: {
        from: "comments",
        let: { question_id: "$_id" },
        pipeline: [
          {
            $match: {
              $expr: {
                $eq: ["$question_id", "$$question_id"],
```

```
        },
      },
    },
    {
      $project: {
        _id: 1,
        // user_id: 1,
        comment: 1,
        created_at: 1,
        // question_id: 1,
      },
    },
  ],
  as: "comments",
 },
},
{
 $lookup: {
  from: "answers",
  let: { question_id: "$_id" },
  pipeline: [
   {
    $match: {
     $expr: {
```

```
            $eq: ["$question_id", "$$question_id"],
          },
        },
      },
      {
        $project: {
          _id: 1,
          // user_id: 1,
          // answer: 1,
          // created_at: 1,
          // question_id: 1,
          // created_at: 1,
        },
      },
    ],
    as: "answerDetails",
  },
},
// {
//   $unwind: {
//     path: "$answerDetails",
//     preserveNullAndEmptyArrays: true,
//   },
// },
```

```
    {
      $project: {
        __v: 0,
        // _id: "$_id",
        // answerDetails: { $first: "$answerDetails" },
      },
    },
  ])
    .exec()
    .then((questionDetails) => {
      res.status(200).send(questionDetails);
    })
    .catch((e) => {
      console.log("Error: ", e);
      res.status(400).send(e);
    });
});


module.exports = router;
```

Router **Comments**

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

const router = express.Router();
```

```
const commentDB = require("../models/Comments");


router.post("/:id", async (req, res) => {
  try {
    await commentDB
      .create({
        question_id: req.params.id,
        comment: req.body.comment,
        user: req.body.user,
      })
      .then((doc) => {
        res.status(201).send({
          message: "Comment added successfully",
        });
      })
      .catch((err) => {
        res.status(400).send({
          message: "Bad format",
        });
      });
  } catch (err) {
    res.status(500).send({
      message: "Error while adding comments",
    });
```

```
  }
});

module.exports = router;
```

Router Question

```
const express = require("express");
const mongoose = require("mongoose");
const router = express.Router();
// const mongoose = require('mongoose')
const QuestionDB = require("../models/Question");

router.post("/", async (req, res) => {
  const questionData = new QuestionDB({
    title: req.body.title,
    body: req.body.body,
    tags: req.body.tag,
    user: req.body.user,
  });

  await questionData
    .save()
    .then((doc) => {
      res.status(201).send(doc);
```

```
    })
    .catch((err) => {
      res.status(400).send({
        message: "Question not added successfully",
      });
    });
});

// router.get("/", async (req, res) => {
//   const questions = await QuestionDB.find({});

//   try {
//     if (questions) {
//       res.status(200).send({ questions });
//     } else {
//       res.status(400).send({
//         message: "question not found",
//       });
//     }
//   } catch (e) {
//     res.status(400).send({
//       message: "Error in getting question",
//     });
//   }
```

```
// });

router.get("/:id", async (req, res) => {
  try {
    // const question = await QuestionDB.findOne({ _id: req.params.id });
    // res.status(200).send(question);
    QuestionDB.aggregate([
      {
        $match: { _id: mongoose.Types.ObjectId(req.params.id) },
      },
      {
        $lookup: {
          from: "answers",
          let: { question_id: "$_id" },
          pipeline: [
            {
              $match: {
                $expr: {
                  $eq: ["$question_id", "$$question_id"],
                },
              },
            },
            {
              $project: {
```

```
        _id: 1,
        user: 1,
        answer: 1,
        // created_at: 1,
        question_id: 1,
        created_at: 1,
      },
    },
  ],
  as: "answerDetails",
 },
},
{
 $lookup: {
  from: "comments",
  let: { question_id: "$_id" },
  pipeline: [
   {
    $match: {
     $expr: {
      $eq: ["$question_id", "$$question_id"],
     },
    },
   },
```

```
      {
        $project: {
          _id: 1,
          question_id: 1,
          user: 1,
          comment: 1,
          // created_at: 1,
          // question_id: 1,
          created_at: 1,
        },
      },
    ],
    as: "comments",
  },
},
// {
//   $unwind: {
//     path: "$answerDetails",
//     preserveNullAndEmptyArrays: true,
//   },
// },
{
  $project: {
    __v: 0,
```

```
        // _id: "$_id",
        // answerDetails: { $first: "$answerDetails" },
      },
    },
  ])
    .exec()
    .then((questionDetails) => {
      res.status(200).send(questionDetails);
    })
    .catch((e) => {
      console.log("Error: ", e);
      res.status(400).send(e);
    });
 } catch (err) {
  res.status(400).send({
    message: "Question not found",
  });
 }
});

router.get("/", async (req, res) => {
 const error = {
  message: "Error in retrieving questions",
  error: "Bad request",
```

```
};

QuestionDB.aggregate([
  {
    $lookup: {
      from: "comments",
      let: { question_id: "$_id" },
      pipeline: [
        {
          $match: {
            $expr: {
              $eq: ["$question_id", "$$question_id"],
            },
          },
        },
        {
          $project: {
            _id: 1,
            // user_id: 1,
            comment: 1,
            created_at: 1,
            // question_id: 1,
          },
        },
```

```
    ],
    as: "comments",
  },
},
{
  $lookup: {
    from: "answers",
    let: { question_id: "$_id" },
    pipeline: [
      {
        $match: {
          $expr: {
            $eq: ["$question_id", "$$question_id"],
          },
        },
      },
      {
        $project: {
          _id: 1,
          // user_id: 1,
          // answer: 1,
          // created_at: 1,
          // question_id: 1,
          // created_at: 1,
```

```
          },
        },
      ],
      as: "answerDetails",
    },
  },
  // {
  //   $unwind: {
  //     path: "$answerDetails",
  //     preserveNullAndEmptyArrays: true,
  //   },
  // },
  {
    $project: {
      __v: 0,
      // _id: "$_id",
      // answerDetails: { $first: "$answerDetails" },
    },
  },
])
  .exec()
  .then((questionDetails) => {
    res.status(200).send(questionDetails);
  })
```

```
    .catch((e) => {
      console.log("Error: ", e);
      res.status(400).send(e);
    });
});

module.exports = router;
```

CHAPTER 8

Conclusion and Future Scope

Conclusion

Buffer Overflow marks a significant milestone as a dynamic platform fostering collaboration and knowledge exchange within the coding community. Rigorous testing ensures reliability, security, and scalability, with key achievements including:

Functionality Verification: Thorough testing validates core features, ensuring robust functionality in user authentication, collaborative coding, and reputation systems.

User Engagement: Integrated features such as commenting, feedback, and collaborative coding tools create an engaging space for the coding community.

Security and Scalability: Emphasis on security measures and scalability testing ensures user data protection and accommodates a growing user base.

Future Scope

Buffer Overflow's journey continues with future enhancements to enrich the coding community experience:
Enhanced Collaboration: Introduce real-time collaborative coding features for live coding sessions and chat functionality.
AI-Powered Assistance: Implement AI-driven tools for problem-solving, code optimization, and resource suggestions.
Gamification Elements: Expand the reputation system with gamification, including coding challenges and competitions.
Extended Learning Resources: Integrate a curated library of coding tutorials and articles to complement interactive features.

Bibliography

1. https://legacy.reactjs.org/docs/getting-started.html
2. https://expressjs.com/en/4x/api.html
3. https://firebase.google.com/docs/guides
4. https://www.youtube.com/playlist?list=PLu71SKxNbfoDqgPchmvIsL4hTnJIrtige
5. https://www.youtube.com/playlist?list=PLu71SKxNbfoBuX3f4EOACle2y-tRC5Q37
6. https://www.mongodb.com/docs/