VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi-590018

A Database Management System Mini Project Report on

“NEWS ARTICLE CRUD WEBSITE”

Submitted in Partial fulfillment of the Requirements for the V Semester of the Degree of

Bachelor of Engineering in

Information Science & Engineering

By

PUSHKAR JHA (1CR21IS122)

Under the Guidance of,

Prof Upasana Mahajan, Assistant Professor, Dept. of ISE

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CMR INSTITUTE OF TECHNOLOGY

Affiliated to VTU, Approved by AICTE, Accredited by NBA and NAAC with “A++” Grade

ITPL MAIN ROAD, BROOKFIELD, BENGALURU-560037, KARNATAKA, INDIA

2023-24

CMR INSTITUTE OF TECHNOLOGY

Affiliated to VTU, Approved by AICTE, Accredited by NBA and NAAC with "A++" Grade

ITPL MAIN ROAD, BROOKFIELD, BENGALURU-560037, KARNATAKA, INDIA

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Database Management System Project work entitled **"News Article CRUD Website"** has been carried out by **Pushkar Jha, 1CR21IS122** bonafide students of CMR Institute of Technology, Bengaluru in partial fulfillment for the award of the Degree of **Bachelor of Engineering in Information Science and Engineering** of the Visvesvaraya Technological University, Belagavi during the year **2023-2024**. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the report deposited in the departmental library. This Database Management System Project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said Degree.

Signature of Guide		**Signature of HOD**
Prof Upasana Mahajan		**Dr Jagadishwari V**
Assistant Professor		**Professor & HoD**
Dept. of ISE, CMRIT	**External Viva**	**Dept. of ISE, CMRIT**

Name of the Examiners	Signature with date
1.	

.

DECLARATION

We, the students of V semester from Department of Information Science and Engineering, CMR Institute of Technology, Bangalore declare that the project work entitled " **News Article CRUD Website** " has been successfully completed under the guidance of Prof Upasana Mahajan, Assistant Professor, Dept. of Information Science and Engineering, CMR Institute of technology, Bengaluru. This project work is submitted in partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in Information Science and Engineering during the academic year 2023-2024. The matter embodied in the project report has not been submitted previously by anybody for the award of any degree or diploma to any university.

Place: Bangalore

Date: 11/03/2024

Team members:

PUSHKAR JHA (1CR21IS122)	
PRAVEEN KUMAR(1CR21IS117)	
PRIYANSHU KUMAR(1CR21IS121)	

ABSTRACT

This project delves into the fusion of modern web technologies with Database Management Systems (DBMS) principles, aiming to develop a full-stack news article CRUD website. Employing React for the frontend, MySQL for the database, and JavaScript for the backend, the project showcases the practical application of DBMS concepts in web development. Through seamless integration, users can navigate, contribute, and manage news articles, underscoring the importance of DBMS in contemporary web applications.

Central to the project is the role of DBMS in structuring and managing data effectively. Serving as the backbone of the application, the database facilitates storage, retrieval, and manipulation of news articles with precision. By implementing CRUD functionalities, users gain dynamic interaction with articles, contributing to a user-centric platform for information dissemination.

Through this endeavor, the project not only demonstrates technical proficiency in web development but also highlights the transformative potential of DBMS in shaping the functionality and performance of web applications. Bridging theory with practice, the project lays the groundwork for future innovation in this dynamic field.

ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratitude and respect to **CMR Institute of Technology, Bengaluru** for providing me a platform to pursue my studies and carry out the Database Management System Project.

It gives me an immense pleasure to express my deep sense of gratitude to **Dr. Sanjay Jain,** Principal, CMRIT, Bengaluru, for his constant encouragement.

I would like to extend my sincere gratitude to **Dr. Jagadishwari V,** HOD, Department of Information Science and Engineering, CMRIT, Bengaluru, who has been a constant support and encouragement throughout the course of this project.

I would like to thank my guide **Prof Upasana Mahajan, Assistant Professor,** Department of Information Science and Engineering, for the valuable guidance throughout the tenure of the project work.

I would also like to thank all the faculty members of Department of Computer Science and Engineering who directly or indirectly encouraged me.

Finally, I thank my parents and friends for all the moral support they have given me during the completion of this work.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Our project introduces a full-stack news article CRUD website, aiming to provide users with a dynamic platform for discovering, sharing, and managing news articles. Leveraging contemporary web technologies such as React for the frontend, MySQL for the database, and JavaScript for the backend, the application is designed to offer a seamless and intuitive user experience. With an emphasis on functionality and usability, the website empowers users to explore a diverse array of articles, contribute their own content, and engage with the community in a collaborative environment.

OBJECTIVES:

1. **Authentication:** Implement robust authentication mechanisms to ensure secure access and user authentication within the platform, enhancing overall user experience and data security.
2. **CRUD Operations:** Facilitate seamless Create, Read, Update, and Delete (CRUD) functionalities for news articles, empowering users to manage and contribute content effortlessly.
3. **User-Friendly UI:** Develop an intuitive and aesthetically pleasing user interface using React, enhancing user engagement and satisfaction while navigating the platform.
4. **Utilization of Relational DB:** Employ MySQL as the relational database management system to efficiently store and manage article data, ensuring scalability and data integrity.
5. **Practical Application of DBMS:** Demonstrate the practical implementation of DBMS principles by integrating MySQL into the backend architecture, showcasing its role in structuring and managing data effectively.

SCOPE OF THE PROJECT:

- **Frontend Using React:** Develop the frontend components of NewsHub using React, enabling the creation of dynamic and responsive user interfaces to enhance the overall user experience.
- **Integration of MySQL for DB Purpose:** Integrate MySQL as the primary database management system for NewsHub, facilitating efficient storage, retrieval, and manipulation of article data.
- **Implementation of Filtering Logic:** Implement advanced filtering logic to enable users to search and filter news articles based on various criteria such as category, date, and author, enhancing user discoverability and content accessibility.
- **Backend Using JavaScript:** Implement the backend logic of NewsHub using JavaScript, allowing seamless communication between the frontend and database while handling user requests and interactions efficiently.

CHAPTER 2

SOFTWARE REQUIREMENTS AND TECH STACK

Software Requirements

- **Code Editor:** A code editor such as Visual Studio Code, Sublime Text, or Atom is essential for writing and editing the project's source code efficiently.
- **Web Browser:** Web browsers like Chrome, Firefox, or Safari are required for testing and previewing the frontend user interface during development.
- **MySQL Database and Workbench:** MySQL Database and Workbench are necessary for managing the relational database used to store news article data and for performing database operations.
- **Terminal:** A terminal or command-line interface is needed for running scripts, executing commands, and managing project files effectively.
- **Operating System:** The project can be developed and deployed on various operating systems such as Windows, macOS, or Linux, depending on the developer's preference and requirements.

Technology Stack

- **React (Frontend):** React.js is employed for building the frontend components of the NewsHub platform, offering a fast, interactive, and component-based user interface.
- **Express and Node.js (Backend):** Express.js, along with Node.js, serves as the backend framework for developing server-side logic and handling HTTP requests and responses efficiently.
- **JWT and Bcrypt (Authentication):** JSON Web Token (JWT) and Bcrypt libraries are utilized for implementing secure authentication and user session management within the application.
- **Multer (File Upload):** Multer is used for handling file uploads within the NewsHub platform, enabling users to upload images or other media files along with their news articles.
- **MySQL (Database):** MySQL is employed as the primary relational database management system (RDBMS) for storing and managing news article data, ensuring scalability, reliability, and data integrity.

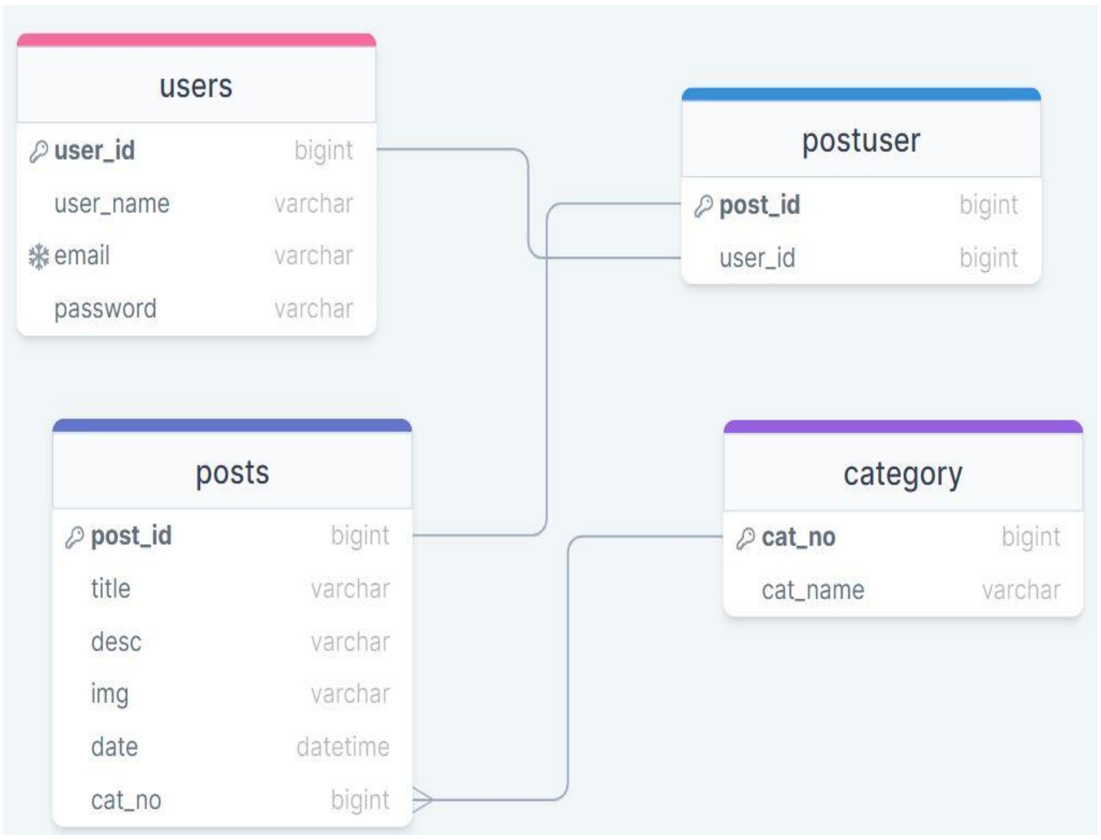
CHAPTER 3

DESIGN

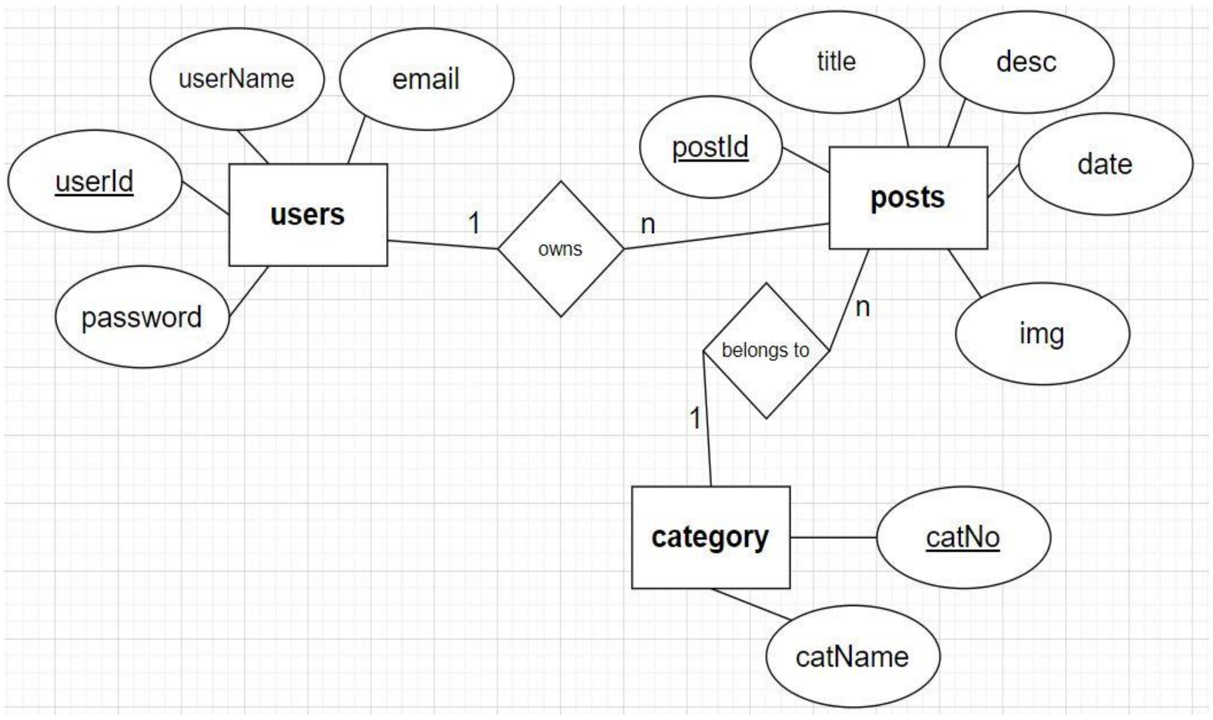
A schema diagram visually represents the structure of a database. It displays tables, their columns, and the relationships between them. It helps in understanding the organization and layout of the database. Schema diagrams are essential for database design and documentation.

An ER diagram depicts entities and their relationships within a database. Entities are represented as rectangles and relationships as lines connecting them. It illustrates the logical structure of a database. ER diagrams aid in conceptualizing and designing database

3.1 Schema Diagram



3.2 E-R Diagram



CHAPTER 4

IMPLEMENTATION

1. **User Authentication :**
    a. Description : Implementing user authentication functionality ensures secure access to the NewsHub platform. Users will be required to sign up and log in to access certain features such as creating, editing, or deleting articles.
    b. Implementation Details :
        i. Utilize JWT (JSON Web Tokens) for secure authentication.
        ii. Implement bcrypt for hashing and encrypting user passwords to enhance security.
        iii. Develop signup and login forms for user registration and authentication.
        iv. Create middleware functions to validate user credentials and authorize access to protected routes.

```
import { db } from "../db.js";
import bcrypt from "bcryptjs";
import jwt from "jsonwebtoken";

export const register = (req, res) => {
  // Input validation
  if (!req.body.email || !req.body.username || !req.body.password) {
    return res.status(400).json("All fields are required.");
  }

  // Check existing user
  const q = "SELECT * FROM users WHERE email = ? OR username = ?";
  db.query(q, [req.body.email, req.body.username], (err, data) => {
    if (err) {
      console.error("Error checking existing user:", err);
      return res.status(500).json("Internal server error.");
    }
    if (data.length) {
      return res.status(409).json("User already exists!");
    }

    // Hash the password and create a user
    const salt = bcrypt.genSaltSync(10);
    const hash = bcrypt.hashSync(req.body.password, salt);

    const insertQuery = "INSERT INTO users(`username`,`email`,`password`)
VALUES (?)";
```

```
      const values = [req.body.username, req.body.email, hash];

      db.query(insertQuery, [values], (err, data) => {
        if (err) {
          console.error("Error creating user:", err);
          return res.status(500).json("Internal server error.");
        }
        return res.status(200).json("User has been created.");
      });
    });
  };

  export const login = (req, res) => {
    // Input validation
    if (!req.body.username || !req.body.password) {
      return res.status(400).json("Username and password are required.");
    }

    // Check user existence
    const selectQuery = "SELECT * FROM users WHERE username = ?";
    db.query(selectQuery, [req.body.username], (err, data) => {
      if (err) {
        console.error("Error checking user:", err);
        return res.status(500).json("Internal server error.");
      }
      if (data.length === 0) {
        return res.status(404).json("User not found!");
      }

      // Check password
      const isPasswordCorrect = bcrypt.compareSync(req.body.password,
  data[0].password);
      if (!isPasswordCorrect) {
        return res.status(400).json("Wrong username or password!");
      }

      // Generate JWT token
      const token = jwt.sign({ id: data[0].id }, process.env.JWT_SECRET, { expiresIn:
  '1h' });

      // Omit sensitive data from response
      const { password, ...other } = data[0];

      // Set JWT token as cookie
      res.cookie("access_token", token, {
        httpOnly: true,
        // Secure cookie only in production
        secure: true,
        // Limit cookie scope to the same site
        sameSite: 'strict'
      }).status(200).json(other);
    });
  };

  export const logout = (req, res) => {
    // Clear JWT token cookie
    res.clearCookie("access_token").status(200).json("User has been logged out.");
  };
```

2. **Article CRUD operation :**
    a. Description: Implementing CRUD (Create, Read, Update, Delete) operations allows users to manage news articles effectively. Users can create, view, edit, and delete articles based on their permissions.
    b. Implementation Details :
        i. Develop API endpoints for handling CRUD operations on articles.
        ii. Implement validation checks to ensure data integrity and consistency.
        iii. Provide appropriate error handling and feedback messages for user interactions.
        iv. Utilize middleware functions for authorization to restrict access based on user roles.

```
import { db } from "../db.js";
import jwt from "jsonwebtoken";

export const getPosts = (req, res) => {
  const { cat } = req.query;
  const query = cat ? "SELECT * FROM posts WHERE cat=?" : "SELECT * FROM posts";

  db.query(query, [cat], (err, data) => {
    if (err) {
      console.error("Error retrieving posts:", err);
      return res.status(500).json("Internal server error.");
    }

    return res.status(200).json(data);
  });
};

export const getPost = (req, res) => {
  const query =
    "SELECT p.id, `username`, `title`, `desc`, p.img, u.img AS userImg, `cat`,`date` FROM users u JOIN posts p ON u.id = p.uid WHERE p.id = ? ";
  db.query(query, [req.params.id], (err, data) => {
    if (err) {
      console.error("Error retrieving post:", err);
      return res.status(500).json("Internal server error.");
    }

    if (data.length === 0) {
      return res.status(404).json("Post not found!");
    }

    return res.status(200).json(data[0]);
  });
};

export const addPost = (req, res) => {
  const token = req.cookies.access_token;
  if (!token) {
    return res.status(401).json("Not authenticated!");
  }

  jwt.verify(token, process.env.JWT_SECRET, (err, userInfo) => {
    if (err) {
      return res.status(403).json("Token is not valid!");
    }

    const query =
      "INSERT INTO posts(`title`, `desc`, `img`, `cat`, `date`,`uid`) VALUES (?)";

    const values = [
      req.body.title,
      req.body.desc,
      req.body.img,
      req.body.cat,
      req.body.date,
      userInfo.id,
    ];

    db.query(query, [values], (err, data) => {
      if (err) {
        console.error("Error adding post:", err);
        return res.status(500).json("Internal server error.");
      }

      return res.json("Post has been created.");
    });
  });
};

export const deletePost = (req, res) => {
  const token = req.cookies.access_token;
  if (!token) {
    return res.status(401).json("Not authenticated!");
  }

  jwt.verify(token, process.env.JWT_SECRET, (err, userInfo) => {
    if (err) {
      return res.status(403).json("Token is not valid!");
    }

    const postId = req.params.id;
    const query = "DELETE FROM posts WHERE `id` = ? AND `uid` = ?";

    db.query(query, [postId, userInfo.id], (err, data) => {
      if (err) {
        console.error("Error deleting post:", err);
        return res.status(500).json("Internal server error.");
      }

      if (data.affectedRows === 0) {
        return res.status(403).json("You can delete only your post!");
      }

      return res.json("Post has been deleted!");
    });
  });
};

export const updatePost = (req, res) => {
  const token = req.cookies.access_token;
  if (!token) {
    return res.status(401).json("Not authenticated!");
  }

  jwt.verify(token, process.env.JWT_SECRET, (err, userInfo) => {
    if (err) {
      return res.status(403).json("Token is not valid!");
    }

    const postId = req.params.id;
    const query =
      "UPDATE posts SET `title`=?,`desc`=?,`img`=?,`cat`=? WHERE `id` = ? AND
`uid` = ?";

    const values = [req.body.title, req.body.desc, req.body.img, req.body.cat];

    db.query(query, [...values, postId, userInfo.id], (err, data) => {
      if (err) {
        console.error("Error updating post:", err);
        return res.status(500).json("Internal server error.");
      }

      if (data.affectedRows === 0) {
        return res.status(403).json("You can update only your post!");
      }

      return res.json("Post has been updated.");
    });
  });
};
```

3. **User Interface Design :**
   a. Description : Designing an intuitive and visually appealing user interface enhances user experience and engagement.
   b. Implemantation Details :
      i. Utilize React components and libraries for building responsive and interactive UI elements.
      ii. Implement consistent layout and styling throughout the application for a cohesive user experience.

```
import React, { useEffect, useState } from "react";
import { Link, useLocation } from "react-router-dom";
import axios from "axios";

const Home = () => {
  const [posts, setPosts] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  const cat = useLocation().search;

  useEffect(() => {
    const fetchData = async () => {
      try {
        const res = await axios.get(`/posts${cat}`);
        setPosts(res.data);
        setLoading(false);
      } catch (err) {
        console.error("Fetch error:", err);
        setError("Error fetching data. Please try again later.");
        setLoading(false);
      }
    };
    fetchData();
  }, [cat]);

  const getText = (html) => {
    const doc = new DOMParser().parseFromString(html, "text/html");
    return doc.body.textContent;
  };

  if (loading) {
    return <p>Loading...</p>;
  }

  if (error) {
    return <p>{error}</p>;
  }

  return (
    <div className="home">
      <div className="posts">
        {posts.map((post) => (
          <div className="post" key={post.id}>
            <div className="img">
              <img src={`/upload/${post.img}`} alt="" />
            </div>
            <div className="content">
              <Link className="link" to={`/post/${post.id}`}>
                <h1>{post.title}</h1>
              </Link>
              <p>{getText(post.desc)}</p>
              <Link className="link" to={`/post/${post.id}`}>
                <button>Read More</button>
              </Link>
            </div>
          </div>
        ))}
      </div>
    </div>
  );
};

export default Home;
import React, { useState } from "react";
import { useContext } from "react";
import { Link, useNavigate } from "react-router-dom";
import { AuthContext } from "../context/authContext";

const Login = () => {
  const [inputs, setInputs] = useState({
    username: "",
    password: "",
  });
  const [err, setError] = useState(null);
```

```
    const navigate = useNavigate();

    const { login } = useContext(AuthContext);

    const handleChange = (e) => {
      setInputs((prev) => ({ ...prev, [e.target.name]: e.target.value }));
    };

    const handleSubmit = async (e) => {
      e.preventDefault();
      try {
        await login(inputs);
        navigate("/");
      } catch (err) {
        setError(err.response.data);
      }
    };
    return (
      <div className="mainAuthContainer">
        <div className="auth">
          <h1>Login</h1>
          <form>
            <input
              required
              type="text"
              placeholder="username"
              name="username"
              onChange={handleChange}
            />
            <input
              required
              type="password"
              placeholder="password"
              name="password"
              onChange={handleChange}
            />
            <button onClick={handleSubmit}>Login</button>
            {err && <p>{err}</p>}
            <span>
              Don't you have an account? <Link to="/register">Register</Link>
            </span>
          </form>
        </div>
      </div>
    );
  };

  export default Login;
  import React from "react";
```

```
import { useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import axios from "axios";

const Register = () => {
  const [inputs, setInputs] = useState({
    username: "",
    email: "",
    password: "",
  });
  const [err, setError] = useState(null);

  const navigate = useNavigate();

  const handleChange = (e) => {
    setInputs((prev) => ({ ...prev, [e.target.name]: e.target.value }));
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      await axios.post("/auth/register", inputs);
      navigate("/login");
    } catch (err) {
      setError(err.response.data);
    }
  };

  return (
    <div className="mainAuthContainer">
      <div className="auth">
        <h1>Register</h1>
        <form>
          <input
            required
            type="text"
            placeholder="username"
            name="username"
            onChange={handleChange}
          />
          <input
            required
            type="email"
            placeholder="email"
            name="email"
            onChange={handleChange}
          />
          <input
            required
            type="password"
            placeholder="password"
            name="password"
            onChange={handleChange}
          />
          <button onClick={handleSubmit}>Register</button>
          {err && <p>{err}</p>}
          <span>
            Do you have an account? <Link to="/login">Login</Link>
          </span>
        </form>
      </div>
    </div>
  );
};

export default Register;
import React, { useEffect, useState, useContext, useCallback } from "react";
import { Link, useLocation, useNavigate } from "react-router-dom";
import Menu from "../components/Menu";
import axios from "axios";
import moment from "moment";
import { AuthContext } from "../context/authContext";
import DOMPurify from "dompurify";

const Single = () => {
  const [post, setPost] = useState(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  const location = useLocation();
  const navigate = useNavigate();
  const postId = location.pathname.split("/")[2];
  const { currentUser } = useContext(AuthContext);

  const fetchData = useCallback(async () => {
    try {
      const res = await axios.get(`/posts/${postId}`);
      setPost(res.data);
      setLoading(false);
      setError(null);
    } catch (err) {
      console.error("Error fetching post:", err);
      setPost(null);
      setLoading(false);
      setError("Failed to fetch post. Please try again later.");
    }
  }, [postId]);
```

```
useEffect(() => {
  fetchData();
}, [fetchData]);

const handleDelete = async () => {
  try {
    await axios.delete(`/posts/${postId}`);
    navigate("/");
  } catch (err) {
    console.error("Error deleting post:", err);
    setError("Failed to delete post. Please try again later.");
  }
};

return (
  <div className="single">
    {loading ? (
      <p>Loading...</p>
    ) : post ? (
      <>
        <div className="content">
          <div className="imgContainer">
            <img src={`/upload/${post?.img}`} alt="" />
          </div>
          <div className="user">
            {post.userImg && <img src={post.userImg} alt="" />}
            <div className="info">
              {post.username && <span>{post.username}</span>}
              {post.date && <p>Posted {moment(post.date).fromNow()}</p>}
            </div>
            {currentUser && currentUser.username === post.username && (
              <div className="edit">
                <Link to={`/write?edit=${postId}`} state={post}>
                  <button className="editBtn">Edit</button>
                </Link>
                <button onClick={handleDelete} className="deleteBtn">
                  Delete
                </button>
              </div>
            )}
          </div>
          <h1>{post.title}</h1>
          <p
            dangerouslySetInnerHTML={{
              __html: DOMPurify.sanitize(post.desc),
            }}
          ></p>
        </div>
        <Menu cat={post.cat} />
        </>
      ) : (
        <p>{error || "No post found"}</p>
      )}
    </div>
  );
};

export default Single;
import React, { useState } from "react";
import ReactQuill from "react-quill";
import "react-quill/dist/quill.snow.css";
import axios from "axios";
import { useLocation, useNavigate } from "react-router-dom";
import moment from "moment";

const Write = () => {
  const state = useLocation().state;
  const [content, setContent] = useState(state?.desc || "");
  const [title, setTitle] = useState(state?.title || "");
  const [file, setFile] = useState(null);
  const [cat, setCat] = useState(state?.cat || "");
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState(null);

  const navigate = useNavigate();

  const upload = async () => {
    try {
      const formData = new FormData();
      formData.append("file", file);
      const res = await axios.post("/upload", formData);
      return res.data;
    } catch (err) {
      console.error("Error uploading file:", err);
      throw new Error("Failed to upload file. Please try again later.");
    }
  };

  const handleClick = async (e) => {
    e.preventDefault();
    if (!title.trim() || !content.trim() || !cat.trim()) {
      setError("Please fill in all fields.");
      return;
    }

    setLoading(true);
    try {
      const imgUrl = file ? await upload() : "";
      const postData = {
        title,
        desc: content,
        cat,
        img: imgUrl,
        date: moment(Date.now()).format("YYYY-MM-DD HH:mm:ss"),
      };

      if (state) {
        await axios.put(`/posts/${state.id}`, postData);
      } else {
        await axios.post(`/posts/`, postData);
      }

      navigate("/");
    } catch (err) {
      console.error("Error publishing post:", err);
      setError("Failed to publish post. Please try again later.");
    } finally {
      setLoading(false);
    }
  };

  return (
    <div className="add">
      <div className="content">
        <input
          type="text"
          placeholder="Title"
          value={title}
          onChange={(e) => setTitle(e.target.value)}
        />
        <div className="editorContainer">
          <ReactQuill
            className="editor"
            theme="snow"
            value={content}
            onChange={setContent}
          />
        </div>
      </div>
      <div className="menu">
        <div className="item">
          <h1>Publish</h1>
          <span>
            <b>Visibility:</b> Public
          </span>
          <input
```

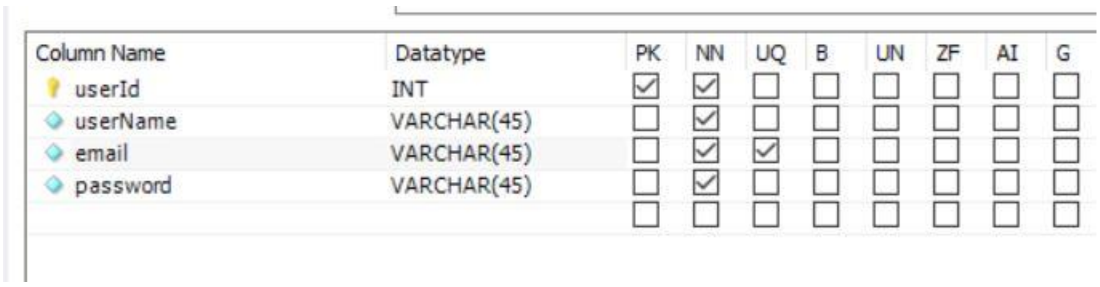
```
              style={{ display: "none" }}
              type="file"
              id="file"
              name=""
              onChange={(e) => setFile(e.target.files[0])}
            />
            <label className="file" htmlFor="file">
              Upload Image
            </label>
            <div className="buttons">
              <button onClick={handleClick} disabled={loading}>
                {loading ? "Publishing..." : "Publish"}
              </button>
            </div>
          </div>
          <div className="item">
            <h1>Category</h1>
            <div className="cat">
              <input
                type="radio"
                checked={cat === "politics"}
                name="cat"
                value="politics"
                id="politics"
                onChange={(e) => setCat(e.target.value)}
              />
              <label htmlFor="politics">Politics</label>
            </div>
            <div className="cat">
              <input
                type="radio"
                checked={cat === "business"}
                name="cat"
                value="business"
                id="business"
                onChange={(e) => setCat(e.target.value)}
              />
              <label htmlFor="business">Business</label>
            </div>
            <div className="cat">
              <input
                type="radio"
                checked={cat === "science"}
                name="cat"
                value="science"
                id="science"
                onChange={(e) => setCat(e.target.value)}
              />
              <label htmlFor="science">Science</label>
            </div>
            <div className="cat">
              <input
                type="radio"
                checked={cat === "sports"}
                name="cat"
                value="sports"
                id="sports"
                onChange={(e) => setCat(e.target.value)}
              />
              <label htmlFor="sports">Sports</label>
            </div>
            <div className="cat">
              <input
                type="radio"
                checked={cat === "tech"}
                name="cat"
                value="tech"
                id="tech"
                onChange={(e) => setCat(e.target.value)}
              />
              <label htmlFor="tech">Tech</label>
            </div>
          </div>
        </div>
        {error && <p className="error">{error}</p>}
      </div>
    );
  };

  export default Write;
```

4. **File Upload Functionality :**
   a. Description **:** Enabling file upload functionality allows users to attach images to their news articles, enhancing content presentation and engagement.
   b. Implementation Details :
      i. Utilize Multer middleware for handling file uploads securely.
      ii. Store uploaded files in a designated directory or cloud storage service for accessibility and scalability.

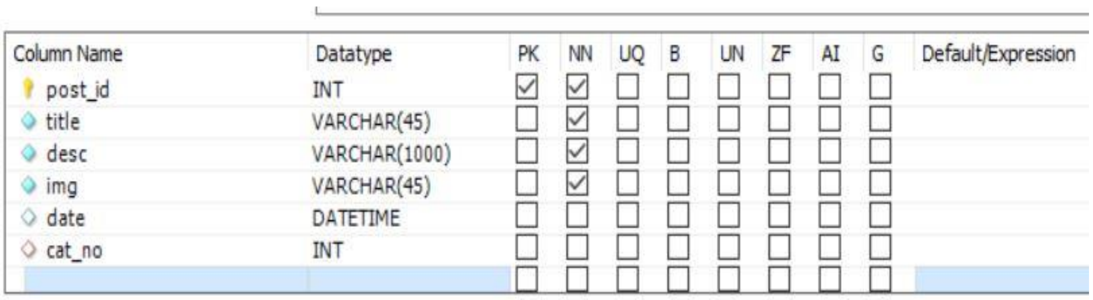
```
import express from "express";
import authRoutes from "./routes/auth.js";

import postRoutes from "./routes/posts.js";
import cookieParser from "cookie-parser";
import multer from "multer";

import {config} from 'dotenv';
```

```
config(
  {
    path:"./config.env"
  }
)

const app = express();


app.use(express.json());
app.use(cookieParser());
const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, "../client/public/upload");
  },
  filename: function (req, file, cb) {
    cb(null, Date.now() + file.originalname);
  },
});

const upload = multer({ storage });

app.post("/api/upload", upload.single("file"), function (req, res) {
  const file = req.file;
  res.status(200).json(file.filename);
});

app.use("/api/auth", authRoutes);
app.use("/api/posts", postRoutes);

app.listen(8800, () => {
  console.log("Connected!");
});
```

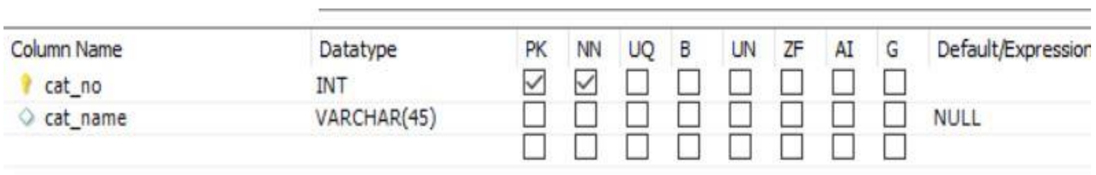
5. **Databse Integration :**
    a. Description: Integrating MySQL database with the backend ensures efficient storage, retrieval, and management of news article data, facilitating seamless user interactions.
    b. Implementation Details:
        i. Set up database schemas and tabels to define data structure and relationships.
        ii. Implement database queries and transactions for CRUD operations on articles.



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
userId	INT	✓	✓						
userName	VARCHAR(45)		✓						
email	VARCHAR(45)		✓	✓					
password	VARCHAR(45)		✓						



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
post_id	INT	✓	✓							
title	VARCHAR(45)		✓							
desc	VARCHAR(1000)		✓							
img	VARCHAR(45)		✓							
date	DATETIME									
cat_no	INT									

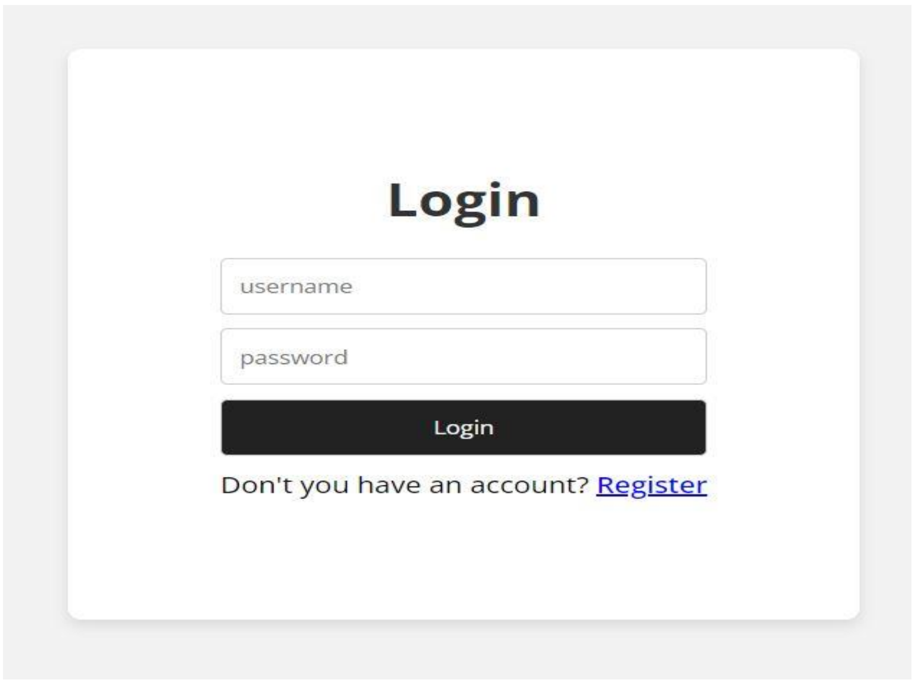


Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
cat_no	INT	✓	✓							
cat_name	VARCHAR(45)									NULL

CHAPTER 5

OUTPUT OF PROJECT

1. Login and Register Page



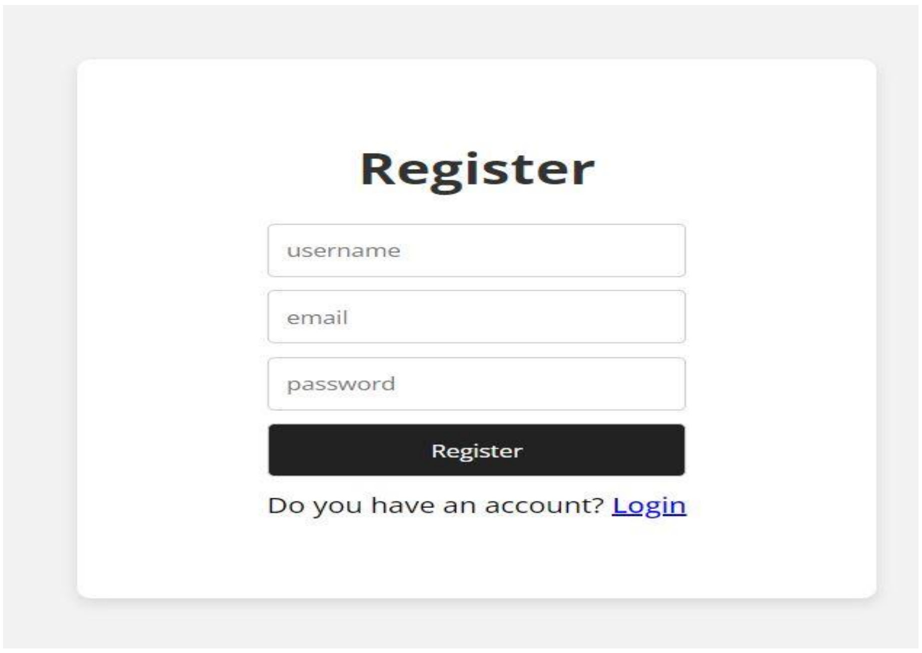
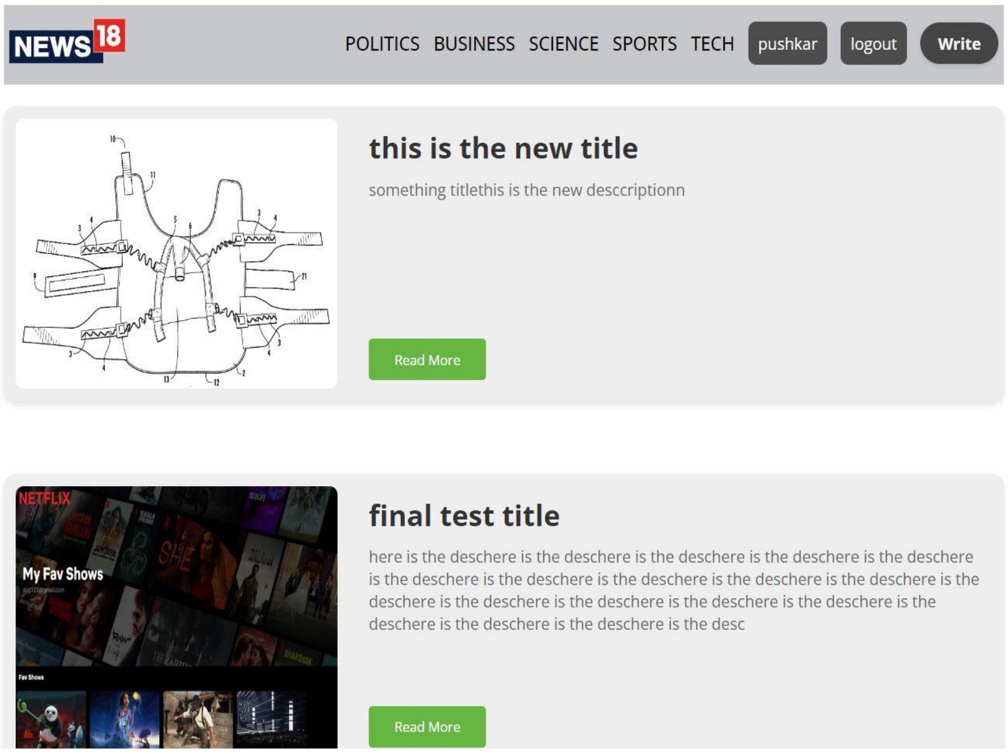


Fig 5.1 Login and register page

2. Home Page



this is the new title

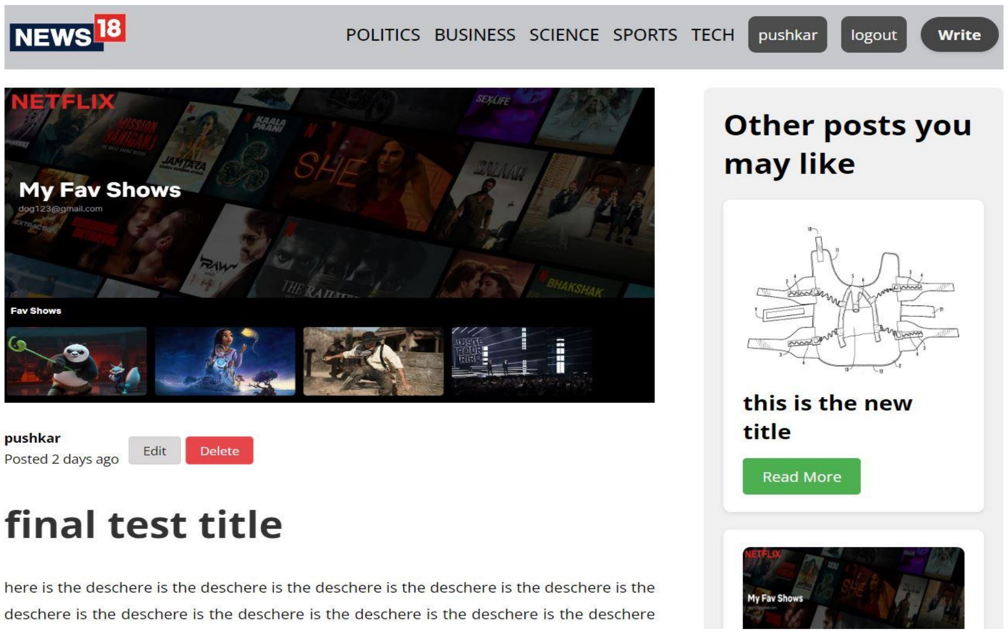
something titlethis is the new descrciptionn

final test title

here is the desc

Fig 5.2 Home page

3. Single Article Page



pushkar
Posted 2 days ago

final test title

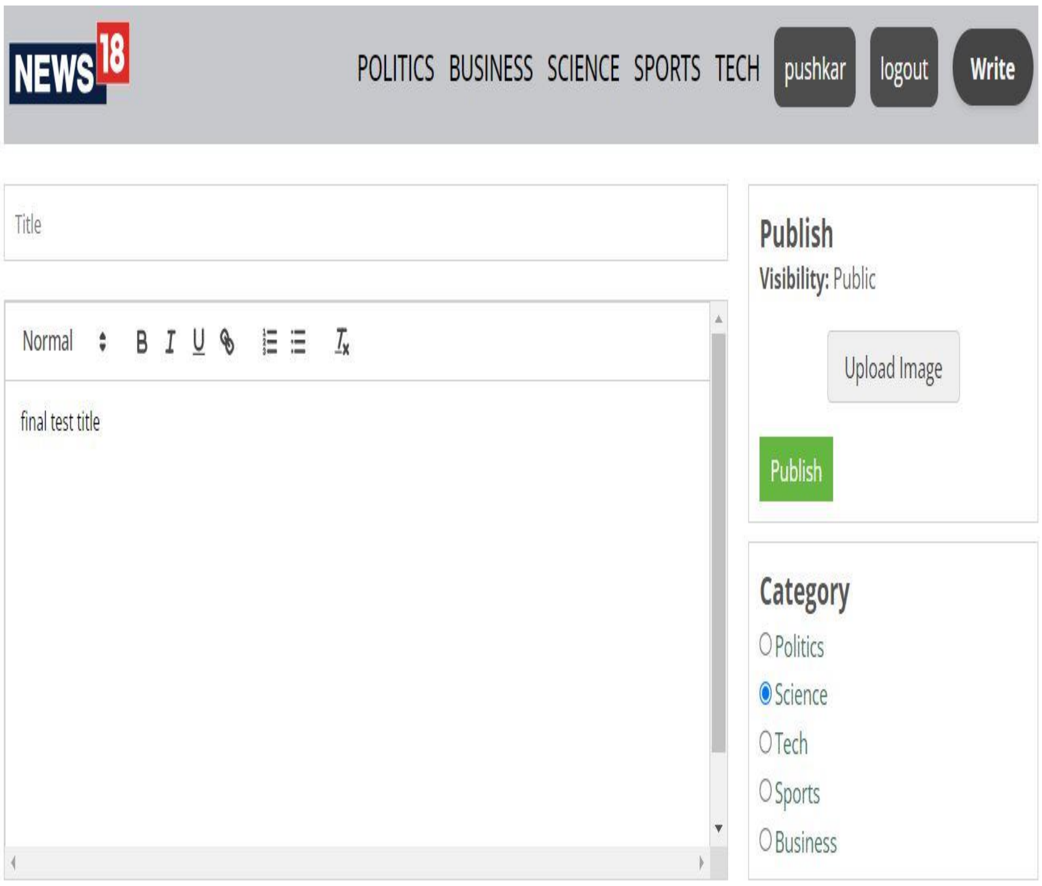
here is the deschere is the deschere is the deschere is the deschere is the deschere is the deschere is the deschere is the deschere is the deschere is the deschere is the deschere is the deschere

Other posts you may like

this is the new title

Fig 5.3 Single Article Page

4. Edit / Write Page



final test title

Visibility: Public

Category: Science [✓]

Fig 5.4 Edit/write page

CHAPTER 6

FUTURE SCOPE OF THE PROJECT

1. **Enhanced User Profiles:**
   - Implement advanced user profile features, allowing users to customize their profiles with bio information, profile pictures, and social media links.
   - Introduce features such as followers/following functionality, user ratings, and user activity feeds to enhance user engagement and community interaction.
2. **Advanced Search Functionality:**
   - Develop advanced search capabilities, allowing users to perform keyword searches across articles, categories, and authors.
   - Implement filters for sorting search results based on relevance, date, popularity, and user ratings to improve search accuracy and usability.
3. **Social Media Integration:**
   - Integrate social media sharing functionalities, enabling users to share articles on popular social media platforms such as Facebook, Twitter, and LinkedIn.
   - Implement social login options, allowing users to sign up or log in using their existing social media accounts for streamlined authentication.
4. **Real-time Updates and Notifications:**
   - Implement real-time updates and notifications to keep users informed about new articles, comments, likes, and other relevant activities on the platform.
   - Utilize WebSocket technology or server-sent events (SSE) for delivering instant notifications to users, enhancing user engagement and retention.
5. **Community Engagement Features:**
   - Introduce community engagement features such as comments, likes, and shares on articles to encourage user interaction and collaboration.
   - Implement moderation tools for managing user-generated content, including comment moderation and flagging inappropriate content.
6. **Monetization Strategies:**
   - Explore monetization strategies such as advertising placements, sponsored content, and premium subscriptions to generate revenue and sustain the platform.
   - Implement analytics tools to track user engagement, article performance, and advertising metrics for data-driven decision-making.

CHAPTER 7

CONCLUSION

In conclusion, the development of a news article CRUD website has been a fulfilling endeavor, resulting in a comprehensive platform for users to engage with news articles. Throughout the development process, we have utilized modern web technologies such as React, Express.js, Node.js, and MySQL to create a seamless and intuitive user experience.
The implementation of features such as user authentication, CRUD operations, and category-based filtering has significantly enhanced the functionality and usability of the platform, providing users with a robust and engaging experience.

Looking ahead, there are numerous opportunities for further expansion and improvement of the news article CRUD website. From enhancing user profiles and search functionality to integrating social media and implementing real-time updates, the future scope of the project is vast and promising. By continuously iterating and innovating, we aim to further elevate the news article CRUD website as a premier destination for news consumption and community engagement.

We extend our gratitude to all contributors, stakeholders, and users who have supported and participated in the development of the news article CRUD website. As we continue to evolve and grow, we remain committed to delivering a valuable and enriching experience for our users.

REFERENCES

1. **W3Schools. Node.js Tutorial**. Retrieved from https://www.w3schools.com/nodejs/

2. **React-** A JavaScript library for building user interfaces. Retrieved from https://reactjs.org/

3. **Express.js** Node.js web application framework. Retrieved from https://expressjs.com/

4. **MySQL:** MySQL 8.0 Reference Manual. Retrieved from https://dev.mysql.com/doc/refman/8.0/en/

5. **JSON Web Token**(JWT). Retrieved from https://jwt.io/

6. **Bcrypt. bcrypt.** Retrieved from https://www.npmjs.com/package/bcrypt

7. **Multer. multer**. Retrieved from https://www.npmjs.com/package/multer

8. **Stack Overflow**. Community-driven question and answer site. Retrieved from https://stackoverflow.com/

9. **GitHub**. The world's leading software development platform. Retrieved from https://github.com/