**Objective:-** Database: Using MongoDB, build the CRUD operations with NodeJS.

**Code :-**

**mongo\_client :-**

**App.jsx**

import { useState, useEffect } from 'react';

import axios from 'axios';

import AddBlogs from './components/AddBlogs';

import BlogsTable from './components/BlogsTable';

import './App.css';

const API\_URL = 'http://localhost:3000/api/blogs';

const App = () => {

  const [blogs, setBlogs] = useState([]);

  const [editingBlog, setEditingBlog] = useState(null);

  const [loading, setLoading] = useState(false);

  const [error, setError] = useState('');

  useEffect(() => {

    fetchBlogs();

  }, []);

  const fetchBlogs = async () => {

    setLoading(true);

    try {

      const response = await axios.get(API\_URL);

      setBlogs(response.data.blogs);

      setError('');

    } catch (error) {

      console.error('Error fetching blogs:', error);

      setError('Failed to fetch blogs. Try again later.');

    } finally {

      setLoading(false);

    }

  };

  const addBlog = async (newBlog) => {

    try {

      if (editingBlog) {

        await axios.put(`${API\_URL}/${editingBlog.\_id}`, newBlog);

        setEditingBlog(null);

      } else {

        await axios.post(API\_URL, newBlog);

      }

      fetchBlogs();

    } catch (error) {

      console.error('Error adding/updating blog:', error);

      setError('Failed to save blog.');

    }

  };

  const deleteBlog = async (id) => {

    if (!window.confirm('Are you sure you want to delete this blog?')) return;

    try {

      if (!id) {

        throw new Error('Blog ID is required');

      }

      const response = await axios.delete(`${API\_URL}/${id}`);

      if (response.data) {

        setBlogs((prevBlogs) => prevBlogs.filter((blog) => blog.\_id !== id));

        setError('');

      }

    } catch (error) {

      console.error('Error deleting blog:', error);

      const errorMessage = error.response?.data?.message || 'Failed to delete blog. Please check if the ID is valid.';

      setError(errorMessage);

    }

  };

  return (

    <div className="container">

      <h1>Blog Management System</h1>

      {loading && <p>Loading...</p>}

      {error && <p className="error">{error}</p>}

      <AddBlogs onAdd={addBlog} editingBlog={editingBlog} />

      <BlogsTable

        blogs={blogs}

        onEdit={setEditingBlog}

        onDelete={deleteBlog}

      />

    </div>

  );

};

export default App;

**App.css**

*/\* Import Poppins Font \*/*

*@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600&display=swap');*

*/\* General Reset \*/*

*\* {*

*box-sizing: border-box;*

*margin: 0;*

*padding: 0;*

*}*

*/\* Full-Screen Layout \*/*

*body {*

*font-family: 'Poppins', sans-serif;*

*background: linear-gradient(135deg, #74EBD5, #ACB6E5);*

*color: #333;*

*min-height: 100vh;*

*display: flex;*

*justify-content: center;*

*align-items: center;*

*padding: 20px;*

*}*

*/\* Main Container \*/*

*.container {*

*background: #fff;*

*box-shadow: 0 10px 20px rgba(0, 0, 0, 0.1);*

*border-radius: 12px;*

*max-width: 1200px;*

*width: 100%;*

*padding: 40px;*

*animation: fadeIn 0.8s ease-in-out;*

*transition: all 0.3s;*

*}*

*/\* Header Styling \*/*

*h1 {*

*color: #4CAF50;*

*font-size: 36px;*

*margin-bottom: 20px;*

*text-align: center;*

*}*

*/\* Form Styling \*/*

*form {*

*display: grid;*

*grid-template-columns: repeat(4, 1fr);*

*gap: 20px;*

*padding: 20px 0;*

*}*

*form input {*

*padding: 12px;*

*border: 1px solid #ccc;*

*border-radius: 8px;*

*font-size: 16px;*

*color: #333;*

*transition: all 0.3s;*

*}*

*form input:focus {*

*outline: none;*

*border-color: #4CAF50;*

*box-shadow: 0 0 8px rgba(76, 175, 80, 0.5);*

*}*

*form button {*

*grid-column: span 4;*

*padding: 14px;*

*background: #4CAF50;*

*color: white;*

*border: none;*

*border-radius: 8px;*

*font-size: 16px;*

*cursor: pointer;*

*transition: background 0.3s, transform 0.2s;*

*}*

*form button:hover {*

*background: #45a049;*

*transform: scale(1.05);*

*}*

*form button:active {*

*transform: scale(0.95);*

*}*

*/\* Table Styling \*/*

*table {*

*width: 100%;*

*border-collapse: collapse;*

*margin-top: 20px;*

*}*

*thead {*

*background: #4CAF50;*

*color: white;*

*}*

*th, td {*

*padding: 10px;*

*text-align: center;*

*border-bottom: 1px solid #ddd;*

*font-size: 16px;*

*}*

*tbody tr:hover {*

*background: #f1f1f1;*

*}*

*/\* Alternating row colors \*/*

*tbody tr:nth-child(even) {*

*background: #f9f9f9;*

*}*

*/\* Button Styling \*/*

*button {*

*cursor: pointer;*

*transition: all 0.3s ease;*

*}*

*/\* Edit and Delete Buttons \*/*

*.edit-btn {*

*background: #2196F3;*

*color: white;*

*border: none;*

*padding: 10px 20px;*

*border-radius: 8px;*

*font-size: 14px;*

*margin-right: 5px;*

*transition: 0.3s;*

*margin-bottom: 2px;*

*width: 90px;*

*}*

*.edit-btn:hover {*

*background: #1976D2;*

*transform: scale(1.05);*

*}*

*.delete-btn {*

*background: #f44336;*

*color: white;*

*border: none;*

*padding: 10px 20px;*

*border-radius: 8px;*

*font-size: 14px;*

*transition: 0.3s;*

*width: 90px;*

*}*

*.delete-btn:hover {*

*background: #d32f2f;*

*transform: scale(1.05);*

*}*

*/\* Animation \*/*

*@keyframes fadeIn {*

*from {*

*opacity: 0;*

*transform: translateY(-30px);*

*}*

*to {*

*opacity: 1;*

*transform: translateY(0);*

*}*

*}*

*/\* Responsive Design \*/*

*@media (max-width: 1024px) {*

*form {*

*grid-template-columns: repeat(2, 1fr);*

*}*

*form button {*

*grid-column: span 2;*

*}*

*th, td {*

*font-size: 14px;*

*padding: 12px;*

*}*

*}*

*@media (max-width: 768px) {*

*form {*

*grid-template-columns: 1fr;*

*}*

*form button {*

*grid-column: span 1;*

*}*

*th, td {*

*font-size: 12px;*

*padding: 10px;*

*}*

*.container {*

*padding: 20px;*

*}*

*h1 {*

*font-size: 28px;*

*}*

*}*

**Components/AddBlogs.jsx**

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

const AddBlogs = ({ editingBlog, onAdd }) => {

  const [id, setId] = useState('');

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

  const [authorName, setAuthorName] = useState('');

  const [date, setDate] = useState('');

  const [content , setContent] = useState('');

  // Pre-fill form with existing data during editing

  useEffect(() => {

    if (editingBlog) {

      setId(editingBlog.\_id);

      setTitle(editingBlog.title);

      setAuthorName(editingBlog.author);

      setDate(editingBlog.date.split('T')[0]);

      setContent(editingBlog.description);

    } else {

      // Reset form when not editing

      setId('');

      setTitle('');

      setAuthorName('');

      setDate('');

      setContent('');

    }

  }, [editingBlog]);

  const handleSubmit = async (e) => {

    e.preventDefault();

    const blogData = {

      \_id : id,

      title: title,

      author: authorName,

      description : content,

      date: date || new Date().toISOString().split('T')[0] // Use current date if none provided

    };

    try {

      await onAdd(blogData);

      alert(`Blog ${editingBlog ? 'updated' : 'added'} successfully!`);

    } catch (error) {

      console.error('Error saving blog:', error);

      alert('Failed to save blog');

    }

    // Clear form after submission

    setId('');

    setTitle('');

    setAuthorName('');

    setDate('');

    setContent('');

  };

  return (

    <form onSubmit={handleSubmit}>

      <input

        type="text"

        placeholder="Title"

        value={title}

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

        required

      />

      <input

        type="text"

        placeholder="Author Name"

        value={authorName}

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

        required

      />

      <input

        type="text"

        placeholder="Content"

        value={content}

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

        required

      />

      <input

        type="date"

        placeholder="Date"

        value={date}

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

        required

      />

      <button type="submit">

        {editingBlog ? 'Update Blog' : 'Add Blog'}

      </button>

    </form>

  );

};

export default AddBlogs;

**Components/BlogsTable.jsx**

import React from 'react';

const BlogsTable = ({ blogs, onEdit, onDelete }) => {

  if (!blogs || !Array.isArray(blogs) || blogs.length === 0) {

    return <p>No blogs available.</p>;

  }

  return (

    <div>

      <h2>Blogs</h2>

      <table>

        <thead>

          <tr>

            <th>ID</th>

            <th>Title</th>

            <th>Author Name</th>

            <th>Content</th>

            <th>Date</th>

            <th>Actions</th>

          </tr>

        </thead>

        <tbody>

          {blogs.map((blog) => (

            <tr key={blog.\_id}>

              <td>{blog.\_id}</td>

              <td>{blog.title}</td>

              <td>{blog.author}</td>

              <td>{blog.description}</td>

              <td>{new Date(blog.date).toLocaleDateString().split('T')[0]}</td>

              <td>

                <button

                  className="edit-btn"

                  onClick={() => onEdit(blog)}

                >

                  Edit

                </button>

                <button

                  className="delete-btn"

                  onClick={() => onDelete(blog.\_id)}

                >

                  Delete

                </button>

              </td>

            </tr>

          ))}

        </tbody>

      </table>

    </div>

  );

};

export default BlogsTable;

**mongo\_server :-**

**index.js :-**

const express = require("express");

const app = express();

const port = 3000;

const cors = require("cors");

app.use(

  cors({

    origin: "http://localhost:5173",

  })

);

//middleware

app.use(express.json());

app.use(express.urlencoded({ extended: true }));

const dbConfig = require("./src/config/db.config");

const mongoose = require("mongoose");

mongoose.Promise = global.Promise;

mongoose

  .connect(dbConfig.url, {

    useNewUrlParser: true,

    useUnifiedTopology: true,

  })

  .then(() => {

    console.log("Sucessfully Connected to the Database");

  })

  .catch((err) => {

    console.log("Error in Database connection ", err);

    process.exit();

  });

const blogRouter = require("./src/blog/blog.routes");

app.use("/api/blogs", blogRouter);

app.get("/", (req, res) => res.send("Welcome to the API of AWT Experiment - 6 API"));

app.listen(port, () => console.log(`Server is running on http://localhost:${port}`));

**config/db.config.js :-**

module.exports = {

url: "mongodb://127.0.0.1:27017/exp\_6\_blog",

};

**blog/blo.controller.js**

const blogService = require("../blog/blog.services");

exports.create = async (req, res) => {

  const { title, description, author , date } = req.body;

  if (!title || !description || !author) {

    return res.status(400).json({ message: "Fields are empty" });

  }

  try {

    const blog = await blogService.createBlog(title, description, author , date);

    res.status(201).json({ message: "Blog created successfully", blog });

  } catch (err) {

    console.error("Error creating blog:", err);

    res.status(500).json({ error: "Internal server error" });

  }

};

exports.findAll = async (req, res) => {

  try {

    const blogs = await blogService.findAllBlogs();

    res.send({ message: "Blogs data", blogs });

  } catch (err) {

    console.error("Error fetching blogs:", err);

    res.status(500).json({ error: "Internal server error" });

  }

};

exports.findOne = async (req, res) => {

  const id = req.params.\_id;

  try {

    const blog = await blogService.findBlogById(id);

    if (!blog) {

      return res.status(400).json({ message: "No blog found with id: " + id });

    }

    res.json({ message: "Data found", blog });

  } catch (err) {

    console.error("Error fetching blog by id:", err);

    res.status(500).json({ error: "Internal server error" });

  }

};

exports.update = async (req, res) => {

  const id = req.params.id;

  const updateData = req.body;

  try {

    const result = await blogService.updateBlog(id, updateData);

    if (!result) {

      return res.status(400).json({ message: "No blog found with id: " + id });

    }

    res.json({ message: "Blog updated successfully" });

  } catch (err) {

    console.error("Error updating blog:", err);

    res.status(500).json({ error: "Internal server error" });

  }

};

exports.delete = async (req, res) => {

  const id = req.params.id;

  try {

    const result = await blogService.deleteBlog(id);

    if (!result) {

      return res.status(400).json({ message: "No blog found with id: " + id });

    }

    res.json({ message: "Blog deleted successfully" });

  } catch (err) {

    console.error("Error deleting blog:", err);

    res.status(500).json({ error: "Internal server error" });

  }

};

**blog/blog.model.js**

const mongoose = require("mongoose");

var schema = new mongoose.Schema(

  {

    title: {

      type: String,

      required: true,

    },

    description: {

      type: String,

      required: true,

    },

    author: {

      type: String,

      required: true,

    },

    date: {

      type: Date,

      default: Date.now(),

      required: true,

    }

  },

  {

    timestamps: true,

  }

);

var blog = new mongoose.model("exp\_6\_blog", schema);

module.exports = blog;

**blog/blog.routes.js**

const blog = require("../blog/blog.controller");

const express = require("express");

const router = express.Router();

router.post("/", blog.create);

router.get("/", blog.findAll);

router.get("/:id", blog.findOne);

router.put("/:id", blog.update);

router.delete("/:id", blog.delete);

module.exports = router;

**blog/blog.services.js**

const BlogModel = require("../blog/blog.model");

exports.createBlog = async (title, description, author , date) => {

  const blog = new BlogModel({

    title,

    description,

    author,

    date

  });

  return await blog.save();

};

exports.findAllBlogs = async () => {

  return await BlogModel.find();

};

exports.findBlogById = async (id) => {

  return await BlogModel.findById(id);

};

exports.updateBlog = async (id, updateData) => {

  return await BlogModel.findByIdAndUpdate(id, updateData, {

    useFindAndModify: false,

  });

};

exports.deleteBlog = async (id) => {

  return await BlogModel.findByIdAndDelete(id);

};

**Output :-**











