**Objective:-** Develop a web application which involves database operations using NodeJS with Error Handling.

**Code :-**

**Frontend :-**

**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);

      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 {

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

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

    } catch (error) {

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

      setError('Failed to delete blog.');

    }

  };

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

*\* {*

*box-sizing: border-box;*

*margin: 0;*

*padding: 0;*

*}*

*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;*

*}*

*.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;*

*}*

*h1 {*

*color: #4CAF50;*

*font-size: 36px;*

*margin-bottom: 20px;*

*text-align: center;*

*}*

*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 {*

*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;*

*}*

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

*background: #f9f9f9;*

*}*

*button {*

*cursor: pointer;*

*transition: all 0.3s ease;*

*}*

*.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);*

*}*

*@keyframes fadeIn {*

*from {*

*opacity: 0;*

*transform: translateY(-30px);*

*}*

*to {*

*opacity: 1;*

*transform: translateY(0);*

*}*

*}*

*@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';

import axios from 'axios';

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\_name);

      setDate(editingBlog.date);

      setContent(editingBlog.content);

    } else {

*// Reset form when not editing*

      setId('');

      setTitle('');

      setAuthorName('');

      setDate('');

      setContent('');

    }

  }, [editingBlog]);

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

    e.preventDefault();

    const blogData = {

      id,

      title,

      author\_name: authorName,

      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

      />

      <textarea

        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)}

      />

      <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>Date</th>

            <th>Content</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\_name}</td>

              <td>{blog.content}</td>

              <td>{new Date(blog.date).toLocaleDateString()}</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;

**Server API :-**

**Index.js :-**

const express = require('express');

const bodyParser = require('body-parser');

const cors = require('cors');

const blogRoutes = require('./src/blogs/blogs.routes');

const db = require('./src/config/db.config');

const app = express();

const PORT = 3000;

app.use(cors());

app.use(bodyParser.json());

app.use('/api/blogs', blogRoutes);

app.get('/', (*req*, *res*) => {

  res.send('Blog Management System API');

});

app.listen(PORT, () => {

  console.log(`Server running at  http://localhost:${PORT}`);

});

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

const mysql = require('mysql2');

const db = mysql.createConnection({

  host: 'localhost',

  user: 'root',

  password: '',

  database: 'student\_results'

});

db.connect((*err*) => {

  if (err) {

    console.error('Database connection failed: ' + err.stack);

    return;

  }

  console.log('Connected to MySQL database.');

});

module.exports = db;

**marks/blogs.controller.js**

const blogService = require('./blogs.services');

exports.createBlog = (*req*, *res*) => {

  const blog = req.body;

  blogService.createBlog(blog, (*err*, *result*) => {

    if (err) {

      res.status(500).send({ message: 'Error creating blog', error: err });

    } else {

      res.status(201).send({ message: 'Blog created', data: result });

    }

  });

};

exports.getAllBlogs = (*req*, *res*) => {

  blogService.getAllBlogs((*err*, *results*) => {

    if (err) {

      res.status(500).send({ message: 'Error fetching blogs', error: err });

    } else {

      res.status(200).send(results);

    }

  });

};

exports.getBlogById = (*req*, *res*) => {

  const id = req.params.id;

  blogService.getBlogById(id, (*err*, *result*) => {

    if (err) {

      res.status(500).send({ message: 'Error fetching blog', error: err });

    } else {

      res.status(200).send(result[0]);

    }

  });

};

exports.updateBlog = (*req*, *res*) => {

  const id = req.params.id;

  const blog = req.body;

  blogService.updateBlog(id, blog, (*err*, *result*) => {

    if (err) {

      res.status(500).send({ message: 'Error updating blog', error: err });

    } else {

      res.status(200).send({ message: 'Blog updated successfully' });

    }

  });

};

exports.deleteBlog = (*req*, *res*) => {

  const id = req.params.id;

  blogService.deleteBlog(id, (*err*, *result*) => {

    if (err) {

      res.status(500).send({ message: 'Error deleting blog', error: err });

    } else {

      res.status(200).send({ message: 'Blog deleted successfully' });

    }

  });

};

**marks/blogs.model.js**

const db = require('../config/db.config');

class Blog {

  static create(*blog*, *callback*) {

    const query = `

      INSERT INTO blogs (id, title, author\_name, content , date)

      VALUES (?, ?, ?, ?, ?)`;

    db.query(query, [blog.id, blog.title, blog.author\_name, blog.content , blog.date || new Date()], callback);

  }

  static findAll(*callback*) {

    const query = 'SELECT \* FROM blogs';

    db.query(query, callback);

  }

  static findById(*id*, *callback*) {

    const query = 'SELECT \* FROM blogs WHERE id = ?';

    db.query(query, [id], callback);

  }

  static update(*id*, *blog*, *callback*) {

    const query = `

      UPDATE blogs

      SET title = ?, author\_name = ?, date = ?

      WHERE id = ?`;

    db.query(query, [blog.title, blog.author\_name, blog.date || new Date(), id], callback);

  }

  static delete(*id*, *callback*) {

    const query = 'DELETE FROM blogs WHERE id = ?';

    db.query(query, [id], callback);

  }

}

module.exports = Blog;

**marks/blogs.routes.js**

const express = require('express');

const router = express.Router();

const blogController = require('./blogs.controller');

router.post('/', blogController.createBlog);

router.get('/', blogController.getAllBlogs);

router.get('/:id', blogController.getBlogById);

router.put('/:id', blogController.updateBlog);

router.delete('/:id', blogController.deleteBlog);

module.exports = router;

**marks/blogs.services.js**

const Blog = require('./blogs.model');

const createBlog = (*data*, *callback*) => {

  Blog.create(data, callback);

};

const getAllBlogs = (*callback*) => {

  Blog.findAll(callback);

};

const getBlogById = (*id*, *callback*) => {

  Blog.findById(id, callback);

};

const updateBlog = (*id*, *data*, *callback*) => {

  Blog.update(id, data, callback);

};

const deleteBlog = (*id*, *callback*) => {

  Blog.delete(id, callback);

};

module.exports = {

  createBlog,

  getAllBlogs,

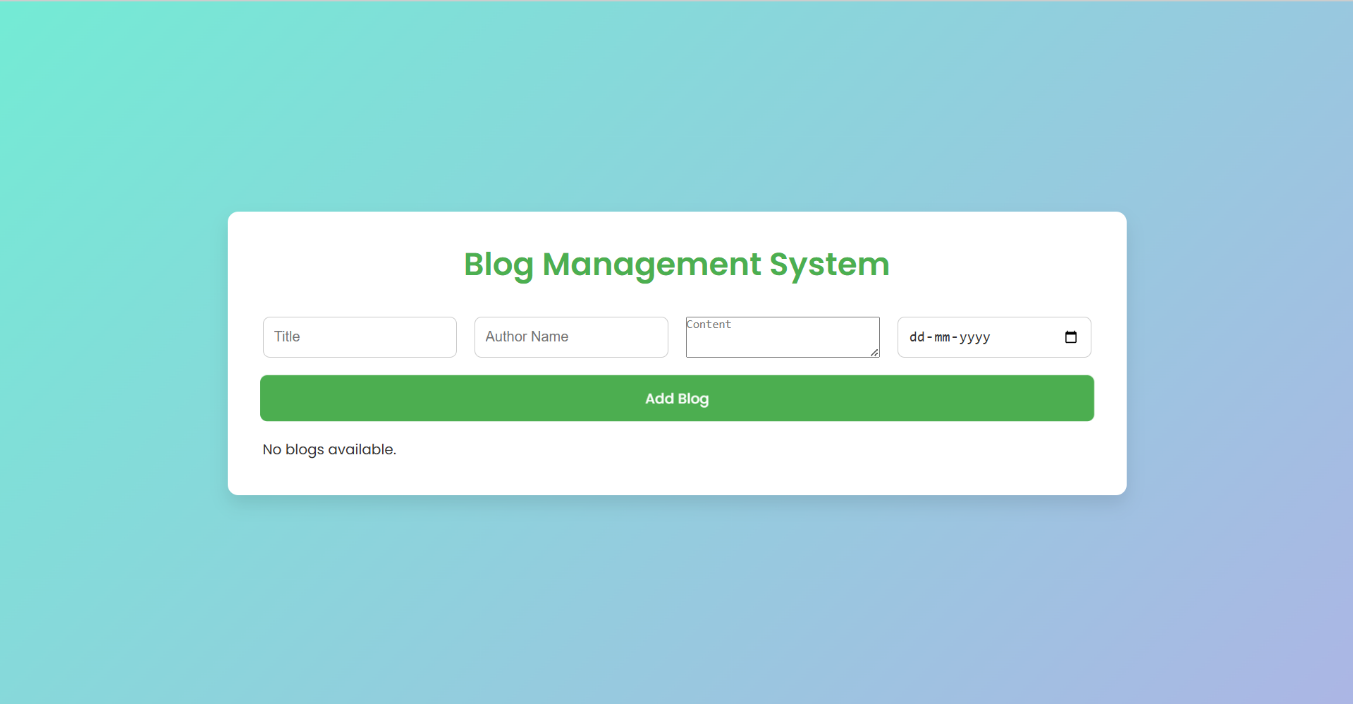
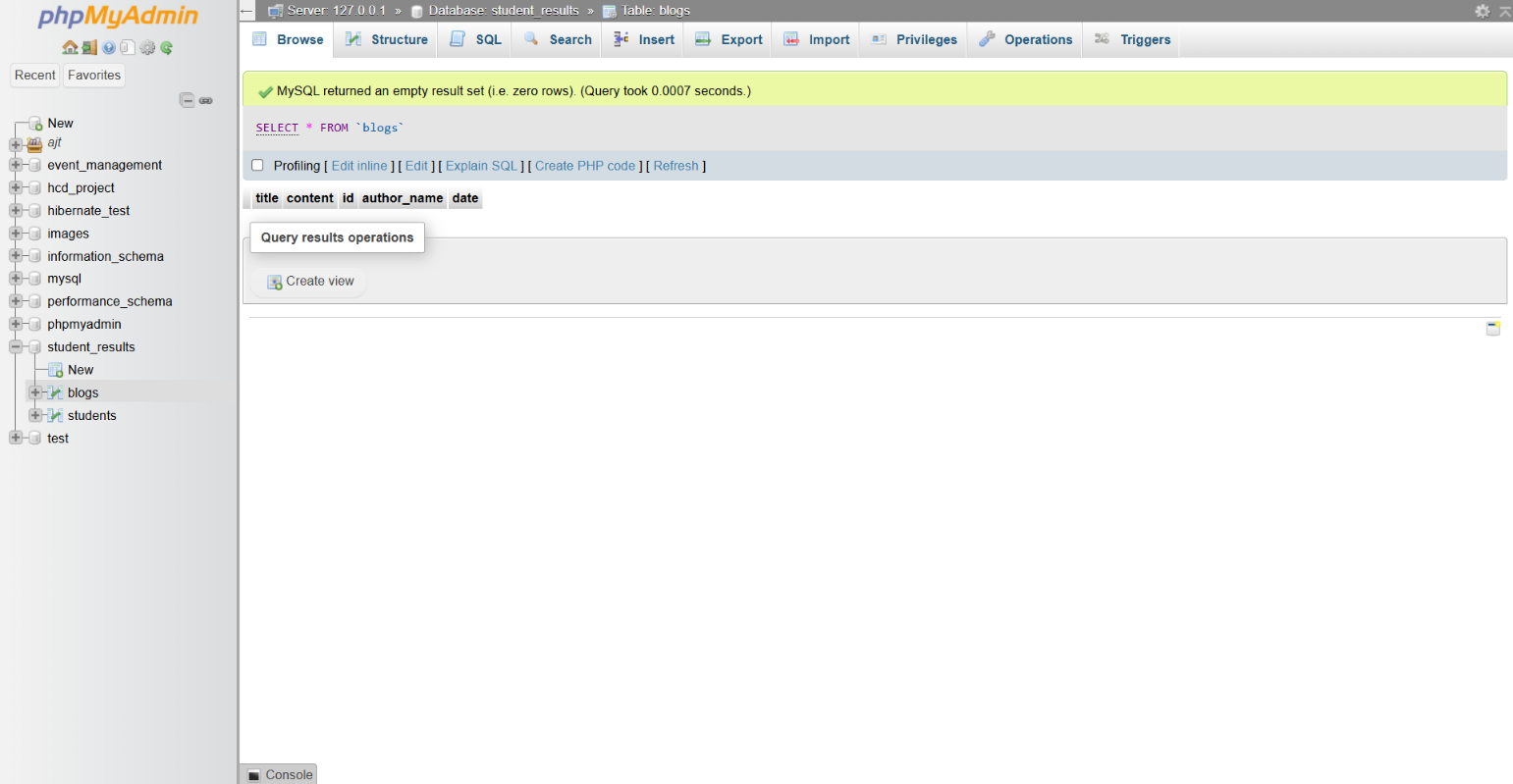
  getBlogById,

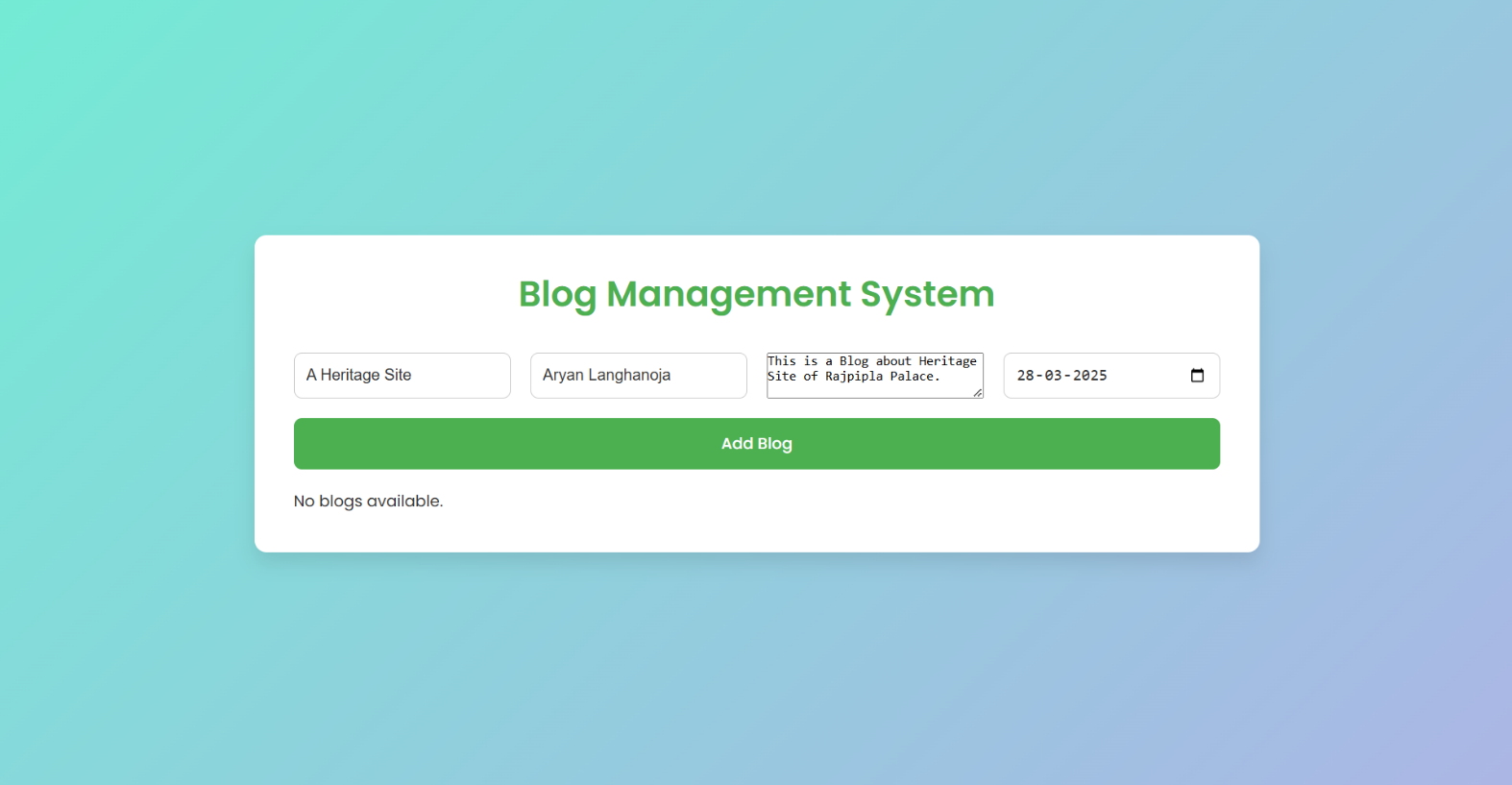
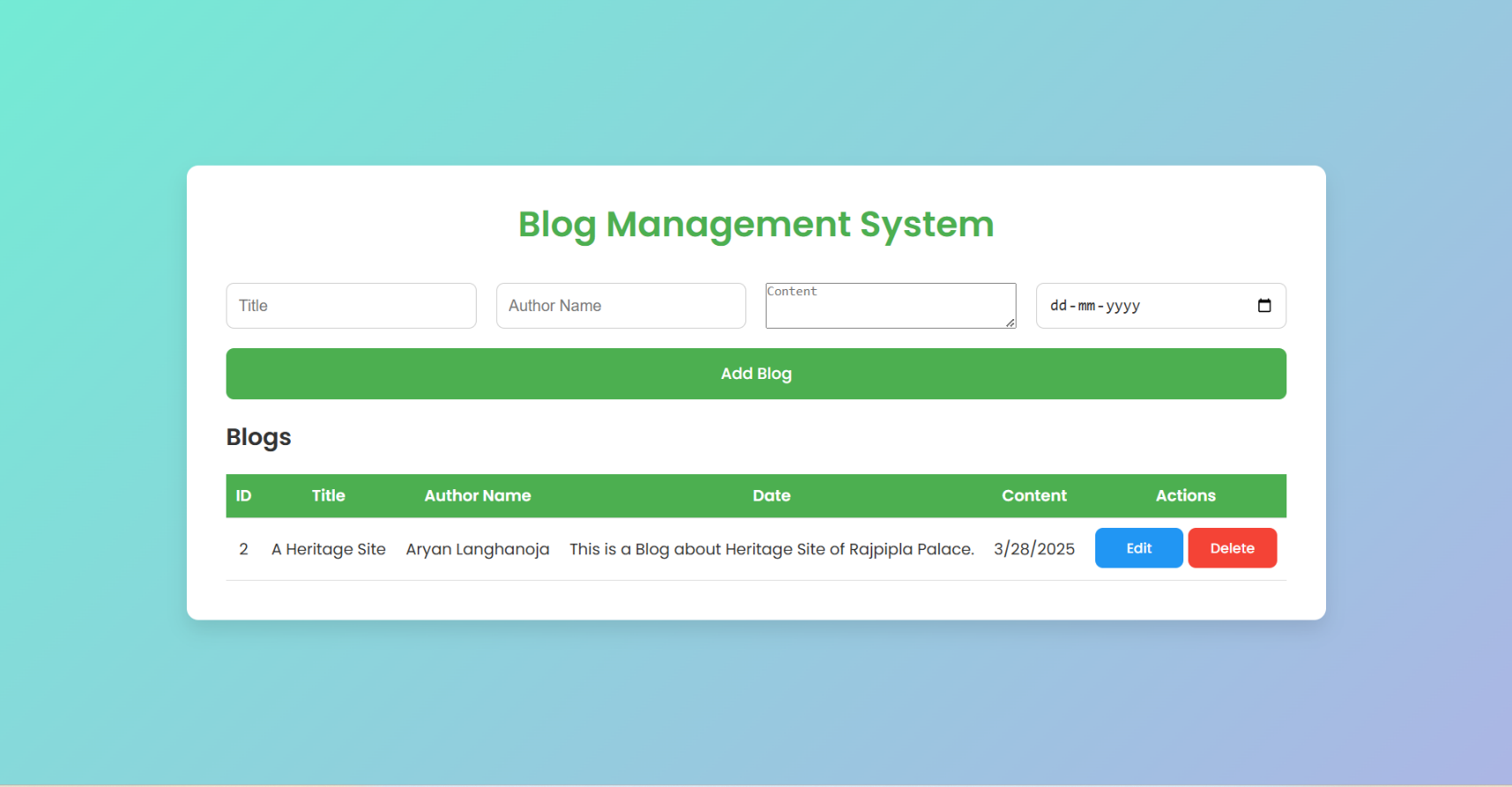
  updateBlog,

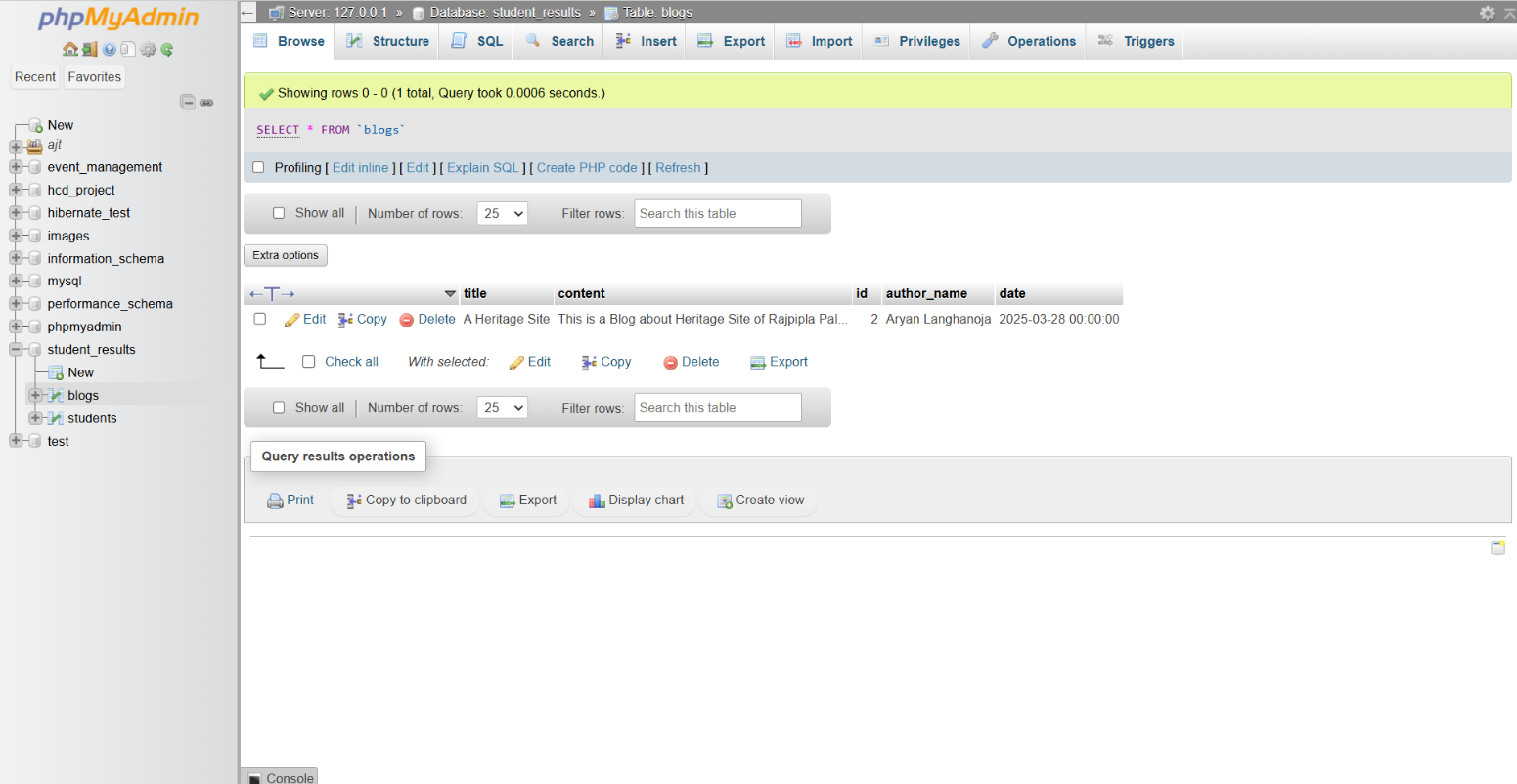
  deleteBlog

};

**Output :-**



****

****

