**Objective:-** Create a react application to perform CRUD operations with Firebase.

**Code :-**

**firebase\_client :-**

**App.jsx**

import { useState, useEffect } from 'react';

import axios from 'axios';

import AddBooks from './components/AddBooks';

import BooksTable from './components/BooksTable';

import './App.css';

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

const App = () => {

  const [Books, setBooks] = useState([]);

  const [editingbook, setEditingbook] = useState(**null**);

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

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

  useEffect(() => {

    fetchBooks();

  }, []);

  const fetchBooks = async () => {

    setLoading(**true**);

    try {

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

      console.log(response.data);

      setBooks(response.data);

      setError('');

    } catch (error) {

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

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

    } finally {

      setLoading(**false**);

    }

  };

  const addbook = async (*newbook*) => {

    try {

      if (editingbook) {

        await axios.put(`${API\_URL}/${editingbook.id}`, newbook);

        setEditingbook(**null**);

      } else {

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

      }

      fetchBooks();

    } catch (error) {

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

      setError('Failed to save book.');

    }

  };

  const deletebook = async (*id*) => {

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

    try {

      if (!id) {

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

      }

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

      if (response.data) {

        setBooks((*prevBooks*) => prevBooks.filter((*book*) => book.id !== id));

        setError('');

      }

    } catch (error) {

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

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

      setError(errorMessage);

    }

  };

  return (

    <div className="container">

      <h1>Book Management System</h1>

      {loading && <p>Loading...</p>}

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

      <AddBooks onAdd={addbook} editingbook={editingbook} />

      <BooksTable

        Books={Books}

        onEdit={setEditingbook}

        onDelete={deletebook}

      />

    </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/AddBooks.jsx**

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

const AddBooks = ({ *editingbook*, *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 (editingbook) {

      setId(editingbook.id);

      setTitle(editingbook.title);

      setAuthorName(editingbook.author);

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

      setContent(editingbook.description);

    } else {

*// Reset form when not editing*

      setId('');

      setTitle('');

      setAuthorName('');

      setDate('');

      setContent('');

    }

  }, [editingbook]);

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

    e.preventDefault();

    const bookData = {

      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(bookData);

      alert(`book ${editingbook ? 'updated' : 'added'} successfully!`);

    } catch (error) {

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

      alert('Failed to save book');

    }

*// 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">

        {editingbook ? 'Update book' : 'Add book'}

      </button>

    </form>

  );

};

export default AddBooks;

**Components/BooksTable.jsx**

import React from 'react';

const BooksTable = ({ *Books*, *onEdit*, *onDelete* }) => {

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

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

  }

  return (

    <div>

      <h2>Books</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>

          {Books.map((*book*) => {

            return (

              <tr key={book.id}>

                <td>{book.id}</td>

                <td>{book.title}</td>

                <td>{book.author}</td>

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

                <td>{book.date}</td>

                <td>

                  <button className="edit-btn" onClick={() => onEdit(book)}>

                    Edit

                  </button>

                  <button className="delete-btn" onClick={() => onDelete(book.id)}>

                    Delete

                  </button>

                </td>

              </tr>

            );

          })}

        </tbody>

      </table>

    </div>

  );

};

export default BooksTable;

**firebase\_server :-**

**index.js :-**

import dotenv from "dotenv";

dotenv.config();

import express from "express";

import bodyParser from "body-parser";

import cors from "cors";

const PORT = process.env.PORT || 3000;

const app = express();

import bookRoutes from "./src/routes/book.routes.js";

app.use(cors())

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

app.use(bodyParser.json());

app.use(express.json());

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

app.use("/api/books", bookRoutes);

try {

    const db = await import("./src/config/firebase\_connection.js");

    console.log("Connected to Firebase Sucessfully");

} catch (error) {

    console.log("Error connecting to Firebase");

    console.log(error.message)

}

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

    res.send("Welcome to Firebase CRUD Operation API!");

});

app.listen(PORT, () => {

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

});

**.src/config/firebase\_connection.js:-**

*// Import the functions you need from the SDKs you need*

import { initializeApp } from "firebase/app";

import { getFirestore , collection } from 'firebase/firestore';

*// 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*

const firebaseConfig = {

  apiKey: "AIzaSyAfp4MUpDuBEZO5cOg97O6Kos3xP9FzWfM",

  authDomain: "awttask1.firebaseapp.com",

  projectId: "awttask1",

  storageBucket: "awttask1.firebasestorage.app",

  messagingSenderId: "711675249544",

  appId: "1:711675249544:web:d81f16120c745da660f03d"

};

const app = initializeApp(firebaseConfig);

const db = getFirestore(app);

const book = collection(db, "books");

*// Error handling for Firebase connection*

try {

  if (!app || !db || !book) {

    throw new Error('Firebase connection failed');

  }

} catch (error) {

  console.error('Firebase initialization error:', error);

*// Re-throw to be handled by calling code*

  throw error;

}

export { db , book };

**./src/routes/book.routes.js**

import express from 'express';

import {

  createBook,

  getBook,

  getBookbyID,

  updateBook,

  deleteBook,

} from '../services/book.service.js';

const bookRoutes = express.Router();

bookRoutes.get('/', getBook);

bookRoutes.post('/', createBook);

bookRoutes.get('/:id', getBookbyID);

bookRoutes.put('/:id', updateBook);

bookRoutes.delete('/:id', deleteBook);

export default bookRoutes;

**./src/services/book.service.js**

import {

  addDoc,

  collection,

  getDoc,

  getDocs,

  doc,

  updateDoc,

  deleteDoc

} from "firebase/firestore";

import { db } from "../config/firebase\_connection.js";

export const createBook = async (*req*, *res*, *next*) => {

  try {

    const data = req.body;

    await addDoc(collection(db, 'books'), data);

    res.status(200).send('Book Created Successfully');

  } catch (error) {

    res.status(400).send(error.message);

  }

};

export const getBook = async (*req*, *res*, *next*) => {

  try {

    const books = await getDocs(collection(db, 'books'));

    const bookArray = [];

    if (books.empty) {

      res.send('No Books found');

    } else {

      books.forEach((*doc*) => {

        const book = {

          id : doc.id,

          title : doc.data().title,

          author: doc.data().author,

          description : doc.data().description,

          date : doc.data().date,

        };

        bookArray.push(book);

      });

      console.log(bookArray)

      res.status(200).send(bookArray);

    }

  } catch (error) {

    res.status(400).send(error.message);

  }

};

export const getBookbyID = async (*req*, *res*, *next*) => {

  try {

    const id = req.params.id;

    const book = doc(db, 'books', id);

    const data = await getDoc(book);

    if (data.exists()) {

      res.status(200).send(data.data());

    } else {

      res.status(404).send('Book not found');

    }

  } catch (error) {

    res.status(400).send(error.message);

  }

};

export const updateBook = async (*req*, *res*, *next*) => {

  try {

    const id = req.params.id;

    const data = req.body;

    const book = doc(db, 'books', id);

    await updateDoc(book, data);

    res.status(200).send('Book updated successfully');

  } catch (error) {

    res.status(400).send(error.message);

  }

};

export const deleteBook = async (*req*, *res*, *next*) => {

  try {

    const id = req.params.id;

    await deleteDoc(doc(db, 'books', id));

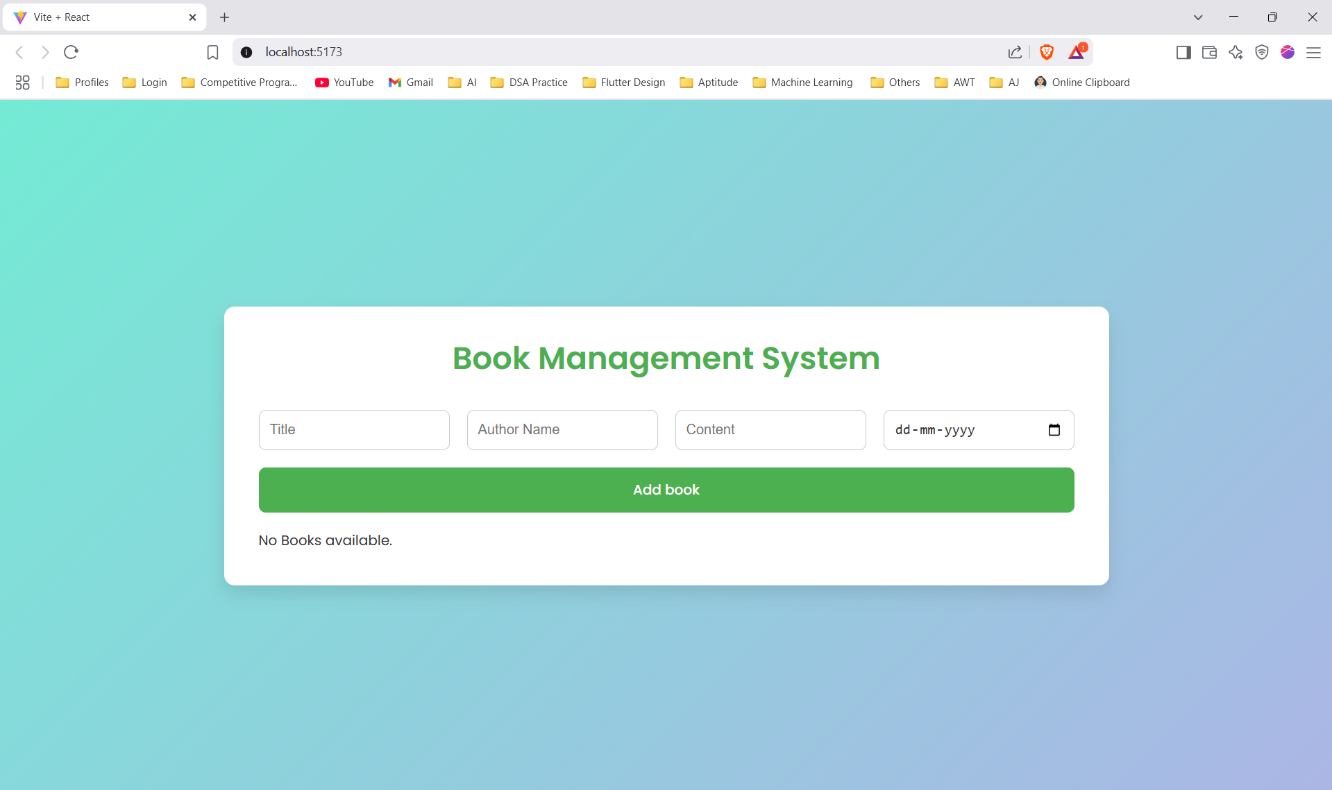
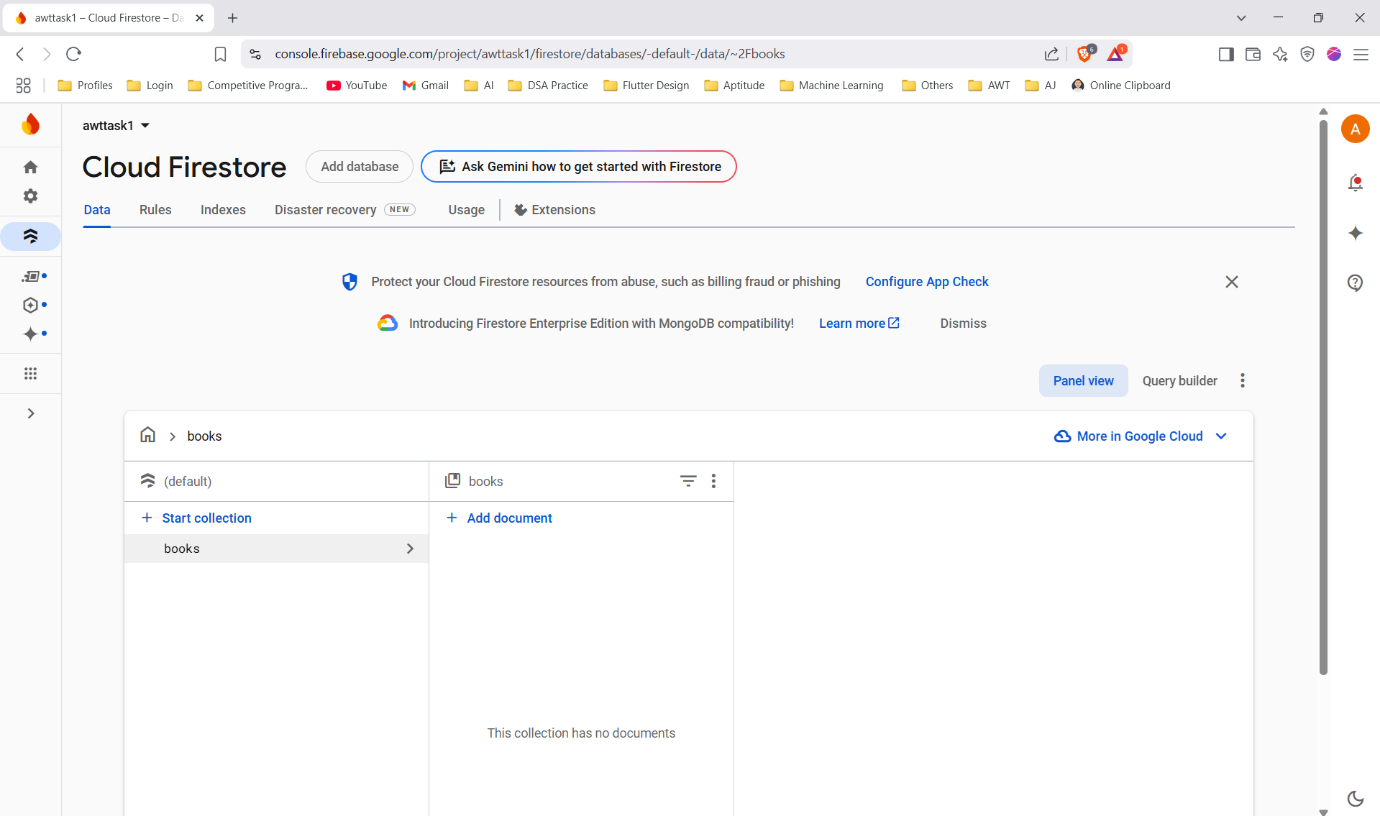
    res.status(200).send('Book deleted successfully');

  } catch (error) {

    res.status(400).send(error.message);

  }

};

**Output :-**

