**WAD Theory Assignment 2**

Name : Parikshit Daivadnya
Roll no. TIA11

Q1) Blogging Platform (User Posts and Comments)
Scenario: You are building a blogging platform where users can create blog posts, comment on them, and like them.
Question: Develop a blog post creation form, a post listing page, and a comment section for each post. Use ReactJS or Angular for the frontend. Implement CRUD operations for posts and comments using a REST API or GraphQL. Add like functionality for posts and comments and pagination for the posts list.

**Solution:**

**src/App.jsx**

```
import React, { useState } from 'react';
import PostForm from './components/PostForm';
import PostList from './components/PostList';
import './App.css';

const App = () => {
  const [refreshPosts, setRefreshPosts] = useState(false);
  const [darkMode, setDarkMode] = useState(false);

  const toggleTheme = () => {
    setDarkMode(!darkMode);
    document.documentElement.setAttribute(
      'data-theme',
      darkMode ? 'light' : 'dark'
    );
  };

  return (
    <div className="main-container">
      <h1>📝 Blogging Platform</h1>
      <button onClick={toggleTheme} className="theme-toggle">
        {darkMode ? '🌞 Light Mode' : '🌙 Dark Mode'}
      </button>
```

```
      <PostForm onPostCreated={() => setRefreshPosts(!refreshPosts)} />
      <PostList refresh={refreshPosts} />
    </div>
  );
};

export default App;
```

---

**src/components/PostList.jsx**

```
import React, { useEffect, useState } from 'react';
import axios from 'axios';
import Post from './PostItem';
import Pagination from './Pagination';
import './PostList.css';

const POSTS_PER_PAGE = 4;

const PostList = ({ refresh }) => {
  const [posts, setPosts] = useState([]);
  const [page, setPage] = useState(1);

  const fetchPosts = async () => {
    try {
      const response = await axios.get('http://localhost:3001/posts');
      setPosts(response.data.reverse());
    } catch (err) {
      console.error('Failed to fetch posts:', err);
    }
  };

  useEffect(() => {
    fetchPosts();
  }, [refresh]);

  const indexOfLast = page * POSTS_PER_PAGE;
  const indexOfFirst = indexOfLast - POSTS_PER_PAGE;
  const currentPosts = posts.slice(indexOfFirst, indexOfLast);
```

```
  return (
    <div className="post-list">
      {currentPosts.map((post) => (
        <Post key={post.id} post={post} onPostUpdate={fetchPosts} />
      ))}
      <Pagination
        totalPosts={posts.length}
        postsPerPage={POSTS_PER_PAGE}
        currentPage={page}
        setCurrentPage={setPage}
      />
    </div>
  );
};

export default PostList;
```

---

**src/components/PostItem.jsx**

```
import React, { useState } from 'react';
import axios from 'axios';
import CommentSection from './CommentSection';
import './PostItem.css';

const Post = ({ post, onPostUpdate }) => {
  const [isEditing, setIsEditing] = useState(false);
  const [title, setTitle] = useState(post.title);
  const [content, setContent] = useState(post.content);

  const handleDelete = async () => {
    try {
      await axios.delete(`http://localhost:3001/posts/${post.id}`);
      onPostUpdate();
    } catch (err) {
      console.error('Error deleting post:', err);
    }
  };
```

```
const handleEdit = async () => {
  try {
    const updatedPost = { ...post, title, content };
    await axios.put(`http://localhost:3001/posts/${post.id}`, updatedPost);
    setIsEditing(false);
    onPostUpdate();
  } catch (err) {
    console.error('Error saving post:', err);
  }
};

const handleLike = async () => {
  try {
    await axios.patch(`http://localhost:3001/posts/${post.id}`, {
      likes: post.likes + 1
    });
    onPostUpdate();
  } catch (err) {
    console.error('Error liking post:', err);
  }
};

return (
  <div className="post">
    {isEditing ? (
      <>
        <input
          className="post-edit-title"
          value={title}
          onChange={(e) => setTitle(e.target.value)}
        />
        <textarea
          className="post-edit-content"
          value={content}
          onChange={(e) => setContent(e.target.value)}
        />
        <button onClick={handleEdit}>💾 Save</button>
      </>
    ) : (
```

```
      <>
        <h2>{post.title}</h2>
        <p>{post.content}</p>
        <div className="post-actions">
          <button onClick={handleLike}>👍 {post.likes}</button>
          <button onClick={() => setIsEditing(true)}>🖍️ Edit</button>
          <button onClick={handleDelete}>🗑️ Delete</button>
        </div>
      </>
    )}
    <CommentSection postId={post.id} />
  </div>
 );
};

export default Post;
```

---

**src/components/PostForm.jsx**

```
import React, { useState } from 'react';
import axios from 'axios';
import './PostForm.css';

const PostForm = ({ onPostCreated }) => {
 const [title, setTitle] = useState('');
 const [content, setContent] = useState('');

 const handleSubmit = async (e) => {
   e.preventDefault();
   if (!title.trim() || !content.trim()) return;

   const newPost = {
     title,
     content,
     likes: 0,
     comments: []
   };
```

```
    try {
      await axios.post('http://localhost:3001/posts', newPost);
      setTitle('');
      setContent('');
      onPostCreated();
    } catch (err) {
      console.error('Error creating post:', err);
    }
  };

  return (
    <form className="post-form" onSubmit={handleSubmit}>
      <input
        type="text"
        placeholder="Post Title"
        value={title}
        onChange={(e) => setTitle(e.target.value)}
        required
      />
      <textarea
        placeholder="Write your content..."
        value={content}
        onChange={(e) => setContent(e.target.value)}
        required
      ></textarea>
      <button type="submit">Create Post</button>
    </form>
  );
};

export default PostForm;
```

---

**src/components/Pagination.jsx**

```
import React from 'react';
import './Pagination.css';

const Pagination = ({ totalPosts, postsPerPage, currentPage, setCurrentPage }) => {
```

```
  const totalPages = Math.ceil(totalPosts / postsPerPage);

  const handlePrev = () => {
    if (currentPage > 1) setCurrentPage(currentPage - 1);
  };

  const handleNext = () => {
    if (currentPage < totalPages) setCurrentPage(currentPage + 1);
  };

  return (
    <div className="pagination">
      <button onClick={handlePrev} disabled={currentPage === 1}>⬅️ Prev</button>
      {[...Array(totalPages)].map((_, idx) => (
        <button
          key={idx}
          className={currentPage === idx + 1 ? 'active' : ''}
          onClick={() => setCurrentPage(idx + 1)}
        >
          {idx + 1}
        </button>
      ))}
      <button onClick={handleNext} disabled={currentPage === totalPages}>Next ➡️</button>
    </div>
  );
};

export default Pagination;
```

---

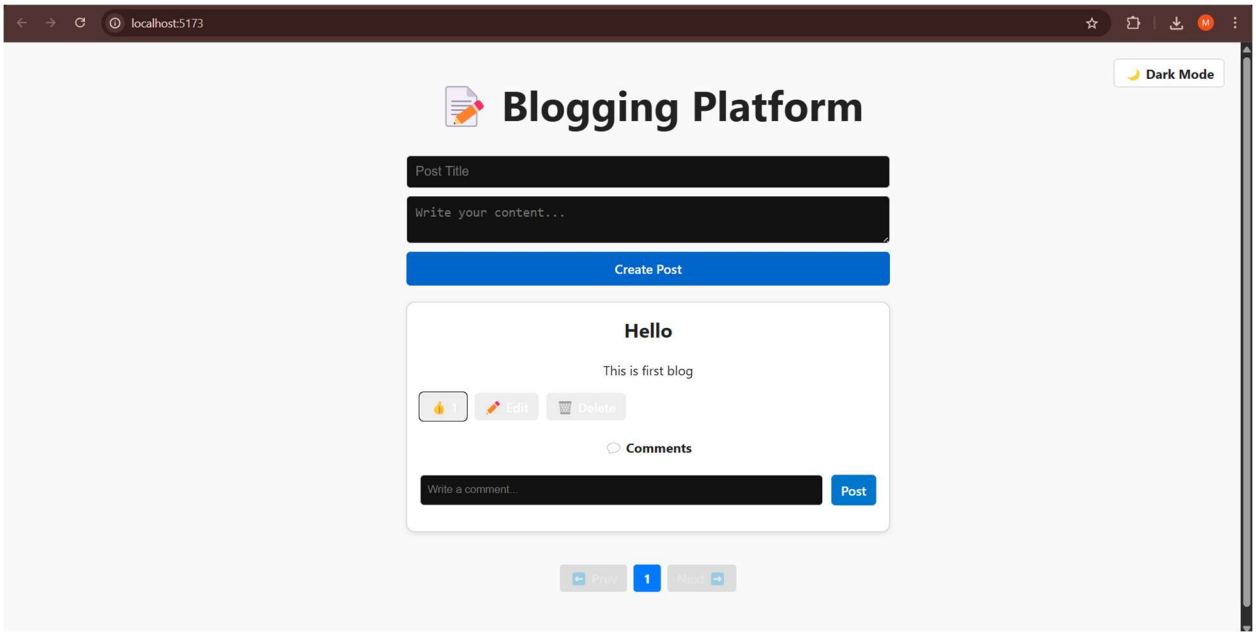
**src/components/CommentSection.jsx**

```
import React, { useEffect, useState } from 'react';
import axios from 'axios';
import './CommentSection.css';

const CommentSection = ({ postId }) => {
  const [comments, setComments] = useState([]);
  const [text, setText] = useState('');
```

```
const fetchComments = async () => {
  try {
    const response = await axios.get(`http://localhost:5000/posts/${postId}`);
    setComments(response.data.comments || []);
  } catch (err) {
    console.error('Error fetching comments:', err);
  }
};

useEffect(() => {
  fetchComments();
}, []);

const handleAddComment = async (e) => {
  e.preventDefault();
  if (!text.trim()) return;

  const newComment = { id: Date.now(), text, likes: 0 };
  const updatedComments = [...comments, newComment];

  try {
    await axios.patch(`http://localhost:5000/posts/${postId}`, {
      comments: updatedComments
    });
    setComments(updatedComments);
    setText('');
  } catch (err) {
    console.error('Error adding comment:', err);
  }
};

const handleLikeComment = async (id) => {
  const updatedComments = comments.map((c) =>
    c.id === id ? { ...c, likes: c.likes + 1 } : c
  );
  try {
    await axios.patch(`http://localhost:5000/posts/${postId}`, {
      comments: updatedComments
    });
```

```
    setComments(updatedComments);
  } catch (err) {
    console.error('Error liking comment:', err);
  }
};

const handleDeleteComment = async (id) => {
  const updatedComments = comments.filter((c) => c.id !== id);
  try {
    await axios.patch(`http://localhost:5000/posts/${postId}`, {
      comments: updatedComments
    });
    setComments(updatedComments);
  } catch (err) {
    console.error('Error deleting comment:', err);
  }
};

return (
  <div className="comment-section">
    <h4>💬 Comments</h4>
    <form onSubmit={handleAddComment} className="comment-form">
      <input
        type="text"
        placeholder="Write a comment..."
        value={text}
        onChange={(e) => setText(e.target.value)}
      />
      <button type="submit">Post</button>
    </form>
    <ul className="comment-list">
      {comments.map((comment) => (
        <li key={comment.id}>
          <span>{comment.text}</span>
          <div className="comment-actions">
            <button onClick={() => handleLikeComment(comment.id)}>👍
{comment.likes}</button>
            <button onClick={() => handleDeleteComment(comment.id)}>🗑️</button>
          </div>
        </li>
```

```
      ))}
    </ul>
  </div>
 );
};

export default CommentSection;
```



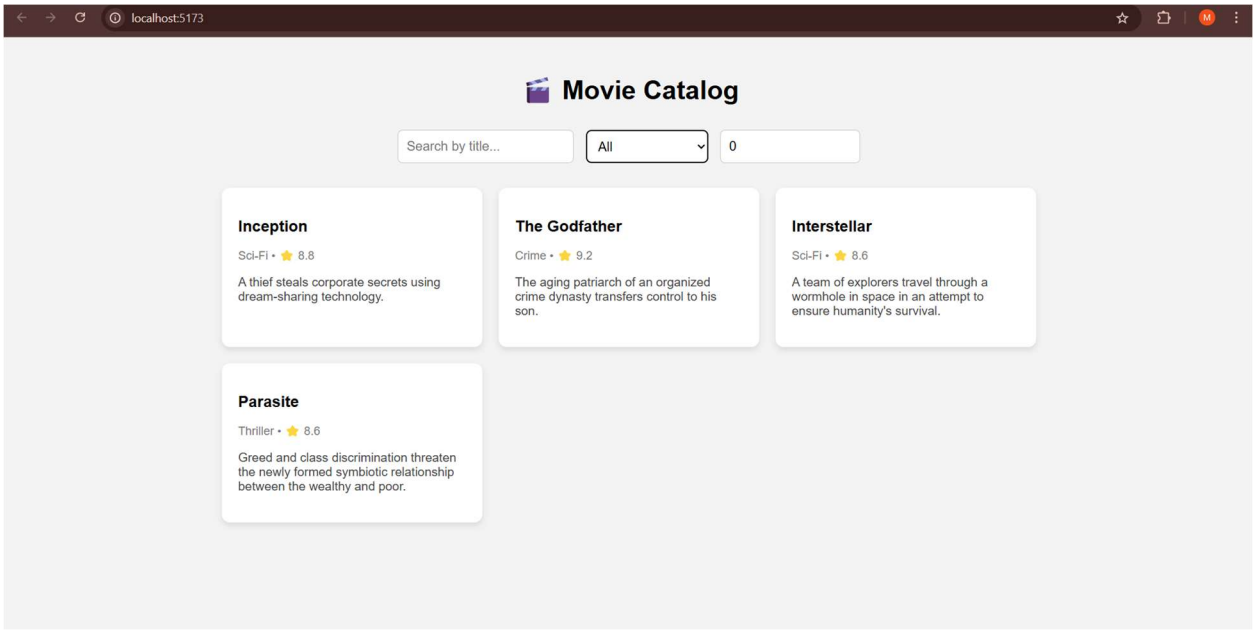

Q2) NoSQL-Based Movie Database
Scenario: A streaming service wants a database to store movies, ratings, and user reviews.
Implementation Task: Develop a MongoDB-based movie catalog with search and filtering capabilities.

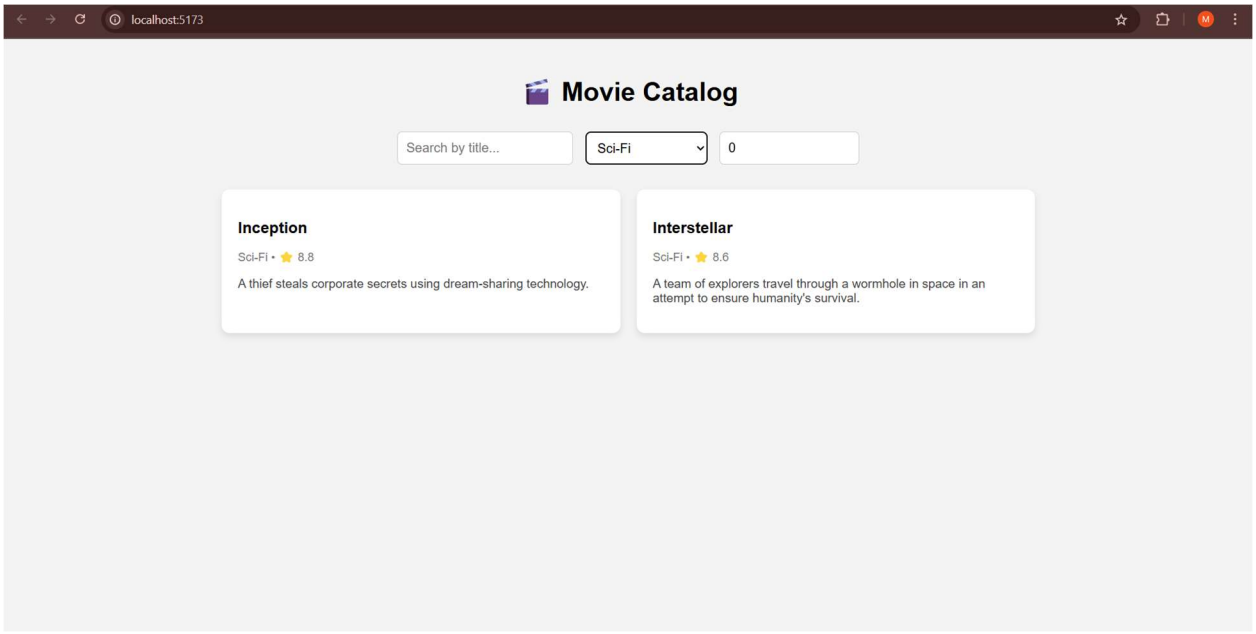
**Solution:**

```
import { useState } from "react";
import "./index.css"; // Import custom styles

function App() {
  const [search, setSearch] = useState("");
  const [genre, setGenre] = useState("All");
  const [minRating, setMinRating] = useState(0);
```

```
const genres = ["All", ...new Set(movies.map(m => m.genre))];

const filteredMovies = movies.filter(m =>
  m.title.toLowerCase().includes(search.toLowerCase()) &&
  (genre === "All" || m.genre === genre) &&
  m.rating >= minRating
);

return (
  <div className="container">
    <h1 className="title">🎬 Movie Catalog</h1>

    <div className="filters">
      <input
        type="text"
        placeholder="Search by title..."
        className="input"
        value={search}
        onChange={e => setSearch(e.target.value)}
      />
      <select
        className="input"
        value={genre}
        onChange={e => setGenre(e.target.value)}
      >
        {genres.map(g => (
          <option key={g}>{g}</option>
        ))}
      </select>
      <input
        type="number"
        min="0"
        max="10"
        step="0.1"
        className="input"
        placeholder="Min rating"
        value={minRating}
        onChange={e => setMinRating(Number(e.target.value))}
      />
```

```
    </div>

    <div className="movie-grid">
      {filteredMovies.map(movie => (
        <div key={movie.id} className="movie-card">
          <h2 className="movie-title">{movie.title}</h2>
                <p className="movie-info">{movie.genre} • ⭐ {movie.rating}</p>
          <p className="movie-desc">{movie.description}</p>
        </div>
      ))}
      {filteredMovies.length === 0 && (
        <p className="no-results">No movies found.</p>
      )}
    </div>
  </div>
 );
}

export default App;
```


localhost:5173
Movie Catalog
Search by title...
All
0
Inception
Sci-Fi • 8.8
A thief steals corporate secrets using dream-sharing technology.
The Godfather
Crime • 9.2
The aging patriarch of an organized crime dynasty transfers control to his son.
Interstellar
Sci-Fi • 8.6
A team of explorers travel through a wormhole in space in an attempt to ensure humanity's survival.
Parasite
Thriller • 8.6
Greed and class discrimination threaten the newly formed symbiotic relationship between the wealthy and poor.


localhost:5173
Movie Catalog
Search by title...
Sci-Fi
0
Inception
Sci-Fi • 8.8
A thief steals corporate secrets using dream-sharing technology.
Interstellar
Sci-Fi • 8.6
A team of explorers travel through a wormhole in space in an attempt to ensure humanity's survival.