```
import React, { useState, useEffect } from 'react';
import './Homepage.css';
import Headerhomepage from '../../common/headerhomepage';
import profileicon from '../../../assets/profileicon.png';
import mediaupload from '../../../assets/mediaupload.png';
import upvoteicon from '../../../assets/upvote.png';
import commenticon from '../../../assets/comment.png';
import Messagepop from '../../popups/messagingpop';
import PostCommentPopup from '../../popups/PostCommentPopup';
import AddPostModal from '../../popups/AddPostModal';
import { io } from 'socket.io-client';
import axiosInstance from '../../../services/axiosInstance';

const socket = io('http://localhost:3001');

const Homepage = () => {
  const [postsData, setPostsData] = useState([]);
  const [profileData, setProfileData] = useState({
    firstName: '',
    middleName: '',
    lastName: '',
    address: '',
  });

  const [newPostContent, setNewPostContent] = useState('');
  const [newPostImage, setNewPostImage] = useState(null);
```

```
const [isPopupOpen, setIsPopupOpen] = useState(false);

const formatTimeAgo = (timestamp) => {
  const now = new Date();
  const postDate = new Date(timestamp);
  const diffInMs = now - postDate;
  const diffInMinutes = Math.floor(diffInMs / 60000);

  if (diffInMinutes < 1) return 'Just now';
  if (diffInMinutes < 60) return `${diffInMinutes} minute${diffInMinutes > 1 ? 's' : ''} ago`;
  const diffInHours = Math.floor(diffInMinutes / 60);
  if (diffInHours < 24) return `${diffInHours} hour${diffInHours > 1 ? 's' : ''} ago`;
  const diffInDays = Math.floor(diffInHours / 24);
  if (diffInDays <= 2) return `${diffInDays} day${diffInDays > 1 ? 's' : ''} ago`;

  // Format the date for posts older than 2 days
  return postDate.toLocaleDateString(undefined, {
    year: 'numeric',
    month: 'short',
    day: 'numeric',
  });
};

const handleUpvote = async (postId) => {
  try {
    const response = await fetch(`http://localhost:3001/api/posts/${postId}/upvote`, {
```

```
      method: "POST",
      headers: {
        "Content-Type": "application/json",
        Authorization: `Bearer ${localStorage.getItem("token")}`,
      },
    });

    if (!response.ok) {
      throw new Error("Failed to toggle upvote");
    }

    const data = await response.json();

    if (data.success) {
      setPostsData((prevPosts) =>
        prevPosts.map((post) =>
          post._id === postId ? { ...post, upvotes: data.post.upvotes } : post
        )
      );
    }
  } catch (err) {
    console.error("Error toggling upvote:", err);
  }
};
```

```
// Fetch posts and initialize `showComments`
const fetchPostsData = async () => {
 try {
  const response = await fetch('http://localhost:3001/api/posts');
  const data = await response.json();

  const updatedPosts = data.map((post) => ({
   ...post,
   showComments: false,
  }));

  setPostsData(updatedPosts);
 } catch (error) {
  console.error('Error fetching posts:', error);
 }
};

// Listen for socket updates
useEffect(() => {
 fetchPostsData();

 socket.on('new_post', (post) => {
  setPostsData((prevPosts) => [{ ...post, showComments: false }, ...prevPosts]);
 });

 socket.on('receive_comment', ({ postId, comment }) => {
```

```
    setPostsData((prevPosts) =>
      prevPosts.map((post) =>
        post._id === postId
          ? { ...post, comments: [...post.comments, comment] }
          : post
      )
    );
  });

  return () => {
    socket.off('new_post');
    socket.off('receive_comment');
  };
}, []);

// Fetch profile data
useEffect(() => {
  const fetchProfileData = async () => {
    try {
      const response = await axiosInstance.get('/api/profile');
      if (response.data.success) {
        setProfileData(response.data.profile.profileDetails || {});
      }
    } catch (error) {
      console.error('Error fetching profile data:', error);
    }
```

```
  };

  fetchProfileData();
}, []);


const profileImageUrl = profileData.profileImg?.startsWith('/')
? `http://localhost:3001${profileData.profileImg}`
: profileData.profileImg || profileicon;

// Toggle comments for a specific post
const toggleComments = (postId) => {
  setPostsData((prevPosts) =>
    prevPosts.map((post) =>
      post._id === postId ? { ...post, showComments: !post.showComments } : post
    )
  );
};

const handleInputChange = (e) => {
  setNewPostContent(e.target.value);
};

const handleImageChange = (e) => {
  const file = e.target.files[0];
  if (file) {
```

```
    const reader = new FileReader();
    reader.onloadend = () => {
      setNewPostImage(reader.result);
    };
    reader.readAsDataURL(file);
  }
};

const handleClosePopup = () => {
  setIsPopupOpen(false);
  setNewPostContent('');
  setNewPostImage(null);
};

const handleAddPost = async () => {
  if (newPostContent.trim()) {
    try {
      const newPost = {
        profileImg: profileData.profileImg || profileicon,
        name: `${profileData.firstName} ${profileData.lastName}` .trim() || 'Student',
        content: newPostContent,
        postImg: newPostImage,
      };

      const response = await axiosInstance.post('/api/posts', newPost);
      if (response.data.success) {
```

```
      // The new post will be automatically added via the 'new_post' socket event
      handleClosePopup();
    } else {
      console.error('Failed to add post:', response.data.message);
    }
   } catch (err) {
    console.error('Error adding post:', err);
   }
  }
 };


 const renderPost = (post, index) => {
  const userId = 'user_id_from_auth'; // Replace this with actual user ID from
authentication
  const hasUpvoted = post.votedUsers.includes(userId); // Check if user has upvoted


  return (
   <div className="post" key={post._id || index}>
    <div className="toppostcontent">
     <img src={post.profileImg || profileicon} alt={post.name} />
     <div className="frompost">
      <h5>{post.name}</h5>
      <p>{formatTimeAgo(post.timestamp)}</p>
     </div>
    </div>
    <div className="postcontent">
```

```
        <p>{post.content}</p>
        {post.postImg && <img src={post.postImg} alt="Post" className="post-image" />}
      </div>
      <div className="downpostcontent">
        <button onClick={() => handleUpvote(post._id)}
          style={{ backgroundColor: hasUpvoted ? 'lightblue' : 'white' }}>
          <img src={upvoteicon} alt="Upvote" /> {post.upvotes}
        </button>
        <button onClick={() => toggleComments(post._id)}>
          <img src={commenticon} alt="Comment" /> {post.comments.length}
        </button>
      </div>
      {post.showComments && (
        <PostCommentPopup
          post={post}
          handleCommentSubmit={() => console.log('Comment submit')} // Update with actual implementation
          toggleComments={toggleComments}
        />
      )}
    </div>
  );
};

  return (
    <div className="Homepage-container">
```

```
    <Headerhomepage />
    <div className="content-container">
     <aside className="sidebar">
      <div className="profile">
       <img src={profileImageUrl} alt="Profile Icon" />
       <h2>{` ${profileData.firstName} ${profileData.middleName || ''}
${profileData.lastName}` .trim()}</h2>
       <p>Student at Technological University of the Philippines</p>
       <p>{profileData.address || 'Not Available'}</p>
      </div>
     </aside>
     <main className="feed">
      <div className="post-input" onClick={() => setIsPopupOpen(true)}>
       <div className='postinputimg-container' >
        <img src={profileImageUrl} alt="Profile Icon" />
       </div>
       <div className="subpost-input">
        <input type="text" placeholder="Start a post" readOnly />
        <button className="media-btn">
         <img src={mediaupload} alt="Media Upload" /> Media
        </button>
       </div>
      </div>
      {postsData.map(renderPost)}
     </main>
    </div>
```

```
    <Messagepop />
    {isPopupOpen && (
      <AddPostModal
        newPostContent={newPostContent}
        handleInputChange={handleInputChange}
        newPostImage={newPostImage}
        handleImageChange={handleImageChange}
        handleClosePopup={handleClosePopup}
        handleAddPost={handleAddPost}
      />
    )}
   </div>
  );
};

export default Homepage;
```