TASK-27

27. Compose React Components:

- *Use composition to **display** different chats in a single `ChatApp` interface.*

---

1. Folder Structure:

```
src/
 ├── components/
 │    ├── ChatApp.jsx
 │    ├── ChatHeader.jsx
 │    ├── MessageList.jsx
 │    ├── MessageInput.jsx
 │    ├── ChatRoom.jsx
 ├── App.js
└── index.js
```

---

2. Creating the Components:

2.1. ChatApp.jsx

- **This component will manage multiple ChatRoom components, each representing a different chat.**

```
import React, { useState } from "react";
import ChatHeader from "./ChatHeader";
import ChatRoom from "./ChatRoom";
import "./ChatApp.css";

const ChatApp = () => {
 const [activeChat, setActiveChat] = useState("general");

 const chats = [
  { id: "general", name: "General Chat" },
  { id: "random", name: "Random Chat" },
  { id: "tech", name: "Tech Chat" }
 ];

 const handleChatChange = (chatId) => {
  setActiveChat(chatId);
 };

 return (
```

```
        <div className="chat-app">
          <ChatHeader chats={chats} activeChat={activeChat}
      onChatChange={handleChatChange} />
          <ChatRoom chatId={activeChat} />
        </div>
      );
    };

export default ChatApp;
```

2.2. ChatHeader.jsx:

- **This component displays the list of available chats and allows switching between them.**

```
import React from "react";

const ChatHeader = ({ chats, activeChat, onChatChange }) => {
 return (
  <div className="chat-header">
   <h1>Chat Application</h1>
   <div className="chat-list">
    {chats.map((chat) => (
     <button
      key={chat.id}
      className={chat.id === activeChat ? "active" : ""}
      onClick={() => onChatChange(chat.id)}
     >
      {chat.name}
     </button>
    ))}
   </div>
  </div>
 );
};

export default ChatHeader;
```

2.3. ChatRoom.jsx:

- **This component represents a single chat room. Each ChatRoom will have its own list of messages and an input field.**

```
import React, { useState } from "react";
import MessageList from "./MessageList";
import MessageInput from "./MessageInput";

const ChatRoom = ({ chatId }) => {
 const [messages, setMessages] = useState([]);

 const handleSendMessage = (message) => {
  setMessages([...messages, { id: messages.length, text: message }]);
 };

 return (
  <div className="chat-room">
   <h2>{chatId} Chat Room</h2>
   <MessageList messages={messages} />
   <MessageInput onSendMessage={handleSendMessage} />
  </div>
 );
};

export default ChatRoom;
```

2.4. MessageList.jsx:

- **This component displays the messages in a chat room.**

```
import React from "react";

const MessageList = ({ messages }) => {
 return (
  <div className="message-list">
   {messages.map((message) => (
    <div key={message.id} className="message">
     {message.text}
    </div>
   ))}
  </div>
 );
};

export default MessageList;
```

2.5. MessageInput.jsx:

– **This component handles the input field and sends messages.**

```
import React, { useState } from "react";

const MessageInput = ({ onSendMessage }) => {
 const [message, setMessage] = useState("");

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

 const handleSendClick = () => {
  if (message.trim()) {
   onSendMessage(message);
   setMessage("");
  }
 };

 return (
  <div className="message-input">
   <input
    type="text"
    value={message}
    onChange={handleInputChange}
    placeholder="Type a message"
   />
   <button onClick={handleSendClick}>Send</button>
  </div>
 );
};

export default MessageInput;
```

---

3. CSS for Styling (Optional):

– **You can add some basic CSS in your ChatApp.css to style the chat application.**

```
.chat-app {
 display: flex;
 flex-direction: column;
 height: 100vh;
 max-width: 600px;
 margin: 0 auto;
 border: 1px solid #ccc;
 border-radius: 8px;
```

```
  overflow: hidden;
}

.chat-header {
  background-color: #007bff;
  color: white;
  text-align: center;
  padding: 10px;
}

.chat-list {
  display: flex;
  justify-content: space-around;
  margin-top: 10px;
}

.chat-list button {
  background-color: #fff;
  color: #007bff;
  padding: 5px 10px;
  border: 1px solid #007bff;
  border-radius: 5px;
  cursor: pointer;
}

.chat-list button.active {
  background-color: #007bff;
  color: white;
}

.chat-room {
  padding: 20px;
  background-color: #f9f9f9;
  flex: 1;
  overflow-y: auto;
}

.message-list {
  margin-bottom: 20px;
}

.message {
  background-color: #007bff;
  color: white;
  padding: 8px;
  margin: 5px 0;
  border-radius: 5px;
```

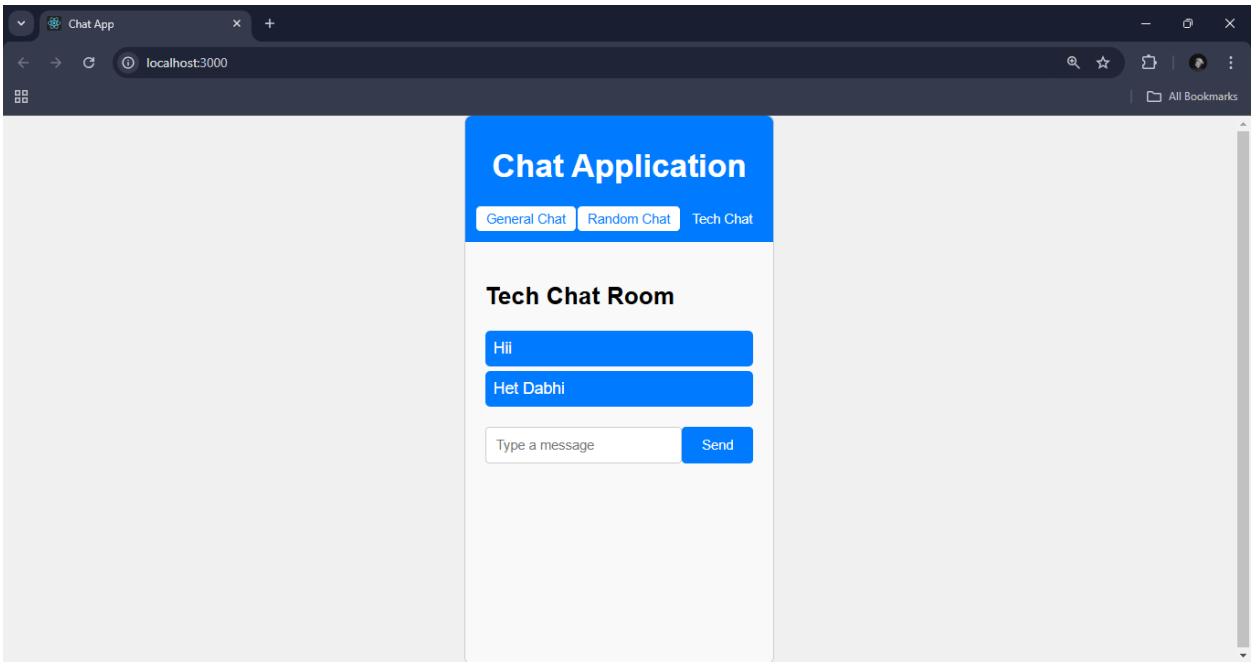
```
        }

        .message-input {
          display: flex;
          justify-content: space-between;
        }

        .message-input input {
          flex: 1;
          padding: 10px;
          border: 1px solid #ccc;
          border-radius: 4px;
        }

        .message-input button {
          background-color: #007bff;
          color: white;
          padding: 10px 20px;
          border: none;
          border-radius: 4px;
          cursor: pointer;
        }

        .message-input button:hover {
          background-color: #0056b3;
}
```

---

4. Run the Application:

```
npm start
```

Output:


Chat App
localhost:3000
All Bookmarks
Chat Application
General Chat
Random Chat
Tech Chat
Tech Chat Room
Hii
Het Dabhi
Type a message
Send