# EXPERIMENT NO: 3

**Aim:** Manage complex state with Redux or Context API

**Theory:**

1. **Redux Overview:** Redux is a predictable state management library for React, centralizing application state in a single, immutable store.
2. **Single Source of Truth:** The entire application state is stored in one JavaScript object, ensuring consistency and accessibility.
3. **Unidirectional Data Flow:** Actions, which are plain JavaScript objects, describe state changes and are dispatched to the store.
4. **Reducers:** Pure functions that process actions to compute the new state, ensuring predictable and traceable updates.
5. **Middleware for Async Operations:** Tools like Redux Thunk or Redux Saga handle asynchronous tasks, such as API calls, within Redux.
6. **API Integration in React:** APIs are typically accessed using Fetch or Axios, with requests managed in useEffect hooks (functional components) or lifecycle methods (class components).
7. **State Management with APIs:** Fetched API data can be stored in component state or the Redux store for global access, enabling dynamic UI updates.
8. **Async Flow in Redux:** A common pattern involves dispatching actions for API request start, success, or failure, allowing Redux to update state and trigger React component re-renders.

**30% Extra** :

1. **WebRTC for Real-Time Communication:** WebRTC (Web Real-Time Communication) is the cornerstone technology for enabling video chatting on websites. It facilitates peer-to-peer audio, video, and data sharing between browsers without requiring plugins. In a Steam-like implementation, WebRTC handles the real-time video and voice streams, ensuring low-latency communication between users. It uses protocols like RTP for media transport and ICE for establishing connections through NATs and firewalls.
2. **Signaling for Connection Setup:** WebRTC requires a signaling mechanism to coordinate communication between peers, such as exchanging session descriptions (SDP) and ICE candidates. In a Steam-inspired system, a signaling server (often built with technologies like Node.js and WebSocket) facilitates this by allowing users to discover and connect with each other. This server manages the initial handshake before the peer-to-peer connection is established.
3. **Integration with Steamworks API:** Steam provides the Steamworks API, which includes tools for enabling voice chats within games or applications. For a website, you can adapt similar functionality by integrating Steam's authentication and user management systems to identify users and initiate video chat sessions. This ensures that only authenticated Steam users can join chats, leveraging Steam's friend lists and group chat features
4. **User Interface and Overlay:** Steam's video chat functionality is often integrated into its in-game overlay, allowing seamless interaction without leaving the application. On a website, this can be replicated using a modular UI component (e.g., a React component) that overlays the video chat window. Users can pin or resize the chat window, similar to Steam's customizable overlay introduced in 2023, enhancing the user experience during multitasking.

5. **Handling Asynchronous Operations:** Video chat involves asynchronous operations, such as initiating streams, handling network changes, or managing connection drops. In a React-based website, these can be managed using state management (e.g., Redux) to track the status of video calls (e.g., connecting, active, or failed). Actions can be dispatched to handle events like stream initialization or errors, ensuring the UI reflects the current state.
6. **Scalability and Server Load:** Unlike Steam's voice chat, which uses a WebRTC-based backend with traffic routed through Steam servers for security and IP privacy, a website implementation may face scalability challenges. To manage server load, you can use a combination of peer-to-peer WebRTC connections and a lightweight signaling server, or leverage third-party services like Twilio or Vonage for video chat infrastructure.
7. **Security and Privacy:** Security is critical in video chat implementations. Steam's voice chat encrypts traffic and masks IP addresses to prevent network attacks. Similarly, a website must implement HTTPS, encrypt WebRTC streams, and ensure user authentication (e.g., via Steam login) to protect privacy. Additionally, handling permissions for camera and microphone access securely enhances user trust

## Source Code:

### 1) Lib/Api.js

```
1   import { axiosInstance } from "./axios";
2
3   export const signup = async (signupData) => {
4     const response = await axiosInstance.post("/auth/signup", signupData);
5     return response.data;
6   };
7
8   export const login = async (loginData) => {
9     const response = await axiosInstance.post("/auth/login", loginData);
10    return response.data;
11  };
12  export const logout = async () => {
13    const response = await axiosInstance.post("/auth/logout");
14    return response.data;
15  };
16
17  export const getAuthUser = async () => {
18    try {
19      const res = await axiosInstance.get("/auth/me");
20      return res.data;
21    } catch (error) {
22      console.log("Error in getAuthUser:", error);
23      return null;
24    }
25  };
26
27  export const completeOnboarding = async (userData) => {
28    const response = await axiosInstance.post("/auth/onboarding", userData);
29    return response.data;
30  };
31
32  export async function getUserFriends() {
33    const response = await axiosInstance.get("/users/friends");
34    return response.data;
35  }
36
```

```
37  export async function getRecommendedUsers() {
38    const response = await axiosInstance.get("/users");
39    return response.data;
40  }
41
42  export async function getOutgoingFriendReqs() {
43    const response = await axiosInstance.get("/users/outgoing-friend-requests");
44    return response.data;
45  }
46
47  export async function sendFriendRequest(userId) {
48    const response = await axiosInstance.post(`/users/friend-request/${userId}`);
49    return response.data;
50  }
51
52  export async function getFriendRequests() {
53    const response = await axiosInstance.get("/users/friend-requests");
54    return response.data;
55  }
56
57  export async function acceptFriendRequest(requestId) {
58    const response = await axiosInstance.put(`/users/friend-request/${requestId}/accept`);
59    return response.data;
60  }
61
62  export async function getStreamToken() {
63    const response = await axiosInstance.get("/chat/token");
64    return response.data;
65  }
```

**2) HomePage.jsx:**

```jsx
import { useMutation, useQuery, useQueryClient } from "@tanstack/react-query";
import { useEffect, useState } from "react";
import {
  getOutgoingFriendReqs,
  getRecommendedUsers,
  getUserFriends,
  sendFriendRequest,
} from "../lib/api";
import { Link } from "react-router";
import { CheckCircleIcon, MapPinIcon, UserPlusIcon, UsersIcon } from "lucide-react";

import { capitialize } from "../lib/utils";

import FriendCard, { getLanguageFlag } from "../components/FriendCard";
import NoFriendsFound from "../components/NoFriendsFound";

const HomePage = () => {
  const queryClient = useQueryClient();
  const [outgoingRequestsIds, setOutgoingRequestsIds] = useState(new Set());

  const { data: friends = [], isLoading: loadingFriends } = useQuery({
    queryKey: ["friends"],
    queryFn: getUserFriends,
  });

  const { data: recommendedUsers = [], isLoading: loadingUsers } = useQuery({
    queryKey: ["users"],
    queryFn: getRecommendedUsers,
  });

  const { data: outgoingFriendReqs } = useQuery({
    queryKey: ["outgoingFriendReqs"],
    queryFn: getOutgoingFriendReqs,
  });

  const { mutate: sendRequestMutation, isPending } = useMutation({
    mutationFn: sendFriendRequest,
    onSuccess: () => queryClient.invalidateQueries({ queryKey: ["outgoingFriendReqs"] }),
  });

  useEffect(() => {
    const outgoingIds = new Set();
    if (outgoingFriendReqs && outgoingFriendReqs.length > 0) {
      outgoingFriendReqs.forEach((req) => {
        outgoingIds.add(req.recipient._id);
```

```
220         });
221         setOutgoingRequestsIds(outgoingIds);
222       }
223     }, [outgoingFriendReqs]);
224
225     return (
226       <div className="min-h-screen bg-base-200 p-4 sm:p-6 lg:p-8">
227         <div className="container mx-auto space-y-10">
228           <div className="flex flex-col sm:flex-row items-start sm:items-center justify-between gap-4">
229             <h2 className="text-2xl sm:text-3xl font-bold tracking-tight">Your Friends</h2>
230             <Link to="/notifications" className="btn btn-outline btn-sm">
231               <UsersIcon className="mr-2 size-4" />
232               Friend Requests
233             </Link>
234           </div>
235
236           {loadingFriends ? (
237             <div className="flex justify-center py-12">
238               <span className="loading loading-spinner loading-lg" />
239             </div>
240           ) : friends.length === 0 ? (
241             <NoFriendsFound />
242           ) : (
243             <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4 gap-4">
244               {friends.map((friend) => (
245                 <FriendCard key={friend._id} friend={friend} />
246               ))}
247             </div>
248           )}
249
250           <section className="pb-12">
251             <div className="mb-6 sm:mb-8">
252               <div className="flex flex-col sm:flex-row items-start sm:items-center justify-between gap-4">
253                 <div>
254                   <h2 className="text-2xl sm:text-3xl font-bold tracking-tight">Meet New Learners</h2>
255                   <p className="opacity-70">
256                     Discover perfect language exchange partners based on your profile
257                   </p>
258                 </div>
259               </div>
260             </div>
```

```jsx
{loadingUsers ? (
  <div className="flex justify-center py-12">
    <span className="loading loading-spinner loading-lg" />
  </div>
) : recommendedUsers.length === 0 ? (
  <div className="card bg-base-200 p-6 text-center">
    <h3 className="font-semibold text-lg mb-2">No recommendations available</h3>
    <p className="text-base-content opacity-70">
      Check back later for new language partners!
    </p>
  </div>
) : (
  <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
    {recommendedUsers.map((user) => {
      const hasRequestBeenSent = outgoingRequestsIds.has(user._id);

      return (
        <div
          key={user._id}
          className="card bg-base-200 hover:shadow-lg transition-all duration-300"
        >
          <div className="card-body p-5 space-y-4">
            <div className="flex items-center gap-3">
              <div className="avatar size-16 rounded-full">
                <img src={user.profilePic} alt={user.fullName} />
              </div>

              <div>
                <h3 className="font-semibold text-lg">{user.fullName}</h3>
                {user.location && (
                  <div className="flex items-center text-xs opacity-70 mt-1">
                    <MapPinIcon className="size-3 mr-1" />
                    {user.location}
                  </div>
                )}
              </div>
            </div>

            {/* Languages with flags */}
            <div className="flex flex-wrap gap-1.5">
              <span className="badge badge-secondary">
                {getLanguageFlag(user.nativeLanguage)}
                Native: {capitialize(user.nativeLanguage)}
              </span>
```

```
                    </span>
                    <span className="badge badge-outline">
                      {getLanguageFlag(user.learningLanguage)}
                      Learning: {capitialize(user.learningLanguage)}
                    </span>
                  </div>

                  {user.bio && <p className="text-sm opacity-70">{user.bio}</p>}

                  {/* Action button */}
                  <button
                    className={`btn w-full mt-2 ${
                      hasRequestBeenSent ? "btn-disabled" : "■btn-primary"
                    } `}
                    onClick={() => sendRequestMutation(user._id)}
                    disabled={hasRequestBeenSent || isPending}
                  >
                    {hasRequestBeenSent ? (
                      <>
                        <CheckCircleIcon className="size-4 mr-2" />
                        Request Sent
                      </>
                    ) : (
                      <>
                        <UserPlusIcon className="size-4 mr-2" />
                        Send Friend Request
                      </>
                    )}
                  </button>
                </div>
              </div>
            );
          })}
        </div>
      )}
    </section>
  </div>
</div>
);
};

export default HomePage;
```

**3) ChatPage.jsx:**

```jsx
import { useEffect, useState } from "react";
import { useParams } from "react-router";
import useAuthUser from "../hooks/useAuthUser";
import { useQuery } from "@tanstack/react-query";
import { getStreamToken } from "../lib/api";

import {
  Channel,
  ChannelHeader,
  Chat,
  MessageInput,
  MessageList,
  Thread,
  Window,
} from "stream-chat-react";
import { StreamChat } from "stream-chat";
import toast from "react-hot-toast";

import ChatLoader from "../components/ChatLoader";
import CallButton from "../components/CallButton";

const STREAM_API_KEY = import.meta.env.VITE_STREAM_API_KEY;

const ChatPage = () => {
  const { id: targetUserId } = useParams();

  const [chatClient, setChatClient] = useState(null);
  const [channel, setChannel] = useState(null);
  const [loading, setLoading] = useState(true);

  const { authUser } = useAuthUser();

  const { data: tokenData } = useQuery({
    queryKey: ["streamToken"],
    queryFn: getStreamToken,
    enabled: !!authUser, // this will run only when authUser is available
  });

  useEffect(() => {
    const initChat = async () => {
      if (!tokenData?.token || !authUser) return;

      try {
        console.log("Initializing stream chat client...");

```

```
        const client = StreamChat.getInstance(STREAM_API_KEY);

        await client.connectUser(
          {
            id: authUser._id,
            name: authUser.fullName,
            image: authUser.profilePic,
          },
          tokenData.token
        );

        //
        const channelId = [authUser._id, targetUserId].sort().join("-");

        // you and me
        // if i start the chat => channelId: [myId, yourId]
        // if you start the chat => channelId: [yourId, myId]  => [myId,yourId]

        const currChannel = client.channel("messaging", channelId, {
          members: [authUser._id, targetUserId],
        });

        await currChannel.watch();

        setChatClient(client);
        setChannel(currChannel);
      } catch (error) {
        console.error("Error initializing chat:", error);
        toast.error("Could not connect to chat. Please try again.");
      } finally {
        setLoading(false);
      }
    };

    initChat();
  }, [tokenData, authUser, targetUserId]);

  const handleVideoCall = () => {
    if (channel) {
      const callUrl = `${window.location.origin}/call/${channel.id}`;

```

```
87          channel.sendMessage({
88            text: `I've started a video call. Join me here: ${callUrl}`,
89          });
90
91          toast.success("Video call link sent successfully!");
92        }
93      };
94
95      if (loading || !chatClient || !channel) return <ChatLoader />;
96
97      return (
98        <div className="h-[93vh]">
99          <Chat client={chatClient}>
100           <Channel channel={channel}>
101             <div className="w-full relative">
102               <CallButton handleVideoCall={handleVideoCall} />
103               <Window>
104                 <ChannelHeader />
105                 <MessageList />
106                 <MessageInput focus />
107               </Window>
108             </div>
109             <Thread />
110           </Channel>
111         </Chat>
112       </div>
113     );
114   };
115   export default ChatPage;
```

**4) Notification.jsx:**

```jsx
import { useMutation, useQuery, useQueryClient } from "@tanstack/react-query";
import { acceptFriendRequest, getFriendRequests } from "../lib/api";
import { BellIcon, ClockIcon, MessageSquareIcon, UserCheckIcon } from "lucide-react";
import NoNotificationsFound from "../components/NoNotificationsFound";

const NotificationsPage = () => {
  const queryClient = useQueryClient();

  const { data: friendRequests, isLoading } = useQuery({
    queryKey: ["friendRequests"],
    queryFn: getFriendRequests,
  });

  const { mutate: acceptRequestMutation, isPending } = useMutation({
    mutationFn: acceptFriendRequest,
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: ["friendRequests"] });
      queryClient.invalidateQueries({ queryKey: ["friends"] });
    },
  });

  const incomingRequests = friendRequests?.incomingReqs || [];
  const acceptedRequests = friendRequests?.acceptedReqs || [];

  return (
    <div className="p-4 sm:p-6 lg:p-8">
      <div className="container mx-auto max-w-4xl space-y-8">
        <h1 className="text-2xl sm:text-3xl font-bold tracking-tight mb-6">Notifications</h1>

        {isLoading ? (
          <div className="flex justify-center py-12">
            <span className="loading loading-spinner loading-lg"></span>
          </div>
        ) : (
          <>
            {incomingRequests.length > 0 && (
              <section className="space-y-4">
                <h2 className="text-xl font-semibold flex items-center gap-2">
                  <UserCheckIcon className="h-5 w-5 text-primary" />
                  Friend Requests
                  <span className="badge badge-primary ml-2">{incomingRequests.length}</span>
                </h2>

                <div className="space-y-3">
                  {incomingRequests.map((request) => (
```

```jsx
            {incomingRequests.map((request) => (
              <div
                key={request._id}
                className="card bg-base-200 shadow-sm hover:shadow-md transition-shadow"
              >
                <div className="card-body p-4">
                  <div className="flex items-center justify-between">
                    <div className="flex items-center gap-3">
                      <div className="avatar w-14 h-14 rounded-full bg-base-300">
                        <img src={request.sender.profilePic} alt={request.sender.fullName} />
                      </div>
                      <div>
                        <h3 className="font-semibold">{request.sender.fullName}</h3>
                        <div className="flex flex-wrap gap-1.5 mt-1">
                          <span className="badge badge-secondary badge-sm">
                            Native: {request.sender.nativeLanguage}
                          </span>
                          <span className="badge badge-outline badge-sm">
                            Learning: {request.sender.learningLanguage}
                          </span>
                        </div>
                      </div>
                    </div>

                    <button
                      className="btn ■btn-primary btn-sm"
                      onClick={() => acceptRequestMutation(request._id)}
                      disabled={isPending}
                    >
                      Accept
                    </button>
                  </div>
                </div>
              </div>
            ))}
          </div>
        </section>
      )}

      {/* ACCEPTED REQS NOTIFICATONS */}
      {acceptedRequests.length > 0 && (
        <section className="space-y-4">
          <h2 className="text-xl font-semibold flex items-center gap-2">
            <BellIcon className="h-5 w-5 text-success" />
```

```
                <BellIcon className="h-5 w-5 text-success" />
                New Connections
              </h2>

              <div className="space-y-3">
                {acceptedRequests.map((notification) => (
                  <div key={notification._id} className="card bg-base-200 shadow-sm">
                    <div className="card-body p-4">
                      <div className="flex items-start gap-3">
                        <div className="avatar mt-1 size-10 rounded-full">
                          <img
                            src={notification.recipient.profilePic}
                            alt={notification.recipient.fullName}
                          />
                        </div>
                        <div className="flex-1">
                          <h3 className="font-semibold">{notification.recipient.fullName}</h3>
                          <p className="text-sm my-1">
                            {notification.recipient.fullName} accepted your friend request
                          </p>
                          <p className="text-xs flex items-center opacity-70">
                            <ClockIcon className="h-3 w-3 mr-1" />
                            Recently
                          </p>
                        </div>
                        <div className="badge badge-success">
                          <MessageSquareIcon className="h-3 w-3 mr-1" />
                          New Friend
                        </div>
                      </div>
                    </div>
                  </div>
                ))}
              </div>
            </section>
          )}

          {incomingRequests.length === 0 && acceptedRequests.length === 0 && (
            <NoNotificationsFound />
          )}
        </>
      )}
    </div>
  </div>
```

**5) LoginPage.jsx:**

```jsx
import { useState } from "react";
import { ShipWheelIcon } from "lucide-react";
import { Link } from "react-router";
import useLogin from "../hooks/useLogin";

const LoginPage = () => {
  const [loginData, setLoginData] = useState({
    email: "",
    password: "",
  });

  const { isPending, error, loginMutation } = useLogin();

  const handleLogin = (e) => {
    e.preventDefault();
    loginMutation(loginData);
  };

  return (
    <div
      className="h-screen flex items-center justify-center p-4 sm:p-6 md:p-8"
      data-theme="forest"
    >
      <div className="border border-primary/25 flex flex-col lg:flex-row w-full max-w-5xl mx-auto bg-base-100 rounded-xl shadow-lg overflow-hidden">
        {/* LOGIN FORM SECTION */}
        <div className="w-full lg:w-1/2 p-4 sm:p-8 flex flex-col">
          {/* LOGO */}
          <div className="mb-4 flex items-center justify-start gap-2">
            <ShipWheelIcon className="size-9 text-primary" />
            <span className="text-3xl font-bold font-mono bg-clip-text text-transparent bg-gradient-to-r from-primary to-secondary  tracking-wider">
              Streamify
            </span>
          </div>

          {/* ERROR MESSAGE DISPLAY */}
          {error && (
            <div className="alert alert-error mb-4">
              <span>{error.response.data.message}</span>
            </div>
          )}

          <div className="w-full">
            <form onSubmit={handleLogin}>
              <div className="space-y-4">
                <div>
```

```jsx
                <div className="form-control w-full space-y-2">
                  <label className="label">
                    <span className="label-text">Password</span>
                  </label>
                  <input
                    type="password"
                    placeholder="••••••••"
                    className="input input-bordered w-full"
                    value={loginData.password}
                    onChange={(e) => setLoginData({ ...loginData, password: e.target.value })}
                    required
                  />
                </div>

                <button type="submit" className="btn ■btn-primary w-full" disabled={isPending}>
                  {isPending ? (
                    <>
                      <span className="loading loading-spinner loading-xs"></span>
                      Signing in...
                    </>
                  ) : (
                    "Sign In"
                  )}
                </button>

                <div className="text-center mt-4">
                  <p className="text-sm">
                    Don't have an account?{" "}
                    <Link to="/signup" className="text-primary hover:underline">
                      Create one
                    </Link>
                  </p>
                </div>
              </div>
            </div>
          </form>
        </div>
      </div>

      {/* IMAGE SECTION */}
      <div className="hidden lg:flex w-full lg:w-1/2 bg-primary/10 items-center justify-center">
        <div className="max-w-md p-8">
          {/* Illustration */}
          <div className="relative aspect-square max-w-sm mx-auto">
```

```jsx
                    <div className="text-center mt-4">
                      <p className="text-sm">
                        Don't have an account?{" "}
                        <Link to="/signup" className="text-primary hover:underline">
                          Create one
                        </Link>
                      </p>
                    </div>
                  </div>
                </div>
              </form>
            </div>
          </div>

          {/* IMAGE SECTION */}
          <div className="hidden lg:flex w-full lg:w-1/2 bg-primary/10 items-center justify-center">
            <div className="max-w-md p-8">
              {/* Illustration */}
              <div className="relative aspect-square max-w-sm mx-auto">
                <img src="/i.png" alt="Language connection illustration" className="w-full h-full" />
              </div>

              <div className="text-center space-y-3 mt-6">
                <h2 className="text-xl font-semibold">Connect with language partners worldwide</h2>
                <p className="opacity-70">
                  Practice conversations, make friends, and improve your language skills together
                </p>
              </div>
            </div>
          </div>
        </div>
      </div>
    );
  };
  export default LoginPage;
```

## 6) CallPage.jsx:

```jsx
import { useEffect, useState } from "react";
import { useNavigate, useParams } from "react-router";
import useAuthUser from "../hooks/useAuthUser";
import { useQuery } from "@tanstack/react-query";
import { getStreamToken } from "../lib/api";

import {
  StreamVideo,
  StreamVideoClient,
  StreamCall,
  CallControls,
  SpeakerLayout,
  StreamTheme,
  CallingState,
  useCallStateHooks,
} from "@stream-io/video-react-sdk";

import "@stream-io/video-react-sdk/dist/css/styles.css";
import toast from "react-hot-toast";
import PageLoader from "../components/PageLoader";

const STREAM_API_KEY = import.meta.env.VITE_STREAM_API_KEY;

const CallPage = () => {
  const { id: callId } = useParams();
  const [client, setClient] = useState(null);
  const [call, setCall] = useState(null);
  const [isConnecting, setIsConnecting] = useState(true);

  const { authUser, isLoading } = useAuthUser();

  const { data: tokenData } = useQuery({
    queryKey: ["streamToken"],
    queryFn: getStreamToken,
    enabled: !!authUser,
  });

  useEffect(() => {
    const initCall = async () => {
      if (!tokenData.token || !authUser || !callId) return;

      try {
        console.log("Initializing Stream video client...");

        const user = {
```

```
45            const user = {
46              id: authUser._id,
47              name: authUser.fullName,
48              image: authUser.profilePic,
49            };
50
51            const videoClient = new StreamVideoClient({
52              apiKey: STREAM_API_KEY,
53              user,
54              token: tokenData.token,
55            });
56
57            const callInstance = videoClient.call("default", callId);
58
59            await callInstance.join({ create: true });
60
61            console.log("Joined call successfully");
62
63            setClient(videoClient);
64            setCall(callInstance);
65          } catch (error) {
66            console.error("Error joining call:", error);
67            toast.error("Could not join the call. Please try again.");
68          } finally {
69            setIsConnecting(false);
70          }
71        };
72
73        initCall();
74      }, [tokenData, authUser, callId]);
75
76      if (isLoading || isConnecting) return <PageLoader />;
77
78      return (
79        <div className="h-screen flex flex-col items-center justify-center">
80          <div className="relative">
81            {client && call ? (
82              <StreamVideo client={client}>
83                <StreamCall call={call}>
84                  <CallContent />
85                </StreamCall>
86              </StreamVideo>
```

```
            ) : (
              <div className="flex items-center justify-center h-full">
                <p>Could not initialize call. Please refresh or try again later.</p>
              </div>
            )}
          </div>
        </div>
      );
    };

    const CallContent = () => {
      const { useCallCallingState } = useCallStateHooks();
      const callingState = useCallCallingState();

      const navigate = useNavigate();

      if (callingState === CallingState.LEFT) return navigate("/");

      return (
        <StreamTheme>
          <SpeakerLayout />
          <CallControls />
        </StreamTheme>
      );
    };

    export default CallPage;
```

## Output:

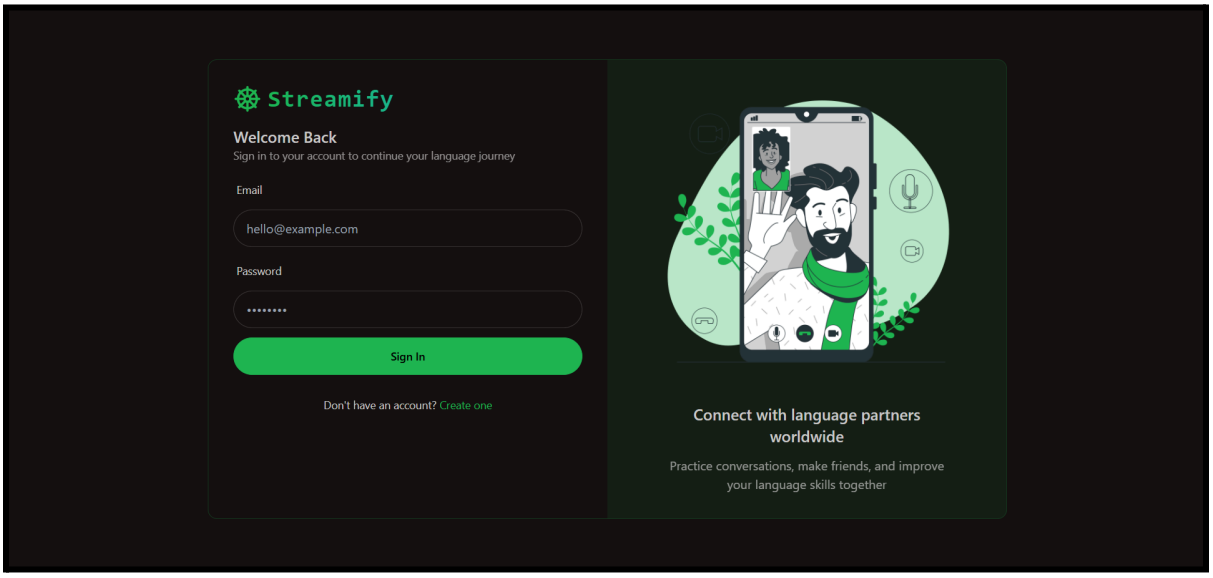


**Figure 1 : Login Page**

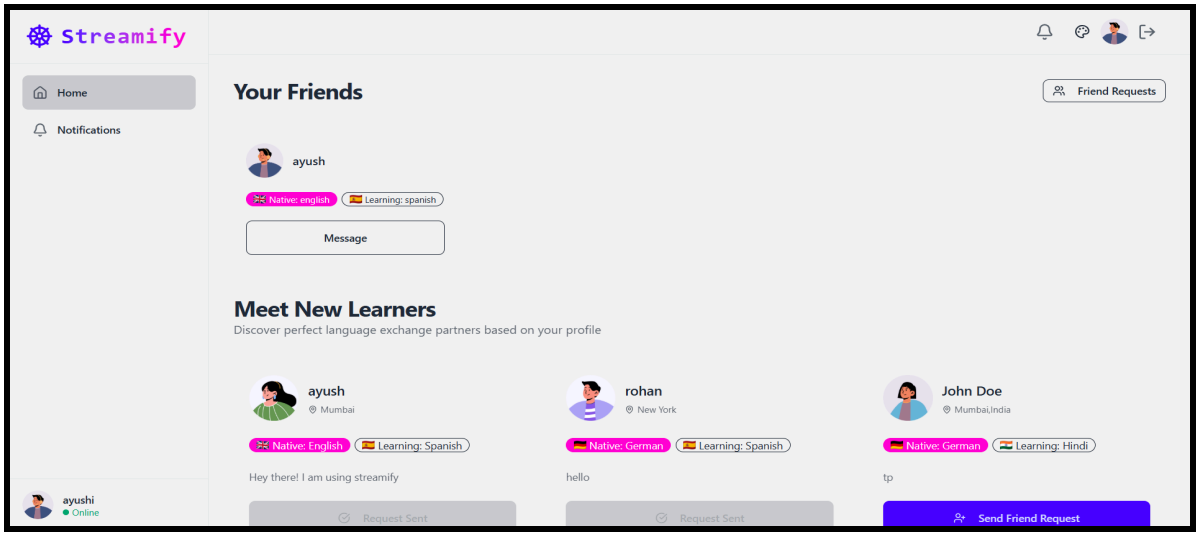


**Figure 2 : Home Page**

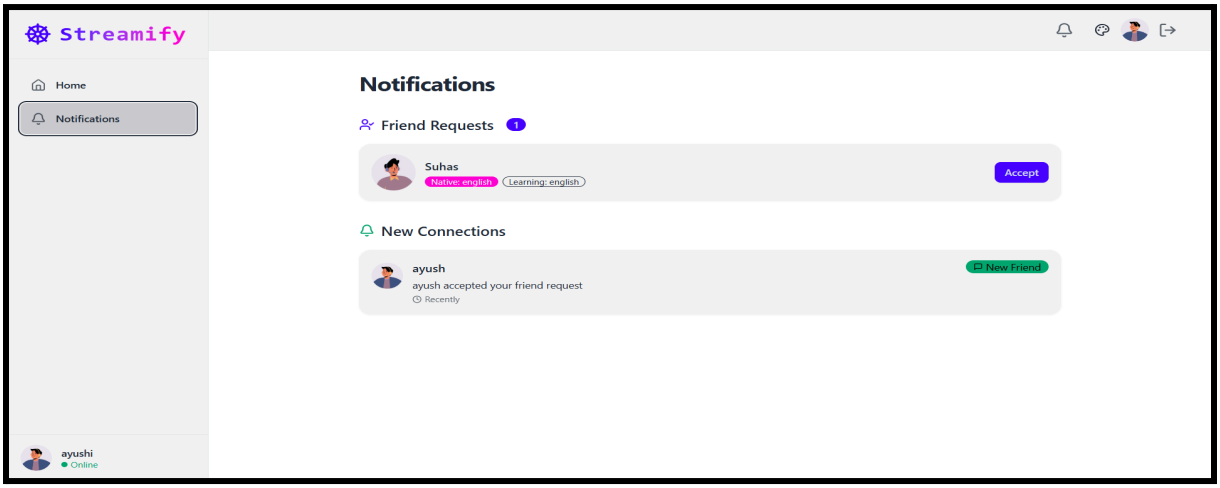


**Figure 3 : Notification Page**

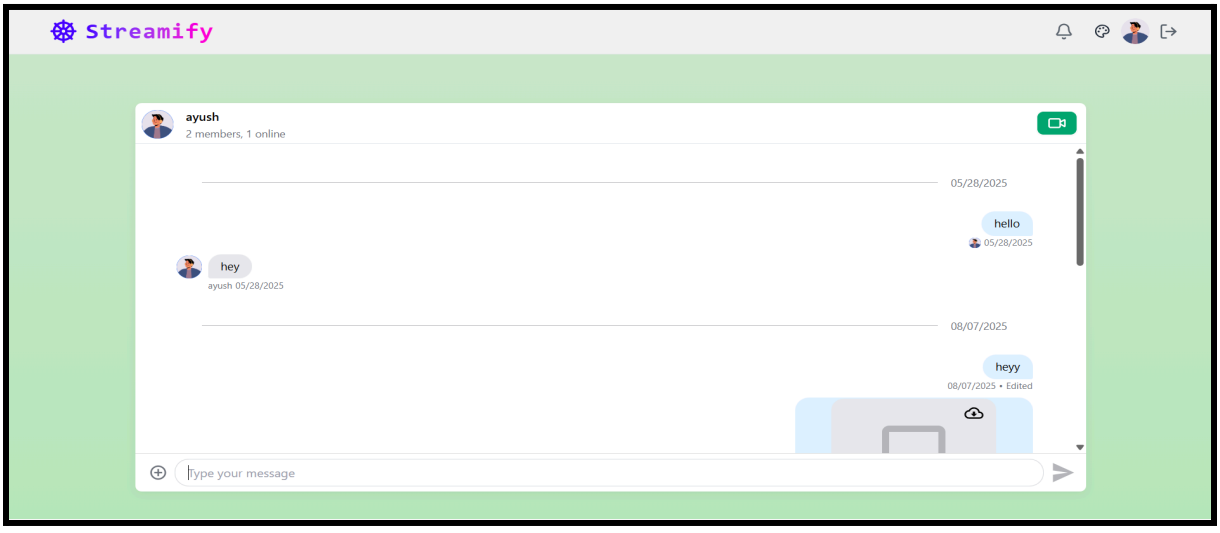

**Figure 4 : Chat Page**

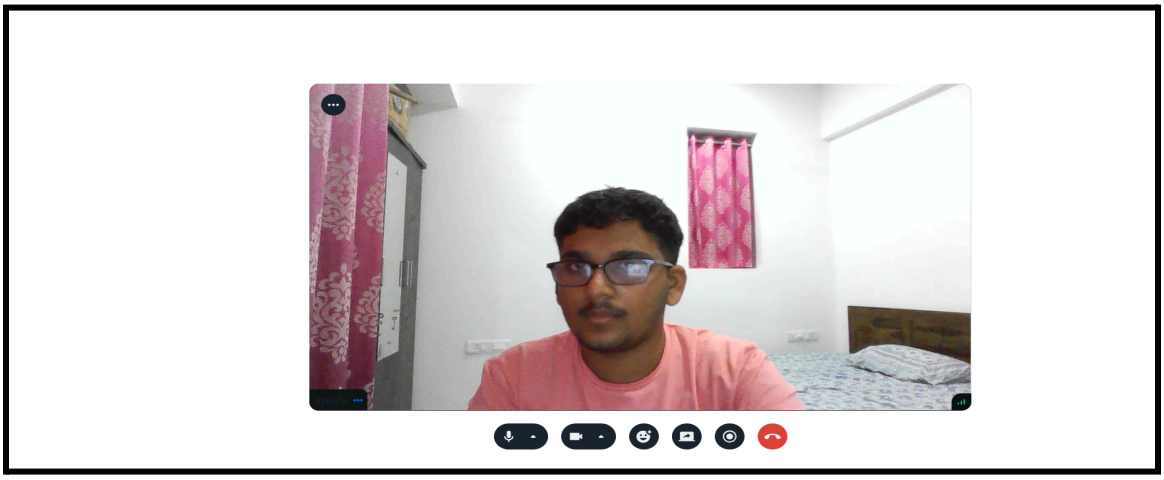

**Figure 5 : Call Page**

To view this project's source code: https://github.com/Ayushkaranth/Streamify
This is the live demo deployed in Vercel: https://streamify-p6p6.onrender.com/login

## Conclusion:

In exploring Redux and API integration in React, we gain a deeper understanding of how to manage state and handle asynchronous data flows in modern web applications. Redux provides a robust framework for maintaining a predictable, centralized state through actions, reducers, and a single store, making it easier to track and debug state changes. When combined with API calls, typically managed via tools like Fetch or Axios within React's useEffect or lifecycle methods, Redux enables seamless handling of asynchronous operations through middleware like Redux Thunk or Saga. This combination ensures that data fetched from external services is efficiently integrated into the application's state, driving dynamic UI updates. By leveraging Redux for state management and carefully structuring API interactions, developers can build scalable, maintainable, and performant React applications. Understanding these tools and their interplay highlights the importance of thoughtful state management and asynchronous data handling for creating responsive and reliable user experiences.