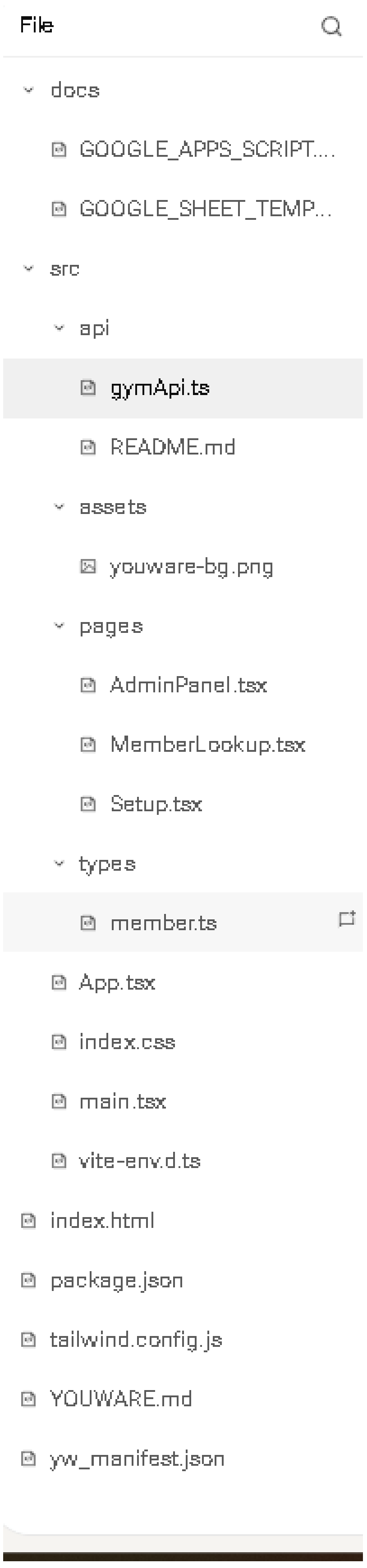

File

- docs
  - GOOGLE_APPS_SCRIPT....
  - GOOGLE_SHEET_TEMP...
- src
  - api
    - gymApi.ts
    - README.md
  - assets
    - youware-bg.png
  - pages
    - AdminPanel.tsx
    - MemberLookup.tsx
    - Setup.tsx
  - types
    - member.ts
  - App.tsx
  - index.css
  - main.tsx
  - vite-env.d.ts
- index.html
- package.json
- tailwind.config.js
- YOUWARE.md
- yw_manifest.json

I am giving u the code according to youwere
Src folder
inside src

gymApi.ts : *// Gym Membership API Client*

```
// Connects to Google Apps Script backend


import type { Member, MemberLookupResponse, MembersListResponse, ApiResponse,
NewMemberData, RenewalData } from '../types/member';


// API Base URL - This will be replaced with your actual Google Apps Script URL

// Instructions: After deploying your Apps Script, replace this URL

const API_BASE_URL = localStorage.getItem('gymApiUrl') || '';


// Helper function to get the API URL

export function getApiUrl(): string {

  return localStorage.getItem('gymApiUrl') || '';

}


// Helper function to set the API URL

export function setApiUrl(url: string): void {

  localStorage.setItem('gymApiUrl', url);

}


// Helper function to get admin password

export function getAdminPassword(): string {

  return sessionStorage.getItem('gymAdminPassword') || '';

}


// Helper function to set admin password (session only)

export function setAdminPassword(password: string): void {

  sessionStorage.setItem('gymAdminPassword', password);

}
```

```
// Helper function to clear admin session
export function clearAdminSession(): void {
  sessionStorage.removeItem('gymAdminPassword');
}

// Check if admin is logged in
export function isAdminLoggedIn(): boolean {
  return !!sessionStorage.getItem('gymAdminPassword');
}

/**
 * Look up a member by ID (Public - Read Only)
 */
export async function lookupMember(memberId: string): Promise<MemberLookupResponse> {
  const apiUrl = getApiUrl();

  if (!apiUrl) {
    throw new Error('API URL not configured. Please set up the Google Apps Script URL first.');
  }

  try {
    const url = `${apiUrl}?action=lookup&id=${encodeURIComponent(memberId)}`;
    const response = await fetch(url);
    const data = await response.json();
    return data;
  } catch (error) {
    console.error('Lookup error:', error);
    throw new Error('Failed to connect to the server. Please check your internet connection.');
  }
}
```

```
/**
 * Get all members (Admin only)
 */
export async function getAllMembers(): Promise<MembersListResponse> {
  const apiUrl = getApiUrl();
  const password = getAdminPassword();

  if (!apiUrl) {
    throw new Error('API URL not configured.');
  }

  if (!password) {
    throw new Error('Admin authentication required.');
  }

  try {
    const url = `${apiUrl}?action=getAll&password=${encodeURIComponent(password)}`;
    const response = await fetch(url);
    const data = await response.json();
    return data;
  } catch (error) {
    console.error('Get all members error:', error);
    throw new Error('Failed to fetch members.');
  }
}

/**
 * Add a new member (Admin only)
 */
export async function addMember(memberData: NewMemberData): Promise<ApiResponse> {
  const apiUrl = getApiUrl();
  const password = getAdminPassword();

  if (!apiUrl || !password) {
    throw new Error('Authentication required.');
  }

  try {
    const response = await fetch(apiUrl, {
      method: 'POST',
      headers: {
        'Content-Type': 'text/plain', // Apps Script requires this
      },
      body: JSON.stringify({
        password,
        action: 'addMember',
        member: memberData,
      }),
    });
    const data = await response.json();
    return data;
  } catch (error) {
    console.error('Add member error:', error);
    throw new Error('Failed to add member.');
  }
}

/**
 * Update member details (Admin only)
 */
```

```
export async function updateMember(memberData: Partial<Member> & { id: string }):
Promise<ApiResponse> {
 const apiUrl = getApiUrl();
 const password = getAdminPassword();


 if (!apiUrl || !password) {
  throw new Error('Authentication required.');
 }


 try {
  const response = await fetch(apiUrl, {
   method: 'POST',
   headers: {
    'Content-Type': 'text/plain',
   },
   body: JSON.stringify({
    password,
    action: 'updateMember',
    member: memberData,
   }),
  });
  const data = await response.json();
  return data;
 } catch (error) {
  console.error('Update member error:', error);
  throw new Error('Failed to update member.');
 }
}

/**
 * Renew membership (Admin only)
 */
export async function renewMembership(renewalData: RenewalData): Promise<ApiResponse> {
  const apiUrl = getApiUrl();
  const password = getAdminPassword();

  if (!apiUrl || !password) {
    throw new Error('Authentication required.');
  }

  try {
    const response = await fetch(apiUrl, {
      method: 'POST',
      headers: {
        'Content-Type': 'text/plain',
      },
      body: JSON.stringify({
        password,
        action: 'renewMember',
        ...renewalData,
      }),
    });
    const data = await response.json();
    return data;
  } catch (error) {
    console.error('Renew membership error:', error);
    throw new Error('Failed to renew membership.');
  }
}

/**
 * Delete member (Admin only)
 */
export async function deleteMember(memberId: string): Promise<ApiResponse> {
  const apiUrl = getApiUrl();
  const password = getAdminPassword();

  if (!apiUrl || !password) {
    throw new Error('Authentication required.');
  }

  try {
    const response = await fetch(apiUrl, {
      method: 'POST',
      headers: {
        'Content-Type': 'text/plain',
      },
      body: JSON.stringify({
        password,
        action: 'deleteMember',
        memberId,
      }),
    });
    const data = await response.json();
    return data;
  } catch (error) {
    console.error('Delete member error:', error);
    throw new Error('Failed to delete member.');
  }
}

/**
 * Verify admin password by attempting to fetch all members
 */
export async function verifyAdminPassword(password: string): Promise<boolean> {
  const apiUrl = getApiUrl();

  if (!apiUrl) {
    throw new Error('API URL not configured.');
  }

  try {
    const url = `${apiUrl}?action=getAll&password=${encodeURIComponent(password)}`;
    const response = await fetch(url);
    const data = await response.json();

    if (data.success) {
      setAdminPassword(password);
      return true;
    }
    return false;
  } catch {
    return false;
  }
}
```

README.md

```
# API Directory

This is an API reserved directory.
```

Assessts folder

Inside assessts

Pages folder

Inside pages file name as per screenshot

AdminPanel.tsx

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

import { motion, AnimatePresence } from 'framer-motion';

import {

  Lock, User, Phone, Calendar, Award, Plus, Edit2, Trash2,

  RefreshCw, LogOut, CheckCircle, XCircle, Loader2, Users,

  ChevronDown, X, Save, AlertTriangle

} from 'lucide-react';

import {

  getAllMembers, verifyAdminPassword, isAdminLoggedIn,

  clearAdminSession, addMember, renewMembership, deleteMember,

  getApiUrl, updateMember

} from '../api/gymApi';

import type { Member, NewMemberData } from '../types/member';


type ModalType = 'add' | 'edit' | 'renew' | 'delete' | null;


export default function AdminPanel() {

  const [isAuthenticated, setIsAuthenticated] = useState(false);

  const [password, setPassword] = useState('');

  const [members, setMembers] = useState<Member[]>([]);

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

  const [error, setError] = useState<string | null>(null);

  const [success, setSuccess] = useState<string | null>(null);
  const [modal, setModal] = useState<ModalType>(null);
  const [selectedMember, setSelectedMember] = useState<Member | null>(null);
  const [filter, setFilter] = useState<'all' | 'active' | 'expired'>('all');

  // Form state for add/edit
  const [formData, setFormData] = useState<Partial<NewMemberData>>({
    id: '',
    name: '',
    phone: '',
    age: 0,
    weight: 0,
    membershipType: '3 Months',
    startDate: new Date().toISOString().split('T')[0],
  });

  // Renewal form state
  const [renewalData, setRenewalData] = useState({
    membershipType: '3 Months' as '3 Months' | '6 Months' | '1 Year',
    startDate: new Date().toISOString().split('T')[0],
  });

  useEffect(() => {
    if (isAdminLoggedIn()) {
      setIsAuthenticated(true);
      fetchMembers();
    }
  }, []);

  const handleLogin = async (e: React.FormEvent) => {
    e.preventDefault();
    setLoading(true);
    setError(null);

    if (!getApiUrl()) {
      setError('API URL not configured. Please set up first.');
      setLoading(false);
      return;
    }

    try {
      const success = await verifyAdminPassword(password);
      if (success) {
        setIsAuthenticated(true);
        fetchMembers();
      } else {
        setError('Invalid password');
      }
    } catch (err) {
      setError(err instanceof Error ? err.message : 'Login failed');
    } finally {
      setLoading(false);
    }
  };

  const handleLogout = () => {
    clearAdminSession();
    setIsAuthenticated(false);
    setMembers([]);
    setPassword('');
  };

  const fetchMembers = async () => {
    setLoading(true);
    setError(null);

    try {
      const response = await getAllMembers();
      if (response.success && response.members) {
        setMembers(response.members);
      } else {
        setError(response.error || 'Failed to fetch members');
      }
    } catch (err) {
      setError(err instanceof Error ? err.message : 'Failed to fetch members');
    } finally {
      setLoading(false);
    }
  };

  const openAddModal = () => {
    setFormData({
      id: `GYM${String(members.length + 1).padStart(3, '0')}`,
      name: '',
      phone: '',
      age: 25,
      weight: 70,
      membershipType: '3 Months',
      startDate: new Date().toISOString().split('T')[0],
    });
    setModal('add');
  };

  const openEditModal = (member: Member) => {
    setSelectedMember(member);
    setFormData({
      id: member.id,
      name: member.name,
      phone: member.phone,
      age: member.age,
      weight: member.weight,
      membershipType: member.membershipType,
      startDate: member.startDate,
    });
    setModal('edit');
  };

  const openRenewModal = (member: Member) => {
    setSelectedMember(member);
    setRenewalData({
      membershipType: member.membershipType,
      startDate: new Date().toISOString().split('T')[0],
    });
    setModal('renew');
  };

  const openDeleteModal = (member: Member) => {
    setSelectedMember(member);
    setModal('delete');
  };

  const closeModal = () => {
    setModal(null);
    setSelectedMember(null);
    setError(null);
```

```
};

const handleAddMember = async () => {
  setLoading(true);
  setError(null);

  try {
    const response = await addMember(formData as NewMemberData);
    if (response.success) {
      setSuccess('Member added successfully!');
      closeModal();
      fetchMembers();
      setTimeout(() => setSuccess(null), 3000);
    } else {
      setError(response.error || 'Failed to add member');
    }
  } catch (err) {
    setError(err instanceof Error ? err.message : 'Failed to add member');
  } finally {
    setLoading(false);
  }
};

const handleEditMember = async () => {
  if (!selectedMember) return;
  setLoading(true);
  setError(null);

  try {
    const response = await updateMember({
      id: selectedMember.id,
      name: formData.name,
      phone: formData.phone,
      age: formData.age,
      weight: formData.weight,
      membershipType: formData.membershipType,
      startDate: formData.startDate,
    });
    if (response.success) {
      setSuccess('Member updated successfully!');
      closeModal();
      fetchMembers();
      setTimeout(() => setSuccess(null), 3000);
    } else {
      setError(response.error || 'Failed to update member');
    }
  } catch (err) {
    setError(err instanceof Error ? err.message : 'Failed to update member');
  } finally {
    setLoading(false);
  }
};

const handleRenewMember = async () => {
  if (!selectedMember) return;
  setLoading(true);
  setError(null);

  try {
    const response = await renewMembership({
      memberId: selectedMember.id,
      membershipType: renewalData.membershipType,
      startDate: renewalData.startDate,
    });
    if (response.success) {
      setSuccess(`Membership renewed! New end date: ${response.newEndDate}`);
      closeModal();
      fetchMembers();
      setTimeout(() => setSuccess(null), 3000);
    } else {
      setError(response.error || 'Failed to renew membership');
    }
  } catch (err) {
    setError(err instanceof Error ? err.message : 'Failed to renew membership');
  } finally {
    setLoading(false);
  }
};

const handleDeleteMember = async () => {
  if (!selectedMember) return;
  setLoading(true);
  setError(null);

  try {
    const response = await deleteMember(selectedMember.id);
    if (response.success) {
      setSuccess('Member deleted successfully!');
      closeModal();
      fetchMembers();
      setTimeout(() => setSuccess(null), 3000);
    } else {
      setError(response.error || 'Failed to delete member');
      }
    } catch (err) {
      setError(err instanceof Error ? err.message : 'Failed to delete member');
    } finally {
      setLoading(false);
    }
  };

  const filteredMembers = members.filter(m => {
    if (filter === 'all') return true;
    if (filter === 'active') return m.status === 'Active';
    if (filter === 'expired') return m.status === 'Expired';
    return true;
  });

  const stats = {
    total: members.length,
    active: members.filter(m => m.status === 'Active').length,
    expired: members.filter(m => m.status === 'Expired').length,
  };

  // Login Screen
  if (!isAuthenticated) {
    return (
      <div className="min-h-screen bg-gradient-to-br from-slate-900 via-purple-900 to-slate-900 flex
items-center justify-center px-4">
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          className="w-full max-w-sm"
        >
        <div className="text-center mb-8">
         <div className="inline-flex items-center justify-center w-16 h-16 rounded-2xl bg-gradient-to-
br from-purple-500 to-pink-500 mb-4 shadow-lg shadow-purple-500/30">
          <Lock className="w-8 h-8 text-white" />
         </div>
         <h1 className="text-2xl font-bold text-white mb-2">Admin Login</h1>
         <p className="text-purple-200/70 text-sm">Enter your password to continue</p>
        </div>

        <form onSubmit={handleLogin} className="space-y-4">
         <div>
          <input
           type="password"
           value={password}
           onChange={(e) => setPassword(e.target.value)}
           placeholder="Enter admin password"
           className="w-full px-4 py-4 bg-white/10 backdrop-blur-lg border border-white/20
rounded-2xl text-white placeholder-purple-300/50 focus:outline-none focus:ring-2 focus:ring-purple-
500"
           disabled={loading}
          />
         </div>

         {error && (
          <motion.div
           initial={{ opacity: 0 }}
           animate={{ opacity: 1 }}
           className="p-3 bg-red-500/20 border border-red-500/30 rounded-xl text-red-300 text-sm
text-center"
          >
           {error}
          </motion.div>
          )}

          <motion.button
            type="submit"
            disabled={loading}
            whileHover={{ scale: 1.02 }}
            whileTap={{ scale: 0.98 }}
            className="w-full py-4 bg-gradient-to-r from-purple-600 to-pink-600 text-white font-semibold rounded-2xl shadow-lg shadow-purple-500/30 hover:shadow-purple-500/50 transition-all disabled:opacity-50 flex items-center justify-center gap-2"
          >
            {loading ? (
              <>
                <Loader2 className="w-5 h-5 animate-spin" />
                Verifying...
              </>
            ) : (
              <>
                <Lock className="w-5 h-5" />
                Login
              </>
            )}
          </motion.button>
        </form>
      </motion.div>
    </div>
  );
}

// Admin Dashboard
return (
  <div className="min-h-screen bg-gradient-to-br from-slate-900 via-purple-900 to-slate-900">
```

```
   {/* Header */}
   <header className="bg-white/5 backdrop-blur-lg border-b border-white/10 px-4 py-4 sticky top-
0 z-40">
    <div className="max-w-6xl mx-auto flex items-center justify-between">
     <div className="flex items-center gap-3">
      <div className="w-10 h-10 rounded-xl bg-gradient-to-br from-purple-500 to-pink-500 flex
items-center justify-center">
       <Users className="w-5 h-5 text-white" />
      </div>
      <div>
       <h1 className="text-lg font-bold text-white">Admin Panel</h1>
       <p className="text-xs text-purple-300/60">Manage Members</p>
      </div>
     </div>
     <button
      onClick={handleLogout}
      className="flex items-center gap-2 px-4 py-2 bg-white/10 hover:bg-white/20 rounded-xl text-
white text-sm transition-all"
     >
      <LogOut className="w-4 h-4" />
      <span className="hidden sm:inline">Logout</span>
     </button>
    </div>
   </header>


   {/* Success Toast */}
   <AnimatePresence>
    {success && (
     <motion.div
      initial={{ opacity: 0, y: -50 }}
      animate={{ opacity: 1, y: 0 }}
      exit={{ opacity: 0, y: -50 }}
```

```
          className="fixed top-20 left-1/2 -translate-x-1/2 z-50 bg-emerald-500/90 backdrop-blur-lg 
text-white px-6 py-3 rounded-2xl shadow-lg flex items-center gap-2"
        >
          <CheckCircle className="w-5 h-5" />
          {success}
        </motion.div>
      )}
    </AnimatePresence>

    <main className="max-w-6xl mx-auto px-4 py-6">
      {/* Stats */}
      <div className="grid grid-cols-3 gap-3 mb-6">
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          className="bg-white/10 backdrop-blur-lg border border-white/20 rounded-2xl p-4 text-
center"
        >
          <div className="text-2xl font-bold text-white">{stats.total}</div>
          <div className="text-xs text-purple-300/60">Total</div>
        </motion.div>
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ delay: 0.1 }}
          className="bg-emerald-500/20 border border-emerald-500/30 rounded-2xl p-4 text-center"
        >
          <div className="text-2xl font-bold text-emerald-300">{stats.active}</div>
          <div className="text-xs text-emerald-300/60">Active</div>
        </motion.div>
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ delay: 0.2 }}
          className="bg-red-500/20 border border-red-500/30 rounded-2xl p-4 text-center"
        >
          <div className="text-2xl font-bold text-red-300">{stats.expired}</div>
          <div className="text-xs text-red-300/60">Expired</div>
        </motion.div>
      </div>

      {/* Actions Bar */}
      <div className="flex flex-wrap items-center justify-between gap-3 mb-6">
        <div className="flex items-center gap-2">
          <button
            onClick={() => setFilter('all')}
            className={`px-4 py-2 rounded-xl text-sm font-medium transition-all ${
              filter === 'all'
                ? 'bg-white/20 text-white'
                : 'bg-white/5 text-purple-300 hover:bg-white/10'
            }`}
          >
            All
          </button>
          <button
            onClick={() => setFilter('active')}
            className={`px-4 py-2 rounded-xl text-sm font-medium transition-all ${
              filter === 'active'
                ? 'bg-emerald-500/30 text-emerald-300'
                : 'bg-white/5 text-purple-300 hover:bg-white/10'
            }`}
          >
            Active
```

```
          </button>
          <button
            onClick={() => setFilter('expired')}
            className={`px-4 py-2 rounded-xl text-sm font-medium transition-all ${
              filter === 'expired'
                ? 'bg-red-500/30 text-red-300'
                : 'bg-white/5 text-purple-300 hover:bg-white/10'
            }`}
          >
            Expired
          </button>
        </div>
        <div className="flex items-center gap-2">
          <button
            onClick={fetchMembers}
            disabled={loading}
            className="p-2 bg-white/10 hover:bg-white/20 rounded-xl text-white transition-all
disabled:opacity-50"
          >
            <RefreshCw className={`w-5 h-5 ${loading ? 'animate-spin' : ''}`} />
          </button>
          <button
            onClick={openAddModal}
            className="flex items-center gap-2 px-4 py-2 bg-gradient-to-r from-purple-600 to-pink-600
text-white font-medium rounded-xl shadow-lg shadow-purple-500/30 hover:shadow-purple-500/50
transition-all"
          >
            <Plus className="w-5 h-5" />
            <span className="hidden sm:inline">Add Member</span>
          </button>
        </div>
      </div>
```

```
        {/* Members List */}
        {loading && members.length === 0 ? (
          <div className="flex items-center justify-center py-20">
            <Loader2 className="w-8 h-8 text-purple-400 animate-spin" />
          </div>
        ) : filteredMembers.length === 0 ? (
          <div className="text-center py-20">
            <Users className="w-12 h-12 text-purple-400/30 mx-auto mb-4" />
            <p className="text-purple-300/50">No members found</p>
          </div>
        ) : (
          <div className="space-y-3">
            {filteredMembers.map((member, index) => (
              <motion.div
                key={member.id}
                initial={{ opacity: 0, y: 20 }}
                animate={{ opacity: 1, y: 0 }}
                transition={{ delay: index * 0.05 }}
                className="bg-white/10 backdrop-blur-lg border border-white/20 rounded-2xl p-4"
              >
                <div className="flex items-start justify-between gap-4">
                  <div className="flex-1 min-w-0">
                    <div className="flex items-center gap-2 mb-2">
                      <span className="font-mono text-xs text-purple-400 bg-purple-500/20 px-2 py-0.5
rounded">
                        {member.id}
                      </span>
                      <span className={`text-xs px-2 py-0.5 rounded-full ${
                        member.status === 'Active'
                          ? 'bg-emerald-500/20 text-emerald-300'
                          : 'bg-red-500/20 text-red-300'
                      }`}>
                        {member.status}
                      </span>
                    </div>
                    <h3 className="text-white font-semibold truncate">{member.name}</h3>
                    <div className="flex flex-wrap items-center gap-x-4 gap-y-1 mt-2 text-sm text-purple-
        300/70">
                      <span className="flex items-center gap-1">
                        <Phone className="w-3.5 h-3.5" />
                        {member.phone}
                      </span>
                      <span className="flex items-center gap-1">
                        <Award className="w-3.5 h-3.5" />
                        {member.membershipType}
                      </span>
                      <span className="flex items-center gap-1">
                        <Calendar className="w-3.5 h-3.5" />
                        {member.endDate}
                      </span>
                    </div>
                  </div>
                  <div className="flex items-center gap-1">
                    <button
                      onClick={() => openRenewModal(member)}
                      className="p-2 bg-emerald-500/20 hover:bg-emerald-500/30 rounded-xl text-emerald-
        300 transition-all"
                      title="Renew"
                    >
                      <RefreshCw className="w-4 h-4" />
                    </button>
                    <button
```

```
                onClick={() => openEditModal(member)}
                className="p-2 bg-blue-500/20 hover:bg-blue-500/30 rounded-xl text-blue-300
transition-all"
                title="Edit"
              >
                <Edit2 className="w-4 h-4" />
              </button>
              <button
                onClick={() => openDeleteModal(member)}
                className="p-2 bg-red-500/20 hover:bg-red-500/30 rounded-xl text-red-300 transition-
all"
                title="Delete"
              >
                <Trash2 className="w-4 h-4" />
              </button>
            </div>
          </div>
        </motion.div>
      ))}
    </div>
  )}
</main>

{/* Modals */}
<AnimatePresence>
  {modal && (
    <motion.div
      initial={{ opacity: 0 }}
      animate={{ opacity: 1 }}
      exit={{ opacity: 0 }}
      className="fixed inset-0 bg-black/70 backdrop-blur-sm z-50 flex items-end sm:items-center
justify-center"
```

```
          onClick={closeModal}
        >
          <motion.div
            initial={{ opacity: 0, y: 100 }}
            animate={{ opacity: 1, y: 0 }}
            exit={{ opacity: 0, y: 100 }}
            className="w-full sm:max-w-md bg-slate-900 border border-white/20 rounded-t-3xl
sm:rounded-3xl p-6 max-h-[90vh] overflow-y-auto"
            onClick={(e) => e.stopPropagation()}
          >
            {/* Modal Header */}
            <div className="flex items-center justify-between mb-6">
              <h2 className="text-xl font-bold text-white">
                {modal === 'add' && 'Add New Member'}
                {modal === 'edit' && 'Edit Member'}
                {modal === 'renew' && 'Renew Membership'}
                {modal === 'delete' && 'Delete Member'}
              </h2>
              <button
                onClick={closeModal}
                className="p-2 hover:bg-white/10 rounded-xl text-purple-300 transition-all"
              >
                <X className="w-5 h-5" />
              </button>
            </div>

            {error && (
              <div className="mb-4 p-3 bg-red-500/20 border border-red-500/30 rounded-xl text-red-
300 text-sm">
                {error}
              </div>
            )}
            {/* Add/Edit Form */}
            {(modal === 'add' || modal === 'edit') && (
              <div className="space-y-4">
                <div>
                  <label className="block text-sm text-purple-300/70 mb-1">Member ID</label>
                  <input
                    type="text"
                    value={formData.id || ''}
                    onChange={(e) => setFormData({ ...formData, id: e.target.value.toUpperCase() })}
                    className="w-full px-4 py-3 bg-white/10 border border-white/20 rounded-xl text-white placeholder-purple-300/50 focus:outline-none focus:ring-2 focus:ring-purple-500"
                    disabled={modal === 'edit'}
                  />
                </div>
                <div>
                  <label className="block text-sm text-purple-300/70 mb-1">Name</label>
                  <input
                    type="text"
                    value={formData.name || ''}
                    onChange={(e) => setFormData({ ...formData, name: e.target.value })}
                    className="w-full px-4 py-3 bg-white/10 border border-white/20 rounded-xl text-white placeholder-purple-300/50 focus:outline-none focus:ring-2 focus:ring-purple-500"
                  />
                </div>
                <div>
                  <label className="block text-sm text-purple-300/70 mb-1">Phone</label>
                  <input
                    type="tel"
                    value={formData.phone || ''}
                    onChange={(e) => setFormData({ ...formData, phone: e.target.value })}
                    className="w-full px-4 py-3 bg-white/10 border border-white/20 rounded-xl text-white placeholder-purple-300/50 focus:outline-none focus:ring-2 focus:ring-purple-500"
                  />
                </div>
                <div className="grid grid-cols-2 gap-4">
                  <div>
                    <label className="block text-sm text-purple-300/70 mb-1">Age</label>
                    <input
                      type="number"
                      value={formData.age || ''}
                      onChange={(e) => setFormData({ ...formData, age: parseInt(e.target.value) || 0 })}
                      className="w-full px-4 py-3 bg-white/10 border border-white/20 rounded-xl text-white placeholder-purple-300/50 focus:outline-none focus:ring-2 focus:ring-purple-500"
                    />
                  </div>
                  <div>
                    <label className="block text-sm text-purple-300/70 mb-1">Weight (kg)</label>
                    <input
                      type="number"
                      value={formData.weight || ''}
                      onChange={(e) => setFormData({ ...formData, weight: parseInt(e.target.value) || 0 })}
                      className="w-full px-4 py-3 bg-white/10 border border-white/20 rounded-xl text-white placeholder-purple-300/50 focus:outline-none focus:ring-2 focus:ring-purple-500"
                    />
                  </div>
                </div>
                <div>
                  <label className="block text-sm text-purple-300/70 mb-1">Membership Type</label>
                  <div className="relative">
                    <select
                      value={formData.membershipType || '3 Months'}
                      onChange={(e) => setFormData({ ...formData, membershipType: e.target.value as any })}
                className="w-full px-4 py-3 bg-white/10 border border-white/20 rounded-xl text-
white appearance-none focus:outline-none focus:ring-2 focus:ring-purple-500"
           >
            <option value="3 Months">3 Months</option>
            <option value="6 Months">6 Months</option>
            <option value="1 Year">1 Year</option>
           </select>
           <ChevronDown className="absolute right-4 top-1/2 -translate-y-1/2 w-4 h-4 text-
purple-300 pointer-events-none" />
          </div>
         </div>
         <div>
          <label className="block text-sm text-purple-300/70 mb-1">Start Date</label>
          <input
           type="date"
           value={formData.startDate || ''}
           onChange={(e) => setFormData({ ...formData, startDate: e.target.value })}
           className="w-full px-4 py-3 bg-white/10 border border-white/20 rounded-xl text-white
focus:outline-none focus:ring-2 focus:ring-purple-500"
          />
         </div>
         <button
          onClick={modal === 'add' ? handleAddMember : handleEditMember}
          disabled={loading}
          className="w-full py-4 bg-gradient-to-r from-purple-600 to-pink-600 text-white font-
semibold rounded-2xl shadow-lg shadow-purple-500/30 hover:shadow-purple-500/50 transition-all
disabled:opacity-50 flex items-center justify-center gap-2"
         >
          {loading ? (
           <Loader2 className="w-5 h-5 animate-spin" />
          ) : (
           <>
            <Save className="w-5 h-5" />
```

```
              {modal === 'add' ? 'Add Member' : 'Save Changes'}
            </>
          )}
        </button>
      </div>
    )}

    {/* Renew Form */}
    {modal === 'renew' && selectedMember && (
      <div className="space-y-4">
        <div className="p-4 bg-white/5 rounded-xl">
          <p className="text-purple-300/70 text-sm">Renewing membership for:</p>
          <p className="text-white font-semibold">{selectedMember.name}
({selectedMember.id})</p>
          <p className="text-purple-300/50 text-sm">Current:
{selectedMember.membershipType} • Expires: {selectedMember.endDate}</p>
        </div>
        <div>
          <label className="block text-sm text-purple-300/70 mb-1">New Membership
Type</label>
          <div className="relative">
            <select
              value={renewalData.membershipType}
              onChange={(e) => setRenewalData({ ...renewalData, membershipType: e.target.value as
any })}
              className="w-full px-4 py-3 bg-white/10 border border-white/20 rounded-xl text-
white appearance-none focus:outline-none focus:ring-2 focus:ring-purple-500"
            >
              <option value="3 Months">3 Months</option>
              <option value="6 Months">6 Months</option>
              <option value="1 Year">1 Year</option>
            </select>
            <ChevronDown className="absolute right-4 top-1/2 -translate-y-1/2 w-4 h-4 text-purple-300 pointer-events-none" />
          </div>
        </div>
        <div>
          <label className="block text-sm text-purple-300/70 mb-1">New Start Date</label>
          <input
            type="date"
            value={renewalData.startDate}
            onChange={(e) => setRenewalData({ ...renewalData, startDate: e.target.value })}
            className="w-full px-4 py-3 bg-white/10 border border-white/20 rounded-xl text-white focus:outline-none focus:ring-2 focus:ring-purple-500"
          />
        </div>
        <button
          onClick={handleRenewMember}
          disabled={loading}
          className="w-full py-4 bg-gradient-to-r from-emerald-600 to-teal-600 text-white font-semibold rounded-2xl shadow-lg shadow-emerald-500/30 hover:shadow-emerald-500/50 transition-all disabled:opacity-50 flex items-center justify-center gap-2"
        >
          {loading ? (
            <Loader2 className="w-5 h-5 animate-spin" />
          ) : (
            <>
              <RefreshCw className="w-5 h-5" />
              Renew Membership
            </>
          )}
        </button>
      </div>
    )}
```

```
          {/* Delete Confirmation */}
          {modal === 'delete' && selectedMember && (
            <div className="space-y-4">
              <div className="p-4 bg-red-500/10 border border-red-500/30 rounded-xl text-center">
                <AlertTriangle className="w-12 h-12 text-red-400 mx-auto mb-3" />
                <p className="text-white font-semibold mb-1">Are you sure?</p>
                <p className="text-red-300/70 text-sm">
                  This will permanently delete <strong>{selectedMember.name}</strong>
({selectedMember.id}) from the system.
                </p>
              </div>
              <div className="grid grid-cols-2 gap-3">
                <button
                  onClick={closeModal}
                  className="py-3 bg-white/10 hover:bg-white/20 text-white font-medium rounded-xl
transition-all"
                >
                  Cancel
                </button>
                <button
                  onClick={handleDeleteMember}
                  disabled={loading}
                  className="py-3 bg-red-600 hover:bg-red-700 text-white font-medium rounded-xl
transition-all disabled:opacity-50 flex items-center justify-center gap-2"
                >
                  {loading ? (
                    <Loader2 className="w-5 h-5 animate-spin" />
                  ) : (
                    <>
                      <Trash2 className="w-4 h-4" />
                      Delete
                    </>
                  )}
                </button>
              </div>
            </div>
          )}
         </motion.div>
</motion.div>
)}
</AnimatePresence>
</div>
);
}
```

MemberLookup.tsx

```
import { useState } from 'react';
import { motion, AnimatePresence } from 'framer-motion';
import { Search, User, Phone, Calendar, Award, Clock, CheckCircle, XCircle, Loader2 } from 'lucide-
react';
import { lookupMember, getApiUrl } from '../api/gymApi';
import type { Member } from '../types/member';

export default function MemberLookup() {
  const [memberId, setMemberId] = useState('');
  const [member, setMember] = useState<Member | null>(null);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState<string | null>(null);
  const [searched, setSearched] = useState(false);
const handleSearch = async (e: React.FormEvent) => {
  e.preventDefault();

  if (!memberId.trim()) {
    setError('Please enter a Member ID');
    return;
  }

  if (!getApiUrl()) {
    setError('System not configured. Please contact the gym administrator.');
    return;
  }

  setLoading(true);
  setError(null);
  setMember(null);
  setSearched(true);

  try {
    const response = await lookupMember(memberId.trim());

    if (response.success && response.member) {
      setMember(response.member);
    } else {
      setError(response.error || 'Member not found');
    }
  } catch (err) {
    setError(err instanceof Error ? err.message : 'An error occurred');
  } finally {
    setLoading(false);
  }
```

```
 };


 const resetSearch = () => {
  setMemberId('');
  setMember(null);
  setError(null);
  setSearched(false);
 };


 return (
  <div className="min-h-screen bg-gradient-to-br from-slate-900 via-purple-900 to-slate-900 flex
flex-col">
   {/* Header */}
   <header className="pt-8 pb-4 px-4">
    <motion.div
     initial={{ opacity: 0, y: -20 }}
     animate={{ opacity: 1, y: 0 }}
     className="text-center"
    >
     <div className="inline-flex items-center justify-center w-16 h-16 rounded-2xl bg-gradient-to-
br from-purple-500 to-pink-500 mb-4 shadow-lg shadow-purple-500/30">
      <Award className="w-8 h-8 text-white" />
     </div>
     <h1 className="text-3xl font-bold text-white mb-2">Gym Membership</h1>
     <p className="text-purple-200/70">Check your membership status</p>
    </motion.div>
   </header>


   {/* Main Content */}
   <main className="flex-1 px-4 pb-8">
    <div className="max-w-md mx-auto">
     {/* Search Form */}
```

```
      <motion.form
        initial={{ opacity: 0, y: 20 }}
        animate={{ opacity: 1, y: 0 }}
        transition={{ delay: 0.1 }}
        onSubmit={handleSearch}
        className="mb-6"
      >
        <div className="relative">
          <div className="absolute inset-y-0 left-0 pl-4 flex items-center pointer-events-none">
            <Search className="w-5 h-5 text-purple-400" />
          </div>
          <input
            type="text"
            value={memberId}
            onChange={(e) => setMemberId(e.target.value.toUpperCase())}
            placeholder="Enter Member ID (e.g., GYM001)"
            className="w-full pl-12 pr-4 py-4 bg-white/10 backdrop-blur-lg border border-white/20
rounded-2xl text-white placeholder-purple-300/50 focus:outline-none focus:ring-2 focus:ring-purple-
500 focus:border-transparent transition-all"
            disabled={loading}
          />
        </div>
        <motion.button
          type="submit"
          disabled={loading}
          whileHover={{ scale: 1.02 }}
          whileTap={{ scale: 0.98 }}
          className="w-full mt-4 py-4 bg-gradient-to-r from-purple-600 to-pink-600 text-white font-
semibold rounded-2xl shadow-lg shadow-purple-500/30 hover:shadow-purple-500/50 transition-all
disabled:opacity-50 disabled:cursor-not-allowed flex items-center justify-center gap-2"
        >
          {loading ? (
            <>
              <Loader2 className="w-5 h-5 animate-spin" />
              Searching...
            </>
          ) : (
            <>
              <Search className="w-5 h-5" />
              Check Status
            </>
          )}
        </motion.button>
      </motion.form>

      {/* Results */}
      <AnimatePresence mode="wait">
        {error && searched && (
          <motion.div
            key="error"
            initial={{ opacity: 0, y: 20 }}
            animate={{ opacity: 1, y: 0 }}
            exit={{ opacity: 0, y: -20 }}
            className="bg-red-500/20 backdrop-blur-lg border border-red-500/30 rounded-2xl p-6 text-center"
          >
            <div className="inline-flex items-center justify-center w-12 h-12 rounded-full bg-red-500/20 mb-4">
              <XCircle className="w-6 h-6 text-red-400" />
            </div>
            <p className="text-red-200 font-medium">{error}</p>
            <button
              onClick={resetSearch}
              className="mt-4 text-sm text-red-300 hover:text-white transition-colors underline"
            >
              Try again
            </button>
          </motion.div>
        )}

        {member && (
          <motion.div
            key="result"
            initial={{ opacity: 0, y: 20 }}
            animate={{ opacity: 1, y: 0 }}
            exit={{ opacity: 0, y: -20 }}
            className="space-y-4"
          >
            {/* Status Card */}
            <div className={`relative overflow-hidden rounded-3xl p-6 ${
              member.status === 'Active'
                ? 'bg-gradient-to-br from-emerald-500/20 to-teal-500/20 border border-emerald-500/30'
                : 'bg-gradient-to-br from-red-500/20 to-orange-500/20 border border-red-500/30'
            }`}>
              <div className="absolute top-0 right-0 w-32 h-32 bg-white/5 rounded-full -translate-y-1/2
translate-x-1/2" />
              <div className="absolute bottom-0 left-0 w-24 h-24 bg-white/5 rounded-full translate-y-
1/2 -translate-x-1/2" />

              <div className="relative">
                <div className="flex items-center justify-between mb-4">
                  <span className="text-sm font-medium text-white/70">Membership Status</span>
                  <div className={`inline-flex items-center gap-1.5 px-3 py-1 rounded-full text-sm font-
semibold ${
                    member.status === 'Active'
                      ? 'bg-emerald-500/30 text-emerald-300'
                      : 'bg-red-500/30 text-red-300'
                  }`}>
                    {member.status === 'Active' ? (
                      <CheckCircle className="w-4 h-4" />
                    ) : (
                      <XCircle className="w-4 h-4" />
                    )}
                    {member.status}
                  </div>
                </div>

                {member.status === 'Active' && member.daysRemaining !== undefined && (
                  <div className="text-center py-4">
                    <div className="text-5xl font-bold text-white mb-1">
                      {member.daysRemaining}
                    </div>
                    <div className="text-emerald-300/70 text-sm">days remaining</div>
                  </div>
                )}

                {member.status === 'Expired' && (
                  <div className="text-center py-4">
                    <div className="text-xl font-semibold text-red-300 mb-1">
                      Membership Expired
                    </div>
                    <div className="text-red-300/70 text-sm">Please renew at the front desk</div>
                  </div>
                )}
              </div>
            </div>
```

```
        {/* Member Details */}
        <div className="bg-white/10 backdrop-blur-lg border border-white/20 rounded-3xl p-6
space-y-4">
          <h3 className="text-lg font-semibold text-white mb-4">Member Details</h3>

          <div className="flex items-center gap-4">
            <div className="w-10 h-10 rounded-xl bg-purple-500/20 flex items-center justify-
center">
              <User className="w-5 h-5 text-purple-400" />
            </div>
            <div>
              <div className="text-xs text-purple-300/60 uppercase tracking-wide">Name</div>
              <div className="text-white font-medium">{member.name}</div>
            </div>
          </div>

          <div className="flex items-center gap-4">
            <div className="w-10 h-10 rounded-xl bg-pink-500/20 flex items-center justify-center">
              <Phone className="w-5 h-5 text-pink-400" />
            </div>
            <div>
              <div className="text-xs text-purple-300/60 uppercase tracking-wide">Phone</div>
              <div className="text-white font-medium">{member.phone}</div>
            </div>
          </div>

          <div className="flex items-center gap-4">
            <div className="w-10 h-10 rounded-xl bg-blue-500/20 flex items-center justify-center">
              <Award className="w-5 h-5 text-blue-400" />
            </div>
            <div>
```

```
                <div className="text-xs text-purple-300/60 uppercase tracking-wide">Membership
Type</div>
                <div className="text-white font-medium">{member.membershipType}</div>
              </div>
            </div>

            <div className="flex items-center gap-4">
              <div className="w-10 h-10 rounded-xl bg-teal-500/20 flex items-center justify-center">
                <Calendar className="w-5 h-5 text-teal-400" />
              </div>
              <div>
                <div className="text-xs text-purple-300/60 uppercase tracking-wide">Valid Until</div>
                <div className="text-white font-medium">{member.endDate}</div>
              </div>
            </div>

            <div className="flex items-center gap-4">
              <div className="w-10 h-10 rounded-xl bg-amber-500/20 flex items-center justify-
center">
                <Clock className="w-5 h-5 text-amber-400" />
              </div>
              <div>
                <div className="text-xs text-purple-300/60 uppercase tracking-wide">Member ID</div>
                <div className="text-white font-medium font-mono">{member.id}</div>
              </div>
            </div>
          </div>

          {/* Search Again Button */}
          <motion.button
            onClick={resetSearch}
            whileHover={{ scale: 1.02 }}
```

```
                whileTap={{ scale: 0.98 }}
                className="w-full py-3 bg-white/10 border border-white/20 text-white font-medium
rounded-2xl hover:bg-white/20 transition-all"
              >
                Search Another Member
              </motion.button>
            </motion.div>
          )}
        </AnimatePresence>

        {/* Initial State */}
        {!searched && !member && !error && (
          <motion.div
            initial={{ opacity: 0 }}
            animate={{ opacity: 1 }}
            transition={{ delay: 0.3 }}
            className="text-center py-8"
          >
            <div className="inline-flex items-center justify-center w-20 h-20 rounded-full bg-white/5
mb-4">
              <Search className="w-10 h-10 text-purple-400/50" />
            </div>
            <p className="text-purple-200/50 text-sm">
              Enter your Member ID to check your membership status
            </p>
          </motion.div>
        )}
      </div>
    </main>

    {/* Footer */}
    <footer className="py-4 px-4 text-center">
```

```
        <p className="text-purple-300/40 text-xs">
          For any issues, please contact the gym front desk
        </p>
      </footer>
    </div>
  );
}
```

Setup.tsx

```
import { useState, useEffect } from 'react';
import { motion } from 'framer-motion';
import { Settings, Link, CheckCircle, ExternalLink, Copy, Check } from 'lucide-react';
import { setApiUrl, getApiUrl } from '../api/gymApi';

interface SetupProps {
  onComplete: () => void;
}

export default function Setup({ onComplete }: SetupProps) {
  const [apiUrl, setApiUrlState] = useState('');
  const [saved, setSaved] = useState(false);
  const [copied, setCopied] = useState(false);

  useEffect(() => {
    const existingUrl = getApiUrl();
    if (existingUrl) {
      setApiUrlState(existingUrl);
    }
  }, []);
  const handleSave = () => {
    if (apiUrl.trim()) {
      setApiUrl(apiUrl.trim());
      setSaved(true);
      setTimeout(() => {
        onComplete();
      }, 1500);
    }
  };

  const copyToClipboard = (text: string) => {
    navigator.clipboard.writeText(text);
    setCopied(true);
    setTimeout(() => setCopied(false), 2000);
  };

  return (
    <div className="min-h-screen bg-gradient-to-br from-slate-900 via-purple-900 to-slate-900 flex
flex-col">
      <header className="pt-8 pb-4 px-4">
        <motion.div
          initial={{ opacity: 0, y: -20 }}
          animate={{ opacity: 1, y: 0 }}
          className="text-center"
        >
          <div className="inline-flex items-center justify-center w-16 h-16 rounded-2xl bg-gradient-to-
br from-purple-500 to-pink-500 mb-4 shadow-lg shadow-purple-500/30">
            <Settings className="w-8 h-8 text-white" />
          </div>
          <h1 className="text-3xl font-bold text-white mb-2">Setup Required</h1>
          <p className="text-purple-200/70">Connect your Google Apps Script</p>
        </motion.div>
      </header>

      <main className="flex-1 px-4 pb-8">
        <div className="max-w-lg mx-auto">
          {/* Instructions */}
          <motion.div
            initial={{ opacity: 0, y: 20 }}
            animate={{ opacity: 1, y: 0 }}
            className="bg-white/10 backdrop-blur-lg border border-white/20 rounded-3xl p-6 mb-6"
          >
            <h2 className="text-lg font-semibold text-white mb-4 flex items-center gap-2">
              <Link className="w-5 h-5 text-purple-400" />
              How to get your API URL
            </h2>
            <ol className="space-y-3 text-purple-200/80 text-sm">
              <li className="flex gap-3">
                <span className="flex-shrink-0 w-6 h-6 rounded-full bg-purple-500/30 flex items-center
justify-center text-purple-300 text-xs font-bold">1</span>
                <span>Create a Google Sheet with the template</span>
              </li>
              <li className="flex gap-3">
                <span className="flex-shrink-0 w-6 h-6 rounded-full bg-purple-500/30 flex items-center
justify-center text-purple-300 text-xs font-bold">2</span>
                <span>Go to <strong>Extensions → Apps Script</strong></span>
              </li>
              <li className="flex gap-3">
                <span className="flex-shrink-0 w-6 h-6 rounded-full bg-purple-500/30 flex items-center
justify-center text-purple-300 text-xs font-bold">3</span>
                <span>Paste the Apps Script code provided</span>
              </li>
              <li className="flex gap-3">
                <span className="flex-shrink-0 w-6 h-6 rounded-full bg-purple-500/30 flex items-center
justify-center text-purple-300 text-xs font-bold">4</span>
                <span>Click <strong>Deploy → New deployment</strong></span>
              </li>
              <li className="flex gap-3">
                <span className="flex-shrink-0 w-6 h-6 rounded-full bg-purple-500/30 flex items-center
justify-center text-purple-300 text-xs font-bold">5</span>
                <span>Select <strong>Web app</strong> and set access to "Anyone"</span>
              </li>
              <li className="flex gap-3">
                <span className="flex-shrink-0 w-6 h-6 rounded-full bg-purple-500/30 flex items-center
justify-center text-purple-300 text-xs font-bold">6</span>
                <span>Copy the deployment URL and paste below</span>
              </li>
            </ol>
          </motion.div>

          {/* API URL Input */}
          <motion.div
            initial={{ opacity: 0, y: 20 }}
            animate={{ opacity: 1, y: 0 }}
            transition={{ delay: 0.1 }}
            className="space-y-4"
          >
            <div>
              <label className="block text-sm text-purple-300/70 mb-2">Google Apps Script URL</label>
              <input
                type="url"
                value={apiUrl}
                onChange={(e) => setApiUrlState(e.target.value)}
                placeholder="https://script.google.com/macros/s/..."
                className="w-full px-4 py-4 bg-white/10 backdrop-blur-lg border border-white/20
rounded-2xl text-white placeholder-purple-300/50 focus:outline-none focus:ring-2 focus:ring-purple-
500 text-sm"
              />
            </div>

            <motion.button
              onClick={handleSave}
              disabled={!apiUrl.trim() || saved}
              whileHover={{ scale: 1.02 }}
              whileTap={{ scale: 0.98 }}
              className={`w-full py-4 font-semibold rounded-2xl shadow-lg transition-all disabled:opacity-
50 flex items-center justify-center gap-2 ${
                saved
                  ? 'bg-emerald-600 text-white shadow-emerald-500/30'
                  : 'bg-gradient-to-r from-purple-600 to-pink-600 text-white shadow-purple-500/30
hover:shadow-purple-500/50'
              }`}
            >
              {saved ? (
                <>
                  <CheckCircle className="w-5 h-5" />
                  Saved! Redirecting...
                </>
              ) : (
                <>
                  <CheckCircle className="w-5 h-5" />
                  Save & Continue
                </>
              )}
            </motion.button>
          </motion.div>
          {/* Help Section */}
          <motion.div
            initial={{ opacity: 0 }}
            animate={{ opacity: 1 }}
            transition={{ delay: 0.3 }}
            className="mt-8 p-4 bg-amber-500/10 border border-amber-500/30 rounded-2xl"
          >
            <h3 className="text-amber-300 font-medium mb-2 text-sm">Need help?</h3>
            <p className="text-amber-200/70 text-xs mb-3">
              Check the documentation files included with this project:
            </p>
            <ul className="text-amber-200/70 text-xs space-y-1">
              <li>• <code className="bg-amber-500/20 px-1 
rounded">docs/GOOGLE_SHEET_TEMPLATE.md</code> - Sheet setup guide</li>
              <li>• <code className="bg-amber-500/20 px-1 
rounded">docs/GOOGLE_APPS_SCRIPT.md</code> - Complete API code</li>
            </ul>
          </motion.div>
        </div>
      </main>
    </div>
   );
}
```

Types folder

Inside folder : member.ts

```
// Member data types for the Gym Membership System

export interface Member {
  id: string;
  name: string;
  phone: string;
  age: number;
  weight: number;
  membershipType: '3 Months' | '6 Months' | '1 Year';
  startDate: string;
  endDate: string;
  status: 'Active' | 'Expired';
  daysRemaining?: number;
  notified?: boolean;
}

export interface MemberLookupResponse {
  success: boolean;
  member?: Member;
  error?: string;
}

export interface MembersListResponse {
  success: boolean;
  members?: Member[];
  error?: string;
}

export interface ApiResponse {
  success: boolean;
  message?: string;
  error?: string;
  newEndDate?: string;
}

export interface NewMemberData {
  id: string;
  name: string;
  phone: string;
  age: number;
  weight: number;
  membershipType: '3 Months' | '6 Months' | '1 Year';
  startDate: string;
}

export interface RenewalData {
  memberId: string;
  membershipType: '3 Months' | '6 Months' | '1 Year';
  startDate: string;
}
```

App.tsx

```
import { useState, useEffect } from 'react';
import { BrowserRouter, Routes, Route, Link, useLocation } from 'react-router-dom';
import { motion, AnimatePresence } from 'framer-motion';
import { Search, Shield, Settings, Menu, X } from 'lucide-react';
import MemberLookup from './pages/MemberLookup';
import AdminPanel from './pages/AdminPanel';
import Setup from './pages/Setup';
import { getApiUrl } from './api/gymApi';

function Navigation() {
  const location = useLocation();
  const [mobileMenuOpen, setMobileMenuOpen] = useState(false);
  const navItems = [
    { path: '/', label: 'Member Lookup', icon: Search },
    { path: '/admin', label: 'Admin Panel', icon: Shield },
    { path: '/setup', label: 'Setup', icon: Settings },
  ];

  return (
    <>
      {/* Desktop Navigation */}
      <nav className="fixed bottom-4 left-1/2 -translate-x-1/2 z-50 hidden sm:block">
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          className="flex items-center gap-2 px-4 py-2 bg-white/10 backdrop-blur-xl border border-
white/20 rounded-2xl shadow-xl"
        >
          {navItems.map(({ path, label, icon: Icon }) => (
            <Link
              key={path}
              to={path}
              className={`flex items-center gap-2 px-4 py-2 rounded-xl transition-all ${
                location.pathname === path
                  ? 'bg-gradient-to-r from-purple-600 to-pink-600 text-white shadow-lg shadow-purple-
500/30'
                  : 'text-purple-200 hover:bg-white/10'
              }`}
            >
              <Icon className="w-4 h-4" />
              <span className="text-sm font-medium">{label}</span>
            </Link>
          ))}
        </motion.div>
      </nav>


      {/* Mobile Navigation */}
      <div className="sm:hidden">
        <button
          onClick={() => setMobileMenuOpen(!mobileMenuOpen)}
          className="fixed bottom-4 right-4 z-50 w-14 h-14 bg-gradient-to-r from-purple-600 to-pink-
600 rounded-2xl shadow-lg shadow-purple-500/30 flex items-center justify-center text-white"
        >
          {mobileMenuOpen ? <X className="w-6 h-6" /> : <Menu className="w-6 h-6" />}
        </button>


        <AnimatePresence>
          {mobileMenuOpen && (
            <motion.div
              initial={{ opacity: 0, scale: 0.9, y: 20 }}
              animate={{ opacity: 1, scale: 1, y: 0 }}
              exit={{ opacity: 0, scale: 0.9, y: 20 }}
              className="fixed bottom-20 right-4 z-50 flex flex-col gap-2"
            >
              {navItems.map(({ path, label, icon: Icon }) => (
                <Link
                  key={path}
                  to={path}
                  onClick={() => setMobileMenuOpen(false)}
                  className={`flex items-center gap-3 px-4 py-3 rounded-xl backdrop-blur-xl border
transition-all ${
                    location.pathname === path
                      ? 'bg-gradient-to-r from-purple-600 to-pink-600 text-white border-purple-500/50
shadow-lg shadow-purple-500/30'
                      : 'bg-white/10 text-purple-200 border-white/20 hover:bg-white/20'
                  }`}
                >
                  <Icon className="w-5 h-5" />
                  <span className="font-medium">{label}</span>
                </Link>
              ))}
            </motion.div>
          )}
        </AnimatePresence>
      </div>
     </>
    );
}

function AppContent() {
     const [isSetup, setIsSetup] = useState(false);
     const location = useLocation();

     useEffect(() => {
      setIsSetup(!!getApiUrl());
     }, [location]);

     const handleSetupComplete = () => {
      setIsSetup(true);
     };

     // Show setup prompt if not configured (except on setup page)
     if (!isSetup && location.pathname !== '/setup') {
      return <Setup onComplete={handleSetupComplete} />;
     }

     return (
    <>
      <Routes>
        <Route path="/" element={<MemberLookup />} />
        <Route path="/admin" element={<AdminPanel />} />
        <Route path="/setup" element={<Setup onComplete={handleSetupComplete} />} />
      </Routes>
      <Navigation />
    </>
  );
}

function App() {
  return (
    <BrowserRouter>
      <AppContent />
    </BrowserRouter>
  );
}

export default App;
```

index.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

main.tsx

```
import { StrictMode } from "react";
import { createRoot } from "react-dom/client";
import "./index.css";
import App from "./App.tsx";

createRoot(document.getElementById("root")!).render(
  <StrictMode>
    <App />
  </StrictMode>
);
```

vite-env.d.ts

```
/* eslint-disable @typescript-eslint/no-explicit-any */
/// <reference types="vite/client" />

export {};
```

index.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Youware App</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.tsx"></script>
  </body>
```

```
</html>
```

pakage.json

```
{
  "name": "youware-react-template",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "@headlessui/react": "^1.7.18",
    "cannon-es": "^0.20.0",
    "clsx": "^2.1.0",
    "framer-motion": "^11.0.8",
    "gsap": "^3.13.0",
    "i18next": "^23.10.1",
    "i18next-browser-languagedetector": "^7.2.0",
    "lucide-react": "^0.533.0",
    "matter-js": "^0.20.0",
    "react": "^18.3.1",
    "react-dom": "^18.3.1",
    "react-i18next": "^14.1.0",
    "react-router-dom": "^6.30.1",
    "react-use": "^17.6.0",
    "three": "^0.179.1",
    "zod": "^3.25.67",
    "zustand": "^4.4.7"
  },
  "devDependencies": {
    "@types/matter-js": "^0.20.0",
    "@types/node": "^24.0.14",
    "@types/react": "^18.3.1",
    "@types/react-dom": "^18.3.1",
    "@vitejs/plugin-react": "^4.5.2",
    "@youware/vite-plugin-react": "^1.0.3",
    "autoprefixer": "^10.4.21",
    "postcss": "^8.5.6",
    "tailwindcss": "^3.4.17",
    "typescript": "~5.8.3",
    "vite": "^7.0.0"
  }
}
```

tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
export default {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

YOUWERE.md

# Gym Membership Lookup System

A complete gym membership management system with Google Sheets as the backend.

## Features

- **Public Member Lookup**: Anyone can check membership status using their Member ID
- **Admin Panel**: Gym owner can add, edit, renew, and delete members
- **Auto-calculated Status**: Membership status (Active/Expired) is calculated automatically
- **Email Notifications**: Owner receives email when memberships expire
- **Mobile-Friendly**: Works seamlessly on both mobile and desktop

## Project Structure

```
 ├── docs/
│   ├── GOOGLE_SHEET_TEMPLATE.md   # Step-by-step Sheet setup guide
│   └── GOOGLE_APPS_SCRIPT.md      # Complete Apps Script code with instructions
 ├── src/
│   ├── api/
│   │   └── gymApi.ts            # API client for Google Apps Script
│   ├── pages/
│   │   ├── MemberLookup.tsx      # Public search page
│   │   ├── AdminPanel.tsx        # Owner admin panel
│   │   └── Setup.tsx            # API configuration page
│   ├── types/
│   │   └── member.ts            # TypeScript interfaces
│   └── App.tsx                 # Main app with routing
```

## Setup Instructions

### 1. Set Up Google Sheet (5 minutes)

Follow the instructions in `docs/GOOGLE_SHEET_TEMPLATE.md`:

- Create a new Google Sheet

- Add column headers

- Set up data validation

- Add the End Date formula

### 2. Deploy Google Apps Script (10 minutes)

Follow the instructions in `docs/GOOGLE_APPS_SCRIPT.md`:

- Open Extensions → Apps Script

- Paste the provided code

- Configure your password and email

- Deploy as Web App

- Copy the deployment URL

### 3. Configure the Frontend

- Open the app

- Paste your Google Apps Script URL in the setup screen

- Start using the system!

## Data Model

Field	Description
Member ID	Unique identifier (e.g., GYM001)
Name	Member's full name
Phone	Contact number
Age	Member's age

Weight	Weight in kg
Membership Type	3 Months / 6 Months / 1 Year
Start Date	When membership began
End Date	Auto-calculated expiry date
Status	Calculated: Active or Expired

## Technology Stack

- **Frontend**: React + TypeScript + Tailwind CSS
- **Backend**: Google Sheets + Google Apps Script
- **Notifications**: Gmail (via Apps Script)
- **Cost**: 100% Free

## Security

- Public lookup only shows masked phone numbers
- Admin operations require password authentication
- Password stored in browser session only
- Google Apps Script handles all data securely

## Build Commands

```bash
npm install    # Install dependencies
npm run build  # Build for production
```

yw_manifest.json

{

```
    "project_type": "react"
}
```

At last this is the document and I want to create the exact type with some changes so take this code at first we will create exact replica of this then will make changes in this
okay