```
import React, { useState, useEffect } from 'react';
import ReactDOM from 'react-dom/client';
import axios from 'axios';
import { useTranslation } from 'react-i18next';
import { TFunction } from 'i18next';
import './i18n';
import './index.css';

type TabType = 'home' | 'payment' | 'status' | 'history';

interface TabContent {
  data: any;
  timestamp: number;
}

interface HeaderProps {
  user?: string;
  activeTab: TabType;
  onTabChange: (tab: TabType) => void;
  t: TFunction<'translation', undefined>;
  userId: string;
  setUserId: (value: string) => void;
  password: string;
  setPassword: (value: string) => void;
  onSignIn: () => Promise<void>;
  isLoading: boolean;
  isSignedIn: boolean;
}

const App: React.FC = () => {
  const { t } = useTranslation();
  const [isSignedIn, setIsSignedIn] = useState(false);
  const [userId, setUserId] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');
  const [activeTab, setActiveTab] = useState<TabType>('home');
  const [isLoading, setIsLoading] = useState(false);
  const [tabContents, setTabContents] = useState<Record<TabType, TabContent>>({
    home: { data: null, timestamp: 0 },
    payment: { data: null, timestamp: 0 },
    status: { data: null, timestamp: 0 },
    history: { data: null, timestamp: 0 }
  });

  const API_BASE_URL = process.env.REACT_APP_API_URL || 'http://localhost:3001/api';

  useEffect(() => {
    if (isSignedIn) {
      fetchTabContent(activeTab);
    }
  }, [activeTab, isSignedIn]);

  const fetchTabContent = async (tab: TabType) => {
    if (tabContents[tab].timestamp > Date.now() - 300000 && tabContents[tab].data) {
      return;
    }

    setIsLoading(true);
    setError('');
    try {
      const endpoint = `/${tab}-content`;
      const response = await axios.get(`${API_BASE_URL}${endpoint}`, {
        headers: {
          Authorization: `Bearer ${localStorage.getItem('authToken')}`
        }
      });

      setTabContents(prev => ({
        ...prev,
        [tab]: {
          data: response.data,
          timestamp: Date.now()
        }
      }));
    } catch (err) {
      setError(t(`errors.${tab}LoadFailed`));
      console.error(`Error loading ${tab} content:`, err);
    } finally {
      setIsLoading(false);
    }
  };

  const handleSignIn = async () => {
    try {
      setIsLoading(true);
      const response = await axios.post(`${API_BASE_URL}/auth/signin`, {
        userId: userId.toUpperCase(),
        password
      });

      localStorage.setItem('authToken', response.data.token);
      setIsSignedIn(true);
      setError('');
      fetchTabContent('home');
    } catch (err) {
      setError(t('errors.invalidCredentials'));
    } finally {
      setIsLoading(false);
    }
  };

  const handleTabChange = (tab: TabType) => {
    setActiveTab(tab);
    setError('');
  };

  const renderContent = () => {
    if (!isSignedIn) return null;
    if (isLoading) return <div className="p-8 text-center">{t('common.loading')}</div>;
    if (error) return <div className="p-8 text-red-500 text-center">{error}</div>;

    switch (activeTab) {
      case 'home':
        return <HomeContent data={tabContents.home.data} t={t} />;
      case 'payment':
        return <div>{t('payment.placeholder')}</div>;
      case 'status':
        return <StatusContent data={tabContents.status.data} t={t} />;
      case 'history':
        return <HistoryContent data={tabContents.history.data} t={t} />;
      default:
        return null;
    }
  };

  return (
    <div className="min-h-screen bg-gray-50 relative">
      <BackgroundDimmed />

      <Header
        isSignedIn={isSignedIn}
        activeTab={activeTab}
        onTabChange={handleTabChange}
        t={t}
        userId={userId}
        setUserId={setUserId}
        password={password}
        setPassword={setPassword}
        onSignIn={handleSignIn}
        isLoading={isLoading}
        user="User"
      />

      <main className="relative z-10 max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-8">
        {renderContent()}
      </main>
    </div>
  );
};

// Component Definitions
const BackgroundDimmed = () => (
  <div className="absolute inset-0 z-0">
    <img
      src="/NdaY_Logo.png"
      alt="App Background"
      className="w-full h-full object-cover opacity-10"
    />
  </div>
);

const Header: React.FC<HeaderProps> = ({
  isSignedIn,
  activeTab,
  onTabChange,
  t,
  userId,
  setUserId,
  password,
  setPassword,
  onSignIn,
  isLoading,
  user = "User"
}) => {
  return (
    <header className="bg-white shadow-md">
      <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
        <div className="flex items-center justify-between py-3 h-16">
          <div className="text-left min-w-[200px]">
          <p className="text-sm font-medium text-gray-600">
            {t('header.welcome', { user })}
          </p>
        </div>

        <div className="mx-4 flex-shrink-0" style={{ height: '150px', width: 'auto' }}>
      <img
        src="/NdaY_Ben_Tanana.png"
        alt="NdaY'Ben'Tanàna Logo"
        className="h-full w-full object-contain"
        style={{ maxWidth: '100%', height: '100%' }}
      />
    </div>

        <div className="text-right min-w-[200px]">
          <p className="text-sm font-medium text-gray-600">
            {t('header.tagline')}
          </p>
        </div>
      </div>

      <div className="border-t border-gray-200">
      <nav className="flex justify-center py-2">
  <ul className="flex space-x-6"> {/* Increased spacing between buttons */}
    {['home', 'payment', 'status', 'history'].map((tab) => (
      <li key={tab}>
        <button
          onClick={() => onTabChange(tab as TabType)}
          className={`text-base font-medium px-4 py-2 rounded-md ${
            activeTab === tab
              ? 'bg-blue-50 text-blue-600'
              : 'text-gray-600 hover:text-blue-600 hover:bg-blue-50'
          }`}
        >
          {t(`header.nav.${tab}`)}
        </button>
      </li>
    ))}
  </ul>
</nav>
      </div>
    </div>
  </header>
  );
};

// (Keep all your other components: HomeContent, PaymentContent, StatusContent,
HistoryContent, SignInForm)

// Home Content Component

interface ContentProps {
  data: any;
  t: (key: string) => string;
}

const HomeContent = ({ data, t }: ContentProps) => {
  return (
    <div className="space-y-6">
      <h2 className="text-2xl font-bold text-gray-800">{t('home.title')}</h2>
      {data ? (
        <div className="grid grid-cols-1 md:grid-cols-3 gap-6">
          {/* Dashboard Cards */}
          <div className="bg-white p-6 rounded-lg shadow">
            <h3 className="font-medium text-gray-700">{t('home.activeUsers')}</h3>
            <p className="text-3xl font-bold text-blue-600 mt-2">
              {data.activeUsers || 0}
            </p>
          </div>

          <div className="bg-white p-6 rounded-lg shadow">
            <h3 className="font-medium text-gray-700">{t('home.pendingPayments')}</h3>
            <p className="text-3xl font-bold text-orange-500 mt-2">
              {data.pendingPayments || 0}
            </p>
          </div>

          <div className="bg-white p-6 rounded-lg shadow">
            <h3 className="font-medium text-gray-700">{t('home.completedTransactions')}</h3>
            <p className="text-3xl font-bold text-green-500 mt-2">
              {data.completedTransactions || 0}
            </p>
          </div>

          {/* Recent Activity */}
          <div className="md:col-span-2 bg-white p-6 rounded-lg shadow">
            <h3 className="font-medium text-gray-700 mb-4">{t('home.recentActivity')}</h3>
            <ul className="space-y-3">
              {data.recentActivity?.map((activity: any) => (
                <li key={activity.id} className="flex items-center">
                  <span className="w-2 h-2 bg-blue-500 rounded-full mr-3"></span>
                  <span className="text-gray-600">{activity.description}</span>
                  <span className="ml-auto text-sm text-gray-500">
                    {new Date(activity.timestamp).toLocaleTimeString()}
                  </span>
                </li>
              ))}
            </ul>
          </div>
        </div>
      ) : (
        <p className="text-gray-500">{t('home.noData')}</p>
      )}
    </div>
  );
};

// Payment Content Component

// Removed duplicate StatusContent declaration

// Tax Status Content Component

const StatusContent = ({ data, t }: ContentProps) => {
  return (
    <div className="space-y-6">
      <h2 className="text-2xl font-bold text-gray-800">{t('status.title')}</h2>

      {data ? (
        <div className="bg-white p-6 rounded-lg shadow">
          <div className="space-y-4">
            {/* Service Status Cards */}
            <div className="flex items-start p-4 border border-gray-200 rounded-lg">
              <div className={`flex-shrink-0 h-5 w-5 rounded-full ${
                data.paymentService.status === 'operational'
                  ? 'bg-green-500'
                  : 'bg-red-500'
              }`}></div>
              <div className="ml-3">
                <h3 className="text-lg font-medium">{t('status.paymentService')}</h3>
                <p className="text-gray-600">
                  {t(`status.${data.paymentService.status}`)} - {data.paymentService.message}
                </p>
                {data.paymentService.incident && (
                  <div className="mt-2 p-2 bg-yellow-50 text-yellow-700 text-sm rounded">
                    {data.paymentService.incident}
                  </div>
                )}
              </div>
            </div>

            {/* Additional service status cards would go here */}

            {/* Maintenance Schedule */}
            <div className="mt-6">
              <h3 className="text-lg font-medium mb-3">{t('status.maintenance')}</h3>
              {data.maintenanceSchedule?.length > 0 ? (
                <ul className="space-y-2">
                  {data.maintenanceSchedule.map((item: any) => (
                    <li key={item.id} className="flex justify-between p-2 bg-gray-50 rounded">
                      <span>{item.description}</span>
                      <span className="text-gray-500">
                        {new Date(item.start).toLocaleString()} - {new Date(item.end).toLocaleTimeString()}
                      </span>
                    </li>
                  ))}
                </ul>
              ) : (
                <p className="text-gray-500">{t('status.noMaintenance')}</p>
              )}
            </div>
          </div>
        </div>
      ) : (
        <p className="text-gray-500">{t('status.noData')}</p>
      )}
    </div>
  );
};

// HistoryContent Component

const HistoryContent = ({ data, t }: ContentProps) => {
  const [filter, setFilter] = useState('all');
  const [searchQuery, setSearchQuery] = useState('');

  const filteredData = data?.filter((item: any) => {
    const matchesFilter = filter === 'all' || item.type === filter;
    const matchesSearch = item.description.toLowerCase().includes(searchQuery.toLowerCase());
    return matchesFilter && matchesSearch;
  });

  return (
    <div className="space-y-6">
      <h2 className="text-2xl font-bold text-gray-800">{t('history.title')}</h2>

      {/* Filters */}
      <div className="bg-white p-4 rounded-lg shadow">
        <div className="flex flex-col sm:flex-row sm:items-center sm:justify-between gap-4">
          <div className="relative">
            <input
              type="text"
              placeholder={t('history.search')}
              className="pl-10 pr-4 py-2 border border-gray-300 rounded-md w-full"
              value={searchQuery}
              onChange={(e) => setSearchQuery(e.target.value)}
            />
            <div className="absolute left-3 top-2.5 text-gray-400">
              <svg className="w-5 h-5" fill="none" stroke="currentColor" viewBox="0 0 24 24">
                <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M21
21l-6-6m2-5a7 7 0 11-14 0 7 7 0 0114 0z" />
              </svg>
            </div>
          </div>

          <select
            value={filter}
            onChange={(e) => setFilter(e.target.value)}
            className="border border-gray-300 rounded-md px-4 py-2"
          >
            <option value="all">{t('history.allActivities')}</option>
            <option value="payment">{t('history.payments')}</option>
            <option value="transfer">{t('history.transfers')}</option>
            <option value="account">{t('history.accountChanges')}</option>
          </select>
        </div>
      </div>

      {/* History List */}
      {filteredData?.length > 0 ? (
        <div className="bg-white rounded-lg shadow overflow-hidden">
          <ul className="divide-y divide-gray-200">
            {filteredData.map((item: any) => (
              <li key={item.id} className="p-4 hover:bg-gray-50">
                <div className="flex items-center">
                  <div className={`flex-shrink-0 h-10 w-10 rounded-full flex items-center justify-center
${
                    item.type === 'payment' ? 'bg-blue-100' :
                    item.type === 'transfer' ? 'bg-green-100' : 'bg-purple-100'
                  }`}>
                    {item.type === 'payment' ? (
                      <svg className="h-5 w-5 text-blue-500" fill="none" stroke="currentColor"
viewBox="0 0 24 24">
                        <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M12
8c-1.657 0-3 .895-3 2s1.343 2 3 2 3 .895 3 2-1.343 2-3 2m0-8c1.11 0 2.08.402 2.599 1M12
8V7m0 1v8m0 0v1m0-1c-1.11 0-2.08-.402-2.599-1M21 12a9 9 0 11-18 0 9 9 0 0118 0z" />
                      </svg>
                    ) : (
                      <svg className="h-5 w-5 text-green-500" fill="none" stroke="currentColor"
viewBox="0 0 24 24">
                        <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M7 16a4
4 0 01-.88-7.903A5 5 0 1115.9 6L16 6a5 5 0 011 9.9M9 19l3 3m0 0l3-3m-3 3V10" />
                      </svg>
                    )}
                  </div>
                  <div className="ml-4 flex-1">
                    <div className="flex items-center justify-between">
                      <p className="text-sm font-medium text-gray-900">{item.description}</p>
                      <p className={`text-sm font-medium ${
                        item.amount > 0 ? 'text-green-600' : 'text-red-600'
                      }`}>
                        {item.amount > 0 ? '+' : ''}{item.amount}
                      </p>
                    </div>
                    <div className="flex items-center justify-between mt-1">
                      <p className="text-sm text-gray-500">{new Date(item.date).toLocaleString()}</p>
                      <span className={`inline-flex items-center px-2.5 py-0.5 rounded-full text-xs font-
medium ${
                        item.status === 'completed' ? 'bg-green-100 text-green-800' :
                        item.status === 'failed' ? 'bg-red-100 text-red-800' : 'bg-yellow-100 text-
yellow-800'
                      }`}>
                        {t(`history.status.${item.status}`)}
                      </span>
                    </div>
                  </div>
                </div>
              </li>
            ))}
          </ul>
        </div>
      ) : (
        <div className="bg-white p-8 rounded-lg shadow text-center">
          <p className="text-gray-500">{t('history.noResults')}</p>
        </div>
      )}
    </div>
   );
};

//SignInForm Component

interface SignInFormProps {
    userId: string;
    setUserId: (value: string) => void;
    password: string;
    setPassword: (value: string) => void;
    onSignIn: () => void;
    isLoading: boolean;
    t: (key: string) => string;
}

const SignInForm = ({
    userId,
    setUserId,
    password,
    setPassword,
    onSignIn,
  isLoading,
  t
}: SignInFormProps) => {
  const [validationError, setValidationError] = useState<string | null>(null);
  const [showPassword, setShowPassword] = useState(false);

  const handleSubmit = (e: React.FormEvent) => {
    e.preventDefault();
    setValidationError(null);

    // Basic validation
    if (!userId.trim()) {
      setValidationError(t('errors.userIdRequired'));
      return;
    }

    if (!password) {
      setValidationError(t('errors.passwordRequired'));
      return;
    }

    if (password.length < 6) {
      setValidationError(t('errors.passwordTooShort'));
      return;
    }

    onSignIn();
  };

  return (
    <div className="flex items-center space-x-4">
      <form onSubmit={handleSubmit} className="flex flex-col sm:flex-row gap-3">
        {/* User ID Field */}
        <div className="relative">
          <div className="absolute inset-y-0 left-0 pl-3 flex items-center pointer-events-none">
            <svg className="h-5 w-5 text-gray-400" fill="none" stroke="currentColor" viewBox="0 0
24 24">
              <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M16 7a4 4 0
11-8 0 4 4 0 018 0zM12 14a7 7 0 00-7 7h14a7 7 0 00-7-7z" />
            </svg>
          </div>
          <input
            type="text"
            value={userId}
            onChange={(e) => setUserId(e.target.value.toUpperCase())}
            placeholder={t('auth.userIdPlaceholder')}
            className={`pl-10 pr-4 py-2 border ${
              validationError?.includes('ID') ? 'border-red-500' : 'border-gray-300'
            } rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 text-sm`}
            disabled={isLoading}
          />
        </div>

        {/* Password Field */}
        <div className="relative">
          <div className="absolute inset-y-0 left-0 pl-3 flex items-center pointer-events-none">
            <svg className="h-5 w-5 text-gray-400" fill="none" stroke="currentColor" viewBox="0 0
24 24">
```

```
       <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M12 15v2m-6
4h12a2 2 0 002-2v-6a2 2 0 00-2-2H6a2 2 0 00-2 2v6a2 2 0 002 2zm10-10V7a4 4 0 00-8
0v4h8z" />
      </svg>
     </div>
     <input
      type={showPassword ? "text" : "password"}
      value={password}
      onChange={(e) => setPassword(e.target.value)}
      placeholder={t('auth.passwordPlaceholder')}
      className={`pl-10 pr-10 py-2 border ${
       validationError?.includes('password') ? 'border-red-500' : 'border-gray-300'
      } rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 text-sm`}
      disabled={isLoading}
     />
     <button
      type="button"
      onClick={() => setShowPassword(!showPassword)}
      className="absolute inset-y-0 right-0 pr-3 flex items-center"
     >
      {showPassword ? (
       <svg className="h-5 w-5 text-gray-400" fill="none" stroke="currentColor" viewBox="0
0 24 24">
        <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M13.875
18.825A10.05 10.05 0 0112 19c-4.478 0-8.268-2.943-9.543-7a9.97 9.97 0
011.563-3.029m5.858.908a3 3 0 114.243 4.243M9.878 9.878l4.242 4.242M9.88
9.88l-3.29-3.29m7.532 7.532l3.29 3.29M3 3l3.59 3.59m0 0A9.953 9.953 0 0112 5c4.478 0 8.268
2.943 9.543 7a10.025 10.025 0 01-4.132 5.411m0 0L21 21" />
       </svg>
      ) : (
       <svg className="h-5 w-5 text-gray-400" fill="none" stroke="currentColor" viewBox="0
0 24 24">
        <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M15 12a3 3 0
11-6 0 3 3 0 016 0z" />
        <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M2.458
12C3.732 7.943 7.523 5 12 5c4.478 0 8.268 2.943 9.542 7-1.274 4.057-5.064 7-9.542 7-4.477
0-8.268-2.943-9.542-7z" />
       </svg>
      )}
     </button>
    </div>

    {/* Submit Button */}
    <button
     type="submit"
     disabled={isLoading}
     className={`bg-blue-600 text-white px-4 py-2 rounded-lg hover:bg-blue-700 text-sm ${
      isLoading ? 'opacity-75 cursor-not-allowed' : ''
     }`}
    >
     {isLoading ? (
      <span className="flex items-center justify-center">
       <svg className="animate-spin -ml-1 mr-2 h-4 w-4 text-white" xmlns="http://
www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">
        <circle className="opacity-25" cx="12" cy="12" r="10" stroke="currentColor"
strokeWidth="4"></circle>
        <path className="opacity-75" fill="currentColor" d="M4 12a8 8 0 018-8V0C5.373 0 0
5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"></path>
       </svg>
       {t('auth.signingIn')}
```

```
            </span>
          ) : (
            t('auth.signInButton')
          )}
        </button>
      </form>

      {/* Error Display */}
      {(validationError) && (
        <div className="absolute mt-16 sm:mt-12 ml-2 text-red-500 text-sm">
          {validationError}
        </div>
      )}
    </div>
  );
};

const root = ReactDOM.createRoot(document.getElementById('root')!);
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

```
import React, { useState, useEffect } from 'react';
import ReactDOM from 'react-dom/client';
import axios from 'axios';
import { useTranslation } from 'react-i18next';
import { TFunction } from 'i18next';
import './i18n';
import './index.css';

// Define Tab Types
type TabType = 'home' | 'payment' | 'status' | 'history';

interface TabContent {
  data: any;
  timestamp: number;
}

interface HeaderProps {
  user?: string;
  activeTab: TabType;
  onTabChange: (tab: TabType) => void;
  t: TFunction<'translation', undefined>;
  userId: string;
  setUserId: (value: string) => void;
  password: string;
  setPassword: (value: string) => void;
  onSignIn: () => Promise<void>;
  isLoading: boolean;
  isSignedIn: boolean;
}

const App: React.FC = () => {
  const { t } = useTranslation();
  const [isSignedIn, setIsSignedIn] = useState(false);
  const [userId, setUserId] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');
  const [activeTab, setActiveTab] = useState<TabType>('home');
  const [isLoading, setIsLoading] = useState(false);
  const [tabContents, setTabContents] = useState<Record<TabType, TabContent>>({
    home: { data: null, timestamp: 0 },
    payment: { data: null, timestamp: 0 },
    status: { data: null, timestamp: 0 },
    history: { data: null, timestamp: 0 }
  });

  const API_BASE_URL = process.env.REACT_APP_API_URL || 'http://localhost:3001/api';

  useEffect(() => {
    if (isSignedIn) {
      fetchTabContent(activeTab);
    }
  }, [activeTab, isSignedIn]);

  const fetchTabContent = async (tab: TabType) => {
    if (tabContents[tab].timestamp > Date.now() - 300000 && tabContents[tab].data) {
      return;
    }

    setIsLoading(true);
    setError('');
    try {
      const endpoint = `/${tab}-content`;
      const response = await axios.get(`${API_BASE_URL}${endpoint}`, {
        headers: {
          Authorization: `Bearer ${localStorage.getItem('authToken')}`
        }
      });

      setTabContents(prev => ({
        ...prev,
        [tab]: {
          data: response.data,
          timestamp: Date.now()
        }
      }));
    } catch (err) {
      setError(t(`errors.${tab}LoadFailed`));
      console.error(`Error loading ${tab} content:`, err);
    } finally {
      setIsLoading(false);
    }
  };

  const handleSignIn = async () => {
    try {
      setIsLoading(true);
      const response = await axios.post(`${API_BASE_URL}/auth/signin`, {
        userId: userId.toUpperCase(),
        password
      });

      localStorage.setItem('authToken', response.data.token);
      setIsSignedIn(true);
      setError('');
      fetchTabContent('home');
    } catch (err) {
      setError(t('errors.invalidCredentials'));
    } finally {
      setIsLoading(false);
    }
  };

  const handleTabChange = (tab: TabType) => {
    setActiveTab(tab);
    setError('');
  };

  const renderContent = () => {
    if (!isSignedIn) return null;
    if (isLoading) return <div className="p-8 text-center">{t('common.loading')}</div>;
    if (error) return <div className="p-8 text-red-500 text-center">{error}</div>;

    switch (activeTab) {
      case 'home':
        return <HomeContent data={tabContents.home.data} t={t} />;
      case 'payment':
        return <div>{t('payment.placeholder')}</div>;
      case 'status':
        return <StatusContent data={tabContents.status.data} t={t} />;
      case 'history':
        return <HistoryContent data={tabContents.history.data} t={t} />;
      default:
        return null;
    }
  };

  return (
    <div className="min-h-screen bg-gray-50 relative">
      <BackgroundDimmed />

      <Header
        isSignedIn={isSignedIn}
        activeTab={activeTab}
        onTabChange={handleTabChange}
        t={t}
        userId={userId}
        setUserId={setUserId}
        password={password}
        setPassword={setPassword}
        onSignIn={handleSignIn}
        isLoading={isLoading}
        user="User"
      />

      <main className="relative z-10 max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-8">
        {renderContent()}
      </main>
    </div>
  );
};

// Component Definitions
const BackgroundDimmed = () => (
  <div className="absolute inset-0 z-0">
    <img
      src="/NdaY_Logo.png"
      alt="App Background"
      className="w-full h-full object-cover opacity-10"
    />
  </div>
);

const Header: React.FC<HeaderProps> = ({
  isSignedIn,
  activeTab,
  onTabChange,
  t,
  userId,
  setUserId,
  password,
  setPassword,
  onSignIn,
  isLoading,
  user = "User"
}) => {
  return (
    <header className="bg-white shadow-md">
      <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
      {/* Top Section: Welcome Text, Logo, and Tagline */}
```

```
        <div className="grid grid-cols-3 items-center py-3 h-16">
        {/* Left Component: Welcome Text */}
          <div className="flex items-center justify-start">
           <p
            className="text-sm font-medium text-gray-600"
            style={{ display: 'flex', flexDirection: 'column',
              alignItems: 'center', justifyContent: 'flex-start',
              fontSize: '30px', fontWeight: 'bold',
              color: '#4A5568', // Text color
              lineHeight: '1.5',
            }}
            >
            {t('header.welcome', { user })}
            </p>
          </div>

        {/* Center Component: Logo */}
          <div className="flex items-center justify-center"
          style={{ height: '180px', width: 'auto', display: 'flex',
            flexDirection: 'column', maxWidth: '100%', maxHeight: '100%', alignItems: 'center',
            }}>
            {/* Logo Image */}
            <img
              src="/NdaY_Ben_Tanana.png"
              alt="NdaY'Ben'Tanàna Logo"
              className="h-[50px] w-auto object-contain"
              style={{ maxWidth: '100%', height: '100%' }} // Fixed height for the logo
            />
          </div>

        {/* Right Component: Tagline */}
          <div className="flex items-center justify-center">
           <p
            className="text-sm font-medium text-gray-600"
            style={{
            fontSize: '10px',
            color: '#4A5568', // Text color
            textAlign: 'right', // Centered text
            margin: '0', padding: '0', // Reset margin and padding
            fontFamily: 'Arial, sans-serif', // Font family
            fontStyle: 'italic', // Italic text
            lineHeight: '1.5', justifyContent: 'flex-end',
            display: 'flex', flexDirection: 'column', maxWidth: '100%', maxHeight: '100%',
            height: '100%', width: 'auto',  // Added alignContent
            alignItems: 'center', // Added alignItems
            }}
            >
            {t('header.tagline')}
            </p>
          </div>
        </div>
        {/* Bottom Section: Navigation Tabs */}
{/* Bottom Section: Navigation Tabs */}
<div className="border-t border-gray-200">
  <nav className="flex justify-center py-2">
    <ul className="flex w-full"> {/* Full width to allow resizing */}
      {['home', 'payment', 'status', 'history'].map((tab) => (
        <li key={tab} className="flex-grow text-center">
          <button
            onClick={() => onTabChange(tab as TabType)}
            className={`w-full text-base font-medium px-4 py-2 rounded-md transition duration-150
ease-in-out ${
              activeTab === tab
                ? 'bg-blue-50 text-blue-600'
                : 'text-gray-600 hover:text-blue-600 hover:bg-blue-50'
            }`}
            style={{
              fontSize: 'clamp(0.8rem, 2vw, 1.2rem)', // Dynamic font size
            }}
            disabled={isLoading}
          >
            {t(`header.nav.${tab}`)}
          </button>
        </li>
      ))}
    </ul>
  </nav>
</div>
      </div>
    </header>
  );
};

// Main Component
const Dashboard = ({ t }: { t: (key: string) => string }) => {
  // State for Active Tab
  const [activeTab, setActiveTab] = useState<TabType>('home');

  // State for Sign-In Modal
  const [isSignInOpen, setIsSignInOpen] = useState(false);

  // Mock Data for Tabs
  const mockData = {
    home: {
      activeUsers: 120,
      pendingPayments: 5,
      completedTransactions: 250,
      recentActivity: [
        { id: 1, description: 'User logged in', timestamp: new Date() },
        { id: 2, description: 'Payment processed', timestamp: new Date() },
      ],
    },
    payment: {}, // Replace with actual data
    status: {}, // Replace with actual data
    history: [], // Replace with actual data
  };

  // Handle Tab Change
  const handleTabChange = (tab: TabType) => {
    setActiveTab(tab);
  };

// (Keep all your other components: HomeContent, PaymentContent, StatusContent,
HistoryContent, SignInForm)

// Home Content Component

interface ContentProps {
  data: any;
  t: (key: string) => string;
```

```
}

const HomeContent = ({ data, t }: ContentProps) => {
  return (
    <div className="space-y-6">
      <h2 className="text-2xl font-bold text-gray-800">{t('home.title')}</h2>

      {data ? (
        <div className="grid grid-cols-1 md:grid-cols-3 gap-6">
          {/* Dashboard Cards */}
          <div className="bg-white p-6 rounded-lg shadow">
            <h3 className="font-medium text-gray-700">{t('home.activeUsers')}</h3>
            <p className="text-3xl font-bold text-blue-600 mt-2">
              {data.activeUsers || 0}
            </p>
          </div>

          <div className="bg-white p-6 rounded-lg shadow">
            <h3 className="font-medium text-gray-700">{t('home.pendingPayments')}</h3>
            <p className="text-3xl font-bold text-orange-500 mt-2">
              {data.pendingPayments || 0}
            </p>
          </div>

          <div className="bg-white p-6 rounded-lg shadow">
            <h3 className="font-medium text-gray-700">{t('home.completedTransactions')}</h3>
            <p className="text-3xl font-bold text-green-500 mt-2">
              {data.completedTransactions || 0}
            </p>
          </div>

          {/* Recent Activity */}
          <div className="md:col-span-2 bg-white p-6 rounded-lg shadow">
            <h3 className="font-medium text-gray-700 mb-4">{t('home.recentActivity')}</h3>
            <ul className="space-y-3">
              {data.recentActivity?.map((activity: any) => (
                <li key={activity.id} className="flex items-center">
                  <span className="w-2 h-2 bg-blue-500 rounded-full mr-3"></span>
                  <span className="text-gray-600">{activity.description}</span>
                  <span className="ml-auto text-sm text-gray-500">
                    {new Date(activity.timestamp).toLocaleTimeString()}
                  </span>
                </li>
              ))}
            </ul>
          </div>
        </div>
      ) : (
        <p className="text-gray-500">{t('home.noData')}</p>
      )}
    </div>
  );
};

// Payment Content Component



// Tax Status Content Component
```

```
const StatusContent = ({ data, t }: ContentProps) => {
  return (
    <div className="space-y-6">
      <h2 className="text-2xl font-bold text-gray-800">{t('status.title')}</h2>

      {data ? (
        <div className="bg-white p-6 rounded-lg shadow">
          <div className="space-y-4">
            {/* Service Status Cards */}
            <div className="flex items-start p-4 border border-gray-200 rounded-lg">
              <div className={`flex-shrink-0 h-5 w-5 rounded-full ${
                data.paymentService.status === 'operational'
                  ? 'bg-green-500'
                  : 'bg-red-500'
              }`}></div>
              <div className="ml-3">
                <h3 className="text-lg font-medium">{t('status.paymentService')}</h3>
                <p className="text-gray-600">
                  {t(`status.${data.paymentService.status}`)} - {data.paymentService.message}
                </p>
                {data.paymentService.incident && (
                  <div className="mt-2 p-2 bg-yellow-50 text-yellow-700 text-sm rounded">
                    {data.paymentService.incident}
                  </div>
                )}
              </div>
            </div>

            {/* Additional service status cards would go here */}

            {/* Maintenance Schedule */}
            <div className="mt-6">
              <h3 className="text-lg font-medium mb-3">{t('status.maintenance')}</h3>
              {data.maintenanceSchedule?.length > 0 ? (
                <ul className="space-y-2">
                  {data.maintenanceSchedule.map((item: any) => (
                    <li key={item.id} className="flex justify-between p-2 bg-gray-50 rounded">
                      <span>{item.description}</span>
                      <span className="text-gray-500">
                        {new Date(item.start).toLocaleString()} - {new Date(item.end).toLocaleTimeString()}
                      </span>
                    </li>
                  ))}
                </ul>
              ) : (
                <p className="text-gray-500">{t('status.noMaintenance')}</p>
              )}
            </div>
          </div>
        </div>
      ) : (
        <p className="text-gray-500">{t('status.noData')}</p>
      )}
    </div>
  );
};

// HistoryContent Component

const HistoryContent = ({ data, t }: ContentProps) => {
  const [filter, setFilter] = useState('all');
  const [searchQuery, setSearchQuery] = useState('');

  const filteredData = data?.filter((item: any) => {
    const matchesFilter = filter === 'all' || item.type === filter;
    const matchesSearch = item.description.toLowerCase().includes(searchQuery.toLowerCase());
    return matchesFilter && matchesSearch;
  });

  return (
    <div className="space-y-6">
      <h2 className="text-2xl font-bold text-gray-800">{t('history.title')}</h2>

      {/* Filters */}
      <div className="bg-white p-4 rounded-lg shadow">
        <div className="flex flex-col sm:flex-row sm:items-center sm:justify-between gap-4">
          <div className="relative">
            <input
              type="text"
              placeholder={t('history.search')}
              className="pl-10 pr-4 py-2 border border-gray-300 rounded-md w-full"
              value={searchQuery}
              onChange={(e) => setSearchQuery(e.target.value)}
            />
            <div className="absolute left-3 top-2.5 text-gray-400">
              <svg className="w-5 h-5" fill="none" stroke="currentColor" viewBox="0 0 24 24">
                <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M21
21l-6-6m2-5a7 7 0 11-14 0 7 7 0 0114 0z" />
              </svg>
            </div>
          </div>

          <select
            value={filter}
            onChange={(e) => setFilter(e.target.value)}
            className="border border-gray-300 rounded-md px-4 py-2"
          >
            <option value="all">{t('history.allActivities')}</option>
            <option value="payment">{t('history.payments')}</option>
            <option value="transfer">{t('history.transfers')}</option>
            <option value="account">{t('history.accountChanges')}</option>
          </select>
        </div>
      </div>

      {/* History List */}
      {filteredData?.length > 0 ? (
        <div className="bg-white rounded-lg shadow overflow-hidden">
          <ul className="divide-y divide-gray-200">
            {filteredData.map((item: any) => (
              <li key={item.id} className="p-4 hover:bg-gray-50">
                <div className="flex items-center">
                  <div className={`flex-shrink-0 h-10 w-10 rounded-full flex items-center justify-center
${
                    item.type === 'payment' ? 'bg-blue-100' :
                    item.type === 'transfer' ? 'bg-green-100' : 'bg-purple-100'
                  }`}>
                    {item.type === 'payment' ? (
                      <svg className="h-5 w-5 text-blue-500" fill="none" stroke="currentColor"
viewBox="0 0 24 24">
```

```
            <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M12
8c-1.657 0-3 .895-3 2s1.343 2 3 2 3 .895 3 2-1.343 2-3 2m0-8c1.11 0 2.08.402 2.599 1M12
8V7m0 1v8m0 0v1m0-1c-1.11 0-2.08-.402-2.599-1M21 12a9 9 0 11-18 0 9 9 0 0118 0z" />
           </svg>
          ) : (
           <svg className="h-5 w-5 text-green-500" fill="none" stroke="currentColor"
viewBox="0 0 24 24">
            <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M7 16a4
4 0 01-.88-7.903A5 5 0 1115.9 6L16 6a5 5 0 011 9.9M9 19l3 3m0 0l3-3m-3 3V10" />
           </svg>
          )}
         </div>
         <div className="ml-4 flex-1">
          <div className="flex items-center justify-between">
           <p className="text-sm font-medium text-gray-900">{item.description}</p>
           <p className={`text-sm font-medium ${
            item.amount > 0 ? 'text-green-600' : 'text-red-600'
           }`}>
            {item.amount > 0 ? '+' : ''}{item.amount}
           </p>
          </div>
          <div className="flex items-center justify-between mt-1">
           <p className="text-sm text-gray-500">{new Date(item.date).toLocaleString()}</p>
           <span className={`inline-flex items-center px-2.5 py-0.5 rounded-full text-xs font-
medium ${
            item.status === 'completed' ? 'bg-green-100 text-green-800' :
            item.status === 'failed' ? 'bg-red-100 text-red-800' : 'bg-yellow-100 text-
yellow-800'
           }`}>
            {t(`history.status.${item.status}`)}
           </span>
          </div>
         </div>
        </div>
       </li>
      ))}
     </ul>
    </div>
   ) : (
    <div className="bg-white p-8 rounded-lg shadow text-center">
     <p className="text-gray-500">{t('history.noResults')}</p>
    </div>
   )}
  </div>
 );
};

//SignInForm Component

interface SignInFormProps {
 userId: string;
 setUserId: (value: string) => void;
 password: string;
 setPassword: (value: string) => void;
 onSignIn: () => void;
 isLoading: boolean;
 t: (key: string) => string;
}

const SignInForm = ({
 userId,
 setUserId,
 password,
 setPassword,
 onSignIn,
 isLoading,
 t
}: SignInFormProps) => {
 const [validationError, setValidationError] = useState<string | null>(null);
 const [showPassword, setShowPassword] = useState(false);

 const handleSubmit = (e: React.FormEvent) => {
  e.preventDefault();
  setValidationError(null);

  // Basic validation
  if (!userId.trim()) {
   setValidationError(t('errors.userIdRequired'));
   return;
  }

  if (!password) {
   setValidationError(t('errors.passwordRequired'));
   return;
  }

  if (password.length < 6) {
   setValidationError(t('errors.passwordTooShort'));
   return;
  }

  onSignIn();
 };

 return (
  <div className="flex items-center space-x-4">
   <form onSubmit={handleSubmit} className="flex flex-col sm:flex-row gap-3">
    {/* User ID Field */}
    <div className="relative">
     <div className="absolute inset-y-0 left-0 pl-3 flex items-center pointer-events-none">
      <svg className="h-5 w-5 text-gray-400" fill="none" stroke="currentColor" viewBox="0 0
24 24">
       <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M16 7a4 4 0
11-8 0 4 4 0 018 0zM12 14a7 7 0 00-7 7h14a7 7 0 00-7-7z" />
      </svg>
     </div>
     <input
      type="text"
      value={userId}
      onChange={(e) => setUserId(e.target.value.toUpperCase())}
      placeholder={t('auth.userIdPlaceholder')}
      className={`pl-10 pr-4 py-2 border ${
       validationError?.includes('ID') ? 'border-red-500' : 'border-gray-300'
      } rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 text-sm`}
      disabled={isLoading}
     />
    </div>

    {/* Password Field */}
    <div className="relative">
    <div className="absolute inset-y-0 left-0 pl-3 flex items-center pointer-events-none">
     <svg className="h-5 w-5 text-gray-400" fill="none" stroke="currentColor" viewBox="0 0 
24 24">
      <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M12 15v2m-6 
4h12a2 2 0 002-2v-6a2 2 0 00-2-2H6a2 2 0 00-2 2v6a2 2 0 002 2zm10-10V7a4 4 0 00-8 
0v4h8z" />
     </svg>
    </div>
    <input
     type={showPassword ? "text" : "password"}
     value={password}
     onChange={(e) => setPassword(e.target.value)}
     placeholder={t('auth.passwordPlaceholder')}
     className={`pl-10 pr-10 py-2 border ${
      validationError?.includes('password') ? 'border-red-500' : 'border-gray-300'
     } rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 text-sm`}
     disabled={isLoading}
    />
    <button
     type="button"
     onClick={() => setShowPassword(!showPassword)}
     className="absolute inset-y-0 right-0 pr-3 flex items-center"
    >
     {showPassword ? (
      <svg className="h-5 w-5 text-gray-400" fill="none" stroke="currentColor" viewBox="0 
0 24 24">
       <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M13.875 
18.825A10.05 10.05 0 0112 19c-4.478 0-8.268-2.943-9.543-7a9.97 9.97 0 
011.563-3.029m5.858.908a3 3 0 114.243 4.243M9.878 9.878l4.242 4.242M9.88 
9.88l-3.29-3.29m7.532 7.532l3.29 3.29M3 3l3.59 3.59m0 0A9.953 9.953 0 0112 5c4.478 0 8.268 
2.943 9.543 7a10.025 10.025 0 01-4.132 5.411m0 0L21 21" />
      </svg>
     ) : (
      <svg className="h-5 w-5 text-gray-400" fill="none" stroke="currentColor" viewBox="0 
0 24 24">
       <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M15 12a3 3 0 
11-6 0 3 3 0 016 0z" />
       <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M2.458 
12C3.732 7.943 7.523 5 12 5c4.478 0 8.268 2.943 9.542 7-1.274 4.057-5.064 7-9.542 7-4.477 
0-8.268-2.943-9.542-7z" />
      </svg>
     )}
    </button>
   </div>

   {/* Submit Button */}
   <button
    type="submit"
    disabled={isLoading}
    className={`bg-blue-600 text-white px-4 py-2 rounded-lg hover:bg-blue-700 text-sm ${
     isLoading ? 'opacity-75 cursor-not-allowed' : ''
    }`}
   >
    {isLoading ? (
     <span className="flex items-center justify-center">
      <svg className="animate-spin -ml-1 mr-2 h-4 w-4 text-white" xmlns="http://
www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">
       <circle className="opacity-25" cx="12" cy="12" r="10" stroke="currentColor" 
strokeWidth="4"></circle>
```

```
                <path className="opacity-75" fill="currentColor" d="M4 12a8 8 0 018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"></path>
              </svg>
              {t('auth.signingIn')}
            </span>
          ) : (
            t('auth.signInButton')
          )}
        </button>
      </form>

      {/* Error Display */}
      {(validationError) && (
        <div className="absolute mt-16 sm:mt-12 ml-2 text-red-500 text-sm">
          {validationError}
        </div>
      )}
    </div>
  );
};

const root = ReactDOM.createRoot(document.getElementById('root')!);
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```