```
import streamlit as st
from ytmusicapi import YTMusic
import logging
import json
import os
import pickle
import subprocess
import sqlite3
import platform
from google.auth.transport.requests import Request
from google.oauth2.credentials import Credentials
from google_auth_oauthlib.flow import Flow
import webbrowser
import time
import shutil

# Configure logging
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

# OAuth 2.0 scopes for YouTube Music
SCOPES = ['https://www.googleapis.com/auth/youtube']

# Page config
st.set_page_config(
    page_title="🎵 YouTube Music Auth Setup",
```

```
    page_icon="🎵",
    layout="wide",
    initial_sidebar_state="expanded"
)

def get_chrome_logged_in_accounts():
    """Get logged-in Google accounts from Chrome"""
    try:
        accounts = []
        system = platform.system()

        if system == "Windows":
            chrome_user_data = os.path.expanduser(r"~\AppData\Local\Google\Chrome\User
Data")
        elif system == "Darwin":  # macOS
            chrome_user_data = os.path.expanduser("~/Library/Application
Support/Google/Chrome")
        else:  # Linux
            chrome_user_data = os.path.expanduser("~/.config/google-chrome")

        if not os.path.exists(chrome_user_data):
            return accounts

        # Look for profile directories
        for item in os.listdir(chrome_user_data):
            profile_path = os.path.join(chrome_user_data, item)
```

```
if os.path.isdir(profile_path) and (item.startswith("Profile") or item == "Default"):

    # Try to get account info from preferences
    prefs_file = os.path.join(profile_path, "Preferences")
    if os.path.exists(prefs_file):
        try:
            with open(prefs_file, 'r', encoding='utf-8') as f:
                prefs = json.load(f)

            # Look for Google account info
            if 'account_info' in prefs:
                for account in prefs['account_info']:
                    if 'email' in account:
                        accounts.append({
                            'email': account['email'],
                            'name': account.get('full_name', account['email'].split('@')[0]),
                            'profile': item
                        })

            # Alternative: look in signin section
            elif 'signin' in prefs and 'allowed_username' in prefs['signin']:
                email = prefs['signin']['allowed_username']
                accounts.append({
                    'email': email,
                    'name': email.split('@')[0],
                    'profile': item
```

```
            })

    except (json.JSONDecodeError, KeyError, FileNotFoundError):
        continue

# Try to get from Login Data (Chrome's saved passwords DB)
login_db = os.path.join(profile_path, "Login Data")
if os.path.exists(login_db):
    try:
        # Make a copy to avoid locking issues
        temp_db = login_db + "_temp"
        shutil.copy2(login_db, temp_db)

        conn = sqlite3.connect(temp_db)
        cursor = conn.cursor()

        # Look for Google accounts in saved logins
        cursor.execute("""
            SELECT origin_url, username_value
            FROM logins
            WHERE origin_url LIKE '%google.com%'
              OR origin_url LIKE '%youtube.com%'
              OR origin_url LIKE '%gmail.com%'
        """)

        for row in cursor.fetchall():
```

```
                if row[1] and '@' in row[1]:  # Valid email
                    email = row[1]
                    if not any(acc['email'] == email for acc in accounts):
                        accounts.append({
                            'email': email,
                            'name': email.split('@')[0],
                            'profile': item
                        })

            conn.close()
            os.remove(temp_db)

        except Exception:
            continue

    return accounts

except Exception as e:
    st.error(f"Error getting Chrome accounts: {e}")
    return []

def get_system_google_accounts():
    """Get Google accounts from system-wide credential stores"""
    try:
        accounts = []
        system = platform.system()
```

```
if system == "Windows":
    # Try Windows Credential Manager
    try:
        import keyring
        stored_creds = keyring.get_credential("google.com", None)
        if stored_creds:
            accounts.append({
                'email': stored_creds.username,
                'name': stored_creds.username.split('@')[0],
                'source': 'Windows Credential Manager'
            })
    except ImportError:
        pass

elif system == "Darwin":  # macOS
    # Try macOS Keychain
    try:
        result = subprocess.run([
            'security', 'find-internet-password',
            '-s', 'accounts.google.com',
            '-g'
        ], capture_output=True, text=True)

        if result.returncode == 0:
            for line in result.stderr.split('\n'):
```

```
                if 'acct' in line and '@' in line:
                    email = line.split('"')[1]
                    accounts.append({
                        'email': email,
                        'name': email.split('@')[0],
                        'source': 'macOS Keychain'
                    })
                    break
            except Exception:
                pass

        return accounts

    except Exception as e:
        st.error(f"Error getting system accounts: {e}")
        return []

def get_detected_google_accounts():
    """Get all detected Google accounts from various sources"""
    accounts = []

    # Get from Chrome
    chrome_accounts = get_chrome_logged_in_accounts()
    accounts.extend(chrome_accounts)

    # Get from system credential stores
```

```
    system_accounts = get_system_google_accounts()
    accounts.extend(system_accounts)

    # Remove duplicates based on email
    unique_accounts = {}
    for account in accounts:
        email = account['email']
        if email not in unique_accounts:
            unique_accounts[email] = account

    return list(unique_accounts.values())

def list_available_google_accounts():
    """List available Google accounts from stored credentials"""
    try:
        accounts = []
        credentials_dir = os.path.expanduser('~/.config/ytmusicapi/')

        if not os.path.exists(credentials_dir):
            os.makedirs(credentials_dir)
            return accounts

        # Look for stored credential files
        for filename in os.listdir(credentials_dir):
            if filename.startswith('credentials_') and filename.endswith('.json'):
                account_name = filename.replace('credentials_', '').replace('.json', '')
```

```
            accounts.append(account_name)

        return accounts

    except Exception as e:
        st.error(f"Error listing accounts: {e}")
        return []

def get_user_info_from_credentials(creds_path):
    """Extract user info from stored credentials"""
    try:
        with open(creds_path, 'r') as f:
            cred_data = json.load(f)

        if 'client_id' in cred_data:
            return cred_data.get('client_id', 'Unknown')
        return "Stored Account"

    except Exception as e:
        return "Unknown Account"

def setup_oauth_page():
    """OAuth authentication setup page"""
    st.header("🔐 OAuth Authentication Setup")

    credentials_dir = os.path.expanduser('~/.config/ytmusicapi/')
```

```
accounts = list_available_google_accounts()

if accounts:
    st.subheader("Existing Accounts")

    account_options = []
    for account in accounts:
        creds_path = os.path.join(credentials_dir, f'credentials_{account}.json')
        user_info = get_user_info_from_credentials(creds_path)
        account_options.append(f"{account} ({user_info})")

    account_options.append("➕ Add new Google account")

    selected_option = st.selectbox(
        "Select an account or add new:",
        account_options,
        key="oauth_account_selection"
    )

    if st.button("Use Selected Account", key="use_oauth_account"):
        if selected_option == "➕ Add new Google account":
            st.session_state.show_new_oauth = True
        else:
            selected_account = accounts[account_options.index(selected_option)]
            creds_path = os.path.join(credentials_dir, f'credentials_{selected_account}.json')
```

```
        with st.spinner("Authenticating..."):
            try:
                # Load and refresh credentials if needed
                creds = Credentials.from_authorized_user_file(creds_path, SCOPES)

                if not creds or not creds.valid:
                    if creds and creds.expired and creds.refresh_token:
                        creds.refresh(Request())
                        with open(creds_path, 'w') as token:
                            token.write(creds.to_json())
                    else:
                        st.error("Credentials invalid, need to re-authenticate...")
                        st.session_state.show_new_oauth = True
                        return

                # Create YTMusic instance
                ytmusic = YTMusic(auth=creds_path)
                ytmusic.get_home()  # Test connection

                st.success(f"✅ Successfully authenticated with account: {selected_account}")
                st.session_state.ytmusic = ytmusic
                st.session_state.current_account = selected_account

            except Exception as e:
                st.error(f"Authentication failed: {e}")
    else:
```

```
        st.info("No existing accounts found. Setting up new account...")
        st.session_state.show_new_oauth = True

    # Show new OAuth setup
    if st.session_state.get('show_new_oauth', False):
        setup_new_oauth_account()

def setup_new_oauth_account():
    """Setup new OAuth account"""
    st.subheader("🆕 Add New Google Account")

    # Detect available Google accounts
    detected_accounts = get_detected_google_accounts()

    if detected_accounts:
        st.info("🔍 Detected Google Accounts in your system:")

        account_display = []
        for i, account in enumerate(detected_accounts):
            source = account.get('source', f"Chrome Profile: {account.get('profile', 'Default')}")
            display_text = f"{account['email']} ({account['name']}) - {source}"
            account_display.append(display_text)
            st.write(f"• {display_text}")

        st.write("---")
```

```
    col1, col2 = st.columns(2)


    with col1:
        use_detected = st.selectbox(
            "Select detected account for naming:",
            ["Enter custom name"] + [f"{acc['name']} ({acc['email']})" for acc in
detected_accounts],
            key="detected_account_selection"
        )


    with col2:
        if use_detected == "Enter custom name":
            account_name = st.text_input(
                "Account name:",
                placeholder="e.g., 'personal', 'work'",
                key="custom_oauth_name"
            )
        else:
            # Extract suggested name from selection
            selected_idx = list(f"{acc['name']} ({acc['email']})" for acc in
detected_accounts).index(use_detected)
            selected_account = detected_accounts[selected_idx]
            suggested_name = selected_account['name']
            account_email = selected_account['email']


            account_name = st.text_input(
                f"Account name (from {account_email}):",
```

```
            value=suggested_name,
            key="suggested_oauth_name"
        )
else:
    st.warning("No Google accounts detected in Chrome or system.")
    account_name = st.text_input(
        "Enter account name:",
        placeholder="e.g., 'personal', 'work'",
        key="manual_oauth_name"
    )

if st.button("🚀 Start OAuth Authentication", key="start_oauth"):
    if not account_name:
        account_name = "default"

    credentials_dir = os.path.expanduser('~/.config/ytmusicapi/')
    if not os.path.exists(credentials_dir):
        os.makedirs(credentials_dir)

    creds_path = os.path.join(credentials_dir, f'credentials_{account_name}.json')

    with st.spinner("Setting up OAuth... A browser window will open for authentication."):
        try:
            # Note: In a real implementation, you'd need proper OAuth client credentials
            ytmusic = YTMusic()
            ytmusic.get_home()  # Test connection
```

```
            st.success(f"✅ New account '{account_name}' authenticated successfully!")
            st.info(f"Credentials saved to: {creds_path}")
            st.session_state.ytmusic = ytmusic
            st.session_state.current_account = account_name
            st.session_state.show_new_oauth = False

        except Exception as e:
            st.error(f"OAuth setup failed: {e}")

def setup_headers_page():
    """Headers authentication setup page"""
    st.header("🌐 Browser Headers Authentication")

    # Detect available Google accounts
    detected_accounts = get_detected_google_accounts()

    if detected_accounts:
        st.info("🔍 Detected Google accounts in your browser:")
        for account in detected_accounts:
            st.write(f"• {account['email']} ({account['name']})")
        st.warning("Make sure you're using one of these accounts in YouTube Music")

    st.subheader("Instructions:")
    st.markdown("""
    1. Open YouTube Music (music.youtube.com) in your browser
```

```
    2. Make sure you're logged into the correct Google account
    3. Open Developer Tools (F12)
    4. Go to Network tab
    5. Refresh the page or click on a song
    6. Find a request to 'music.youtube.com/youtubei/v1/'
    7. Right-click → Copy → Copy as cURL
    8. Paste the cURL command below
    """)

    headers_raw = st.text_area(
        "Paste cURL command here:",
        height=150,
        placeholder="curl 'https://music.youtube.com/youtubei/v1/...' -H 'authorization: ...' ...",
        key="headers_input"
    )

    col1, col2 = st.columns(2)

    with col1:
        if detected_accounts:
            account_selection = st.selectbox(
                "Which Google account are you using?",
                ["Enter custom name"] + [f"{acc['name']} ({acc['email']})" for acc in
detected_accounts],
                key="headers_account_selection"
            )
```

```
        else:
            account_selection = "Enter custom name"

    with col2:
        if account_selection == "Enter custom name":
            account_name = st.text_input(
                "Account name:",
                placeholder="e.g., 'personal', 'work'",
                key="headers_custom_name"
            )
        else:
            # Extract suggested name from selection
            selected_idx = list(f"{acc['name']} ({acc['email']})" for acc in
detected_accounts).index(account_selection)
            selected_account = detected_accounts[selected_idx]
            suggested_name = selected_account['name']
            account_email = selected_account['email']

            account_name = st.text_input(
                f"Account name (from {account_email}):",
                value=suggested_name,
                key="headers_suggested_name"
            )

    if st.button("🔗 Setup Headers Authentication", key="setup_headers"):
        if not headers_raw.strip():
```

```
    st.error("Please paste the cURL command")
    return

if not account_name:
    account_name = "default"

with st.spinner("Setting up headers authentication..."):
    try:
        headers_dir = os.path.expanduser('~/.config/ytmusicapi/')
        if not os.path.exists(headers_dir):
            os.makedirs(headers_dir)

        headers_path = os.path.join(headers_dir, f'headers_{account_name}.json')

        # Initialize with headers
        ytmusic = YTMusic(auth=headers_raw)
        ytmusic.get_home()  # Test connection

        st.success(f"✅ Headers authentication successful for account: {account_name}")
        st.info(f"Headers saved to: {headers_path}")
        st.session_state.ytmusic = ytmusic
        st.session_state.current_account = account_name

    except Exception as e:
        st.error(f"Headers authentication failed: {e}")
```

```
def setup_cookies_page():
    """Cookie authentication setup page"""
    st.header("🍪 Cookie-based Authentication")

    # Check if browser_cookie3 is installed
    try:
        import browser_cookie3
        cookies_available = True
    except ImportError:
        cookies_available = False
        st.error("browser_cookie3 not installed. Install with: `pip install browser_cookie3`")
        return

    # Detect available Google accounts
    detected_accounts = get_detected_google_accounts()

    if detected_accounts:
        st.info("🔍 Detected Google accounts in your browser:")
        for account in detected_accounts:
            st.write(f"• {account['email']} ({account['name']})")

    st.subheader("Instructions:")
    st.markdown("""
    1. Make sure you're logged into the correct YouTube Music account in your browser
    2. Close other Google account tabs to avoid conflicts
    3. Select your browser below
```

```
    """)

    col1, col2 = st.columns(2)

    with col1:
        browser_choice = st.selectbox(
            "Select your browser:",
            ["Chrome", "Firefox", "Edge", "Safari"],
            key="browser_selection"
        )

    with col2:
        if detected_accounts:
            account_selection = st.selectbox(
                "Which Google account are you using?",
                ["Enter custom name"] + [f"{acc['name']} ({acc['email']})" for acc in
detected_accounts],
                key="cookies_account_selection"
            )
        else:
            account_selection = "Enter custom name"

    if account_selection == "Enter custom name":
        account_name = st.text_input(
            "Account name:",
            placeholder="e.g., 'personal', 'work'",
```

```
            key="cookies_custom_name"
        )
    else:
        # Extract suggested name from selection
        selected_idx = list(f"{acc['name']} ({acc['email']})" for acc in
detected_accounts).index(account_selection)
        selected_account = detected_accounts[selected_idx]
        suggested_name = selected_account['name']
        account_email = selected_account['email']

        account_name = st.text_input(
            f"Account name (from {account_email}):",
            value=suggested_name,
            key="cookies_suggested_name"
        )

    if st.button("🍪 Setup Cookie Authentication", key="setup_cookies"):
        if not account_name:
            account_name = "cookie_account"

        with st.spinner("Setting up cookie authentication..."):
            try:
                import browser_cookie3

                if browser_choice == "Chrome":
                    cookies = browser_cookie3.chrome(domain_name='music.youtube.com')
```

```
            elif browser_choice == "Firefox":
                cookies = browser_cookie3.firefox(domain_name='music.youtube.com')
            elif browser_choice == "Edge":
                cookies = browser_cookie3.edge(domain_name='music.youtube.com')
            elif browser_choice == "Safari":
                cookies = browser_cookie3.safari(domain_name='music.youtube.com')

            ytmusic = YTMusic(auth=cookies)
            ytmusic.get_home()  # Test connection

            st.success(f"✅ Cookie authentication successful for account: {account_name}")
            st.session_state.ytmusic = ytmusic
            st.session_state.current_account = account_name

        except Exception as e:
            st.error(f"Cookie authentication failed: {e}")

def list_accounts_page():
    """List all configured accounts"""
    st.header("📋 Configured Accounts")

    credentials_dir = os.path.expanduser('~/.config/ytmusicapi/')

    if not os.path.exists(credentials_dir):
        st.info("No configured accounts found.")
        return
```

```
oauth_accounts = []
header_accounts = []

for filename in os.listdir(credentials_dir):
    if filename.startswith('credentials_') and filename.endswith('.json'):
        account_name = filename.replace('credentials_', '').replace('.json', '')
        oauth_accounts.append(account_name)
    elif filename.startswith('headers_') and filename.endswith('.json'):
        account_name = filename.replace('headers_', '').replace('.json', '')
        header_accounts.append(account_name)

col1, col2 = st.columns(2)

with col1:
    if oauth_accounts:
        st.subheader("🔐 OAuth Accounts")
        for account in oauth_accounts:
            st.write(f"• {account}")
    else:
        st.info("No OAuth accounts configured")

with col2:
    if header_accounts:
        st.subheader("🌐 Header-based Accounts")
        for account in header_accounts:
```

```
            st.write(f"• {account}")
        else:
            st.info("No header-based accounts configured")

    # Show detected accounts
    detected_accounts = get_detected_google_accounts()
    if detected_accounts:
        st.subheader("🔍 Detected Google Accounts (available for setup)")
        for account in detected_accounts:
            source = account.get('source', f"Chrome Profile: {account.get('profile', 'Default')}")
            st.write(f"• {account['email']} ({account['name']}) - {source}")

def test_search():
    """Test search functionality"""
    st.subheader("🔍 Test Search")

    if 'ytmusic' not in st.session_state:
        st.warning("Please authenticate first using one of the methods above.")
        return

    search_term = st.text_input(
        "Enter song/artist to search:",
        placeholder="e.g., 'Bohemian Rhapsody Queen'",
        key="search_input"
    )
```

```
    if st.button("🔍 Search", key="test_search"):
        if search_term:
            with st.spinner("Searching..."):
                try:
                    results = st.session_state.ytmusic.search(search_term, limit=5)

                    st.write(f"**Search results for '{search_term}':**")

                    for i, result in enumerate(results[:5], 1):
                        title = result.get('title', 'Unknown')
                        artist = 'Unknown Artist'

                        if result.get('artists'):
                            artist = result['artists'][0].get('name', 'Unknown Artist')

                        st.write(f"{i}. **{title}** - {artist}")

                except Exception as e:
                    st.error(f"Search failed: {e}")
        else:
            st.warning("Please enter a search term")

def setup_oauth_with_account_selection() -> YTMusic:
    """Setup OAuth authentication with account selection"""
    try:
        credentials_dir = os.path.expanduser('~/.config/ytmusicapi/')
```

```
accounts = list_available_google_accounts()

if not accounts:
    # No existing accounts, setup new OAuth
    return setup_new_oauth_account()

print("\nAvailable accounts:")
for i, account in enumerate(accounts, 1):
    print(f"{i}. {account}")
print(f"{len(accounts) + 1}. Add new account")

choice = input("\nSelect account number: ")
try:
    choice_num = int(choice)
    if 1 <= choice_num <= len(accounts):
        selected_account = accounts[choice_num - 1]
        creds_path = os.path.join(credentials_dir, f'credentials_{selected_account}.json')

        # Create and test YTMusic instance
        ytmusic = YTMusic(auth=creds_path)
        ytmusic.get_home()  # Test connection
        print(f"\n ✅ Successfully authenticated with account: {selected_account}")
        return ytmusic

    elif choice_num == len(accounts) + 1:
        return setup_new_oauth_account()
```

```
        except (ValueError, IndexError):
            print("Invalid selection")
            return None

    except Exception as e:
        print(f"Error in OAuth setup: {e}")
        return None

def setup_headers_auth() -> YTMusic:
    """Setup authentication using browser headers"""
    try:
        print("\n🌐 Browser Headers Authentication")
        print("\nInstructions:")
        print("1. Open YouTube Music in your browser")
        print("2. Open Developer Tools (F12)")
        print("3. Go to Network tab")
        print("4. Refresh the page")
        print("5. Find request to 'music.youtube.com'")
        print("6. Copy as cURL")

        headers_raw = input("\nPaste cURL command: ")
        if not headers_raw.strip():
            print("No headers provided")
            return None
```

```
        # Initialize with headers
        ytmusic = YTMusic(auth=headers_raw)
        ytmusic.get_home()  # Test connection
        print("\n✅ Headers authentication successful")
        return ytmusic

    except Exception as e:
        print(f"Headers authentication failed: {e}")
        return None

def setup_cookie_auth() -> YTMusic:
    """Setup authentication using browser cookies"""
    try:
        print("\n🍪 Cookie Authentication")
        print("\nSelect browser:")
        print("1. Chrome")
        print("2. Firefox")
        print("3. Edge")
        print("4. Safari")

        browser_choice = input("\nEnter browser number (1-4): ")

        try:
            import browser_cookie3

            if browser_choice == "1":
```

```
            cookies = browser_cookie3.chrome(domain_name='music.youtube.com')
        elif browser_choice == "2":
            cookies = browser_cookie3.firefox(domain_name='music.youtube.com')
        elif browser_choice == "3":
            cookies = browser_cookie3.edge(domain_name='music.youtube.com')
        elif browser_choice == "4":
            cookies = browser_cookie3.safari(domain_name='music.youtube.com')
        else:
            print("Invalid browser selection")
            return None

        ytmusic = YTMusic(auth=cookies)
        ytmusic.get_home()  # Test connection
        print("\n✅ Cookie authentication successful")
        return ytmusic

    except ImportError:
        print("browser_cookie3 not installed. Install with: pip install browser_cookie3")
        return None

    except Exception as e:
        print(f"Cookie authentication failed: {e}")
        return None

def main():
    """Main Streamlit app"""
```

```
# Initialize session state
if 'show_new_oauth' not in st.session_state:
    st.session_state.show_new_oauth = False

# Title and description
st.title("🎵 YouTube Music Authentication Setup")
st.markdown("Configure authentication for YouTube Music API access")

# Sidebar navigation
st.sidebar.title("🎛️ Navigation")

page = st.sidebar.selectbox(
    "Choose authentication method:",
    [
        "🔐 OAuth Authentication",
        "🌐 Browser Headers",
        "🍪 Cookie Authentication",
        "📋 List Accounts"
    ]
)

# Show current authentication status
if 'ytmusic' in st.session_state and 'current_account' in st.session_state:
    st.sidebar.success(f"✅ Authenticated as: {st.session_state.current_account}")
```

```
    else:
        st.sidebar.info("Not authenticated")

    # Main content area
    if page == "🔐 OAuth Authentication":
        setup_oauth_page()
    elif page == "🌐 Browser Headers":
        setup_headers_page()
    elif page == "🍪 Cookie Authentication":
        setup_cookies_page()
    elif page == "📋 List Accounts":
        list_accounts_page()

    # Test section (always visible if authenticated)
    if 'ytmusic' in st.session_state:
        st.markdown("---")
        test_search()

    # Dependencies info
    st.sidebar.markdown("---")
    st.sidebar.subheader("📦 Required Dependencies")
    st.sidebar.code("""
pip install streamlit
pip install ytmusicapi
pip install google-auth
pip install google-auth-oauthlib
```

```
pip install google-auth-httplib2
pip install browser_cookie3
pip install keyring
    """)

    # Instructions
    st.sidebar.markdown("---")
    st.sidebar.subheader("ℹ️ How to run")
    st.sidebar.code("streamlit run youtube_auth_app.py")

if __name__ == "__main__":
    main()
```