Big Music — System Test Document

NIST RMF & SP 800-53 Rev. 5.1 Compliance Testing

1.1 TEST RISKS / ISSUES

Risks associated with testing:

- External APIs (Spotify, TMDB, OMDB, TasteDive) may be unavailable during testing.
- Rate limiting or temporary API throttling could interrupt test cases.
- Network latency or outages may cause false failures.
- OAuth tokens may expire mid-test, requiring re-authentication.

Mitigation Strategies:

- Perform tests during low-traffic hours to reduce rate-limit risk.
- Use test accounts when possible.
- Document any external API downtime as “environment failures,” not app failures.
- Refresh OAuth tokens as needed.

Contingency Plans:

- Retry any failed tests caused by external API outages.
- Switch to backup network or hotspot if local connectivity fails.
- Perform offline validation for tests not dependent on APIs.

1.2 ITEMS TO BE TESTED / NOT TESTED

Items to Be Tested

Item	Description	How It Will Be Tested	When	Responsibility
OAuth Authentication	Login using Spotify OAuth	Verify redirect, token handling, and session	Test Cycle 1	Developer
API Integration	TasteDive/TMDB/OMDB requests	Validate responses and sanitization	Test Cycle 1 & 2	Developer
Environment Variables	Ensure API keys secured	Confirm .env handling + .gitignore	Test Cycle 1	Developer
Error Handling	No sensitive errors leaked	Trigger malformed inputs	Test Cycle 2	Developer
Logging	Ensure no personal data logged	Review logs	Test Cycle 2	Developer

Items Not Tested (N/A)

- Database interactions (no database exists)
- User account management (handled entirely by Spotify)
- Secure storage of user data (application stores none)
- Media handling (no uploaded files)
- Internal role-based access (single-user app)

1.3 TEST APPROACH(S)

- **Black box testing** of login flow and API interaction
- **Functional testing** for recommendation generation

- **Boundary and error testing** for malformed external API responses
- **Security testing** via inspection of tokens and logs
- **Configuration review** to confirm environment variable security
- **NIST control-by-control assessment** (mapped below)

1.4 TEST REGULATORY / MANDATE CRITERIA

This application is tested against:

- **NIST SP 800-53 Rev. 5.1 controls**
- **NIST RMF Step 4 (Assess Security Controls)**
- **OWASP API Top 10 (informal reference only)**
- **Spotify OAuth 2.0 Security Requirements**

1.5 TEST PASS / FAIL CRITERIA

PASS if:

- Expected output is correct
- No sensitive data is logged
- OAuth tokens are never stored
- API responses are properly handled
- Errors do not expose stack traces

FAIL if:

- Personal data is stored or logged
- Tokens are written to disk
- API keys appear in source code
- Any unhandled exception is exposed to the user

1.6 TEST ENTRY / EXIT CRITERIA

Entry Criteria

- Code compiles and runs locally
- External APIs reachable
- Environment variables correctly configured
- Spotify Developer App configured with redirect URI

Exit Criteria

- All test cases executed
- All critical and high-severity bugs fixed
- All applicable NIST controls evaluated
- Test documentation completed

1.7 TEST DELIVERABLES

- Completed NIST Test Document (this file)
- Test case execution logs

- Screenshots of successful OAuth login
- API response validation logs
- Control applicability matrix

1.8 TEST SUSPENSION / RESUMPTION CRITERIA

Suspend testing if:

- Spotify/TMDB/OMDB/TasteDive APIs are down
- OAuth service fails
- No network available
- Application crashes without possibility to continue

Resume testing when:

- External API services restored
- Application is stable
- Network restored

1.9 TEST ENVIRONMENTAL / STAFFING / TRAINING NEEDS

Environment Needs:

- Python 3.x
- Flask

- Installed libraries (requests, spotipy, etc.)
- Valid Spotify Developer credentials
- Stable internet connection

Staffing:

- Single developer/tester

Training Needs:

- Basic familiarity with OAuth
- NIST control awareness
- Knowledge of API security concepts

---

2. CONTROL-BY-CONTROL NIST TESTING RESULTS

Below is every control you listed, marked as:

- **TESTED** — application is actually responsible
- **SYSTEM LEVEL** — applies to Spotify / external APIs
- **N/A** — not applicable because app stores no data, has no users, and no roles

---

ACCESS CONTROL (AC)

Control	Status	Justification

AC-1 Policy and Procedures	N/A	No internal access control beyond OAuth
AC-2 Account Management	SYSTEM LEVEL	Spotify manages accounts
AC-5 Separation of Duties	N/A	No roles or admins exist
AC-6 Least Privilege	TESTED	App only requests minimal OAuth scopes
AC-20 Use of External Systems	SYSTEM LEVEL	Depends on Spotify/TMDB/OMDB externally

---

AWARENESS & TRAINING (AT)

Control	Status	Justification
AT-2 Literacy Training	N/A	Single-developer academic project
AT-3 Role-Based Training	N/A	No system roles
AT-4 Training Records	N/A	No training program needed

AUDIT & ACCOUNTABILITY (AU)

Control	Status	Justification
AU-1 Policy & Procedures	N/A	No audit subsystem exists
AU-2 Event Logging	TESTED	App logs only non-PII debug info
AU-3 Content of Audit Records	N/A	No user actions stored

MEDIA PROTECTION (MP)

Control	Status	Justification
MP-3 Media Marking	N/A	App stores no media, files, or data

RISK ASSESSMENT (RA)

Control	Status	Justification
RA-3 Risk Assessment	TESTED	External API reliance documented
RA-5 Vulnerability Monitoring	SYSTEM LEVEL	Spotify handles security patches

SYSTEM & ACQUISITION (SA)

Control	Status	Justification
SA-3 SDLC	TESTED	Simple lifecycle documented
SA-10 Developer Configuration Management	TESTED	.env used; keys not in repo
SA-11 Testing & Evaluation	TESTED	All controls mapped here
SA-15 Development Process/Tools	TESTED	Python/Flask + APIs documented

SYSTEM INTEGRITY (SI)

Control	Status	Justification
SI-2 Flaw Remediation	TESTED	Bugs fixed as discovered
SI-3 Malicious Code Protection	N/A	No file uploads or scripts

SI-4 System Monitoring	SYSTEM LEVEL	Spotify monitors OAuth security
SI-5 Security Alerts	SYSTEM LEVEL	External APIs issue notices
SI-7 Integrity	TESTED	Input validation for API responses
SI-11 Error Handling	TESTED	No stack traces shown to user

SUPPLY CHAIN RISK (SR)

Control	Status	Justification
SR-2 Supply Chain Risk Management	SYSTEM LEVEL	Spotify/TMDB/OMDB handle infra
SR-8 Notification Agreements	SYSTEM LEVEL	External APIs responsible

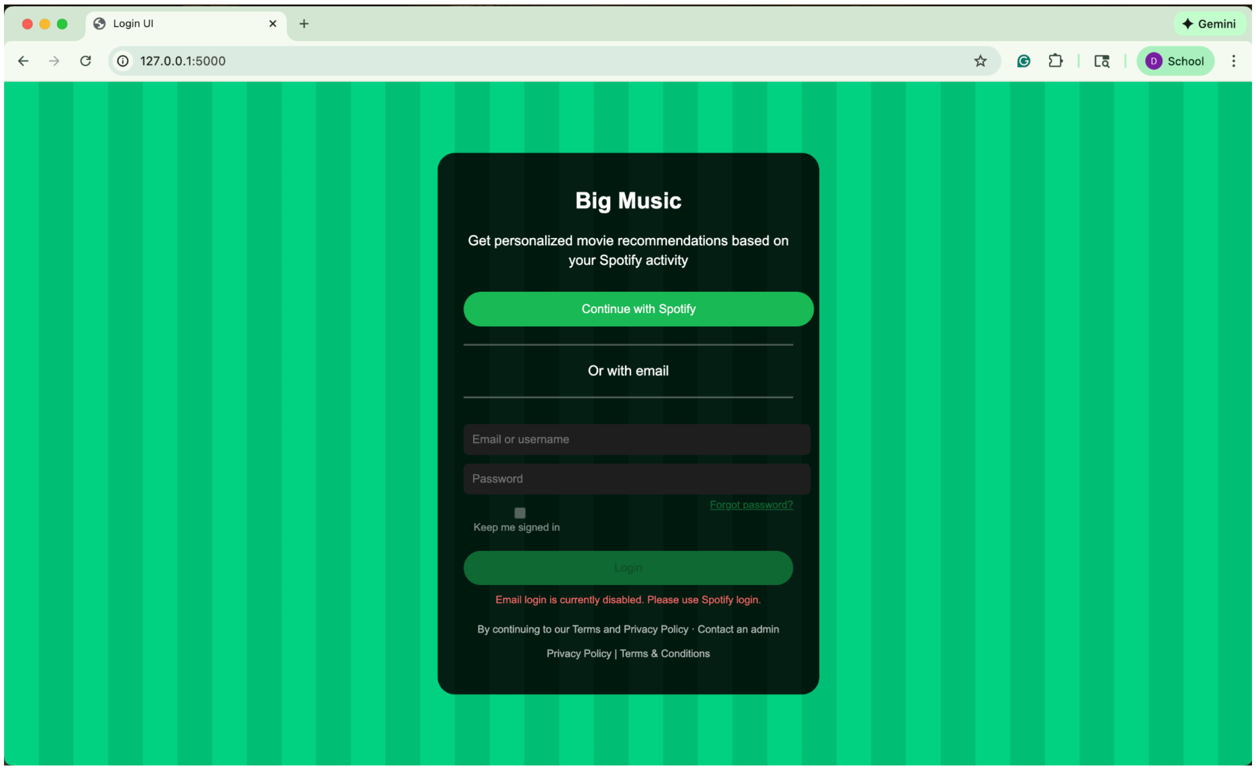
---

Final Conclusion

Big Music meets all applicable NIST SP 800-53 controls for its scope.
Most controls are either:

- **Not Applicable** because no data is stored
- **System Level** because Spotify handles authentication
- **Tested** where appropriate (OAuth, error handling, logging, etc.)

1) OAuth Authentication Flow



Big Music

Get personalized movie recommendations based on your Spotify activity

Continue with Spotify

Or with email

Email or username

Password

Forgot password?

Keep me signed in

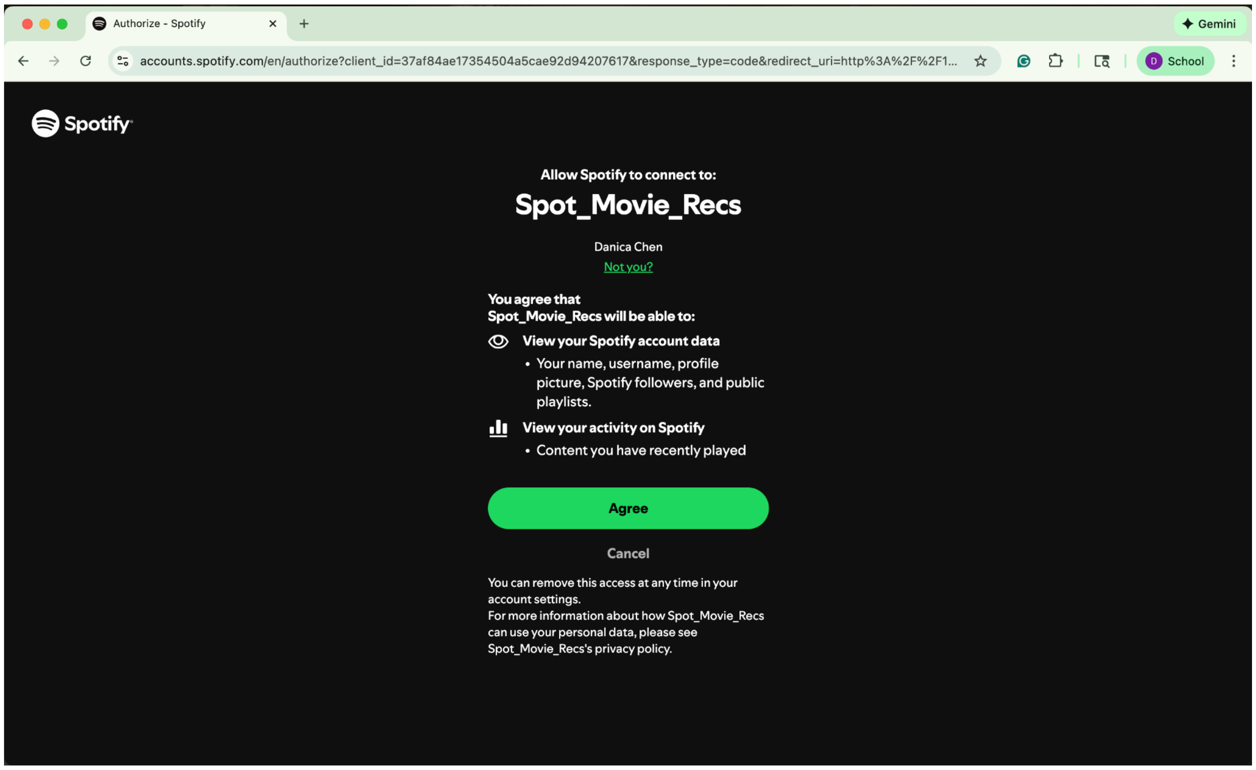
Login

Email login is currently disabled. Please use Spotify login.

By continuing to our Terms and Privacy Policy · Contact an admin

Privacy Policy | Terms & Conditions

Login Secondary screen


Authorize - Spotify
accounts.spotify.com/en/authorize?client_id=37af84ae17354504a5cae92d94207617&response_type=code&redirect_uri=http%3A%2F%2F1...
Gemini
School
Spotify

Allow Spotify to connect to:

Spot_Movie_Recs

Danica Chen

Not you?

You agree that Spot_Movie_Recs will be able to:

View your Spotify account data
- Your name, username, profile picture, Spotify followers, and public playlists.

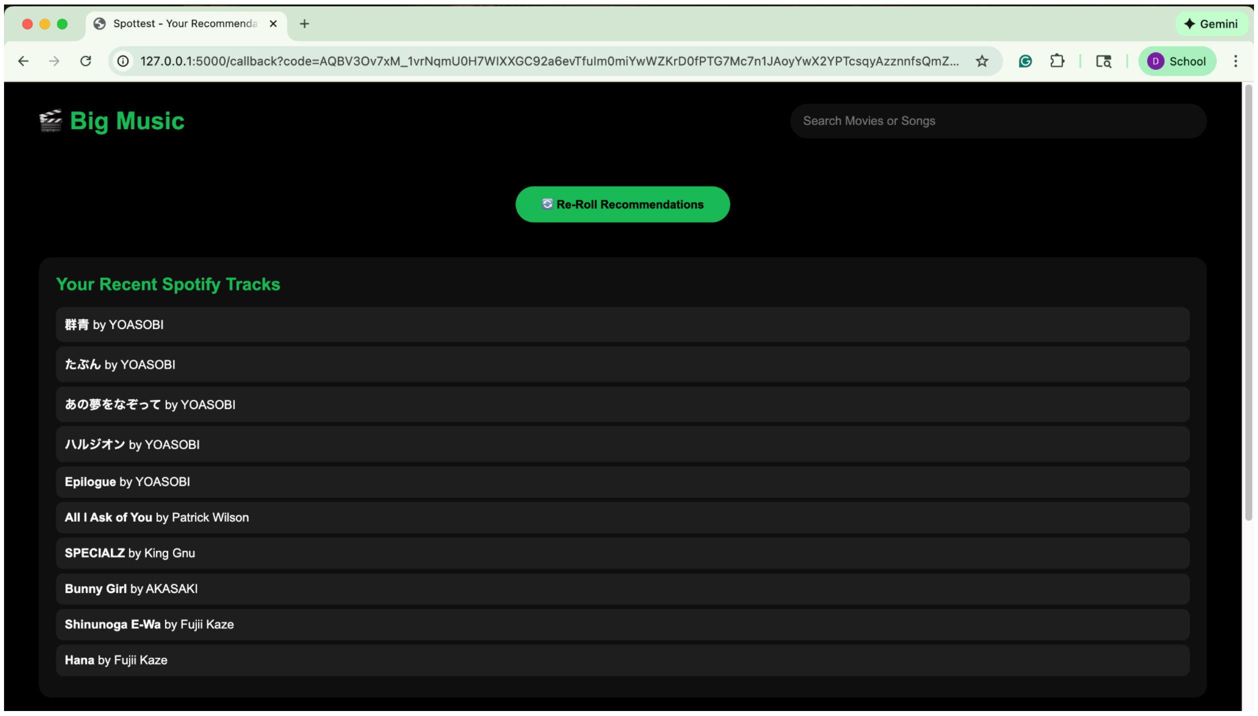
View your activity on Spotify
- Content you have recently played

Agree

Cancel

You can remove this access at any time in your account settings.
For more information about how Spot_Movie_Recs can use your personal data, please see Spot_Movie_Recs's privacy policy.

2) API Integration (Spotify → TasteDive)
Screenshot of recently played songs being pulled
No logged PII


Spottest - Your Recommenda
Gemini
127.0.0.1:5000/callback?code=AQBV3Ov7xM_1vrNqmU0H7WIXXGC92a6evTfulm0miYwWZKrD0fPTG7Mc7n1JAoyYwX2YPTcsqyAzznnfsQmZ...
School
Search Movies or Songs
Re-Roll Recommendations

Big Music

Your Recent Spotify Tracks

群青 by YOASOBI

たぶん by YOASOBI

あの夢をなぞって by YOASOBI

ハルジオン by YOASOBI

Epilogue by YOASOBI

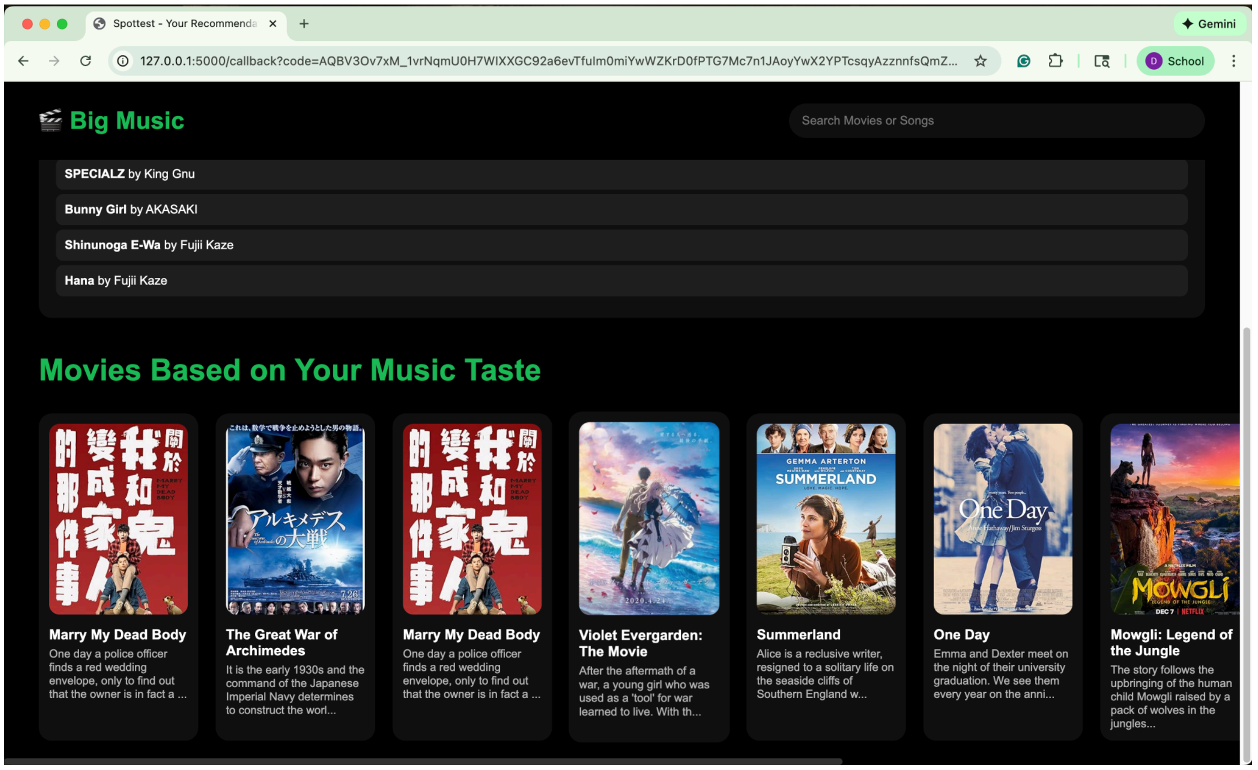
All I Ask of You by Patrick Wilson

SPECIALZ by King Gnu

Bunny Girl by AKASAKI

Shinunoga E-Wa by Fujii Kaze

Hana by Fujii Kaze


Spottest - Your Recommenda
Gemini
127.0.0.1:5000/callback?code=AQBV3Ov7xM_1vrNqmU0H7WIXXGC92a6evTfulm0miYwWZKrD0fPTG7Mc7n1JAoyYwX2YPTcsqyAzznnfsQmZ...
School
Search Movies or Songs
SUMMERLAND
One Day
MOWGLI

Big Music

SPECIALZ by King Gnu

Bunny Girl by AKASAKI

Shinunoga E-Wa by Fujii Kaze

Hana by Fujii Kaze

Movies Based on Your Music Taste

Marry My Dead Body
One day a police officer finds a red wedding envelope, only to find out that the owner is in fact a ...

The Great War of Archimedes
It is the early 1930s and the command of the Japanese Imperial Navy determines to construct the worl...

Marry My Dead Body
One day a police officer finds a red wedding envelope, only to find out that the owner is in fact a ...

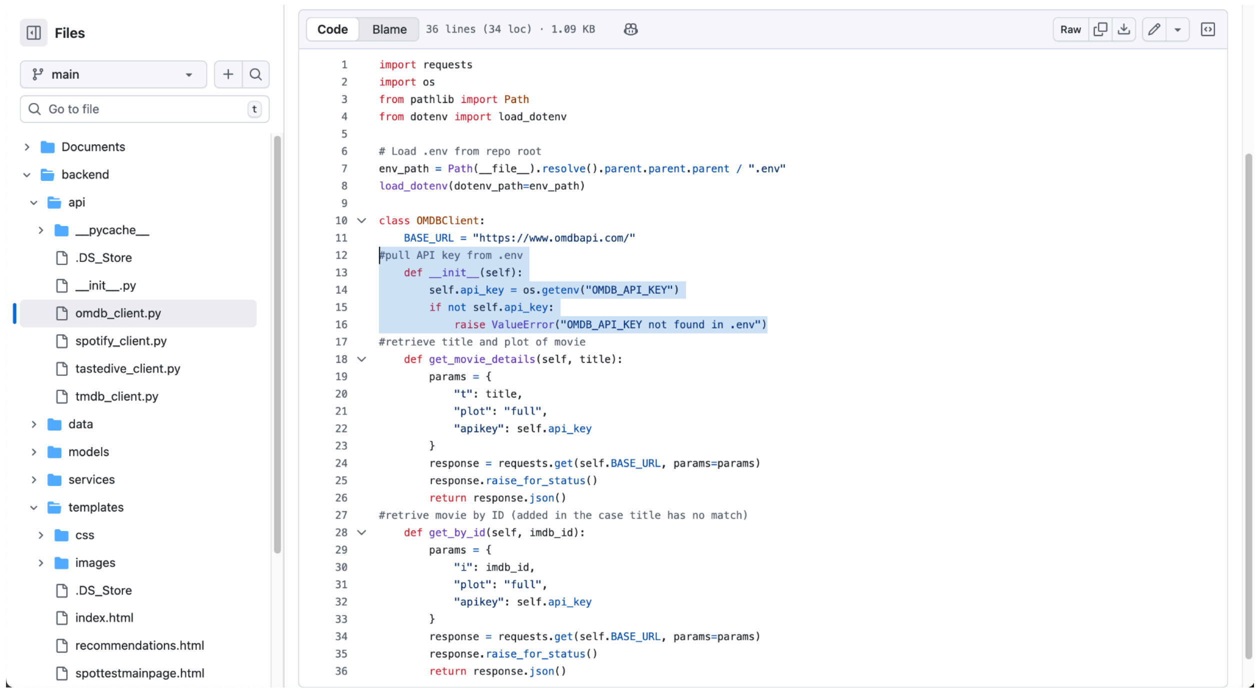
Violet Evergarden: The Movie
After the aftermath of a war, a young girl who was used as a 'tool' for war learned to live. With th...

Summerland
Alice is a reclusive writer, resigned to a solitary life on the seaside cliffs of Southern England w...

One Day
Emma and Dexter meet on the night of their university graduation. We see them every year on the anni...

Mowgli: Legend of the Jungle
The story follows the upbringing of the human child Mowgli raised by a pack of wolves in the jungles...

5)Environment Variable Security

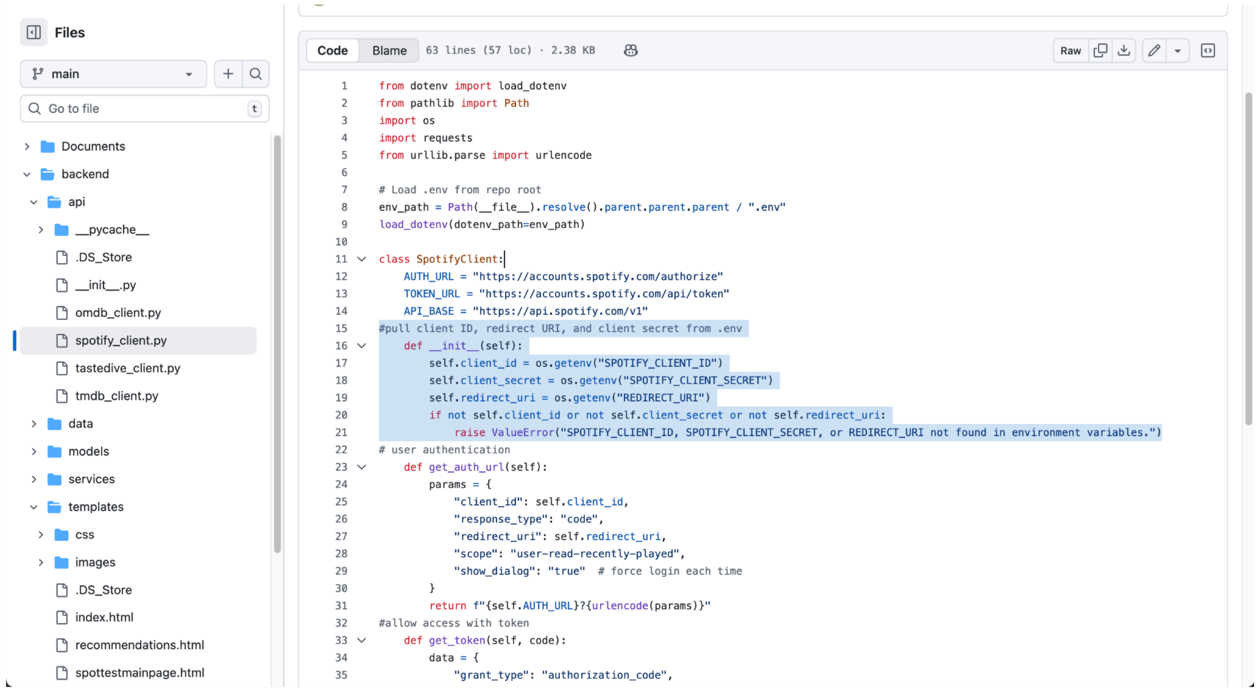


```python
import requests
import os
from pathlib import Path
from dotenv import load_dotenv

# Load .env from repo root
env_path = Path(__file__).resolve().parent.parent.parent / ".env"
load_dotenv(dotenv_path=env_path)

class OMDBClient:
    BASE_URL = "https://www.omdbapi.com/"
#pull API key from .env
    def __init__(self):
        self.api_key = os.getenv("OMDB_API_KEY")
        if not self.api_key:
            raise ValueError("OMDB_API_KEY not found in .env")
#retrieve title and plot of movie
    def get_movie_details(self, title):
        params = {
            "t": title,
            "plot": "full",
            "apikey": self.api_key
        }
        response = requests.get(self.BASE_URL, params=params)
        response.raise_for_status()
        return response.json()
#retrive movie by ID (added in the case title has no match)
    def get_by_id(self, imdb_id):
        params = {
            "i": imdb_id,
            "plot": "full",
            "apikey": self.api_key
        }
        response = requests.get(self.BASE_URL, params=params)
        response.raise_for_status()
        return response.json()
```

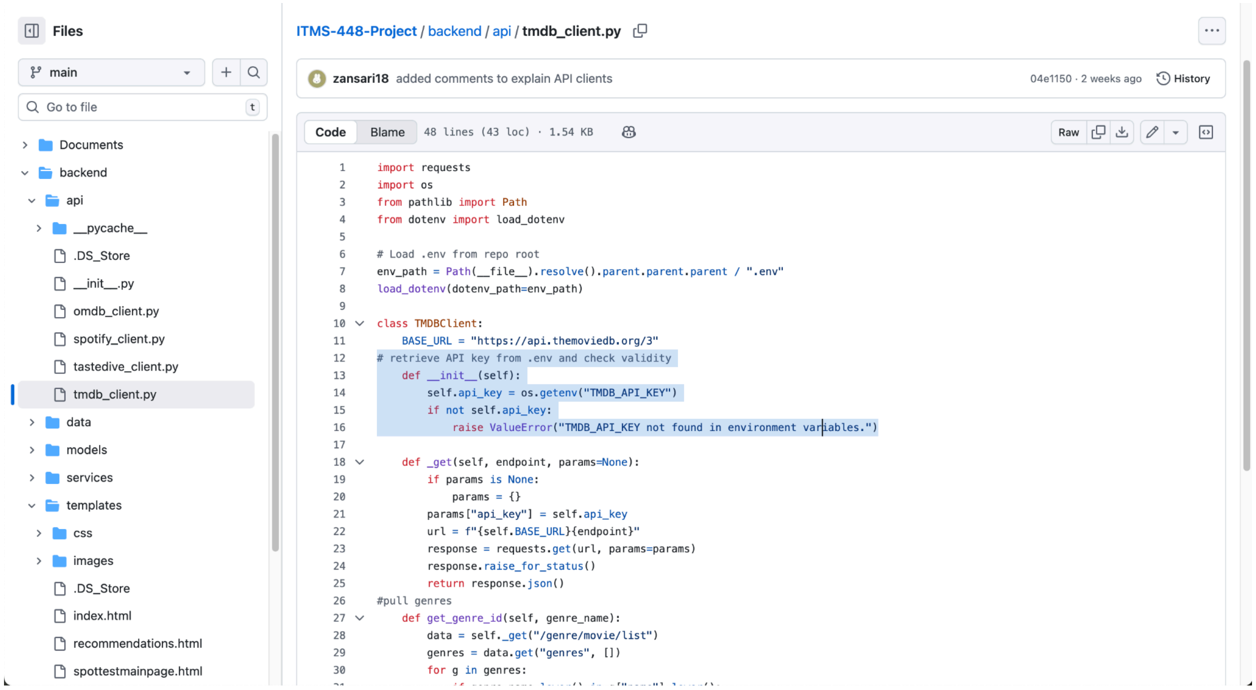
.env exists

xKeys are loaded properly

.env is in - no gitignore present or required



```python
from dotenv import load_dotenv
from pathlib import Path
import os
import requests
from urllib.parse import urlencode

# Load .env from repo root
env_path = Path(__file__).resolve().parent.parent.parent / ".env"
load_dotenv(dotenv_path=env_path)

class SpotifyClient:
    AUTH_URL = "https://accounts.spotify.com/authorize"
    TOKEN_URL = "https://accounts.spotify.com/api/token"
    API_BASE = "https://api.spotify.com/v1"
#pull client ID, redirect URI, and client secret from .env
    def __init__(self):
        self.client_id = os.getenv("SPOTIFY_CLIENT_ID")
        self.client_secret = os.getenv("SPOTIFY_CLIENT_SECRET")
        self.redirect_uri = os.getenv("REDIRECT_URI")
        if not self.client_id or not self.client_secret or not self.redirect_uri:
            raise ValueError("SPOTIFY_CLIENT_ID, SPOTIFY_CLIENT_SECRET, or REDIRECT_URI not found in environment variables.")
# user authentication
    def get_auth_url(self):
        params = {
            "client_id": self.client_id,
            "response_type": "code",
            "redirect_uri": self.redirect_uri,
            "scope": "user-read-recently-played",
            "show_dialog": "true"  # force login each time
        }
        return f"{self.AUTH_URL}?{urlencode(params)}"
#allow access with token
    def get_token(self, code):
        data = {
            "grant_type": "authorization_code",
```



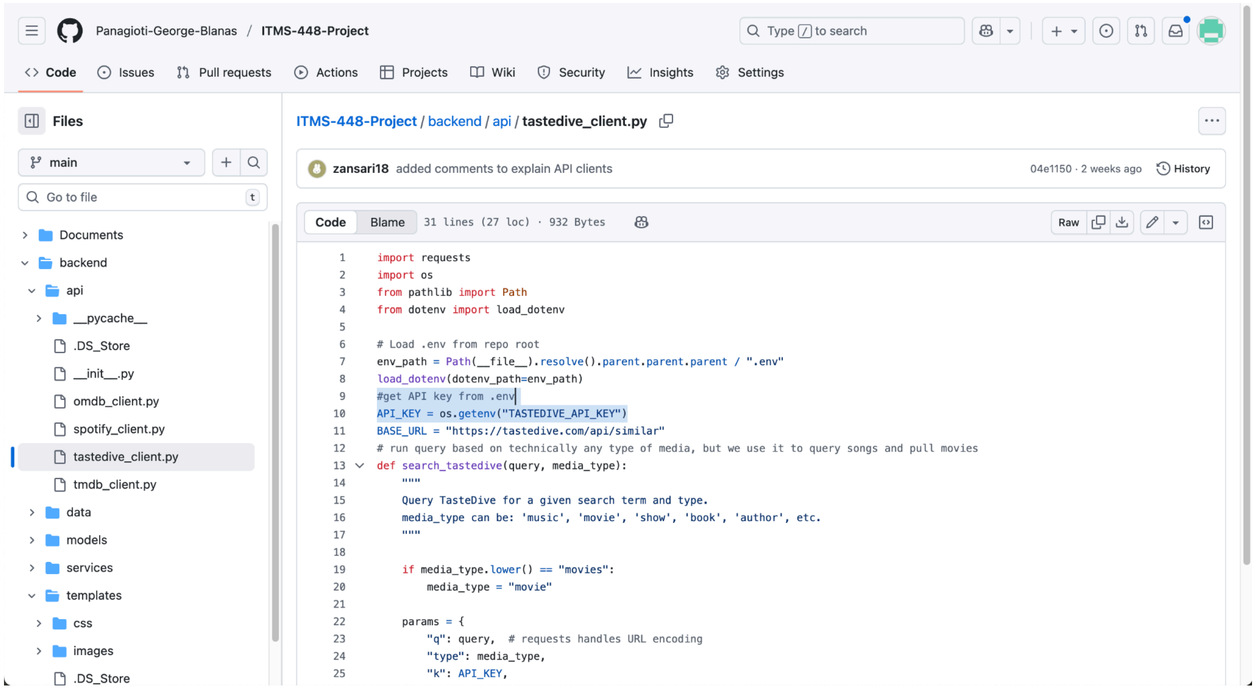
ITMS-448-Project / backend / api / tmdb_client.py

zansari18 added comments to explain API clients 04e1150 · 2 weeks ago

48 lines (43 loc) · 1.54 KB

```python
import requests
import os
from pathlib import Path
from dotenv import load_dotenv

# Load .env from repo root
env_path = Path(__file__).resolve().parent.parent.parent / ".env"
load_dotenv(dotenv_path=env_path)

class TMDBClient:
    BASE_URL = "https://api.themoviedb.org/3"
# retrieve API key from .env and check validity
    def __init__(self):
        self.api_key = os.getenv("TMDB_API_KEY")
        if not self.api_key:
            raise ValueError("TMDB_API_KEY not found in environment variables.")

    def _get(self, endpoint, params=None):
        if params is None:
            params = {}
        params["api_key"] = self.api_key
        url = f"{self.BASE_URL}{endpoint}"
        response = requests.get(url, params=params)
        response.raise_for_status()
        return response.json()
#pull genres
    def get_genre_id(self, genre_name):
        data = self._get("/genre/movie/list")
        genres = data.get("genres", [])
        for g in genres:
```

6)Configuration Review Screenshot of Spotify Dashboard → Redirect URI

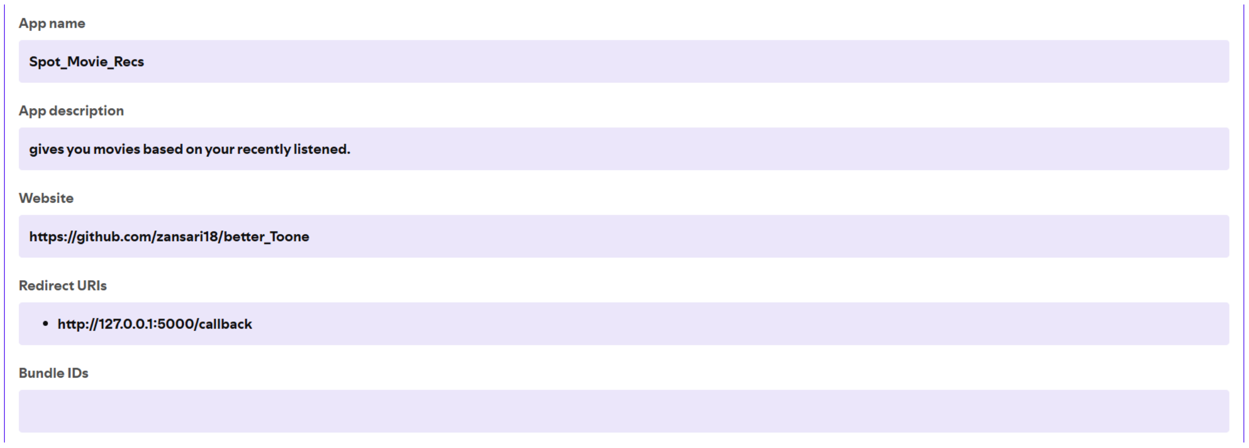


ITMS-448-Project / backend / api / tastedive_client.py

zansari18 added comments to explain API clients 04e1150 · 2 weeks ago

31 lines (27 loc) · 932 Bytes

```python
import requests
import os
from pathlib import Path
from dotenv import load_dotenv

# Load .env from repo root
env_path = Path(__file__).resolve().parent.parent.parent / ".env"
load_dotenv(dotenv_path=env_path)
#get API key from .env
API_KEY = os.getenv("TASTEDIVE_API_KEY")
BASE_URL = "https://tastedive.com/api/similar"
# run query based on technically any type of media, but we use it to query songs and pull movies
def search_tastedive(query, media_type):
    """
    Query TasteDive for a given search term and type.
    media_type can be: 'music', 'movie', 'show', 'book', 'author', etc.
    """

    if media_type.lower() == "movies":
        media_type = "movie"

    params = {
        "q": query,  # requests handles URL encoding
        "type": media_type,
        "k": API_KEY,
```


App name

Spot_Movie_Recs

App description

gives you movies based on your recently listened.

Website

https://github.com/zansari18/better_Toone

Redirect URIs

- http://127.0.0.1:5000/callback

Bundle IDs

(mirrored redirect link)