Soundtrack Your Brand API Authentication Fix

Problem Summary

The application was making excessive authentication requests to the Soundtrack Your Brand API, potentially exhausting the available token budget and hitting rate limits.

Rate Limits (from Soundtrack API Documentation)

- **Maximum tokens**: 3600
- **Token regeneration**: 50 tokens per second
- **Token deduction**: Based on query complexity
- Simple queries: ~10-20 tokens
- Complex queries: ~100-1000 tokens
- **Rate limit headers**:
- `x-ratelimiting-cost`: Tokens used for the request
- `x-ratelimiting-tokens-available`: Remaining tokens

Root Cause

The previous implementation was:
1. Creating new API instances repeatedly
2. Not caching the authenticated token
3. Making multiple authentication attempts
4. No rate limiting between requests
5. Potentially re-authenticating on every API call

This could quickly exhaust the 3600 token budget, especially with complex queries that cost 100-1000 tokens each.

Solution Implemented

1. Authenticate ONCE Strategy

- Token is validated **only once** when the user saves it in the configuration
- Token is stored in the database (`SoundtrackConfig` table)
- All subsequent API calls reuse the cached token from the database
- No repeated authentication attempts

2. Rate Limiting

Implemented a `RateLimiter` class that:
- Limits requests to 10 per second (conservative)
- Enforces minimum 100ms interval between requests
- Tracks request timestamps in a sliding window
- Automatically waits when rate limit is approached

3. Token Caching

- Token stored in database on initial save
- `getSoundtrackAPI()` function reuses the same API instance
- Token only re-validated if 401 error occurs (token expired)

4. Improved Error Handling

- Detects 429 (rate limit exceeded) errors
- Logs rate limit information from response headers
- Provides clear error messages for different failure scenarios

Code Changes

`/src/lib/soundtrack-your-brand.ts`

- Added `RateLimiter` class with intelligent request throttling
- Added `tokenValidated` flag to track authentication state
- Added `waitIfNeeded()` method called before each request
- Added rate limit status logging
- Improved error messages for rate limit errors (429)

`/src/app/api/soundtrack/config/route.ts`

- Added comments clarifying "AUTHENTICATE ONCE" strategy
- Token validation happens only during initial save
- Added logging for successful token validation
- Added logging for sound zone sync operations

`/src/app/api/soundtrack/players/route.ts`

- Added comments clarifying use of cached token
- No authentication on player fetch/control operations
- Token loaded from database, not re-validated

`/src/app/api/soundtrack/test/route.ts`

- Added logging for token testing
- Clarified this is for validation before saving

Authentication Flow

Initial Setup (ONCE)

```
User enters token → POST /api/soundtrack/config
  ↓
Test connection (validates token)
  ↓
Fetch account info
  ↓
Fetch sound zones
  ↓
Save token to database
  ↓
Token is now CACHED
```

Subsequent Operations (NO RE-AUTH)

```
User controls player → GET/PATCH /api/soundtrack/players
  ↓
Load token from database (cached)
  ↓
Rate limiter checks/waits if needed
  ↓
Make API request with cached token
  ↓
If 401 error → mark token as invalid, user must re-save
```

Benefits

1. **Reduced Token Usage**: Only authenticate once instead of repeatedly
2. **Rate Limit Compliance**: Automatic throttling prevents hitting limits
3. **Better Performance**: No authentication overhead on every request
4. **Improved Reliability**: Less chance of token exhaustion
5. **Clear Error Messages**: Users know when rate limits are hit

Testing Recommendations

1. Save a new Soundtrack API token - should authenticate once
2. Control players multiple times - should NOT re-authenticate
3. Monitor console logs for rate limit information
4. Verify no 429 errors occur during normal usage

Monitoring

The system now logs:
- `[Soundtrack] Token validated successfully - will be cached in database`
- `[Soundtrack] Synced X sound zones`
- `[Soundtrack] Rate limit - Cost: X, Available: Y`
- `[Soundtrack] Testing API token...`

Future Improvements

1. Add token expiration tracking in database
2. Implement automatic token refresh before expiration
3. Add dashboard showing current rate limit status
4. Implement request queuing for burst scenarios
5. Add metrics tracking for API usage patterns