NFHS Network Authentication & Scraping Implementation

Overview

This document describes the implementation of NFHS Network authentication and web scraping for the Sports Bar TV Controller application. The implementation allows the system to automatically fetch real high school sports game data from NFHS Network and store it in the database.

Implementation Date

October 2, 2025

Architecture

Components

1. **Authentication Module** (`authenticateNFHS`)
   - Handles login to NFHS Network
   - Manages CSRF tokens and cookies
   - Returns session cookies for authenticated requests

2. **School Search Module** (`searchSchools`)
   - Searches for schools in a specific location
   - Parses school listings from search results
   - Returns school metadata (ID, name, city, state)

3. **Game Fetching Modules**
   - `fetchGamesFromEventsPage` : Scrapes main events page for broad coverage
   - `fetchSchoolGames` : Fetches games for specific schools
   - `fetchNFHSGames` : Orchestrates both strategies and deduplicates results

4. **Data Storage Module** (`storeNFHSData`)
   - Upserts school records in NFHSSchool table
   - Upserts game records in NFHSGame table
   - Handles errors gracefully with detailed logging

Authentication Flow

Step 1: Fetch Login Page

```
GET https://www.nfhsnetwork.com/users/sign_in
```

- Retrieves initial cookies
- Extracts CSRF token from HTML form
- Looks for token in: `authenticity_token` , `_csrf` , or `csrf-token` meta tag

Step 2: Submit Login Credentials

```
POST https://www.nfhsnetwork.com/users/sign_in
Content-Type: application/x-www-form-urlencoded

user[email]={username}
user[password]={password}
user[remember_me]=0
authenticity_token={csrf_token}
```

Step 3: Validate Authentication

- Checks for 302/301 redirect (successful login)
- Validates presence of session cookies
- Looks for error messages in response HTML
- Returns cookies for subsequent authenticated requests

Web Scraping Strategy

Dual-Strategy Approach

Strategy 1: Events Page Scraping

URL: `https://www.nfhsnetwork.com/watch-events`

Advantages:
- Broad coverage of games across all locations
- Faster than individual school queries
- Captures featured and popular games

Data Extracted:
- Event ID from URL (`/events/{association}/{eventId}`)
- Sport, level, and gender from card text
- Team names from “Team A vs. Team B” format
- Date and time in UTC
- Location (city, state)
- Stream availability status

HTML Structure:

```
<a href="/events/association/gameId">
  Varsity Boys Football
  Team A vs. Team B
  Oct 4, 2025 | 2:00 AM UTC
  City, ST
</a>
```

Strategy 2: School-Specific Scraping

URL: `https://www.nfhsnetwork.com/schools/{school-id}`

Advantages:
- Targeted to user’s location
- Captures all games for local schools
- More accurate school metadata

Process:
1. Search for schools in target location
2. For each school (limited to 5):
- Fetch school page
- Parse upcoming events
- Extract game details
3. Combine with events page results

Data Parsing

Game Information Extraction

Sport and Level:

```
const sportMatch = fullText.match(/(Varsity|JV|Freshman)\s+(Boys|Girls)?\s*(\w+)/i)
level = sportMatch[1]  // "Varsity", "JV", "Freshman"
gender = sportMatch[2] // "Boys", "Girls"
sport = sportMatch[3]  // "Football", "Basketball", etc.
```

Team Names:

```
const teamsMatch = fullText.match(/([^vs]+)\s+vs\.?\s+([^0-9]+)/i)
homeTeam = teamsMatch[1].trim()
awayTeam = teamsMatch[2].trim()
```

Date and Time:

```
const dateMatch = fullText.match(/([A-Z][a-z]{2}\s+\d{1,2},\s+\d{4})/i)
const timeMatch = fullText.match(/(\d{1,2}:\d{2}\s+[AP]M\s+UTC)/i)
```

Location:

```
const locationMatch = fullText.match(/([^,]+),\s+([A-Z]{2})\s*$/i)
venue = locationMatch[1].trim()
state = locationMatch[2].trim()
```

Rate Limiting

To avoid being blocked by NFHS Network, the implementation includes:

1. **Request Delays**:
   - 1 second between authentication steps
   - 1.5 seconds between school page requests
   - 2 seconds between major operations

2. **Request Limits**:
   - Maximum 10 schools from search results
   - Maximum 5 schools for detailed scraping
   - Deduplication to avoid redundant requests

3. **User-Agent Spoofing**:
   - Uses realistic Chrome browser user agent

- Includes standard browser headers
- Mimics normal browser behavior

Cookie Management

Cookie Handling

- Uses `tough-cookie` library for cookie jar management
- Combines cookies from multiple requests
- Maintains session across all authenticated requests
- Includes cookies in all subsequent requests

Cookie Types

- Session cookies (authentication)
- CSRF protection cookies
- Remember-me cookies (optional)
- Tracking/analytics cookies

Database Schema

NFHSSchool Model

```
model NFHSSchool {
  id          String     @id @default(cuid())
  nfhsId      String     @unique
  name        String
  city        String
  state       String
  district    String?
  conferences String?
  sports      String?
  isActive    Boolean    @default(true)
  createdAt   DateTime   @default(now())
  updatedAt   DateTime   @updatedAt

  homeGames   NFHSGame[] @relation("HomeSchool")
  awayGames   NFHSGame[] @relation("AwaySchool")
}
```

NFHSGame Model

```
model NFHSGame {
  id                String     @id @default(cuid())
  nfhsEventId       String?    @unique
  homeSchoolId      String
  awaySchoolId      String
  homeTeamName      String
  awayTeamName      String
  sport             String
  level             String?
  gender            String?
  gameDate          DateTime
  gameTime          String
  venue             String
  status            String     @default("scheduled")
  isNFHSNetwork     Boolean    @default(false)
  streamUrl         String?
  lastSynced        DateTime   @default(now())
  createdAt         DateTime   @default(now())
  updatedAt         DateTime   @updatedAt

  homeSchool        NFHSSchool @relation("HomeSchool")
  awaySchool        NFHSSchool @relation("AwaySchool")
}
```

Environment Variables

Required in `.env` file:

```
# NFHS Network Credentials
NFHS_USERNAME="your-nfhs-email@example.com"
NFHS_PASSWORD="your-nfhs-password"
NFHS_LOCATION="Green Bay, Wisconsin"
```

API Endpoints

POST /api/nfhs/sync

Triggers a full sync of NFHS Network data.

Request: No body required

Response:

```
{
  "success": true,
  "message": "Successfully synced 25 games from NFHS Network",
  "data": {
    "gamesFound": 25,
    "schoolsCreated": 8,
    "schoolsUpdated": 2,
    "gamesCreated": 20,
    "gamesUpdated": 5,
    "errors": []
  }
}
```

GET /api/nfhs/sync

Returns information about the sync endpoint.

Response:

```
{
  "message": "NFHS Network Sync Endpoint",
  "description": "Use POST method to trigger NFHS Network data sync",
  "endpoint": "/api/nfhs/sync",
  "method": "POST",
  "credentialsConfigured": true,
  "implementation": "Web scraping with authentication",
  "features": [...],
  "limitations": [...]
}
```

Error Handling

Authentication Errors

- Missing credentials → 400 Bad Request
- Invalid credentials → 401 Unauthorized
- Network errors → Detailed error message

Scraping Errors

- Failed requests → Logged and skipped
- Parse errors → Logged, game skipped
- Database errors → Logged in results.errors array

Graceful Degradation

- If events page fails, tries school-specific scraping
- If school scraping fails, returns events page results
- Partial failures don't stop entire sync

Limitations

1. **HTML Structure Dependency**
   - Implementation depends on current NFHS Network HTML structure

  - May break if NFHS Network redesigns their website
  - Requires maintenance when site changes

2. **Rate Limiting**
   - Limited to 10 schools per sync
   - Delays between requests slow down sync
   - May not capture all games in large areas

3. **Authentication Requirements**
   - Requires valid NFHS Network subscription
   - Credentials must be kept secure
   - Session may expire, requiring re-authentication

4. **Data Accuracy**
   - Parsing relies on text patterns
   - May miss games with unusual formatting
   - Location data may be incomplete

5. **Performance**
   - Sync can take 30-60 seconds
   - Not suitable for real-time updates
   - Should be run on a schedule (e.g., daily)

Security Considerations

1. **Credential Storage**
   - Store credentials in environment variables
   - Never commit credentials to version control
   - Use secure .env file with restricted permissions

2. **Rate Limiting**
   - Respect NFHS Network's servers
   - Implement delays between requests
   - Limit number of concurrent requests

3. **Error Logging**
   - Don't log sensitive information
   - Sanitize error messages
   - Use structured logging

4. **Terms of Service**
   - Review NFHS Network Terms of Use
   - Ensure scraping is permitted
   - Use data responsibly

Testing

Manual Testing

1. Configure credentials in .env
2. Call POST /api/nfhs/sync
3. Check console logs for detailed progress

4. Verify data in database
5. Check for errors in response

Validation

- Verify authentication succeeds
- Check game count matches expectations
- Validate data quality in database
- Test error handling with invalid credentials

Maintenance

Regular Tasks

1. Monitor for NFHS Network site changes
2. Update selectors if HTML structure changes
3. Review error logs for patterns
4. Optimize rate limiting based on performance

Troubleshooting

- Check credentials are correct
- Verify NFHS Network site is accessible
- Review console logs for detailed errors
- Test authentication separately
- Check database connectivity

Future Improvements

1. **Enhanced Parsing**
   - Use more robust parsing techniques
   - Add fallback patterns for edge cases
   - Improve location detection

2. **Better Error Recovery**
   - Retry failed requests
   - Cache successful results
   - Resume from last successful point

3. **Performance Optimization**
   - Parallel requests where safe
   - Cache school data
   - Incremental updates

4. **Monitoring**
   - Add metrics collection
   - Alert on sync failures
   - Track data quality over time

5. **API Alternative**
   - Monitor for official NFHS Network API
   - Migrate to API when available
   - Maintain scraping as fallback

Dependencies

```json
{
  "cheerio": "^1.0.0",
  "tough-cookie": "^5.0.0"
}
```

Support

For issues or questions:
1. Check console logs for detailed error messages
2. Verify credentials and configuration
3. Review this documentation
4. Check NFHS Network site for changes
5. Contact development team

License

This implementation is part of the Sports Bar TV Controller application and follows the same license terms.

Last Updated: October 2, 2025
Version: 1.0.0
Author: AI Development Team