NFHS Network Puppeteer Solution

Problem

The NFHS Network website was returning 403 Forbidden errors when attempting to scrape game data using standard HTTP requests with enhanced headers. This is due to sophisticated anti-bot detection mechanisms.

Solution

Implemented a Puppeteer-based browser automation solution that:
1. Uses a real Chrome browser to bypass anti-bot detection
2. Authenticates with NFHS Network credentials
3. Maintains session cookies for efficient scraping
4. Extracts game data from the rendered HTML

Implementation Details

New Files Created

1. **`src/lib/sports-apis/nfhs-puppeteer-scraper.ts`**
   - Core Puppeteer scraper with stealth plugin
   - Handles login, session management, and data extraction
   - Implements browser resource optimization

2. **`src/lib/sports-apis/nfhs-api-puppeteer.ts`**
   - API wrapper around the Puppeteer scraper
   - Provides same interface as original NFHS API
   - Manages session expiry and re-authentication

3. **`scripts/test_nfhs_puppeteer.js`**
   - Test script to verify Puppeteer implementation
   - Tests login, scraping, and cookie management

Modified Files

1. **`package.json`**
   - Added dependencies:
     - `puppeteer`: ^23.5.0
     - `puppeteer-extra`: ^3.3.6
     - `puppeteer-extra-plugin-stealth`: ^2.11.2
2. **`src/app/api/nfhs-streams/route.ts`**
   - Updated to use Puppeteer API by default
   - Falls back to standard API if configured
   - Environment variable: `NFHS_USE_PUPPETEER` (default: true)

Features

Anti-Bot Evasion

- **Stealth Plugin**: Masks automation indicators
- **Realistic Headers**: Uses genuine browser headers
- **Human-like Behavior**: Adds delays between actions
- **Resource Blocking**: Blocks images/CSS for faster scraping

Session Management

- **Persistent Sessions**: Maintains authentication for 30 minutes
- **Cookie Storage**: Saves and restores session cookies
- **Auto Re-authentication**: Refreshes session when expired

Performance Optimization

- **Headless Mode**: Runs without GUI for server deployment
- **Resource Blocking**: Disables unnecessary resources
- **Connection Pooling**: Reuses browser instances
- **Memory Management**: Proper cleanup on close

Configuration

Environment Variables

```
# NFHS Network Credentials
NFHS_EMAIL=lhoople@graystonealehouse.com
NFHS_PASSWORD=Graystone#1

# Enable/Disable Puppeteer (default: true)
NFHS_USE_PUPPETEER=true
```

Server Requirements

- Node.js 18+
- Chrome/Chromium browser (installed automatically by Puppeteer)
- Sufficient memory for browser instances (recommended: 2GB+)

Usage

API Endpoints

The existing NFHS API endpoints work unchanged:

```
# Get games by state
GET /api/nfhs-streams?state=WI&sport=Football

# Get live streams
GET /api/nfhs-streams?liveOnly=true

# Search with location
POST /api/nfhs-streams
{
  "location": {
    "city": "Madison",
    "state": "WI"
  },
  "sport": "Basketball"
}
```

Testing

```
# Install dependencies
npm install

# Run test script
node scripts/test_nfhs_puppeteer.js
```

Deployment

Server Deployment (Ubuntu)

```
# Install Chrome dependencies
sudo apt-get update
sudo apt-get install -y \
  chromium-browser \
  fonts-liberation \
  libasound2 \
  libatk-bridge2.0-0 \
  libatk1.0-0 \
  libatspi2.0-0 \
  libcups2 \
  libdbus-1-3 \
  libdrm2 \
  libgbm1 \
  libgtk-3-0 \
  libnspr4 \
  libnss3 \
  libwayland-client0 \
  libxcomposite1 \
  libxdamage1 \
  libxfixes3 \
  libxkbcommon0 \
  libxrandr2 \
  xdg-utils

# Install Node dependencies
npm install

# Build application
npm run build

# Start server
npm start
```

PM2 Configuration

```
# Start with PM2
pm2 start npm --name "sports-bar-tv" -- start

# Set environment variables
pm2 set sports-bar-tv:NFHS_EMAIL "lhoople@graystonealehouse.com"
pm2 set sports-bar-tv:NFHS_PASSWORD "Graystone#1"
pm2 set sports-bar-tv:NFHS_USE_PUPPETEER "true"

# Save configuration
pm2 save
```

Troubleshooting

Common Issues

1. Chrome/Chromium Not Found

```
# Install Chromium manually
sudo apt-get install chromium-browser

# Or let Puppeteer download Chrome
npx puppeteer browsers install chrome
```

2. Memory Issues

```
# Increase Node.js memory limit
NODE_OPTIONS="--max-old-space-size=4096" npm start
```

3. Login Failures

- Verify credentials in environment variables
- Check NFHS Network website is accessible
- Review browser console logs for errors

4. Slow Performance

- Ensure resource blocking is enabled
- Check server has sufficient memory
- Consider increasing timeout values

Debug Mode

Enable headless: false to see browser actions:

```
const scraper = new NFHSPuppeteerScraper(credentials, {
  headless: false, // Show browser window
  timeout: 60000   // Increase timeout
})
```

Performance Metrics

Expected Performance

- **Login Time**: 3-5 seconds
- **Game Scraping**: 5-10 seconds per page
- **Memory Usage**: 200-400 MB per browser instance
- **Session Duration**: 30 minutes

Optimization Tips

1. Reuse browser instances across requests
2. Implement caching for frequently accessed data
3. Use connection pooling for concurrent requests
4. Schedule scraping during off-peak hours

Security Considerations

Credentials Storage

- Store credentials in environment variables
- Never commit credentials to version control
- Use secrets management in production

Rate Limiting

- Implement delays between requests
- Respect NFHS Network's terms of service
- Monitor for rate limit responses

Error Handling

- Graceful degradation on failures
- Automatic retry with exponential backoff
- Comprehensive error logging

Future Enhancements

Potential Improvements

1. **Browser Pool**: Maintain multiple browser instances
2. **Proxy Support**: Rotate IPs for distributed scraping
3. **CAPTCHA Solving**: Integrate CAPTCHA solving services
4. **Data Caching**: Cache scraped data with TTL
5. **Webhook Support**: Real-time notifications for live games

Monitoring

- Track scraping success rates
- Monitor browser memory usage
- Alert on authentication failures
- Log performance metrics

Support

For issues or questions:
1. Check the troubleshooting section
2. Review browser console logs
3. Test with the provided test script
4. Contact the development team

License