Test Matrix & Reproduction Guide

Manual Test Cases (15 scenarios)

1. Local Auth Success

Steps:

1. Start local server on `http://localhost:8000`
2. Update `config.js` with your CLIENT_ID
3. Navigate to `/index.html`
4. Click "Start Playing"
5. Complete OAuth flow

Expected:

- Redirects to Spotify auth
- Returns to `/auth.html` with code
- Shows "Success! Redirecting to game..."
- Lands on `/rhythm.html` with bootstrap parameter

Network calls: OAuth authorize, token exchange, `/me` profile

---

2. Production Auth Success

Steps:

1. Deploy to production domain
2. Update `config.js` PROD_HOST and Spotify redirect URIs
3. Navigate to production `/index.html`
4. Complete auth flow

Expected:

- Same as local but using production URLs
- All redirects use HTTPS production domain

Network calls: Same as local test

---

3. Refresh on Auth Page (Before Code Exchange)

Steps:

1. Start auth flow normally
2. Complete Spotify OAuth (get redirected back)
3. Before page finishes loading, refresh browser
4. Page should reload with same URL parameters

Expected:

- Shows "Session expired. Please start over."
- code_verifier missing from sessionStorage
- URL cleared of parameters

Network calls: No token exchange attempted

---

4. Back Button from Rhythm Page

Steps:

1. Complete successful auth flow
2. Land on rhythm.html
3. Use browser back button
4. Try to navigate forward again

Expected:

- Back to auth.html triggers bootstrap validation failure
- Redirects to fresh auth (preserves country if set)
- No redirect loops

Network calls: Possible `/me` call, then redirect

---

5. Expired Token Auto-Refresh

Steps:

1. Complete auth flow
2. Manually set `expires_at` in sessionStorage to past time
3. Try to start session or make API call
4. Should auto-refresh token

Expected:

- Shows brief loading
- Token refreshed automatically
- API call succeeds
- New `expires_at` stored

Network calls: Token refresh, then original API call

---

6. Active Device Found

Steps:

1. Open Spotify app/Web Player
2. Play any song briefly
3. Complete auth flow to rhythm page
4. Observe device detection

Expected:

- "Searching for active devices..." appears
- Within 3-6 seconds: "✅ Found active device: [Device Name]"
- "Start Rhythm Session" button enabled
- Device activation section hidden after 2 seconds

Network calls: Repeated `/me/player/devices` every 3 seconds

---

7. No Active Device Found

Steps:

1. Ensure no Spotify apps are playing
2. Complete auth flow to rhythm page
3. Wait for device detection timeout (2 minutes)

Expected:

- "Searching for active devices..." for 2 minutes
- Then: "⚠ No active device found. Please open Spotify..."
- "Start Rhythm Session" remains disabled

- "Open Spotify App" button remains visible

Network calls: Repeated device checks, then stops

---

8. Premium vs Free Account

Steps:

1. Use Spotify Free account
2. Complete auth flow

Expected:

- Profile loads normally
- Warning message: "Spotify Premium is required for playback control"
- Can still attempt to start session
- Playback controls will fail with Premium error

Network calls: `/me` shows `"product": "free"`

---

9. Mobile Deep-Link Success

Steps:

1. Use mobile browser
2. Complete auth flow
3. Click "Open Spotify App"
4. Should attempt `spotify://` deep link

Expected:

- Tries `spotify://` protocol first
- Falls back to `https://open.spotify.com/` after 1.5s
- Device detection restarts after 3s
- May successfully find mobile device

Network calls: Device detection resumes

---

10. Desktop Web Player Fallback

Steps:

1. Use desktop browser without Spotify app
2. Complete auth flow
3. Click "Open Spotify App"
4. Should open web player

Expected:

- Hidden iframe tries `spotify:` protocol
- Opens `https://open.spotify.com/` in new tab after 1s
- User can play song in web player
- Device detection finds web player device

Network calls: Device detection continues

---

11. Playlist Creation Success

Steps:

1. Have listening history on Spotify account
2. Complete device activation
3. Click "Start Rhythm Session"
4. Observe playlist creation

Expected:

- Shows "Starting rhythm session..."
- Playlist created from top tracks
- "Current Playlist" section shows track count and current track
- Playback begins automatically
- Progress bar appears

Network calls: `/me/top/tracks`, `/me/player/play`, device transfer

---

12. Playlist Creation Failure/Fallback

Steps:

1. Use fresh Spotify account with no listening history
2. Start session

Expected:

- Falls back to featured playlists
- Still creates 3-track playlist
- Session starts normally with fallback content

Network calls: `/me/top/tracks` (empty), `/browse/featured-playlists`, playlist tracks

---

13. Early Finish Path

Steps:

1. Start successful session
2. Let 1 track partially play
3. Click "Finish Session"

Expected:

- Session ends immediately
- "Session finished early. Complete more tracks next time!"
- Buttons reset (Start enabled, Save/Finish disabled)
- Playback stops

Network calls: Player pause, telemetry (if configured)

---

14. Completion Code Generated

Steps:

1. Start session successfully
2. Let tracks play through designated gaps
3. Reach 70% completion threshold

Expected:

- Completion code section appears
- Code format: `RG-[TIMESTAMP_BASE36]`
- "🎉 Session Complete!" message
- Session automatically finishes
- Code remains visible

Network calls: Telemetry event, session finish

---

15. Save Track Success

Steps:

1. Start active session
2. During playback, click "Save Current Track"
3. Check Spotify library

Expected:

- Success message: "[Track Name] saved to your library!"
- Track appears in Spotify Liked Songs
- Message auto-hides after 5 seconds

Network calls: `PUT /me/tracks` with track ID

Telemetry Test Cases

Telemetry Disabled (Default)

Steps:

1. Ensure `CONFIG.TELEMETRY_URL` is `null`
2. Complete any session actions

Expected:

- No telemetry network calls
- `sendTelemetry()` returns immediately
- No errors in console

Telemetry Enabled

Steps:

1. Set `CONFIG.TELEMETRY_URL` to test endpoint
2. Complete session with events

Expected:

- POST requests to telemetry URL
- Payloads include: event, timestamp, session_id, user_agent
- Failures are silent (no user-visible errors)

Network calls: POST to telemetry endpoint for each event