Snake Evolution - Maintenance & Support Guide v1.0

Document Header

Project: Snake Evolution Game
Document Type: Maintenance & Support Guide
Version: 1.0
Date: November 2025
Status: Production Ready
Prepared by: Senior Software Architect
Audience: DevOps Teams, SRE Engineers, Support Staff
Standard: Based on IEEE 1063-2001 System Maintenance Guide

Executive Summary

This guide provides **operational procedures** for maintaining Snake Evolution in production, managing known issues, handling customer support requests, and planning future evolution of the system. It is the reference manual for DevOps teams, site reliability engineers, and technical support personnel.

Document Purpose

- **Who should read:** DevOps engineers, SRE teams, support staff, maintenance personnel
- **What's covered:** Known issues, dependencies, troubleshooting, maintenance procedures, update strategies, customer support
- **What's NOT covered:** Development procedures (see Implementation Guide)

1. Known Issues & Workarounds

1.1 Current Known Issues (v2.1)

Issue #1: Audio Context Suspension on Mobile (HIGH)

Severity: HIGH
Affects: Mobile browsers (iOS Safari, Chrome Mobile)
Symptoms: No sound on mobile devices; audio context returns `suspended` state
Root Cause: Browser security policy requires user interaction before playing audio
Status: Known issue, documented behavior

Workaround for Users:

```
1. Tap anywhere on the game screen
2. Audio context will unlock
3. Restart game to hear sound
```

Workaround for Developers:

```
// Already implemented in AudioManager
document.addEventListener('touchstart', () =&gt; {
  audioContext.resume(); // Unlock audio context
});
```

Permanent Fix: Planned for v2.1 patch

Issue #2: localStorage Quota Exceeded Error (MEDIUM)

Severity: MEDIUM
Affects: Users with limited storage quota
Symptoms: High score saves fail; message: "Save failed"
Root Cause: Browser localStorage typically has 5-10MB limit; rarely exceeded
Status: Graceful fallback implemented

Workaround for Users:

```
1. Clear browser cache:
   Settings → Privacy → Clear browsing data
2. Select "All time"
3. Check "Cached images and files"
4. Click "Clear data"
5. Retry save
```

Workaround for Developers:

```
// Already implemented in StorageManager
if (error.code === 'QuotaExceededError') {
  Logger.warn('Storage quota exceeded - using memory fallback');
  this.useMemoryLeaderboard(); // In-memory backup
}
```

Issue #3: Low FPS on Low-End Devices (MEDIUM)

Severity: MEDIUM
Affects: Devices with <2GB RAM, older browsers
Symptoms: Game runs at 20-30 FPS instead of 60 FPS
Root Cause: Canvas rendering bottleneck; CPU/GPU limitations
Status: Known limitation; performance profiling added

Workaround for Users:

```
1. Close other browser tabs
2. Disable browser extensions (esp. ad blockers)
3. Update browser to latest version
4. Lower screen resolution (if on desktop)
5. Use hardware acceleration:
   Settings → Performance → Enable hardware acceleration
```

Monitoring:

```
// FPS monitor alerts if &lt; 30 FPS
if (performanceMonitor.getAverageFPS() &lt; 30) {
  Logger.warn('LOW_FPS', { currentFPS: fpsValue });
  notifyDevOps('Low FPS detected', { device, browser });
}
```

Issue #4: Canvas Not Fully Rendered on Initial Load (LOW)

Severity: LOW
Affects: Rare edge case; some browsers
Symptoms: Black screen for 1-2 seconds on first load
Root Cause: Canvas initialization timing race condition
Status: Fixed in v2.1; still monitor

Workaround:

```
Hard refresh (Ctrl+Shift+R) usually resolves
```

1.2 Issue Resolution Procedure

For Support Staff:

```
1. IDENTIFY:
   - Get exact error message from user
   - Note browser, device, OS
   - Ask about reproducibility (always? sometimes?)

2. SEARCH:
   - Check this Known Issues section
   - Search GitHub issues: github.com/yourusername/snake-evolution/issues
   - Check community discussions

3. REPRODUCE:
   - Test on your device with same browser/OS
   - Try different devices if needed
   - Document exact steps
```

```
4. RESPOND:
   - If known issue: Provide workaround from this guide
   - If new issue: Create GitHub issue with full details
   - Set priority (P1=blocking, P2=important, P3=nice-to-have)
   - Assign to dev team for investigation

5. TRACK:
   - Follow up after user applies workaround
   - Collect data for pattern analysis
   - Escalate if critical
```

2. System Dependencies

2.1 Browser Requirements

Supported Browsers

Browser	Desktop	Mobile	Min Version	Status
Chrome	✅	✅	90+	Fully supported
Firefox	✅	✅	88+	Fully supported
Safari	✅	✅	14+	Fully supported
Edge	✅	✅	90+	Fully supported
Opera	✅	✅	76+	Fully supported
IE 11	❌	❌	N/A	Not supported

Required Browser Features

```
HTML5 Canvas API
Web Audio API
localStorage (5MB+)
ES2020 JavaScript support
Promise/async-await
Fetch API (for v2+)
```

Compatibility Check:

```
// Run in browser console to verify
const canRun =
  !!window.HTMLCanvasElement &amp;&amp;
  !!window.AudioContext &amp;&amp;
  !!window.localStorage &amp;&amp;
  !!Promise &amp;&amp;
  !!fetch;

console.log('Snake Evolution compatible:', canRun);
```

2.2 Device Requirements

Hardware

Component	Minimum	Recommended
RAM	512 MB	2 GB+
CPU	Dual-core 1GHz	Quad-core 2GHz+
Storage	50 MB	100 MB
Screen	320x240	1024x768+

Operating Systems

- ✅ Windows 10+
- ✅ macOS 10.12+
- ✅ Ubuntu 18.04+
- ✅ iOS 12+
- ✅ Android 8+

2.3 Server/CDN Dependencies

Netlify (Primary Host)

- **Status:** Essential
- **Availability SLA:** 99.95%
- **Fallback:** CloudFlare global CDN
- **Escalation:** If Netlify down, switch to backup host

Check Netlify Status:

```
https://www.netlifystat.us/
```

DNS Provider

- **Service:** Netlify DNS (included)
- **TTL:** 3600 seconds
- **Update time:** Up to 1 hour for propagation

2.4 Optional Dependencies (Roadmap v2)

Service	Purpose	Status
Sentry.io	Error tracking	Optional v2
Google Analytics	User analytics	Optional v2
SpeedCurve	Performance trending	Optional v2

3. Troubleshooting Procedures

3.1 Customer Support Flowchart

```
User reports issue
    ↓
Is it in Known Issues (Section 1)?
    ├─ YES → Provide documented workaround
    │         → Follow up in 24hrs
    │         → Track resolution
    └─ NO → Continue diagnosis
            ↓
Can you reproduce locally?
    ├─ YES → Create GitHub issue (P1/P2/P3)
    │         → Assign to dev team
    │         → Communicate ETA to user
    └─ NO → Ask for more details:
            - Screenshot/video
            - Browser version
            - Device specs
            - Network speed
            - Exact steps to reproduce
            ↓
Collect info, try again
    ├─ Can reproduce now?
    │  ├─ YES → Follow YES path above
    │  └─ NO → Continue to next section
    └─ Issue may be intermittent
      → Increase logging
      → Monitor error tracking
      → Escalate to DevOps
```

3.2 Performance Troubleshooting

Symptom: Game running slow (< 30 FPS)

```
DIAGNOSIS STEPS:

1. Check device specs:
   - RAM: 2GB minimum
   - CPU: 1GHz+ minimum
```

```
   - Browser: Latest version?

2. Check browser state:
   - How many tabs open? (close extra)
   - How many extensions active? (disable ad blockers)
   - Any large downloads? (pause)
   - Virtual machine? (native usually faster)

3. Check network:
   - Ping latency &lt; 100ms? (irrelevant for offline game)
   - But CDN performance affects initial load

4. Check game state:
   - How long has game been running? (memory leak?)
   - How large is snake? (physics get harder)
   - DevTools Performance tab: Record 10 seconds, check FPS graph

5. Reproduce steps:
   - Fresh browser window
   - Single tab, minimal extensions
   - Take screenshot/video
   - Note exact FPS shown in HUD
```

Resolution:

```
If FPS consistently &lt; 30:
  → Not a bug, device limitation
  → Suggest using better device
  → Consider feature flags for perf-limited devices

If FPS sometimes low (intermittent):
  → Could be other processes
  → Suggest closing other apps
  → Monitor for memory leak
```

3.3 Data Loss Troubleshooting

Symptom: High scores disappeared

```
DIAGNOSIS:

1. Check if intentional:
   - Did user clear browsing data?
   - Did they use incognito/private mode?

2. Check storage:
   - Open DevTools (F12)
   - Application → Local Storage
   - Search for "snakeEvolution"
   - Should see entries: leaderboard, settings, etc.
   - If empty: Storage was cleared
```

```
3. Check checksum:
   - Log shows: "Checksum validation: PASS" or "FAIL"
   - If FAIL: Data was corrupted, recovery fallback triggered

4. Recovery attempt:
   - Check localStorage backup (v2+)
   - Restore from browser history if available
```

Prevention:

```
- Document how to back up scores (see User Guide)
- Add export feature (planned v2)
- Add automatic cloud backup (planned v2)
```

4. Maintenance Procedures

4.1 Daily Maintenance Checklist

```
Every morning, verify:

[ ] Game accessible at https://snake-evolution.netlify.app
    Command: curl -I https://snake-evolution.netlify.app
    Should return: HTTP 200

[ ] Netlify deployment status
    Check: https://app.netlify.com/
    Should show: "Published"

[ ] Error rate &lt; 0.1%
    Check: Netlify Analytics dashboard
    Should show: Error rate trending down

[ ] No critical errors in logs
    Check: Sentry.io dashboard (v2+)
    Should show: 0 critical issues

[ ] Uptime monitor still active
    Command: ping https://snake-evolution.netlify.app

[ ] Performance metrics stable
    Check: Lighthouse score (target: 90+)
    Should show: No regression
```

4.2 Weekly Maintenance Checklist

```
Every Friday, perform:

[ ] Review GitHub issues
    - Any new bug reports?
    - Any support requests pending?
    - Any performance issues emerging?

[ ] Check dependency updates
    - npm outdated (in dev)
    - Any security vulnerabilities?
    - Any major updates to consider?

[ ] Backup user data
    - Export leaderboard snapshots
    - Archive logs for compliance
    - Verify backups are readable

[ ] Performance trending
    - Compare this week vs last week
    - Any performance regressions?
    - Any unusual patterns?

[ ] User metrics
    - DAU (Daily Active Users)
    - Average session duration
    - Retention D1/D7
```

4.3 Monthly Maintenance Window

```
First Monday of month (off-peak hours):

PLANNED MAINTENANCE:

1. Dependency updates (npm)
   - Review security patches
   - Test thoroughly before deploying
   - Deploy to staging first
   - Verify in production

2. Browser compatibility check
   - Test on latest browser versions
   - Verify on mobile (iOS + Android)
   - Check for deprecation warnings

3. Performance optimization
   - Run Lighthouse full audit
   - Profile with DevTools
   - Optimize any bottlenecks
   - Document improvements

4. Backup rotation
```

```
    - Archive logs older than 90 days
    - Test backup restore
    - Document backup inventory

 5. Documentation updates
    - Update this guide with any changes
    - Update troubleshooting section
    - Update known issues
    - Communicate to team
```

5. Monitoring & Alerting

5.1 Key Metrics to Monitor

Metric	Target	Alert Threshold	Check Frequency
Uptime	99.9%	< 99.5%	Every 5 min
Error Rate	< 0.1%	> 0.5%	Every 5 min
Page Load Time	< 2s	> 5s	Every 10 min
FPS (P95)	55+	< 30	Every 1 min (in-game)
Crash Rate	< 0.5%	> 1%	Every 10 min
CDN Performance	< 200ms	> 1000ms	Every 5 min

5.2 Alert Configuration

High Priority (Page/SMS):

- Uptime < 99%
- Error rate > 1%
- Crash rate > 2%
- CDN down

Medium Priority (Email/Slack):

- Performance degradation
- Dependency vulnerability
- Backup failure
- Storage quota warning

Low Priority (Dashboard only):

- Minor updates available
- Documentation needs refresh
- Non-critical suggestions

6. Compatibility & Evolution Planning

6.1 Browser Compatibility Strategy

```
Current (v2.1):
  Supports: Chrome 90+, Firefox 88+, Safari 14+, Edge 90+

v2.1 Maintenance:
  Continue supporting above versions
  Drop support for versions &lt; N-24 months old

v3.0 (future):
  May require Chrome 100+, Firefox 100+, etc.
  Plan deprecation cycle: 3 months notice
```

6.2 Feature Deprecation Policy

```
Lifecycle:

ACTIVE (v2.1)
  - Fully supported
  - Bug fixes applied
  - Performance optimizations

DEPRECATED (announced)
  - Final release with feature
  - 3-month notice period
  - Users get warning

SUNSET (removed)
  - Feature no longer exists
  - Users guided to alternative
  - No support provided
```

6.3 Data Migration for Future Updates

Scenario: Upgrading localStorage format

```
OLD FORMAT (v1.0):
{
  "score": 100,
  "stage": 3
}

NEW FORMAT (v2.0):
{
  "score": 100,
  "stage": 3,
  "timestamp": "2025-11-06T14:30:00Z",
```

```
  "version": 2
}

MIGRATION LOGIC:
if (loadedData.version === undefined) {
  // Upgrade v1 → v2
  loadedData.version = 2;
  loadedData.timestamp = Date.now();
  save(loadedData);
}
```

7. Incident Response

7.1 Incident Severity Classification

Severity	Impact	Response Time	Escalation
P1 - Critical	Game down or unplayable	15 minutes	On-call engineer
P2 - Major	Feature broken	1 hour	Team lead
P3 - Minor	Cosmetic or workaround exists	4 hours	Next work day
P4 - Trivial	Documentation or non-issue	Next sprint	No escalation

7.2 Incident Response Procedure

```
STEP 1: DETECT &amp;amp; VERIFY (5 min)
  [ ] Alert triggered
  [ ] Confirm it's real (not false positive)
  [ ] Gather initial data
  [ ] Page on-call engineer

STEP 2: ASSESS IMPACT (5 min)
  [ ] How many users affected?
  [ ] Are new users completely blocked?
  [ ] Can existing users continue?
  [ ] Estimate business impact
  [ ] Set severity level (P1-P4)

STEP 3: COMMUNICATE (Immediate)
  [ ] Post status: "Investigating issue"
  [ ] Notify affected users (email/SMS)
  [ ] Start incident in tracking system
  [ ] Create Slack channel #incident-2025-11-06

STEP 4: INVESTIGATE (Ongoing)
  [ ] Check application logs
  [ ] Check infrastructure (Netlify, DNS, CDN)
  [ ] Check third-party services (Sentry, etc.)
  [ ] Identify root cause
  [ ] Run diagnostics
```

```
STEP 5: REMEDIATE (Depends on severity)
  P1: IMMEDIATE
    - Rollback to previous version if needed
    - Or apply emergency fix
    - Deploy and verify

  P2: URGENT (within 1 hour)
    - Prepare fix or workaround
    - Test in staging
    - Deploy to production

  P3: PLANNED
    - Schedule fix for next sprint
    - Or apply next release
    - Monitor for now

STEP 6: VERIFY RESOLUTION (5 min)
  [ ] Confirm fix worked
  [ ] Monitor for regressions
  [ ] Check error logs cleared
  [ ] Verify users not affected

STEP 7: COMMUNICATE RESOLUTION (Immediate)
  [ ] Post: "Issue resolved"
  [ ] Thank users for patience
  [ ] Provide details (what happened, why, what we're doing)
  [ ] Close Slack channel

STEP 8: POST-INCIDENT (Within 24 hours)
  [ ] Write post-mortem
  [ ] Root cause analysis
  [ ] Action items to prevent recurrence
  [ ] Share learnings with team
  [ ] Update documentation
```

7.3 Incident Post-Mortem Template

```
INCIDENT: [Severity] [Brief description]
DATE: [Date/Time when detected]
DURATION: [How long it lasted]

IMPACT:
  - Users affected: [Number]
  - Revenue impact: [If applicable]
  - Reputation impact: [Assessment]

ROOT CAUSE:
  [Detailed explanation of what happened and why]

TIMELINE:
  14:30 - Alert triggered
  14:35 - Incident confirmed
  14:45 - Root cause identified
  14:55 - Fix deployed
```

```
  15:05 - Issue resolved

WHAT WENT WELL:
  ✅ [Good things to celebrate]

WHAT DIDN'T GO WELL:
  ❌ [Things to improve]

ACTION ITEMS:
  1. [Specific action to prevent recurrence]
  2. [Owner assigned]
  3. [Due date]

LESSONS LEARNED:
  - [Key insight #1]
  - [Key insight #2]
```

8. Customer Support Guidelines

8.1 Support Channel Response Times

Channel	Response Time	Owner
Critical Issues	15 min	On-call
GitHub Issues	4 hours	Dev team
Email	24 hours	Support
Feedback Form	48 hours	Product

8.2 Support Response Template

```
RESPONSE TEMPLATE:

Subject: [RE] Your issue - [Brief solution]

Hi [User],

Thank you for reporting this issue!

WHAT YOU REPORTED:
  [Paraphrase their issue]

WHAT WE FOUND:
  [Explanation or known issue info]

WHAT YOU CAN TRY:
  [Step-by-step workaround]

NEXT STEPS:
  If this doesn't work, please:
  1. [Specific next action]
```

```
   2. Reply with result

Best regards,
Snake Evolution Support Team
```

8.3 Common Support Requests

"How do I get my scores back?"

Response:
Scores are saved automatically to your device. They won't be lost unless:

1. You clear browser data (data can't be recovered)
2. You use private/incognito mode (scores not saved)
3. Storage quota exceeded (check if you need space)

Prevention: Screenshot your leaderboard for backup.

"Why does the game crash on my phone?"

Response:
This could be due to:

1. Low device RAM (try closing other apps)
2. Outdated browser (update to latest version)
3. Known issue (check if in list)

Troubleshooting: [Provide steps from Section 3.2]

"Can I play offline?"

Response:
In v1.0, the game loads online but can play offline after caching. Full offline support is planned for v2.0.

9. Performance Baselines & Targets

9.1 Current Baseline (v2.1)

Metric	Actual	Target	Status
Page Load Time	1.2s	< 2s	✅ Pass
First Paint	0.8s	< 1s	✅ Pass
FPS (Average)	58.5	58+	✅ Pass
FPS (P99)	56	55+	✅ Pass

Metric	Actual	Target	Status
Bundle Size	512 KB	< 1 MB	✅ Pass
Input Latency	16ms	< 50ms	✅ Pass
Memory Usage	30 MB	< 50 MB	✅ Pass

9.2 Performance Regression Procedure

```
WEEKLY PERFORMANCE CHECK:

Compare current week vs last week:

[ ] Lighthouse score (track trending)
[ ] Bundle size (alert if &gt; 5% increase)
[ ] Load time (alert if &gt; 10% increase)
[ ] FPS metrics (alert if &lt; 55 FPS P95)

If regression detected:
1. Identify commit that introduced it
2. Run profiler to find bottleneck
3. Create P2 issue for fix
4. Schedule optimization work
```

10. Disaster Recovery

10.1 Disaster Recovery Plan

Scenario 1: Netlify Goes Down

```
IMPACT: Game completely unavailable

RESPONSE TIME: &lt; 30 minutes

STEPS:
1. Activate backup host (AWS S3 + CloudFront)
2. Update DNS to point to backup
3. Verify game accessible
4. Communicate status to users
5. Wait for Netlify recovery
6. Switch back when stable

RTO (Recovery Time Objective): &lt; 1 hour
RPO (Recovery Point Objective): &lt; 5 min (auto-sync)
```

Scenario 2: Data Corruption in localStorage

```
IMPACT: Users can't access their scores
```

```
RESPONSE TIME: &lt; 4 hours

MITIGATION:
1. Enable automatic backups (v2+)
2. Use checksum validation (already implemented)
3. Provide recovery tool (v2+)
4. Manual data export/restore (supported)
```

Scenario 3: Critical Code Bug in Production

```
IMPACT: Depends on severity (P1/P2/P3)

RESPONSE TIME: &lt; 15 min for P1

PROCEDURES:
1. Identify bug and severity
2. If critical:
   a. Rollback to previous version
   b. Deploy fix
   c. Verify in staging first
3. If non-critical:
   a. Schedule for next release
   b. Provide workaround if needed
```

10.2 Backup & Recovery Checklist

```
DAILY:
  [ ] Automated backup of leaderboard (v2+)
  [ ] Verify backup file size is reasonable
  [ ] Test one backup is readable
  [ ] Monitor backup job completion

WEEKLY:
  [ ] Restore from backup in staging
  [ ] Verify data integrity
  [ ] Document restore time

MONTHLY:
  [ ] Full disaster recovery drill
  [ ] Time the entire restore process
  [ ] Document procedures
  [ ] Update runbooks if needed
```

11. Support Escalation Matrix

```
TIER 1 - SUPPORT TEAM:
  Handles: Common issues, known workarounds
  Authority: Provide documentation, suggest steps
  Escalate to Tier 2 if:
    - User requires code change
```

```
    - Issue not in knowledge base
    - Emergency/critical severity

TIER 2 - DEV TEAM:
  Handles: Bug reproduction, root cause analysis
  Authority: Create issues, assign priorities
  Escalate to Tier 3 if:
    - Infrastructure change needed
    - Security concern
    - Architectural change required

TIER 3 - DEVOPS / SENIOR ENGINEER:
  Handles: Infrastructure, deployment, architecture
  Authority: Change production, approve major releases
  Escalate to Tier 4 if:
    - Business impact very high
    - External stakeholder approval needed

TIER 4 - MANAGEMENT:
  Handles: Business decisions, customer escalations
  Authority: Commit resources, negotiate timelines
```

12. Document Information

Item	Details
Document Type	Maintenance & Support Guide
Version	1.0
Date Published	November 2025
Standard	IEEE 1063-2001
Language	English
Audience	DevOps, SRE, Support Staff

Document Version History

Version	Date	Changes
1.0	2025-11-06	Initial complete maintenance guide with incident response

End of Maintenance & Support Guide

For user support, refer to User Guide v1.0.
For development procedures, see Implementation Guide v2.0.
For DevOps procedures, see DevOps & CI/CD Guide v2.0.
[1] [2]

⁂

1. PRD-v2-1-EN.pdf

2. Functional-Analysis-EN.pdf