Green Man Tavern - Validation Strategy Master Summary

Your Comprehensive Quality Assurance Roadmap

📊 Executive Overview

You're implementing a **three-tier quality assurance system** for Green Man Tavern using Claude Code:

Tier 1: Real-Time Standards (During Development)

- Follow `.claude/project_standards.md` conventions
- Check work before committing
- Maintain consistency across all modules

Tier 2: Periodic Checks (Weekly/Monthly)

- Weekly health checks (5 min)
- Bi-weekly architecture drift detection (10 min)
- Monthly integration tests (15 min)
- Quarterly deep audits (30 min)

Tier 3: Gate-Based Checks (Before Deployments)

- Security audits
- Performance profiling
- Pre-deployment verification
- Go/no-go decision criteria

🎯 The Three Documents You Now Have

Document 1: Full Strategy (50+ pages when printed)

File: `Structural Validation & Consistency Strategy.md`

Contains:

- Complete architecture decision framework
- 6 types of initial deep audits
- 4 types of ongoing periodic checks

- Setup and maintenance procedures
- Detailed explanations of each check
- How to interpret results

Use When: You want to understand the full methodology or train someone else

---

Document 2: Ready-to-Use Prompts (30+ pages)

File: Claude Code Prompts (Ready-to-Use).md

Contains:

- 1 Master Initial Audit prompt
- 6 Ongoing check prompts (Weekly/Bi-weekly/Monthly/Quarterly)
- 6 Special-purpose prompts (Security, Performance, Testing, Documentation, Refactoring, Custom)
- Copy/paste ready
- No modification needed

Use When: You want to run an audit — just copy/paste

Document 3: Quick Start (10-15 minutes)

File: Quick Start Guide.md

Contains:

- 7-step setup process
- How to create reference files
- What to do with results
- Troubleshooting guide
- Learning paths for different experience levels

Use When: You're getting started today

Document 4: This Summary

File: Validation Strategy Master Summary.md

Contains:

- High-level overview (this document)
- What you're building
- How the three documents fit together
- Decision matrix
- Implementation checklist
- Timeline
- Next actions

Use When: You need the 30,000-foot view

🗺️ How the Documents Work Together

```
YOU ARE HERE
  ↓
┌──────────────────────────────────────┐
│ Read: Master Summary (this document) │
│ Time: 5 minutes                      │
│ Purpose: Understand what you're doing │
└──────────────────────────────────────┘
  ↓
  DECIDE: Am I ready to start TODAY?
  ├─ YES → Go to Next Step
  └─ NO → Read Full Strategy for deeper understanding
  ↓
┌──────────────────────────────────────┐
│ Follow: Quick Start Guide            │
│ Time: 30 minutes                     │
│ Purpose: Set everything up           │
└──────────────────────────────────────┘
  ↓
┌──────────────────────────────────────┐
│ Use: Claude Code Prompts             │
│ Time: As needed (5-30 min per audit) │
│ Purpose: Run actual audits           │
└──────────────────────────────────────┘
  ↓
┌──────────────────────────────────────┐
│ Reference: Full Strategy             │
│ Time: As questions arise             │
│ Purpose: Deep dives on specific checks │
└──────────────────────────────────────┘
```

✅ Implementation Checklist

Week 1: Setup

☐ Day 1 (30 min):
- ☐ Read this Master Summary
- ☐ Read Quick Start Guide sections 1-4

☐ Day 2 (30 min):
- ☐ Follow Quick Start: Steps 1-4 (create files)
- ☐ Create .claude/audit_results/ folder

☐ Day 3 (15 min):
- ☐ Quick Start Step 5: Run initial audit with Claude Code

☐ Day 4 (30 min):
- ☐ Process initial audit results
- ☐ Create tickets for CRITICAL/HIGH issues
- ☐ Update AUDIT_BASELINE.md

☐ Day 5 (5 min):
- ☐ Add weekly/monthly reminders to calendar
- ☐ Bookmark .claude/audit_results/
- ☐ Share approach with team (if applicable)

Week 2+: Ongoing

☐ Every Monday 9:00 AM (5 min):
- ☐ Run "Weekly Structural Health Check"
- ☐ Note any 🔴 issues
- ☐ Fix immediately if found

☐ Every Thursday 9:00 AM (10 min):
- ☐ Run "Bi-Weekly Architecture Drift"
- ☐ Verify patterns are consistent

☐ 1st of month 9:00 AM (15 min):
- ☐ Run "Monthly Integration Test"
- ☐ Test complete user journey

☐ End of each quarter (30 min):
- ☐ Run "Quarterly Deep Audit"
- ☐ Compare to baseline
- ☐ Update IMPLEMENTATION_LOG.md

☐ Before any production deployment:
  ☐ Run "Pre-Deployment Checklist"
  ☐ Get green light before shipping

Optional: Advanced Monitoring

☐ When users report bugs:
  ☐ Run "Security Deep-Dive" (if security-related)
  ☐ Run "Custom Module Audit" on affected code

☐ When code feels messy:
  ☐ Run "Refactoring Guidance"
  ☐ Prioritize improvements

☐ When performance degrades:
  ☐ Run "Performance Profiling"
  ☐ Identify bottlenecks

☐ Before hiring/onboarding:
  ☐ Run "Documentation Verification"
  ☐ Ensure developer can ramp up

📈 Timeline & Effort

Initial Setup: 2.5 hours

Creating files: 15 min
First audit: 15 min
Processing results: 30 min
Fixing CRITICAL issues: 60-90 min
Setting up calendar reminders: 5 min

Weekly Ongoing: 15 min/week

Monday health check: 5 min
Thursday drift check: 10 min

Monthly Additional: 15 min/month

Full integration test: 15 min

Quarterly Additional: 30 min/quarter

```
Deep audit: 30 min
```

Before Deployment: 20-30 min/release

```
Pre-deployment checklist: 20-30 min
```

Total ongoing commitment: ~1 hour per week

🎯 Decision Matrix: Which Document to Use When?

Situation	Document	Time
"What am I building?"	Master Summary	5 min
"How do I start today?"	Quick Start	30 min
"I need an audit prompt"	Prompts (Ready-to-Use)	1 min
"Why does this approach work?"	Full Strategy	20 min
"How do I handle this specific issue?"	Full Strategy (relevant section)	10 min
"I'm training a team member"	Quick Start + Full Strategy	60 min
"I want to customize the approach"	Full Strategy	30 min

🚨 Critical Success Factors

Your validation system will **succeed** if:

1. ✅ **You run audits on schedule** (not "when you remember")
2. ✅ **You fix CRITICAL issues immediately** (not "next sprint")
3. ✅ **You review results (don't ignore)** (not "archive and forget")
4. ✅ **You track trends over time** (not "point-in-time snapshots")
5. ✅ **You use results for deployment decisions** (not "deploy anyway")

Your validation system will **fail** if:

1. ❌ You skip weekly checks ("we're too busy")
2. ❌ You ignore CRITICAL issues ("we'll fix it later")
3. ❌ You don't update IMPLEMENTATION_LOG.md ("no time")
4. ❌ You deploy without pre-deployment checklist ("it's just a small change")

5. ❌ You treat alerts as noise ("another false positive")

🎓 For Different Roles

Solo Developer

```
- Run all audits yourself
- Follow the Quick Start guide
- Spend 1 hour/week on validation
- Use results to guide your development priorities
```

Small Team (2-3 developers)

```
- Assign one person to coordinate audits (rotates weekly)
- Share results in team standup
- Discuss priorities together
- Use IMPLEMENTATION_LOG.md as shared truth
- Add pre-deployment gate (no deploy without green light)
```

Larger Team (4+ developers)

```
- Assign QA/DevOps lead to manage system
- Integrate audits into CI pipeline (if possible)
- Share results in sprint planning
- Create team standards document
- Use audit results as hiring/onboarding metrics
- Consider weekly QA meetings to review findings
```

With a Product Manager/Stakeholder

```
- Run quarterly deep audits (comprehensive)
- Show trend graphs (coverage, issue count)
- Explain deployment risk percentage
- Use metrics to justify refactoring/tech debt work
- Document ROI: "Caught X bugs before users saw them"
```

🔄 Integration with Your Workflow

Before Code Review

```
Run personal pre-commit check:
```

```
claude-code "Quick review of my changes today"
```

Before Merge to Main

```
Run CI-equivalent check:
claude-code "Architecture consistency check"
```

Before Release

```
Run pre-deployment:
claude-code "Pre-deployment checklist"
```

After Deployment

```
Run post-deployment:
claude-code "Weekly health check" (next Monday as usual)
```

🛠️ Customization Points

You can customize this approach for your needs:

Audit Frequency

- Too frequent (daily)? → Too much noise
- Current (weekly/bi-weekly/monthly)? → Recommended sweet spot
- Less frequent (monthly only)? → Miss issues between checks
- Recommended: Keep weekly + monthly at minimum

Issue Severity Ratings

- Default: 🔴 Critical / 🟡 High / 🟠 Medium / 🟢 Low
- Customize: Add your team's own criteria for each level
- Document: In `.claude/project_standards.md`

Deployment Gates

- Minimum criteria: ✅ Tests pass + No Critical issues
- Optional additions: Coverage >75% + No High issues
- Document: In your deployment guide

Report Format

- Default: Markdown files in `.claude/audit_results/`
- Alternative: Email summaries, dashboard, spreadsheet
- Choose: Whatever your team prefers

🚀 Get Started Today: 3 Action Items

Action 1 (Now): Read

- Read this Master Summary (already done! ✓)
- Read Quick Start Guide section 1-2 (5 min)

Action 2 (Next 30 min): Setup

- Follow Quick Start steps 1-4
- Create three reference files
- Create .claude/audit_results/ folder

Action 3 (Next 15 min): Run

- Copy the Master Initial Audit prompt
- Open Claude Code
- Paste and run the audit
- Watch it analyze your codebase

Total time investment: 50 minutes. Returns: Clarity on your codebase health.

📞 Support & Questions

If you get stuck:

1. Check Quick Start troubleshooting section
2. Review Full Strategy for the specific check
3. Look at example audit output
4. Add notes to IMPLEMENTATION_LOG.md

If you want to go deeper:

1. Read Full Strategy completely

2. Customize prompts for your team
3. Integrate with your CI/CD pipeline
4. Train team members on the approach

If you have suggestions:

1. Document in IMPLEMENTATION_LOG.md
2. Discuss with team
3. Update .claude/project_standards.md if needed
4. Share learnings

✨ What Success Looks Like

After 3 months using this system:

Week 1:

- Found 12 issues (3 Critical, 5 High, 4 Medium)
- Started fixing immediately
- Updated documentation

Week 4:

- Fixed all Critical issues
- Fixed 4 of 5 High issues
- Added 30 new tests
- Coverage: 45% → 58%

Week 8:

- All High issues fixed
- Started addressing Medium issues
- Added more tests
- Coverage: 58% → 68%

Week 12:

- Zero Critical issues for 4 weeks straight
- Zero High issues for 2 weeks

- New deployments feel confident
- Coverage: 68% → 76%
- Team trusts the codebase

🎯 Long-Term Benefits

After 6-12 months:

✅ **Code Quality**: Consistently high, trends upward
✅ **Development Speed**: New features ship faster (less debugging)
✅ **Confidence**: Everyone trusts the codebase
✅ **Onboarding**: New developers ramp up in days, not weeks
✅ **Bugs**: Caught before users see them
✅ **Deployments**: Simple and routine (not scary)
✅ **Technical Debt**: Minimal and managed
✅ **Knowledge**: Clear documentation of decisions

📚 Your Complete Toolkit

You now have:

Document	Purpose	Length	When to Use
Master Summary	High-level overview	This page	First thing
Quick Start	Get going in 30 min	5 pages	Setup day
Claude Code Prompts	Ready-to-use audits	30 pages	Every audit
Full Strategy	Deep understanding	50 pages	Reference
`.claude/project_standards.md`	Team conventions	1 page	Daily coding
`AUDIT_BASELINE.md`	Track progress	2 pages	After audits
`IMPLEMENTATION_LOG.md`	What you've built	3 pages	Ongoing

🎉 You're Ready!

Everything you need is documented. The approach is proven. The tool (Claude Code) is ready.

What remains is execution.

Your Next Step:

Follow the Quick Start Guide and run your first audit today.

It will take 50 minutes total and give you immediate clarity on your codebase.

This Strategy Was Built For:

- Phoenix LiveView + Elixir projects
- Database-driven applications
- AI/LLM integration (MindsDB, etc.)
- Real-world progressive web apps
- Team-based development
- Production quality standards

Version: 1.0
Date: Today
Status: Complete and ready to implement
Maintenance: Update quarterly as your project evolves