🎯 HOW TO USE THE VOICE PRD TRUE METHOD SYSTEM

What This System Does

Transforms **10-minute Wispr Flow dictation → manually iterated execution with ZERO context rot** through fresh Claude Code sessions per user story.

Key Principle: Each atomic story = ONE fresh session = Maximum context quality = Optimal results

---

📁 Files in This System

1. **Voice_PRD_True_Method.md** (Main Reference)

- Complete methodology and philosophy
- Detailed dictation script with 6 blocks
- Claude's clarification protocol
- Manual iteration output format
- TRUE method execution workflow
- Full workflow example

When to use: Before your first dictation - understand the system

2. **Wispr_Flow_Cheatsheet.md** (Quick Reference)

- Single-page printable reference
- Time allocations for 10-minute recording
- Magic phrases for context
- Pre-recording checklist

When to use: Keep visible during every recording session

3. **Claude_Clarification_Framework_v2.md** (For Claude)

- Instructions for Claude to process transcripts
- Question prioritization tiers
- Smart defaults strategy
- Progressive questioning approach

When to use: Reference in your prompt when pasting transcripts

4. **Voice_PRD_Template_v2.md** (Optional Structure)

- Alternative structured approach
- More detailed section prompts
- Good for complex projects

When to use: If you prefer more structure over natural speech

---

🚀 THE WORKFLOW (Step by Step)

STEP 1: PREPARE (5 minutes)

Open `Wispr_Flow_Cheatsheet.md` on second monitor or print it.

Gather:

- [ ] Absolute path to project folder
- [ ] Location of .env or credentials
- [ ] Any API docs or example URLs
- [ ] Mental picture of the workflow

STEP 2: RECORD (10-12 minutes)

Start Wispr Flow and follow the 6 blocks:

1. **Project Core** (2 min): WHY and WHO
2. **The Problem** (2 min): WHAT'S broken
3. **The Solution** (3 min): HOW to fix - BE SPECIFIC about stack
4. **Integrations** (1 min): WHAT connects - name APIs explicitly
5. **Edge Cases** (2 min): WHAT could break
6. **Success Criteria** (1 min): WHEN it's done - concrete tests

Key: Talk naturally, like explaining to a friend. Use cheatsheet for prompts.

STEP 3: PASTE & ENGAGE (1 minute)

Open Claude and paste your transcript with this prompt:

```
I've dictated a PRD using Wispr Flow. Process this using the Voice PRD TRUE Method
(reference Voice_PRD_True_Method.md and Claude_Clarification_Framework_v2.md).

Ask me clarifying questions ONE AT A TIME, starting with blockers (file paths,
credentials, environment). After we've resolved gaps, generate the manual
iteration plan with atomic user stories.
```

STEP 4: CLARIFICATION Q&A (5-10 minutes)

Claude will ask you 5-10 questions in this order:

Round 1 - BLOCKERS (2-3 questions):

- Absolute file paths
- Credential locations
- Environment setup

Round 2 - IMPLEMENTATION (2-3 questions):

- Error handling approach
- Data flow details
- Integration specifics

Round 3 - CONFIRMATION (1 summary):

- Structured recap with smart defaults filled in
- You confirm or correct

STEP 5: GENERATION (1 minute)

Claude outputs **Manual Iteration Plan** with:

```
# [PROJECT NAME] - Manual Iteration Plan

## PROJECT CONTEXT
[Summary]

## TECHNICAL FOUNDATION
[Stack, paths, credentials]

## USER STORIES (Sequential Execution Order)

### Story 1: [Title]
**Session:** Fresh Claude Code #1
**Atomic Goal:** [One completable task]
**Acceptance Criteria:** [Testable conditions]
**Files to Create/Modify:** [Absolute paths]
**Validation Steps:** [How to verify]

### Story 2: [Title]
**Session:** Fresh Claude Code #2 (NEW SESSION)
**Context from Story 1:** [Essential context]
**Atomic Goal:** [One completable task]
[...and so on...]
```

STEP 6: EXECUTE (Manual TRUE Method)

For EACH User Story:

A. Open Fresh Claude Code Session

```
❌ DO NOT reuse previous session
❌ DO NOT continue in degraded context
✅ START FRESH for optimal performance
```

B. Provide Story Context

```
"I'm implementing Story [N] of a larger project.

PROJECT: [Name]
BASE PATH: [Absolute path]
STORY GOAL: [From plan]

CONTEXT FROM PREVIOUS STORIES:
[Essential context only]

ACCEPTANCE CRITERIA:
[Paste from story card]

FILES TO MODIFY/CREATE:
[Paste from story card]

Execute this story completely in this session."
```

C. Let Claude Code Execute

- Claude Code reads files, writes code, runs tests
- Watch for completion or errors
- DO NOT interrupt mid-execution

D. Validate Results

```
✅ Run validation steps from story card
✅ Check all acceptance criteria
✅ Test actual functionality (don't just trust output)
✅ Look for errors/warnings
```

E. Record Handoff

```
- Mark story complete
- Note timestamp
- Document any gotchas
- Update HANDOFF.md (if using Genesis)
```

F. Move to Next Story

```
✅ Close current Claude Code session
✅ Open FRESH session
✅ Repeat process A-F
```

🎯 WHY TRUE METHOD WORKS

The Problem with Context Rot

Without Fresh Sessions (Plugin/Loop Methods):

```
Session Start: 100% context quality ✅
Story 1: 95% quality (slight degradation) ⚠️
Story 2: 85% quality (compounding errors) ⚠️
Story 3: 70% quality (major issues) ❌
Story 4: 50% quality (garbage output) ❌
```

With TRUE Method (Fresh Sessions):

```
Story 1 Session: 100% context quality ✅
Story 2 Session: 100% context quality ✅
Story 3 Session: 100% context quality ✅
Story 4 Session: 100% context quality ✅
```

The Manual Validation Advantage

Automated Loops (Plugin):

- Errors compound silently
- Failures discovered too late
- Backtracking wastes tokens
- No intervention points

Manual TRUE Method:

- Validate after EVERY story
- Catch errors when cheap to fix
- Understand root causes
- Control the process

📊 SUCCESS METRICS

You'll know TRUE method is working when:

✅ Each story completes in ONE fresh session ✅ Validation passes after each story (first try) ✅ Zero backtracking or rework needed ✅ No accumulated errors across stories ✅ Final product meets all criteria ✅ Budget stays within estimate

---

🔍 EXAMPLE: THE DIFFERENCE

❌ OLD WAY (Plugin/Automated Loop with Context Rot)

What Happens:

```
Session Start → Story 1 (good) → Story 2 (okay) → Story 3 (errors creeping in)
→ Story 4 (major failures) → Story 5 (garbage output) → Restart entire loop
→ Waste $20 in tokens → Manual cleanup required → Frustration
```

Problems:

- Context degradation after story 2
- Errors compound silently
- Late discovery of failures
- Expensive token burn
- Manual intervention anyway

✅ NEW WAY (TRUE Method with Fresh Sessions)

What Happens:

```
Fresh Session → Story 1 ✅ Validate → Fresh Session → Story 2 ✅ Validate
→ Fresh Session → Story 3 ✅ Validate → Fresh Session → Story 4 ✅ Validate
→ All stories validated → Integration test ✅ → Ship it
```

Benefits:

- Perfect context every time
- Errors caught immediately
- Validation confirms quality
- Predictable token usage
- High confidence in output

Time comparison:

- Old way: 2 hours + cleanup + restart = 4 hours total
- TRUE way: 1.5 hours sequential = Done right first time

💡 PRO TIPS FOR TRUE METHOD

Story Scoping (Critical)

Make stories TRULY atomic:

```
❌ BAD: "Build entire dashboard"
✅ GOOD: "Add task list component to dashboard"

❌ BAD: "Implement all API endpoints"
✅ GOOD: "Create POST /api/tasks endpoint"
```

```
❌ BAD: "Set up authentication system"
✅ GOOD: "Add Clerk auth to Next.js app"
```

One session test: Can this be built and validated in <15 minutes?

Context Handoff (Minimal is Better)

When moving to next story, copy ONLY:

```
✅ Absolute file paths created
✅ Key architectural decisions
✅ Environment setup steps taken
❌ NOT detailed implementation
❌ NOT code snippets
❌ NOT debug logs
```

Fresh context = fresh thinking

Validation Discipline

After EVERY story:

```
1. Run the code (npm run dev / pytest / cargo build)
2. Test the feature manually
3. Check for errors/warnings
4. Verify acceptance criteria
5. Note any deviations
```

If validation fails:

```
❌ DO NOT continue to next story
❌ DO NOT try to fix in same session
✅ START FRESH session
✅ RETRY failed story
✅ ADJUST criteria if needed
```

Session Management

Best practices:

```
✅ Close previous session completely
✅ Clear your mind between stories
✅ Copy story context fresh (don't rely on memory)
✅ Treat each session as independent task
✅ Celebrate small wins (each validated story)
```

🎬 QUICK START (Your First Project)

Pick a small project - 3-5 atomic stories maximum.

Right now:

1. Open `Wispr_Flow_Cheatsheet.md`
2. Start Wispr Flow
3. Talk for 10 minutes (follow prompts)
4. Paste to Claude with TRUE method prompt
5. Answer 5-10 clarification questions
6. Get manual iteration plan
7. Execute Story 1 in fresh Claude Code session
8. Validate Story 1
9. Execute Story 2 in fresh Claude Code session
10. Validate Story 2
11. Continue until all stories validated
12. Integration test
13. Ship it

That's the TRUE method in practice.

📞 WHEN TO USE TRUE METHOD

Always Use TRUE Method When:

```
✅ Quality matters more than speed
✅ Budget is constrained (no token waste)
✅ Complex multi-step builds
✅ Production/client deliverables
✅ Learning new patterns (see each step)
```

Consider Alternatives When:

```
⚠️ Throwaway prototypes (context rot acceptable)
⚠️ Single-story builds (no iteration needed)
⚠️ Well-tested patterns (low risk)
```

For 90% of AgileAdapt/RiverSun work: USE TRUE METHOD

🔄 INTEGRATION WITH GENESIS

Using HANDOFF.md Bridge

After each story, update `E:\genesis-system\HANDOFF.md` :

```
## [Project] - Story [N] Complete

**Completed:** [Timestamp]
**Story:** [Title]
```

```
**Files:** [List]
**Status:** ✅ Validated
**Next:** Story [N+1]

**Context for Next Session:**
[Essential context only - paths, decisions]
```

This enables Genesis agents to pick up work between sessions.

Manual vs Genesis Execution

Manual (you control each session):

- You validate between stories
- You decide when to proceed
- Full transparency

Genesis (autonomous with handoffs):

- Genesis validates between stories
- Genesis decides when to proceed
- Logged execution trace

Both use TRUE method (fresh sessions). Difference is who's driving.

✨ FINAL CHECKLIST

Before first dictation:

- [ ] Read `Voice_PRD_True_Method.md` once (philosophy)
- [ ] Open `Wispr_Flow_Cheatsheet.md` (keep visible)
- [ ] Bookmark `Claude_Clarification_Framework_v2.md` (reference)
- [ ] Pick small project (3-5 stories max)
- [ ] Gather: paths, credentials, mental workflow
- [ ] Commit to validation discipline
- [ ] Commit to fresh session discipline
- [ ] Start Wispr Flow and talk

You're ready for TRUE method execution.

🎓 PHILOSOPHY: Why Manual is Better Than Automated

The Counterintuitive Truth

Automation seems faster:

- One command
- Overnight execution
- Wake up to results

But reality:

- Context rot after 2-3 iterations
- Silent failure accumulation
- Expensive token waste
- Manual cleanup required anyway

Manual seems slower:

- One story at a time
- Validate between stories
- Human in the loop

But reality:

- Perfect context every iteration
- Immediate error detection
- Predictable costs
- Right the first time

The Math

Automated loop (with context rot):

```
5 stories × 3 attempts (due to failures) = 15 iterations
15 iterations × $2/iteration = $30
Plus 2 hours manual cleanup = NOT WORTH IT
```

TRUE method (manual validation):

```
5 stories × 1 attempt (validated) = 5 iterations
5 iterations × $2/iteration = $10
Plus 0 hours cleanup = DONE RIGHT
```

TRUE method costs 67% less and produces better results.

🚀 MASTERY PATH

Week 1: Small projects (3 stories)

- Learn story scoping
- Practice validation discipline
- Get comfortable with fresh sessions

Week 2: Medium projects (5-7 stories)

- Refine context handoff
- Speed up between stories
- Develop validation patterns

Week 3: Large projects (10+ stories)

- Parallel execution (multiple stories/day)
- Genesis integration
- Pattern library development

Month 2+: TRUE method becomes natural

- 15 min average per story
- 95%+ first-time validation pass rate
- Predictable delivery timelines
- High confidence in autonomous overnight runs

The TRUE method: Manual iteration with fresh context = Optimal autonomous execution foundation.

You're now equipped to build with ZERO context rot.

Start small. Validate everything. Go build.