Agent Orchestration Standard Operating Procedure (SOP)

PURPOSE

MANDATORY: Every development prompt MUST include agent orchestration patterns. This prevents having to repeatedly remind prompt builders to include specialist agent usage.

SCOPE

Applies to ALL TaylorDash development prompts: main system, plugins, infrastructure, testing, documentation.

RULE: NEVER SEND A PROMPT WITHOUT AGENT ORCHESTRATION

STANDARD AGENT ORCHESTRATION BLOCK (COPY TO EVERY PROMPT)

```
AVAILABLE SPECIALIST AGENTS (use these exact agents):
- @task-completion-validator: Detects incomplete, superficial, or fraudulent implementations. Expert at identifying when developers claim completion but haven't delivered working functionality.
- @ui-comprehensive-tester: Thorough UI testing across platforms. Intelligently selects best testing approach (Puppeteer/Playwright/Mobile MCP) based on requirements.
- @Jenny: Implementation vs specification validator. Compares actual code against project requirements and identifies gaps between what was specified and what was built.
- @code-quality-pragmatist: Expert at identifying shortcuts, placeholder code, TODO comments, and over-engineering that doesn't provide real value. Hunts for fake implementations.
- @claude-md-compliance-checker: Verifies adherence to project governance rules, add-only constraints, and documented principles.

AGENT ORCHESTRATION STRATEGY:
Use multiple agents simultaneously for different aspects:
- Have @[agent-1] validate [specific aspect] while @[agent-2] checks [different aspect]
- Use @[agent-3] for [function] while @[agent-4] reviews [quality-concern]
- Deploy @[agent-5] to ensure [governance] throughout

MANDATORY VALIDATION SEQUENCE:
After building each major component, use agents systematically:

1. **Development Phase**: Use @Jenny continuously
   - [Specific compliance requirement for this task]
   - [Architectural constraint to verify]

2. **Implementation Quality**: Use @code-quality-pragmatist during development
   - [Specific anti-pattern to hunt for]
   - [Quality concern to identify]

3. **Functionality Testing**: Use @ui-comprehensive-tester with [appropriate tool]
   - [Specific workflow to test]
   - [Interaction requirement to validate]

4. **End-to-End Validation**: Use @task-completion-validator
   - [Complete user workflow to verify]
   - [Integration requirement to confirm]

5. **Governance Compliance**: Use @claude-md-compliance-checker
   - [Project constraint to ensure]
   - [Governance requirement to check]

EVIDENCE REQUIREMENTS:
- [Agent name] must provide [specific evidence type] showing [what to prove]
- Multiple agents must reach consensus on [critical requirement]
- All validation agents must confirm [completion criteria] before proceeding
```

ORCHESTRATION PATTERNS FOR SPECIFIC SCENARIOS

Pattern 1: New Feature Development

```
AGENT ORCHESTRATION FOR NEW FEATURES:
- @Jenny: Validate requirements compliance throughout development
- @code-quality-pragmatist: Hunt for shortcuts and ensure proper implementation
- @ui-comprehensive-tester: Test user workflows as features are built
- @task-completion-validator: Verify end-to-end functionality
- @claude-md-compliance-checker: Ensure governance adherence
```

Pattern 2: Bug Fixes and Issues

```
AGENT ORCHESTRATION FOR BUG FIXES:
- @task-completion-validator: Verify bug is actually fixed with evidence
- @ui-comprehensive-tester: Test affected workflows thoroughly
- @code-quality-pragmatist: Ensure fix doesn't introduce technical debt
- @claude-md-compliance-checker: Verify fix follows project constraints
```

Pattern 3: Integration and Testing

```
AGENT ORCHESTRATION FOR INTEGRATION:
- @task-completion-validator: Verify integration actually works
- @Jenny: Check integration matches specifications
- @ui-comprehensive-tester: Test cross-system workflows
- @code-quality-pragmatist: Ensure integration is robust, not hacky
```

Pattern 4: Documentation and Close-out

```
AGENT ORCHESTRATION FOR DOCUMENTATION:
- @claude-md-compliance-checker: Verify documentation follows standards
- @task-completion-validator: Confirm documentation matches actual implementation
- @Jenny: Ensure documentation covers all specified requirements
```

ANTI-PATTERNS (FORBIDDEN)

❌ **NEVER DO THIS:**

- Send prompts without agent orchestration instructions
- Use only one agent when multiple are appropriate
- Accept claims without evidence from validation agents
- Skip agent validation for "simple" tasks
- Use generic "check everything" agent instructions

✅ **ALWAYS DO THIS:**

- Include specific agent assignments for the task
- Require evidence from appropriate validation agents
- Use multiple agents simultaneously when possible
- Specify exactly what each agent should validate
- Require agent consensus on critical decisions

PROMPT TEMPLATE WITH ORCHESTRATION

```
ROLE: [Specific role for the task]
Operate add-only, idempotent, local-only. Do not modify Git auth/remotes.

CONTEXT: [Specific context for this task]

[INCLUDE FULL AGENT ORCHESTRATION BLOCK FROM ABOVE]

TASK: [Specific task description]

SPECIFIC REQUIREMENTS: [Task-specific requirements]

CONSTRAINTS: [Project constraints and governance]

EVIDENCE REQUIRED: [What agents must provide as proof]

VALIDATION: [How agents will verify completion]
```

QUALITY CHECKPOINTS

Before sending any prompt, verify:

- ☐ Agent orchestration block included
- ☐ Specific agent assignments for this task

- [ ] Evidence requirements defined
- [ ] Multiple agents assigned where appropriate
- [ ] Validation sequence specified

FAILURE RECOVERY

If a prompt is sent without proper agent orchestration:

1. **Stop the session** and add orchestration immediately
2. **Redirect the agent** to use specialist agents
3. **Require evidence** for any claims made
4. **Update this SOP** if new patterns are discovered

CONTINUOUS IMPROVEMENT

Track agent effectiveness:

- Document which agent combinations work best
- Note any gaps in agent capabilities
- Update orchestration patterns based on results
- Refine evidence requirements based on experience

REMEMBER: Agent orchestration is not optional - it's mandatory for all development prompts.