Mission Control Documentation

Mission Control: Multi-Agent Project Management System

> *"Like having a team of AI specialists coordinated by AI project managers—with built-in safeguards against groupthink."*

Table of Contents

What is Mission Control?

Mission Control is a **multi-agent coordination system** that manages AI-powered project execution. It combines:

- **AI Project Managers** (Chhotu & Cheenu) running on Opus 4.5
- **Specialist AI Agents** for specific tasks (development, design, writing, research)
- **Anti-Groupthink Protocols** to ensure diverse, quality decisions
- **Real-time Dashboard** for human oversight and intervention
- **Supabase Backend** for persistent state and real-time updates

Think of it as a virtual software company where AI agents handle the work, AI PMs coordinate, and you maintain strategic oversight.

The Problem It Solves

1. Single-Agent Limitations

Traditional AI assistants are generalists. They can do many things, but: - Context gets polluted across different task types - No specialization means mediocre results in complex domains - Single point of failure—one confused agent derails everything

2. Coordination Overhead

Managing multiple AI tools manually is exhausting: - Copy-pasting between Claude, ChatGPT, Cursor, etc. - Losing context between sessions - No unified view of project progress - You become the bottleneck

3. AI Sycophancy & Groupthink

AI models tend to agree with each other (and you) too readily: - First opinion anchors all subsequent opinions - No genuine debate or critique - Important concerns get buried - Decisions lack rigor

4. Cost Inefficiency

Using Opus for everything is expensive: - Simple tasks don't need frontier models - No intelligent routing based on task complexity - Token waste on boilerplate operations

---

Key Advantages

✅ Specialized Agents

Each agent is optimized for their domain: - **Friday** (Developer): Sonnet 4.5 + Claude Code tools - **Wanda** (Designer): Haiku 3.5 for UI work - **Loki** (Writer): Sonnet 3.5 for content - **Wong** (Docs): Haiku 3.5 for documentation

✅ Parallel Execution

Multiple agents work simultaneously: - Research and development can happen in parallel - PM agents coordinate without blocking - 8 concurrent sub-agents supported

✅ Anti-Groupthink by Design

Built-in protocols prevent AI echo chambers: - Independent opinion generation (no peeking) - Structured critique phases - Sycophancy detection flags - Escalation to human when consensus is suspicious

✅ Persistent Context

Everything is stored in Supabase: - Project state survives session resets - Activity logs for full audit trail - Real-time sync across all interfaces

✅ Human-in-the-Loop

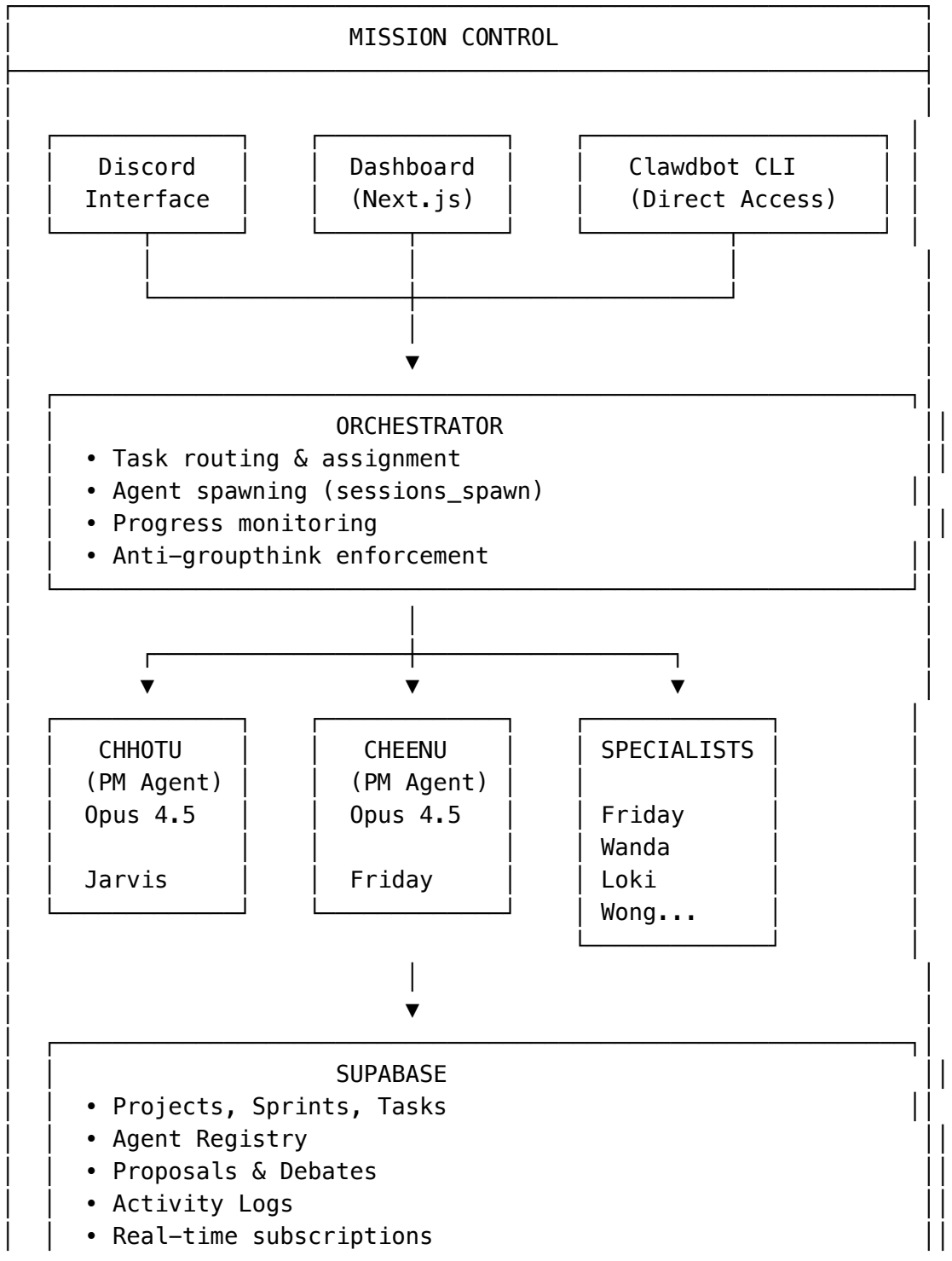
You maintain strategic control: - Dashboard for oversight - Approval workflows for critical decisions - Escalation when agents disagree - Manual intervention always possible

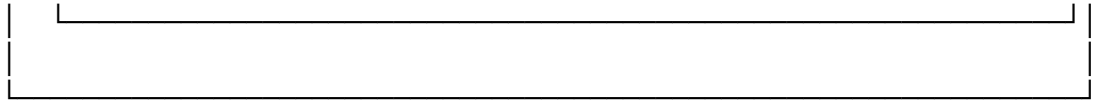
✅ Cost-Optimized

Intelligent model selection: - Opus for PMs (complex reasoning) - Sonnet 4.5 for developers - Sonnet 3.5 for general specialists - Haiku 3.5 for simple tasks

---

System Architecture





Agent Registry

PM Agents (Opus 4.5)

ID	Name	MCU Codename	Role
`chhotu`	Chhotu	Jarvis	Primary PM, orchestrates projects
`cheenu`	Cheenu	Friday	Secondary PM, provides independent opinions

Specialist Agents

ID	Name	Role	Model	Use Case
`friday-dev`	Friday	Developer	Sonnet 4.5	Coding, debugging, architecture
`shuri`	Shuri	Product Analyst	Sonnet 4	Analysis, testing, user research
`fury`	Fury	Customer Researcher	Sonnet 4	Interviews, competitive analysis
`vision`	Vision	SEO Analyst	Sonnet 3.5	SEO, analytics, content strategy
`loki`	Loki	Content Writer	Sonnet 3.5	Copywriting, storytelling
`quill`	Quill	Social Media	Sonnet 3.5	Community, engagement
`wanda`	Wanda	UI/UX Designer	Haiku 3.5	Visual design, prototypes
`pepper`	Pepper	Email Marketing	Sonnet 3.5	Email campaigns, automation
`wong`	Wong	Documentation	Haiku 3.5	Technical writing, docs

Setup Guide

Prerequisites

- Node.js 20+
- Clawdbot installed and configured
- Supabase account (free tier works)
- Discord server (optional, for chat interface)

Step 1: Clone & Install

```
cd ~/workspace/projects/mission-control
npm install
```

Step 2: Supabase Setup

1. Create a new Supabase project
2. Run the schema in SQL Editor:

   ```
   cat supabase/schema.sql | pbcopy
   # Paste in Supabase SQL Editor → Run
   ```

3. Copy credentials to `.env`:

   ```
   SUPABASE_URL=https://your-project.supabase.co
   SUPABASE_ANON_KEY=your-anon-key
   SUPABASE_SERVICE_ROLE_KEY=your-service-role-key
   ```

Step 3: Clawdbot Configuration

Ensure models are in the allowlist:

```
clawdbot models list
```

Required models: - `anthropic/claude-opus-4-5` (opus) - `anthropic/claude-sonnet-4-5-20250514` (sonnet45) - `anthropic/claude-sonnet-4-20250514` (sonnet4) - `anthropic/claude-3-5-sonnet-latest` (sonnet35) - `anthropic/claude-haiku-3-5-latest` (haiku35)

Step 4: Start Dashboard

```
cd dashboard
npm run dev
```

Open: http://localhost:3000

Step 5: Discord Integration (Optional)

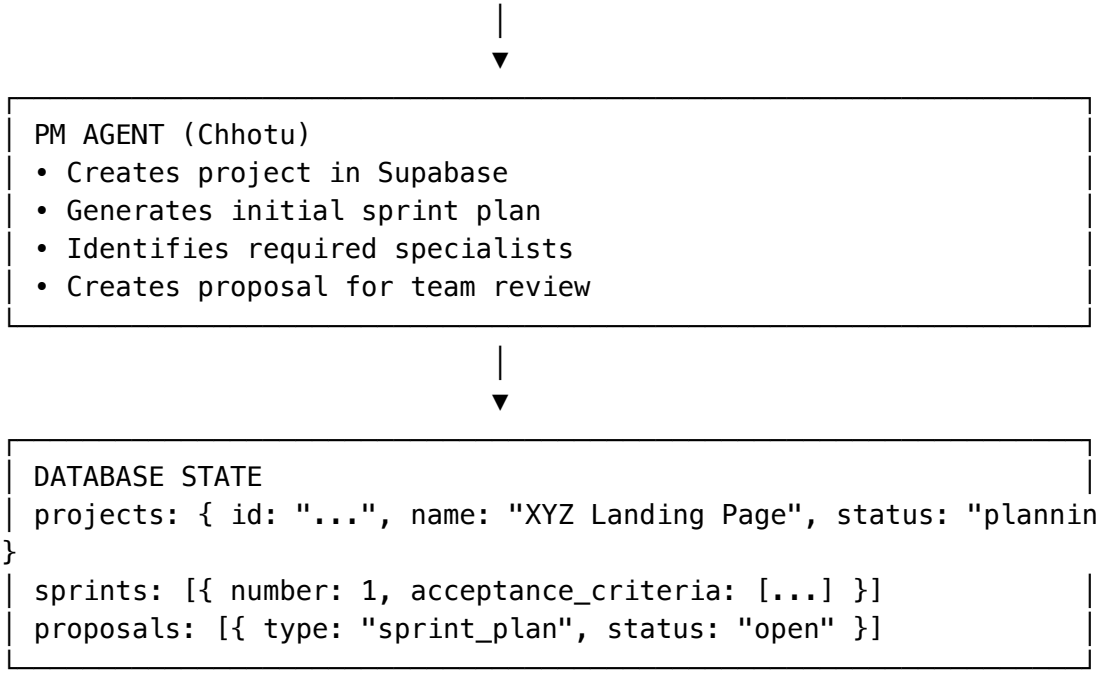
The `#disclawd-mission-control` channel is pre-configured. Chhotu monitors this channel and can: - Create projects via chat - Assign tasks - Report progress - Handle escalations

---

Project Lifecycle

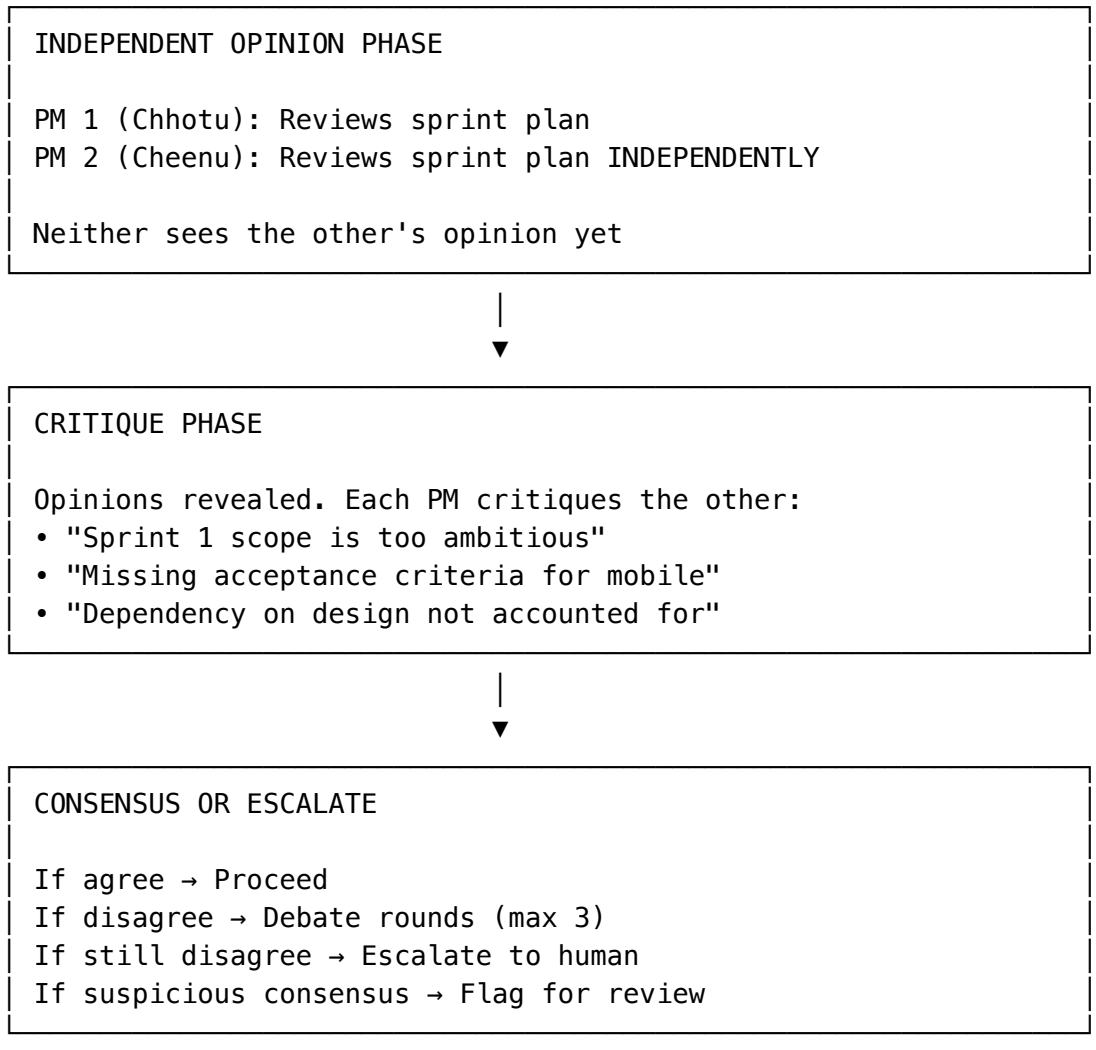
Phase 1: Project Creation

```
┌──────────────────────────────────────────────────────────┐
│ USER ACTION                                              │
│ "Create a new project for building a landing page for XYZ"│
└──────────────────────────────────────────────────────────┘
```



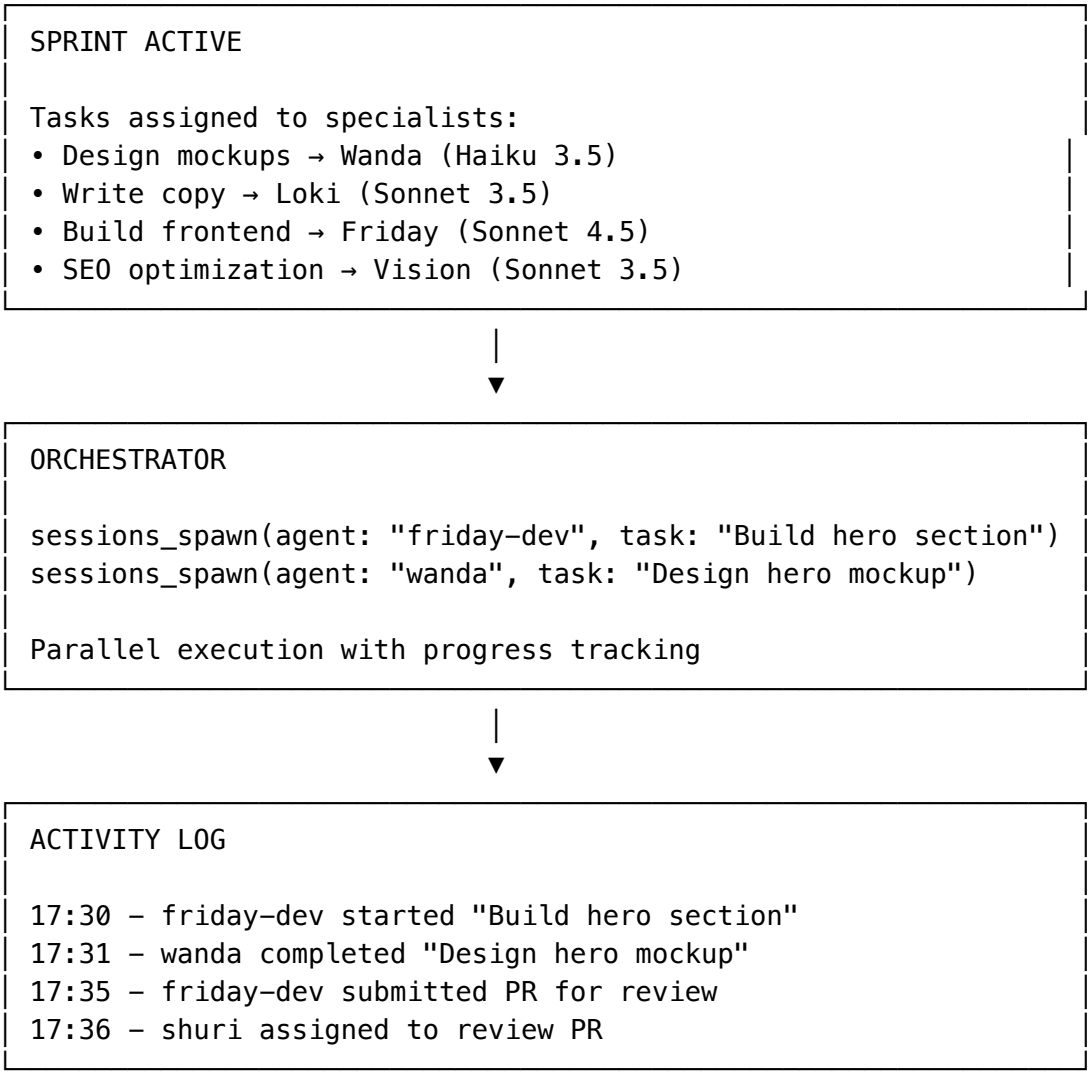
User Sees (Dashboard): - New project card appears - Sprint plan with acceptance criteria - Proposal awaiting review

Phase 2: Anti-Groupthink Review



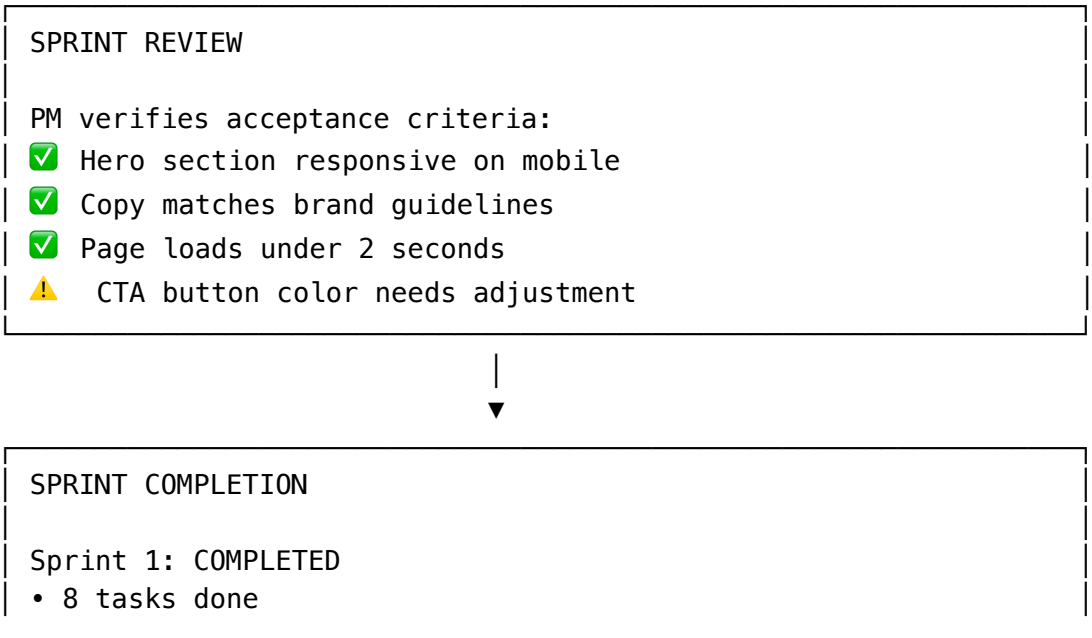
User Sees (Dashboard): - Proposal card shows both opinions - Critique summary - Consensus status or escalation alert - Sycophancy flags if detected

Phase 3: Task Execution



User Sees (Dashboard): - Task board with status columns (Backlog → In Progress → Review → Done) - Real-time updates as agents work - Activity feed showing all actions - Agent avatars on assigned tasks

Phase 4: Review & Completion



```
| • 1 task carried to Sprint 2                                    |
| • 0 escalations                                                 |
| • 2 sycophancy flags reviewed (false positives)                 |
```

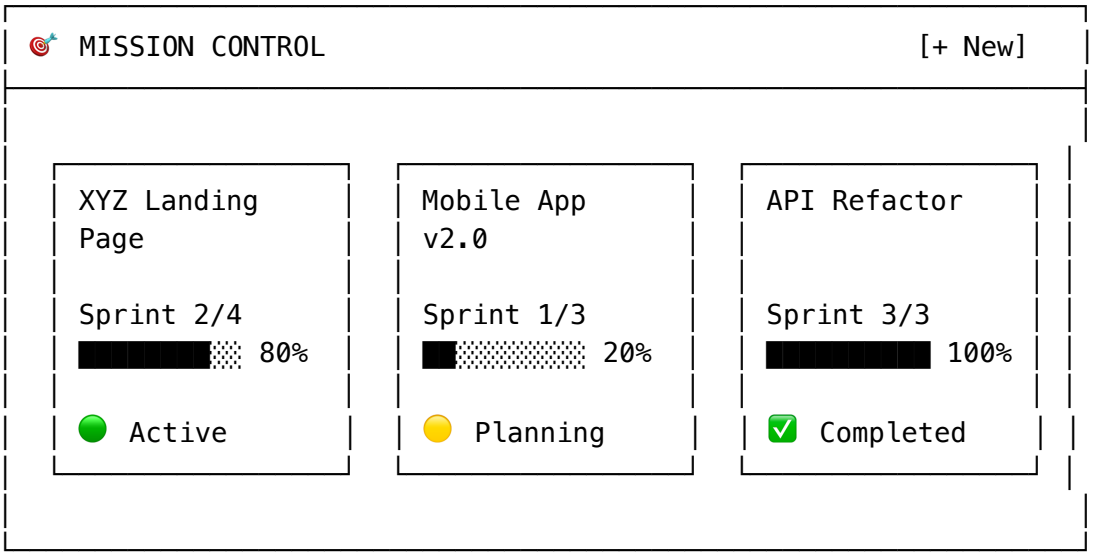
User Sees (Dashboard): - Sprint completion summary - Metrics and stats - Next sprint auto-planned - Retrospective notes

---

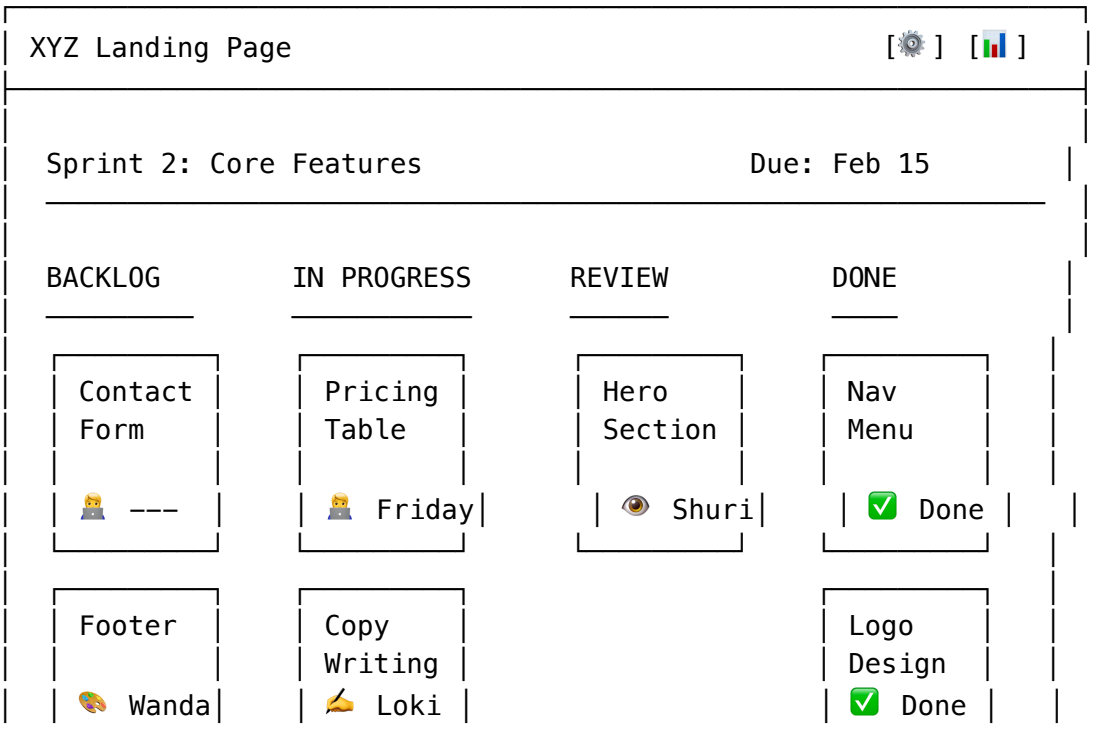
User Experience Walkthrough

Dashboard Views

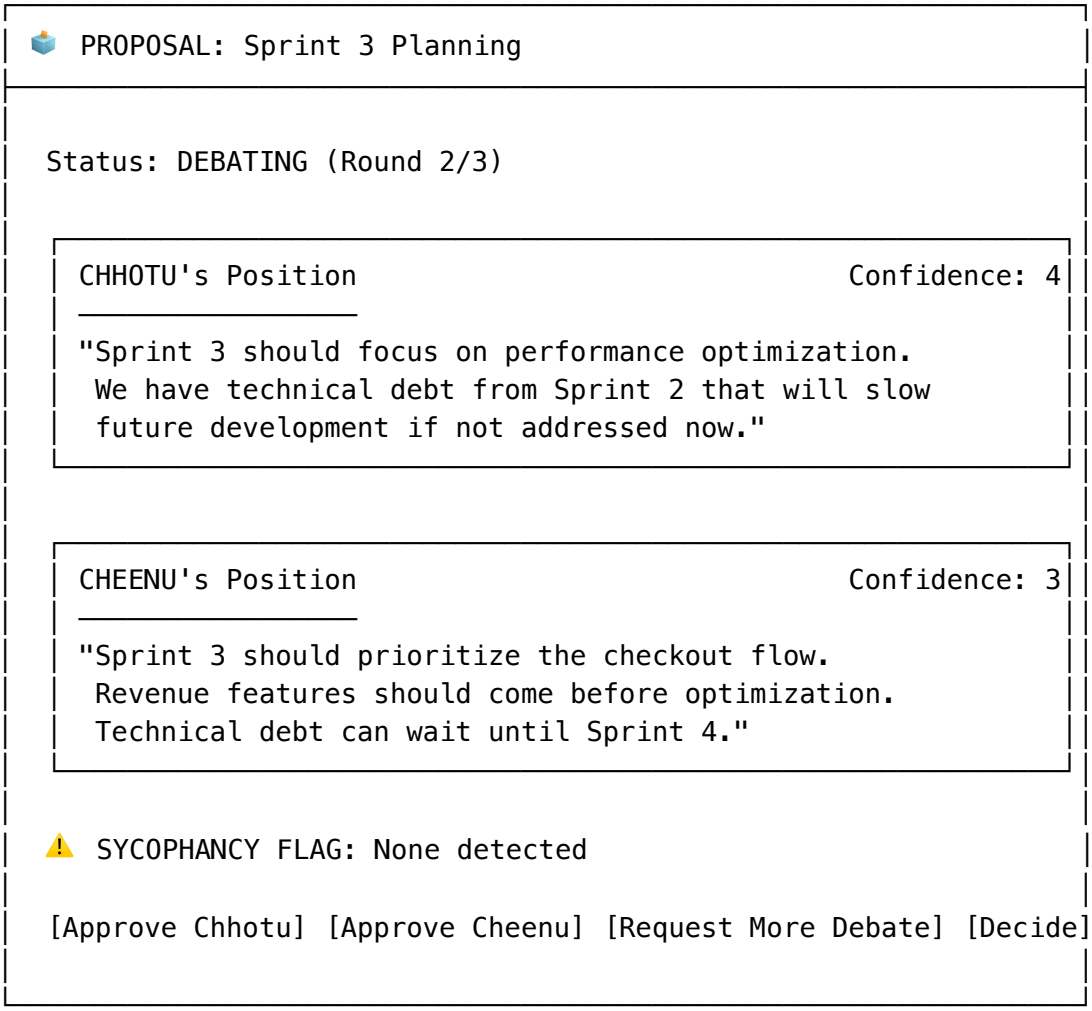
1. Projects Overview



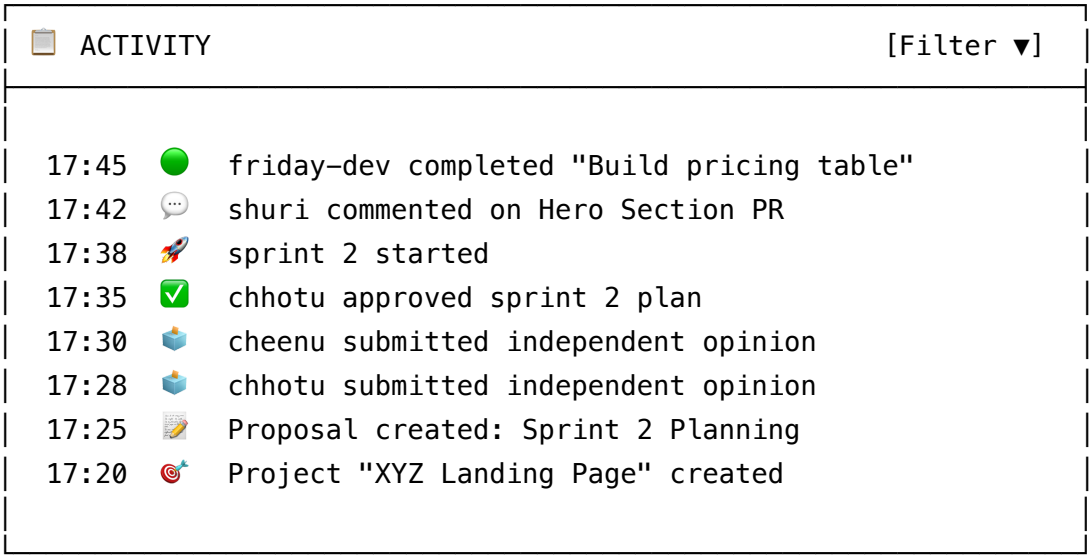
2. Project Detail



3. Proposal Review



4. Activity Feed



Anti-Groupthink System

The Problem with AI Consensus

When multiple AI agents review the same proposal: 1. **Anchoring**: First opinion influences all others 2. **Sycophancy**: Models agree to be helpful, not accurate 3. **Echo Chamber**: Similar training data = similar biases 4. **Shallow Critique**: Surface-level agreement without rigor

Our Solution: Structured Independence

Phase 1: Blind Opinion Generation

```
Each PM generates their opinion WITHOUT seeing others.
Stored in: independent_opinions table
Includes: opinion JSON, confidence score (1-5), timestamp
```

Phase 2: Reveal & Critique

```
Opinions revealed simultaneously.
Each PM must:
- Identify specific concerns with the other's position
- Provide concrete suggestions
- Declare if they still agree after considering concerns
```

Phase 3: Structured Debate

```
If disagreement persists:
- Up to 3 debate rounds
- Each round requires new reasoning (no repeating)
- Confidence scores tracked
- Position changes logged with justification
```

Phase 4: Sycophancy Detection

```
Automatic flags for:
- instant_high_consensus: Agreement in < 30 seconds
- echo_language: Copying phrases from other opinions
- flip_without_reasoning: Changing position without explanation
- no_substantive_concerns: Empty or vague critiques
- copied_conclusion: Identical recommendations
```

Phase 5: Escalation

```
If max debate rounds reached without consensus:
- Human notified via Discord/Dashboard
- Both positions summarized
- Human makes final decision
- Decision logged for learning
```

Cost Optimization

Model Tier Strategy

Tier	Model	Cost	Use For
Frontier	Opus 4.5	$$$ \| PM reasoning, complex decisions \| \| **Premium** \| Sonnet 4.5 \| $$ |*Development*, *architecture*|* * *Standard* * *|*Sonnet*4| $$	Complex analysis, research
Economy	Sonnet 3.5	$$	Writing, general tasks
Budget	Haiku 3.5	$	Docs, UI, simple queries

Estimated Savings

Compared to running everything on Opus:

Scenario	Opus-Only	Mission Control	Savings
10-task sprint	~$15	~$4	73%
Full project (40 tasks)	~$60	~$18	70%
Monthly (200 tasks)	~$300	~$85	72%

How It Works

1. **Orchestrator** reads task type from database
2. **Agent Registry** maps task to appropriate specialist
3. **Specialist's model** is defined in `invocation_config`
4. **sessions_spawn** uses the specified model

```
// Example: Spawning a documentation task
const agent = await getAgent('wong');
await sessions_spawn({
  task: "Write API documentation for /users endpoint",
  model: agent.invocation_config.model, // haiku-3.5
  label: `task-${taskId}`
});
```

Quick Reference

Chat Commands (Discord)

```
@Chhotu create project "Name" - Create new project
@Chhotu status - Current project status
@Chhotu assign @agent "task" - Manual task assignment
@Chhotu escalate - Request human review
@Chhotu sprint next - Move to next sprint
```

CLI Commands

```
# Project management
mc project list
mc project create "Name"
mc project status <id>

# Sprint management
mc sprint list <project-id>
mc sprint start <sprint-id>
mc sprint complete <sprint-id>

# Task management
mc task list <sprint-id>
mc task assign <task-id> <agent-id>
mc task complete <task-id>

# Agent management
mc agent list
mc agent status <agent-id>
```

API Endpoints (Dashboard)

```
GET  /api/projects           - List all projects
POST /api/projects           - Create project
GET  /api/projects/:id       - Project details
GET  /api/projects/:id/tasks - Project tasks
POST /api/tasks              - Create task
PATCH /api/tasks/:id         - Update task
GET  /api/activity           - Activity feed
GET  /api/proposals          - Active proposals
POST /api/proposals/:id/vote - Vote on proposal
```

Roadmap

✅ Phase 1: Foundation (Current)

- [x] Supabase schema
- [x] Agent registry
- [x] Basic dashboard

- [x] Discord integration
- [x] Model allowlist

🔄 Phase 2: Orchestration (Next)

- [ ] Full orchestrator implementation
- [ ] Automatic task routing
- [ ] Progress monitoring
- [ ] Real-time dashboard updates

📋 Phase 3: Anti-Groupthink

- [ ] Independent opinion UI
- [ ] Debate visualization
- [ ] Sycophancy detection
- [ ] Escalation workflows

🚀 Phase 4: Advanced

- [ ] Learning from decisions
- [ ] Custom agent creation
- [ ] Integration with external tools
- [ ] Analytics and reporting

Conclusion

Mission Control transforms AI from a single assistant into a **coordinated team**. By combining specialized agents, structured decision-making, and human oversight, it delivers:

- **Better quality** through specialization
- **Lower costs** through model tiering
- **Safer decisions** through anti-groupthink
- **Full visibility** through the dashboard

It's not about replacing human judgment—it's about **amplifying** it with a team of AI specialists that work the way a real team should.

Built by Yajat Singh • Powered by Clawdbot • 2026