Shadow AI Agent - Complete Capabilities Guide

This document provides a comprehensive overview of the Shadow AI agent's features, capabilities, and components to help you understand what you can develop, improve, or modify.

Table of Contents

Architecture Overview

Shadow AI is an **Electron-based** desktop AI coding assistant with:

Layer	Technology	Description
Frontend	React + TypeScript	Rich UI with chat, code editor, dashboards
Backend	Electron Main Process	Node.js runtime with full system access
AI Providers	OpenAI, Anthropic, Gemini, Mistral, Ollama	Multi-provider model support
Storage	electron-store, ChromaDB	Persistent settings + vector database

Directory Structure

```
src/main/ai/          # 1017+ AI modules
src/main/services/    # 18 core services
src/main/ipc/         # 52 IPC handler files
src/renderer/         # React UI components
```

Core Agent System

BaseAgent (`src/main/ai/agents/BaseAgent.ts`)

The foundation class all agents inherit from, providing:

- Task execution lifecycle

- Memory integration
- Event emission
- Handoff capabilities (new!)

Agent Types

Agent	Purpose
`coder`	Code generation & editing
`reviewer`	Code review & quality
`debugger`	Bug fixing & analysis
`architect`	System design
`designer`	UI/UX design
`devops`	Infrastructure & deployment
`tester`	Test generation & QA
`security`	Security analysis
`analyst`	Requirements analysis

AgentHandoff (`src/main/ai/agents/AgentHandoff.ts`)

Enables structured agent-to-agent communication:

- Request/Accept/Reject/Complete lifecycle
- Policy-based routing
- Concurrent handoff limits
- Statistics tracking

AI Model Integration

ModelManager (`src/main/ai/ModelManager.ts`)

Central hub for AI model management:

Feature	Description
Multi-Provider	OpenAI, Anthropic, Gemini, Mistral, Cohere, Ollama
Auto-Selection	Routes to best available model
Fallback Chain	Automatic failover on errors
Token Tracking	Usage monitoring and limits
Streaming	Real-time token streaming

EnhancedProviders (`src/main/ai/EnhancedProviders.ts`)

Extended provider implementations with:

- Rate limiting
- Retry logic
- Error handling
- Response caching

Tool System

ToolRegistry (`src/main/ai/tools/ToolRegistry.ts`)

Central registry for all agent tools:

Tool Category	Examples
File System	Read, Write, List, Search
Code Analysis	AST parsing, Semantic search
Git Operations	Commit, Branch, Merge
Terminal	Execute commands
Web	Browser automation, Fetch
Database	Query, Migrate

ToolChainExecutor (`src/main/ai/tools/ToolChainExecutor.ts`)

NEW: Declarative tool pipelines:

```
const chain = executor.createChain('analyze-fix', [
  { toolName: 'readFile', params: { path: './file.ts' } },
  { toolName: 'analyzeCode', inputMapping: { content: 'step_0' } },
  { toolName: 'applyFix', inputMapping: { issues: 'step_1' } }
]);
```

Key features:

- Input mapping between steps
- Middleware transformations
- Execution history
- Chain statistics

Agentic Workflows

AgenticLoop (`src/main/ai/agentic/AgenticLoop.ts`)

Autonomous task execution with reflection:

- Task decomposition
- Self-correction
- Goal tracking
- Progress monitoring

PlanActController (`src/main/ai/agentic/PlanActController.ts`)

Two-phase execution model:

1. **Plan Mode**: Generate execution plan
2. **Act Mode**: Execute with approval

GoalTracker (`src/main/ai/agentic/GoalTracker.ts`)

Progress monitoring and adjustment:

- Success criteria validation
- Milestone tracking
- Auto-adjustment on failures

Context & Memory

ContextCompressor (`src/main/ai/context/ContextCompressor.ts`)

NEW: Smart context window management:

- Token-based sliding window
- Priority decay over time
- Semantic grouping
- Checkpoint/restore

DeepContextEngine (`src/main/ai/context/DeepContextEngine.ts`)

Repository-aware context gathering:

- AST-based code analysis
- Symbol extraction
- Dependency mapping
- Query-relevant context

AgentMemory (`src/main/ai/memory/AgentMemory.ts`)

Long-term agent memory:

- Semantic memory storage
- Pattern recognition
- Preference learning
- Context recall

VectorStore (`src/main/ai/memory/VectorStore.ts`)

ChromaDB integration for embeddings:

- Code embeddings
- Semantic search
- Similarity matching

Streaming & Real-time

StreamingPipeline (`src/main/ai/streaming/StreamingPipeline.ts`)

NEW: Composable streaming data processing:

Transformer	Purpose
`TokenizerTransformer`	Split text into tokens
`JSONParserTransformer`	Parse streaming JSON
`ValidatorTransformer`	Validate against schema
`AccumulatorTransformer`	Collect final output

Features:

- SSE event streams
- Stage timeout handling
- Per-stage statistics

StreamingService (`src/main/services/StreamingService.ts`)

Token streaming management:

- Pipeline integration
- Real-time UI updates
- Error handling

Orchestration Layer

AgentOrchestrator (`src/main/ai/orchestration/AgentOrchestrator.ts`)

ENHANCED: Unified coordination layer:

Component	Integration
PlanActController	Execution planning
DeepContextEngine	Context gathering
GitHubAgent	Issue/PR automation
MCPClient	Model Context Protocol
CodeProvenance	Origin tracking
AgentMemory	Long-term memory
ToolChainExecutor	Tool pipelines (new)
AgentHandoff	Agent coordination (new)
ContextCompressor	Context management (new)
MCTSPlanner	Intelligent planning (new)

MCTSPlanner (`src/main/ai/reasoning/MCTSPlanner.ts`)

NEW: Monte Carlo Tree Search for planning:

- UCB1 selection strategy
- Configurable search depth
- Value backpropagation
- Best path extraction

Specialized Agents

VisionAgent (`src/main/ai/agents/VisionAgent.ts`)

Multimodal image understanding:

- Screenshot analysis
- UI detection
- Visual debugging

RedTeamAgent (`src/main/ai/agents/RedTeamAgent.ts`)

Security-focused adversarial testing:

- Vulnerability scanning
- Attack simulation
- Security recommendations

SelfEvolutionEngine (`src/main/ai/evolution/SelfEvolutionEngine.ts`)

Agent self-improvement:

- Performance analysis
- Capability enhancement
- Learning from feedback

Services

Service	File	Purpose
VoiceControlService	`VoiceControlService.ts`	Voice commands
WhisperService	`WhisperService.ts`	Speech-to-text
GitHubIntegration	`GitHubIntegration.ts`	GitHub API
PluginManager	`PluginManager.ts`	Plugin system
CloudSyncService	`CloudSyncService.ts`	Cloud sync
CollaborationService	`CollaborationService.ts`	Real-time collab
InlineCodeReviews	`InlineCodeReviews.ts`	Code review
FileHandler	`FileHandler.ts`	File operations
StreamingService	`StreamingService.ts`	Token streaming
FigmaService	`FigmaService.ts`	Figma integration
CanvaService	`CanvaService.ts`	Canva integration

UI Components

React Components (`src/renderer/components/`)

Component	Purpose
`AgenticDashboard`	Agent status & control
`CodeEditor`	Monaco-based editor
`PreviewArea`	Live preview
`ChatPanel`	AI conversation
`AIChat`	Streaming chat interface
`TaskQueuePanel`	Task management
`AutonomousMonitor`	Workflow monitoring
`TestingPanel`	Test runner UI

Shadow API (`src/renderer/shadowAPI.ts`)

IPC bridge for renderer access (540+ lines):

- Model management
- Tool execution
- Agent coordination
- Streaming
- **NEW**: toolChaining, handoff, context, planner, streaming APIs

React Hooks (`src/renderer/hooks/useAgentEnhancements.ts`)

NEW: 5 hooks for agent features:

- `useToolChaining()`
- `useAgentHandoff()`
- `useContextCompression()`
- `useMCTSPlanner()`
- `useStreamingPipeline()`

Areas for Improvement

High Priority

Area	Description	Files
E2E Testing	Playwright test suite	New
Performance	Response latency optimization	ModelManager, StreamingService
Error Handling	Better error recovery	All services

Medium Priority

Area	Description	Files
React 19 Migration	Update React deps	package.json, all components
Tailwind 4	CSS framework upgrade	package.json, styles
Plugin Marketplace	Plugin discovery UI	New component
Analytics Dashboard	Usage insights	New component

Feature Enhancements

Feature	Description	Impact
Multi-file Refactoring	Cross-file changes	High
Proactive Insights	Auto-suggestions	High
Code Coverage	Visual coverage	Medium
Browser Automation	Playwright integration	Medium
Voice Commands	Enhanced voice UI	Low

Code Quality

Area	Description
Unit Test Coverage	Expand test suite (currently 59 tests)
API Documentation	TSDoc for all modules
Type Safety	Reduce `any` usage
Code Splitting	Reduce bundle size

Quick Start for Development

Run the agent

```
cd /Users/shadyawayda/Desktop/antigravity/shadow-ai
npm run dev
```

Run tests

```
npm test
```

Build for production

```
npm run build
```

Key configuration files

- `package.json` - Dependencies
- `tsconfig.main.json` - Main process TypeScript
- `vite.config.ts` - Renderer build
- `.env.example` - Environment variables

Statistics

Metric	Count
AI Modules	1017+
IPC Handlers	52 files
Services	18
React Components	50+
Lines of AI Code	~200,000+
Unit Tests	59 passing