AI Code Editor - ASCII Project Structure Diagrams

🏗️ Overall Project Structure

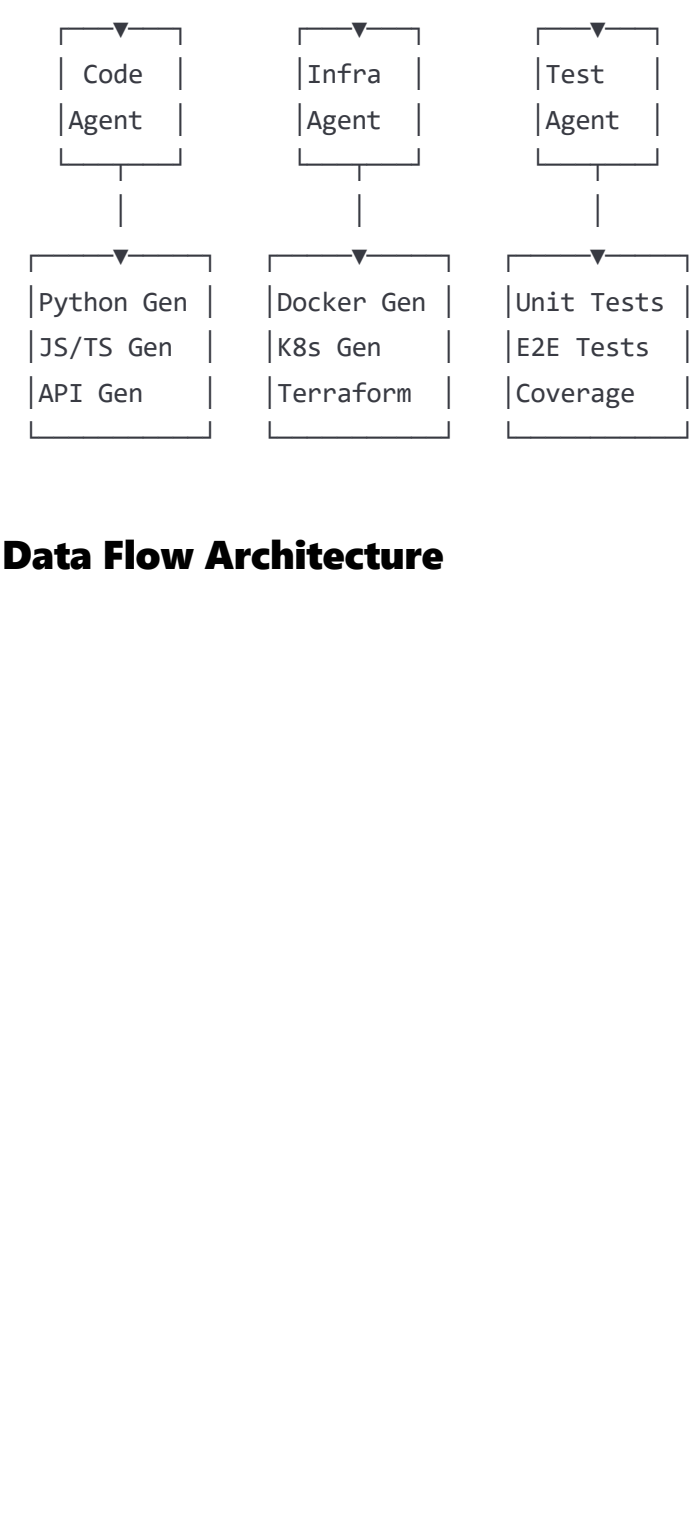
```
ai-code-editor/
├── 📁 .github/                    # GitHub workflows & templates
│   ├── workflows/                 # CI/CD automation
│   └── ISSUE_TEMPLATE/            # Issue templates
├── 📁 docs/                       # Documentation
│   ├── architecture/              # System design docs
│   ├── api/                       # API documentation
│   └── deployment/                # Deployment guides
├── 📁 frontend/                   # React Frontend Application
│   ├── public/                    # Static assets
│   ├── src/                       # Source code
│   │   ├── components/            # React components
│   │   ├── services/              # API clients
│   │   ├── hooks/                 # Custom React hooks
│   │   └── types/                 # TypeScript definitions
│   └── package.json               # Dependencies
├── 📁 backend/                    # ⭐ Python Backend (MAIN)
│   ├── models/                    # 🎯 Data models
│   ├── core/                      # 🧠 Core systems
│   ├── agents/                    # 🤖 AI agents
│   ├── api/                       # 🌐 REST endpoints
│   ├── config/                    # ⚙️ Configuration
│   └── requirements/              # 📦 Dependencies
├── 📁 infrastructure/             # Deployment configs
│   ├── docker/                    # Container configs
│   ├── kubernetes/                # K8s manifests
│   └── terraform/                 # IaC definitions
├── 📁 shared/                     # Cross-platform code
├── 📁 tools/                      # Development tools
├── 📁 tests/                      # Integration tests
└── 📄 Configuration Files
    ├── .env.example               # Environment template
    ├── docker-compose.yml         # Local development
    ├── Makefile                   # Build commands
    └── README.md                  # Getting started
```

🧠 Backend Architecture (Detailed)

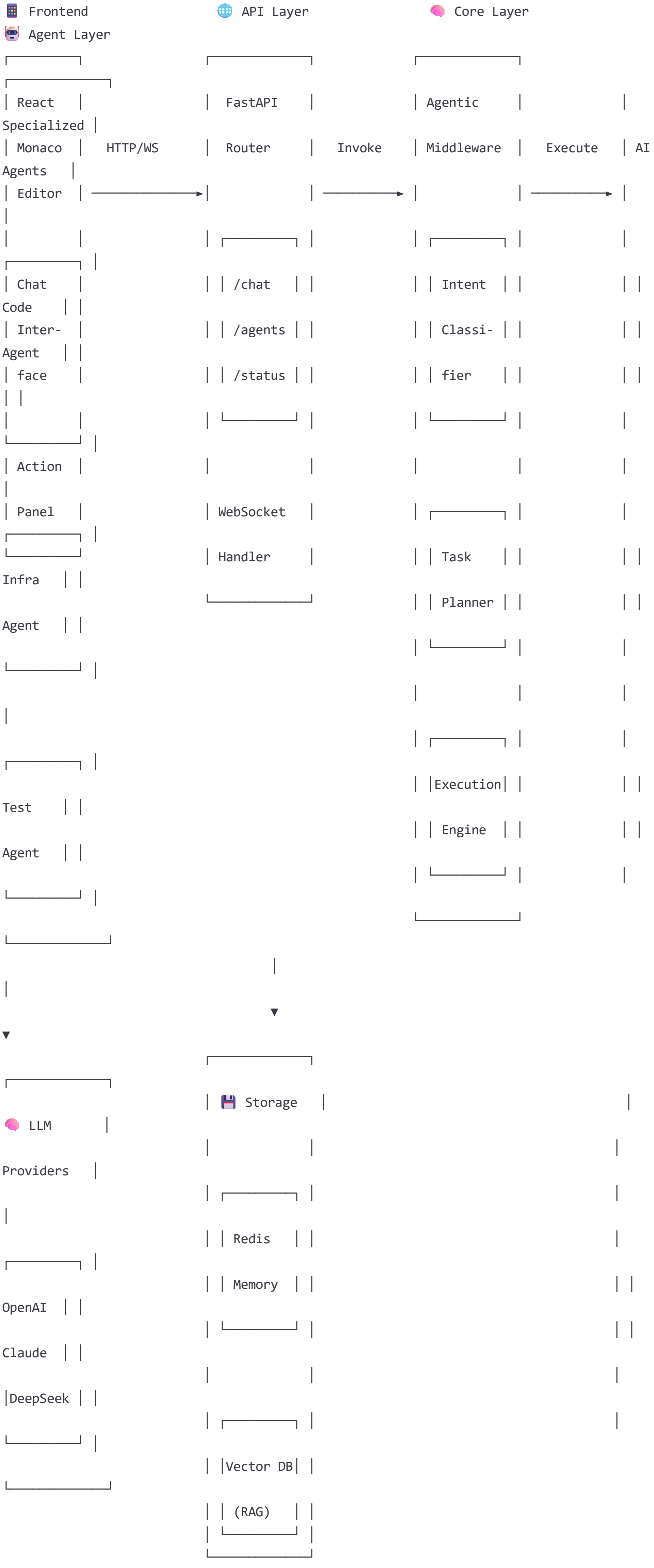
```
backend/
├── 📁 models/                    # 🎯 DATA FOUNDATION
│   ├── agent_models.py           # Core models (TaskRequest, TaskResult, etc.)
│   ├── api_models.py             # Request/response models
│   └── __init__.py               # Package exports
│
├── 📁 core/                      # 🧠 CORE ORCHESTRATION
│   ├── orchestration/            # Main middleware logic
│   │   ├── middleware.py         # ⭐ AgenticMiddleware (MAIN CLASS)
│   │   ├── intent_classifier.py  # User intent detection
│   │   ├── task_planner.py       # Complex task planning
│   │   ├── execution_engine.py   # Task execution
│   │   ├── agent_registry.py     # Agent management
│   │   └── coordinator.py        # Multi-agent coordination
│   ├── memory/                   # Context & session management
│   │   ├── context_manager.py    # Session context
│   │   ├── conversation_history.py # Chat history
│   │   ├── project_memory.py     # Project-specific data
│   │   └── vector_store.py       # RAG integration
│   └── llm/                      # LLM provider abstraction
│       ├── gateway.py            # Multi-provider gateway
│       ├── rate_limiter.py       # API rate limiting
│       ├── cost_tracker.py       # Usage monitoring
│       └── providers/            # LLM implementations
│           ├── openai_provider.py
│           ├── claude_provider.py
│           └── deepseek_provider.py
│
├── 📁 agents/                    # 🤖 SPECIALIZED AI AGENTS
│   ├── base/                     # Base classes
│   │   ├── agent.py              # BaseAgent abstract class
│   │   ├── llm_agent.py          # LLM-powered base
│   │   └── protocols.py          # Communication protocols
│   ├── code/                     # Code generation
│   │   ├── code_agent.py         # Main code agent
│   │   ├── generators/           # Language-specific generators
│   │   ├── parsers/              # Code analysis
│   │   └── validators/           # Code validation
│   ├── infrastructure/           # Infrastructure as code
│   │   ├── infra_agent.py        # Main infra agent
│   │   ├── docker_generator.py   # Docker configs
│   │   ├── kubernetes_generator.py # K8s manifests
│   │   └── terraform_generator.py # Terraform configs
│   ├── testing/                  # Test generation
│   │   ├── test_agent.py         # Main test agent
│   │   ├── generators/           # Test generators
│   │   └── runners/              # Test execution
│   ├── devops/                   # CI/CD automation
│   │   ├── devops_agent.py       # Main DevOps agent
│   │   └── ci_generators/        # Pipeline generators
│   ├── documentation/            # Doc generation
│   ├── security/                 # Security scanning
│   └── factory.py                # Agent factory
│
├── 📁 api/                       # 🌐 REST API LAYER
│   └── v1/                       # API version 1
│       ├── middleware_api.py     # Main API routes
│       ├── websocket.py          # Real-time communication
│       ├── chat.py               # Chat endpoints
│       ├── agents.py             # Agent management
│       └── health.py             # Health checks
│
├── 📁 services/                  # 💼 BUSINESS SERVICES
│   ├── file_manager.py           # File operations
│   ├── git_service.py            # Git integration
│   ├── project_service.py        # Project management
│   └── workspace_service.py      # Workspace handling
│
├── 📁 config/                    # ⚙️ CONFIGURATION
│   ├── settings.py               # Environment-based config
│   └── __init__.py               # Config exports
```

```
│
├── 📁 app/                          # 🚀 APPLICATION ENTRY
│   ├── main.py                     # FastAPI application
│   ├── dependencies.py             # Dependency injection
│   └── __init__.py
│
└── 📁 tests/                        # 🧪 TESTING
    ├── unit/                       # Unit tests
    ├── integration/                # Integration tests
    └── e2e/                        # End-to-end tests
```

🤖 Agent Architecture

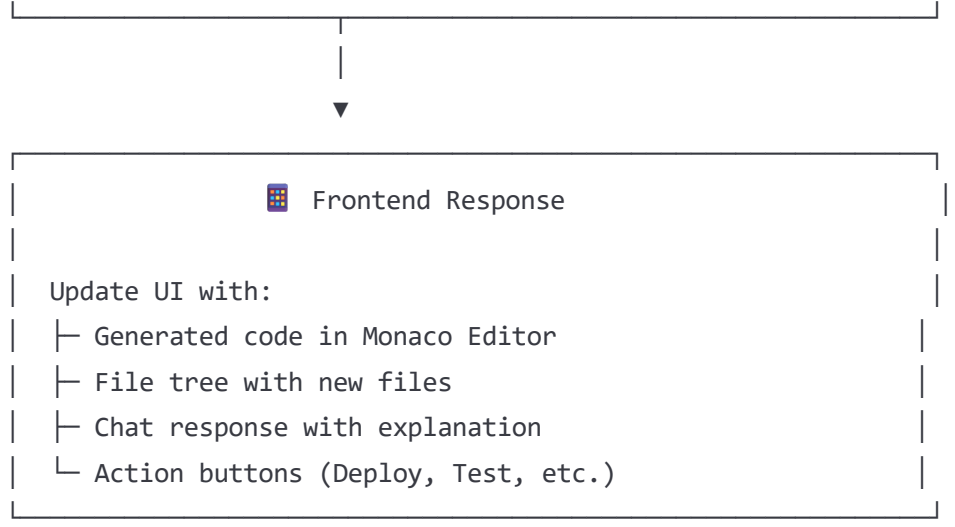


🔄 Data Flow Architecture


Frontend
API Layer
Core Layer
Agent Layer
React
Monaco
Editor
Chat
Inter-
face
Action
Panel
Handler
FastAPI
Router
/chat
/agents
/status
WebSocket
Agentic
Middleware
Intent
Classi-
fier
Task
Planner
Execution
Engine
HTTP/WS
Invoke
Execute
AI
Specialized
Agents
Code
Agent
Infra
Agent
Test
Agent
LLM
Providers
OpenAI
Claude
DeepSeek
Storage
Redis
Memory
Vector DB
(RAG)

📋 Request Processing Flow

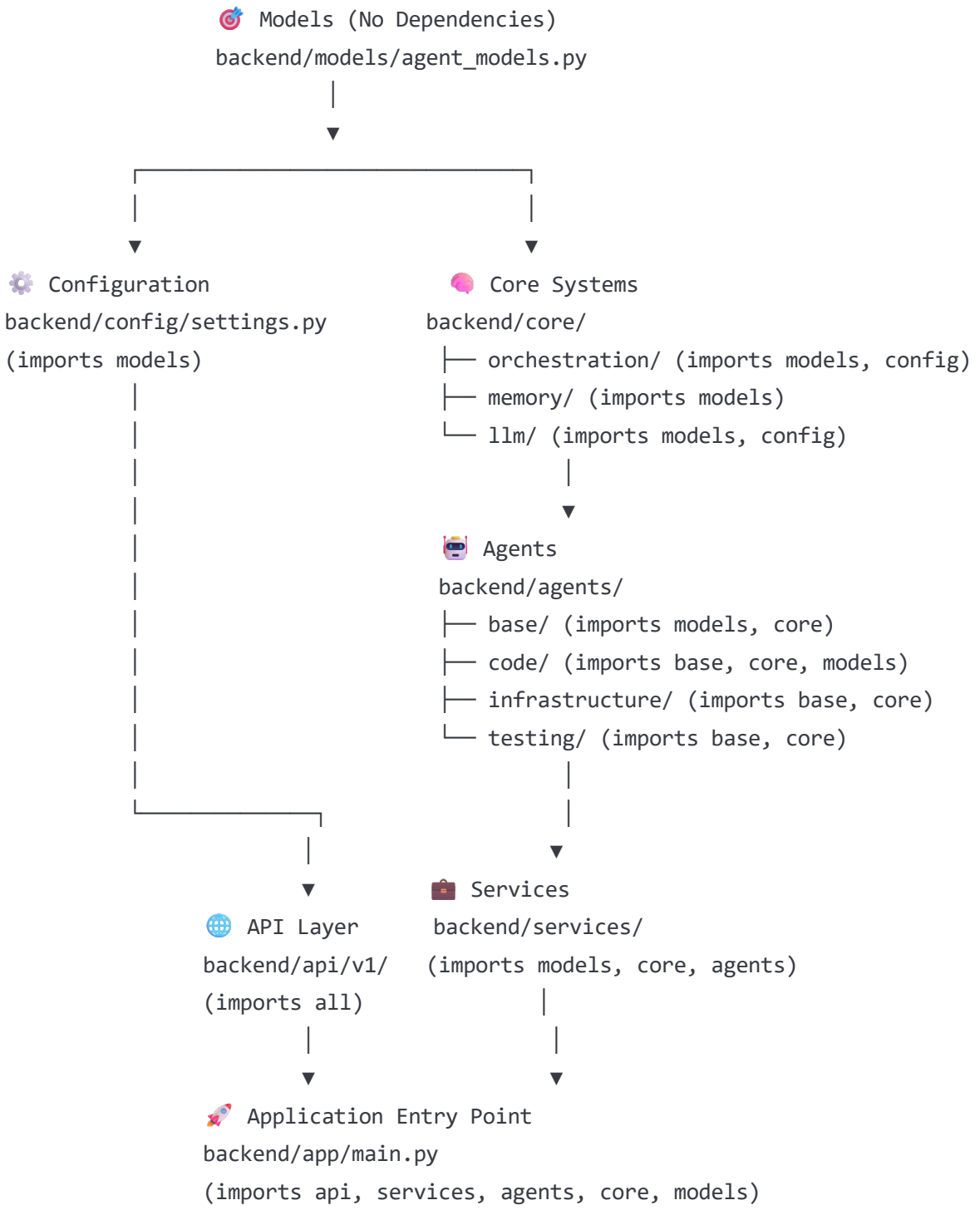
```
1. User Input
   │
   ▼
┌──────────────────────────────────────────────────────────────┐
│  "Create a FastAPI app with JWT auth and deploy to AWS"       │
└──────────────────────────────────────────────────────────────┘
                      │
                      ▼
┌──────────────────────────────────────────────────────────────┐
│                  🌐 API Layer                                  │
│  POST /api/v1/chat                                            │
│  ├─ Validate ChatMessage                                      │
│  ├─ Extract session context                                   │
│  └─ Forward to AgenticMiddleware                              │
└──────────────────────────────────────────────────────────────┘
                      │
                      ▼
┌──────────────────────────────────────────────────────────────┐
│               🧠 Core Orchestration                            │
│                                                               │
│  1. Intent Classification                                     │
│     ├─ "fastapi" → CODE_GENERATION                           │
│     ├─ "jwt auth" → SECURITY + CODE                          │
│     └─ "deploy aws" → INFRASTRUCTURE_SETUP                   │
│                                                               │
│  2. Task Planning                                             │
│     ├─ Subtask 1: Generate FastAPI code                      │
│     ├─ Subtask 2: Add JWT authentication                     │
│     ├─ Subtask 3: Create AWS infrastructure                  │
│     └─ Subtask 4: Generate deployment config                 │
│                                                               │
│  3. Execution Orchestration                                   │
│     └─ Route to appropriate agents                           │
└──────────────────────────────────────────────────────────────┘
                      │
                      ▼
┌──────────────────────────────────────────────────────────────┐
│                 🤖 Agent Execution                             │
│                                                               │
│  Sequential Execution:                                        │
│                                                               │
│  ┌─────────────┐    ┌─────────────┐    ┌─────────────┐       │
│  │ Code Agent  │    │Security     │    │Infra Agent  │       │
│  │             │───▶│Agent        │───▶│             │       │
│  │• FastAPI    │    │• JWT impl   │    │• Terraform  │       │
│  │• Routes     │    │• Auth       │    │• Docker     │       │
│  │• Models     │    │• Middleware │    │• K8s        │       │
│  └─────────────┘    └─────────────┘    └─────────────┘       │
│                                                               │
│  Each agent:                                                  │
│  ├─ Receives TaskRequest                                      │
│  ├─ Calls LLM with specialized prompt                        │
│  ├─ Validates generated output                                │
│  └─ Returns TaskResult                                        │
└──────────────────────────────────────────────────────────────┘
                      │
                      ▼
┌──────────────────────────────────────────────────────────────┐
│                  📦 Result Aggregation                         │
│                                                               │
│  Combine all agent results:                                   │
│  ├─ Code files (main.py, auth.py, models.py)                │
│  ├─ Infrastructure (Dockerfile, terraform/, k8s/)            │
│  ├─ Documentation (README.md, API docs)                      │
│  └─ Deployment instructions                                   │
│                                                               │
│  Format as ChatResponse:                                      │
│  ├─ Status: "completed"                                       │
│  ├─ Files: {...}                                             │
│  ├─ Explanation: "Created FastAPI app with..."               │
│  └─ Next steps: "Run `docker build` to..."                   │
```



🗂️ File Organization by Purpose

```
📊 Data Models (Foundation)
├── backend/models/agent_models.py    # Core business models
├── backend/models/api_models.py      # API request/response
└── shared/types/                     # Cross-platform types

🧠 Core Logic (Brain)
├── backend/core/orchestration/       # Main middleware logic
├── backend/core/memory/              # Context & history
└── backend/core/llm/                 # LLM abstraction

🤖 AI Agents (Workers)
├── backend/agents/code/              # Code generation
├── backend/agents/infrastructure/    # Infrastructure as code
├── backend/agents/testing/           # Test automation
├── backend/agents/devops/            # CI/CD pipelines
├── backend/agents/documentation/     # Documentation
└── backend/agents/security/          # Security scanning

🌐 API Layer (Interface)
├── backend/api/v1/                   # REST endpoints
└── backend/app/main.py               # FastAPI application

💼 Business Services (Utilities)
├── backend/services/                 # File, Git, Project services
└── backend/utils/                    # Helper functions

⚙️ Configuration (Settings)
├── backend/config/                   # Environment config
├── .env.example                      # Environment template
└── backend/requirements/             # Dependencies

🎨 Frontend (User Interface)
├── frontend/src/components/          # React components
├── frontend/src/services/            # API clients
└── frontend/src/hooks/               # Custom hooks

🏗️ Infrastructure (Deployment)
├── infrastructure/docker/            # Container configs
├── infrastructure/kubernetes/        # K8s manifests
└── infrastructure/terraform/         # Cloud infrastructure

🧪 Testing (Quality)
├── backend/tests/                    # Backend tests
├── tests/                            # Integration tests
└── frontend/src/__tests__/           # Frontend tests
```

🎯 Import Dependency Graph



This ASCII structure shows how the AI Code Editor is organized as a **scalable, modular system** where each component has a clear responsibility and the data flows logically from user input through the agentic middleware to specialized AI agents and back to the user! 🚀