AI Collaboration Hub - Project Plan

Created: March 18, 2025

Project Vision

AI Collaboration Hub aims to create an integrated development environment that bridges multiple AI capabilities with human-directed development processes. The system will optimize the use of AI resources by intelligently routing tasks between web-based AI services and local AI models, while maintaining session continuity and context.

This tool will serve as a central hub for AI-assisted development, allowing developers to leverage the right AI tool for each task while maintaining a coherent workflow and development history.

Objectives

1. **Resource Optimization**
    - Reduce costs associated with web-based AI services
    - Minimize unnecessary consumption of online computational resources
    - Improve response times by using local models for appropriate tasks
2. **Workflow Integration**
    - Provide a seamless interface for communicating with multiple AI systems
    - Maintain context across development sessions
    - Integrate with existing development tools and workflows
3. **User Control**
    - Allow users to decide which AI model handles specific types of tasks
    - Provide transparency in how tasks are routed
    - Enable customization of prompts and interaction patterns
4. **Development Continuity**
    - Ensure session persistence across crashes or interruptions
    - Maintain development history and context
    - Enable collaboration between AI systems on complex tasks

Scope

Phase 1: Standalone Application (Core Hub)

1. **User Interface Components**

   - Conversation panel for AI interactions
   - Code editor/viewer for sharing code context
   - Settings panel for AI routing preferences
   - Session management interface
2. **AI Integration**
   - Connection to web-based Claude (with appropriate ToS compliance)
   - Integration with at least one local AI model (e.g., Ollama)
   - Task routing logic based on content type and user preferences
3. **State Management**
   - Session persistence using AIDEV-State principles
   - Conversation history management
   - Context tracking between sessions
4. **Basic Workflow Support**
   - Copy/paste support for code and text
   - File import/export capabilities
   - Session summary generation

Phase 2: Integration Capabilities

1. **External Tool Integration**
   - File system watching for project changes
   - Git integration for version tracking
   - AIDEV-Validate integration for code standards
2. **Advanced AI Routing**
   - Machine learning for optimal task routing
   - Custom prompt templates for specific tasks
   - Performance tracking for AI model efficiency
3. **Collaboration Features**
   - Shareable AI sessions
   - Team-based AI preference settings
   - Multi-user session support

Phase 3: Editor Integration (Optional)

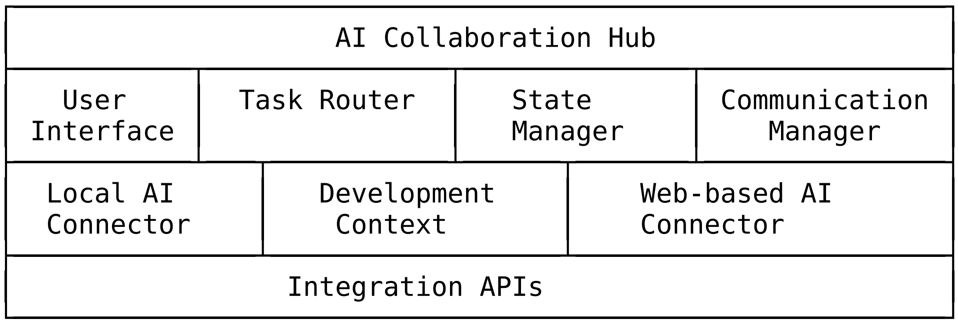
1. **VS Code Extension**
   - Integration with VS Code's interface
   - Code action menu for direct AI interaction
   - Inline suggestions from AI models
2. **Other Editor Support**
   - Similar integrations for other popular editors
   - API for custom editor integration

Architecture

Core Components

1. **Communication Manager**
    - Handles connections to web-based AI services
    - Manages authentication and session tokens
    - Implements rate limiting and usage tracking
2. **Local AI Connector**
    - Interfaces with locally installed AI models
    - Manages model loading and resource allocation
    - Standardizes interaction across different models
3. **Task Router**
    - Analyzes content to determine appropriate AI
    - Implements routing rules based on user preferences
    - Tracks performance metrics for optimization
4. **State Manager**
    - Persists conversation and session state
    - Recovers from crashes or interruptions
    - Manages context windows across AI systems
5. **User Interface**
    - Provides intuitive access to AI capabilities
    - Displays conversation history and context
    - Enables configuration of AI routing preferences

System Diagram



Technical Considerations

AI Service Integration

1. **Claude Integration**
    - Explore official API options with proper attribution
    - Investigate Terms of Service implications
    - Implement usage monitoring and cost controls

2. **Local AI Models**
    - Support for Ollama-based models
    - Compatibility with other local inference engines
    - Resource management to prevent system overload

State Management

1. **Persistence Mechanisms**
    - SQLite database for state storage
    - JSON export/import capabilities
    - Backup and recovery functionality
2. **Context Management**
    - Tracking conversation threads
    - Maintaining code context
    - Managing token limits across models

User Experience

1. **Interface Design**
    - Clean, intuitive conversation interface
    - Clear indication of active AI model
    - Transparent settings for AI routing
2. **Performance Considerations**
    - Minimize latency for local operations
    - Efficient management of background processes
    - Responsive UI during AI processing

Implementation Approach

Development Methodology

1. **Iterative Development**
    - Start with minimal viable product
    - Short development cycles with regular testing
    - Early user feedback incorporation
2. **Modular Architecture**
    - Separate concerns between components
    - Well-defined interfaces between modules
    - Plugin system for extensibility
3. **Open Source Strategy**
    - Clear documentation and contribution guidelines
    - Permissive licensing model
    - Active community engagement

Technology Stack

1. **Application Framework**
    - Electron for cross-platform desktop application
    - React for user interface components
    - Node.js for backend services
2. **Storage and State**
    - SQLite for persistent storage
    - Redux for state management
    - File system integration for project context
3. **AI Integration**
    - REST API clients for web services
    - Ollama integration for local models
    - Stream processing for real-time responses

Project Timeline

Phase 1 (3-4 months)

1. **Month 1: Core Framework**
    - Set up application architecture
    - Implement basic UI components
    - Create state management system
2. **Month 2: AI Integration**
    - Implement Claude connection
    - Add local AI integration
    - Develop basic task routing
3. **Month 3: User Experience**
    - Refine interface design
    - Implement session persistence
    - Add configuration capabilities
4. **Month 4: Testing & Release**
    - User testing and feedback
    - Bug fixing and optimization
    - Initial public release

Phase 2 (3-4 months)

1. **Month 5-6: External Integration**
    - Implement file system watching
    - Add Git integration
    - Create AIDEV-Validate connection
2. **Month 7-8: Advanced Features**

    - Develop ML-based task routing
    - Implement custom prompt templates
    - Add performance tracking

Phase 3 (Ongoing)

- Editor integration research
- Community-driven development
- Continuous improvement based on user feedback

Success Metrics

1. **User Adoption**
    - Number of active users
    - Installation retention rate
    - Community engagement
2. **Performance Metrics**
    - Response time improvements
    - Cost savings on web AI services
    - Session recovery success rate
3. **Development Metrics**
    - Code quality improvements
    - Development time reduction
    - User satisfaction ratings

Potential Challenges

1. **Technical Challenges**
    - Ensuring compatibility across different AI models
    - Managing state persistence reliably
    - Performance optimization for real-time interaction
2. **External Dependencies**
    - Changes to AI service APIs or terms
    - Local AI model availability and compatibility
    - Integration with evolving development tools
3. **Community Adoption**
    - Building awareness of the tool
    - Managing user expectations
    - Balancing features with simplicity

Conclusion

The AI Collaboration Hub represents a significant opportunity to improve the AI-assisted development process. By intelligently combining web-based and local

AI capabilities, this tool can provide developers with the best of both worlds while maintaining workflow continuity and resource efficiency.

The phased approach allows for incremental development and validation, with each phase building on the proven value of the previous one. Starting with a standalone application provides the quickest path to demonstrating the core value proposition, while keeping future integration options open.

This project has the potential to fundamentally change how developers interact with AI assistants, making the process more efficient, cost-effective, and tailored to individual workflows.

"Code is not merely functional—it is a visual medium that developers interact with for extended periods. The choices made in these standards prioritize the axis of symmetry, character distinction, readability at scale, and visual hierarchy."

— Herbert J. Bowers