Project Plan - Research Agent Development

AI-Powered Research Agent with Dynamic Model Selection

Document Information

- **Version**: 1.0
- **Date**: June 1, 2025
- **Author**: [Your Name]
- **Project Code**: RA-2025-001-PLAN

---

1. Executive Summary

1.1 Project Overview

Development of an intelligent research agent system with dynamic LLM model selection, multi-agent orchestration, and RAG capabilities. The system will intelligently route tasks to optimal models based on complexity, cost, and performance requirements.

1.2 Project Timeline

- **Total Duration**: 12 weeks
- **Version 1.0 (Local)**: Weeks 1-8
- **Version 2.0 (Cloud)**: Weeks 9-12
- **Documentation & Demo**: Ongoing

---

2. Work Breakdown Structure

2.1 Phase 1: Foundation & Architecture (Weeks 1-2)

Week 1: Environment Setup & Core Infrastructure

Sprint Goal: Establish development environment and basic architecture

Tasks:

- [ ] **ENV-001**: Setup development environment (Cursor IDE, Python 3.9+, Docker)
- [ ] **ENV-002**: Configure OpenRouter API access and test model connectivity
- [ ] **ENV-003**: Setup Milvus vector database with Docker Compose
- [ ] **ENV-004**: Initialize project structure with proper Git workflow

- [ ] **ENV-005**: Create basic FastAPI application skeleton
- [ ] **ENV-006**: Setup logging and configuration management

Deliverables:

- Working development environment
- Basic API structure with health checks
- Database connectivity established

Week 2: Model Selection Conductor Agent

Sprint Goal: Build intelligent model routing system

Tasks:

- [ ] **MSC-001**: Design model selection decision matrix
- [ ] **MSC-002**: Implement Conductor Agent with task complexity analysis
- [ ] **MSC-003**: Create cost optimization algorithms
- [ ] **MSC-004**: Build model performance tracking system
- [ ] **MSC-005**: Implement fallback mechanisms for model failures
- [ ] **MSC-006**: Add model usage analytics and reporting

Deliverables:

- Conductor Agent with intelligent routing
- Model selection documentation
- Cost tracking dashboard

2.2 Phase 2: Core Agent Development (Weeks 3-5)

Week 3: Document Processing & RAG Foundation

Sprint Goal: Implement document ingestion and vector storage

Tasks:

- [ ] **DOC-001**: Build document upload and validation system
- [ ] **DOC-002**: Implement PDF/DOCX parsing with PyMuPDF4LLM
- [ ] **DOC-003**: Create intelligent chunking strategies
- [ ] **DOC-004**: Setup embedding generation with OpenRouter
- [ ] **DOC-005**: Implement Milvus integration for vector storage
- [ ] **DOC-006**: Build document metadata management

Deliverables:

- Document processing pipeline
- Vector storage system
- RAG query capabilities

Week 4: Web Research & Content Extraction

Sprint Goal: Build web research capabilities

Tasks:

- [ ] **WEB-001**: Integrate Brave Search API with rate limiting
- [ ] **WEB-002**: Implement Jina Reader for clean content extraction
- [ ] **WEB-003**: Build source credibility scoring system
- [ ] **WEB-004**: Create content deduplication logic
- [ ] **WEB-005**: Implement parallel search processing
- [ ] **WEB-006**: Add search result ranking and filtering

Deliverables:

- Web research agent
- Content extraction pipeline
- Source validation system

Week 5: CrewAI Integration & Agent Orchestration

Sprint Goal: Implement multi-agent coordination

Tasks:

- [ ] **CREW-001**: Setup CrewAI framework integration
- [ ] **CREW-002**: Define agent roles and responsibilities
- [ ] **CREW-003**: Implement agent communication protocols
- [ ] **CREW-004**: Build task delegation and coordination
- [ ] **CREW-005**: Create agent performance monitoring
- [ ] **CREW-006**: Implement error handling and recovery

Deliverables:

- Multi-agent system
- Agent coordination workflows

- Performance monitoring

2.3 Phase 3: Analysis & Report Generation (Weeks 6-7)

Week 6: Analysis & Synthesis Engine

Sprint Goal: Build intelligent analysis capabilities

Tasks:

- [ ] **ANA-001**: Implement cross-source information synthesis
- [ ] **ANA-002**: Build pattern recognition and insight generation
- [ ] **ANA-003**: Create source conflict resolution logic
- [ ] **ANA-004**: Implement confidence scoring algorithms
- [ ] **ANA-005**: Build gap analysis for incomplete research
- [ ] **ANA-006**: Add analytical reasoning and recommendations

Deliverables:

- Analysis engine
- Synthesis algorithms
- Insight generation system

Week 7: Report Generation & Formatting

Sprint Goal: Create professional report output

Tasks:

- [ ] **REP-001**: Build report template system
- [ ] **REP-002**: Implement citation management and formatting
- [ ] **REP-003**: Create PDF generation with WeasyPrint
- [ ] **REP-004**: Build executive summary generation
- [ ] **REP-005**: Implement multiple output formats (MD, PDF, DOCX)
- [ ] **REP-006**: Add report quality validation

Deliverables:

- Report generation system
- Multiple output formats
- Professional templates

2.4 Phase 4: Integration & Testing (Week 8)

Week 8: System Integration & Version 1.0 Completion

Sprint Goal: Complete end-to-end integration and testing

Tasks:

- [ ] **INT-001**: End-to-end workflow integration testing
- [ ] **INT-002**: Performance optimization and tuning
- [ ] **INT-003**: Error handling and recovery testing
- [ ] **INT-004**: User interface completion (Streamlit)
- [ ] **INT-005**: Documentation and user guides
- [ ] **INT-006**: Demo preparation and showcase development

Deliverables:

- Complete Version 1.0 system
- User documentation
- Demo materials

2.5 Phase 5: Cloud Deployment (Weeks 9-12)

Week 9: Containerization & Microservices

Sprint Goal: Prepare for cloud deployment

Tasks:

- [ ] **CON-001**: Dockerize all application components
- [ ] **CON-002**: Create Kubernetes manifests
- [ ] **CON-003**: Implement service mesh architecture
- [ ] **CON-004**: Setup container registry and CI/CD
- [ ] **CON-005**: Implement health checks and monitoring
- [ ] **CON-006**: Create deployment automation scripts

Week 10: Scalability & Performance

Sprint Goal: Optimize for production scale

Tasks:

- [ ] **SCA-001**: Implement horizontal scaling capabilities
- [ ] **SCA-002**: Optimize database performance and indexing
- [ ] **SCA-003**: Add caching layers (Redis)
- [ ] **SCA-004**: Implement load balancing

- [ ] **SCA-005**: Setup monitoring and alerting (Prometheus/Grafana)
- [ ] **SCA-006**: Performance testing and optimization

Week 11: Security & Compliance

Sprint Goal: Implement production security

Tasks:

- [ ] **SEC-001**: Implement JWT authentication system
- [ ] **SEC-002**: Add role-based access control
- [ ] **SEC-003**: Setup TLS certificates and HTTPS
- [ ] **SEC-004**: Implement data encryption at rest
- [ ] **SEC-005**: Add audit logging and compliance features
- [ ] **SEC-006**: Security testing and vulnerability assessment

Week 12: Final Deployment & Documentation

Sprint Goal: Complete Version 2.0 and project wrap-up

Tasks:

- [ ] **FIN-001**: Production deployment to cloud infrastructure
- [ ] **FIN-002**: Complete technical documentation
- [ ] **FIN-003**: Create business presentation materials
- [ ] **FIN-004**: Build portfolio showcase and demos
- [ ] **FIN-005**: Setup project maintenance procedures
- [ ] **FIN-006**: Post-mortem and lessons learned documentation

---

3. Model Selection Strategy

3.1 Conductor Agent Decision Matrix

Task Complexity Analysis:

python

```
TASK_COMPLEXITY_MATRIX = {
    "simple": {
        "description": "Basic operations, straightforward tasks",
        "examples": ["document parsing", "simple queries", "status updates"],
        "recommended_models": [
            "anthropic/claude-3-haiku",     # Primary - fast & cheap
            "openai/gpt-3.5-turbo",         # Fallback
            "meta-llama/llama-3-8b"         # Budget option
        ],
        "cost_priority": "high",
        "speed_priority": "high"
    },
    "moderate": {
        "description": "Multi-step reasoning, analysis tasks",
        "examples": ["source validation", "content synthesis", "basic analysis"],
        "recommended_models": [
            "anthropic/claude-3.5-sonnet",  # Primary - balanced
            "openai/gpt-4o-mini",           # Alternative
            "anthropic/claude-3-haiku"      # Budget fallback
        ],
        "cost_priority": "medium",
        "speed_priority": "medium"
    },
    "complex": {
        "description": "Advanced reasoning, creative tasks",
        "examples": ["deep analysis", "report writing", "complex synthesis"],
        "recommended_models": [
            "anthropic/claude-3.5-sonnet",  # Primary
            "anthropic/claude-3-opus",      # High-quality option
            "openai/gpt-4o"                 # Alternative
        ],
        "cost_priority": "low",
        "speed_priority": "low"
    },
    "critical": {
        "description": "Mission-critical, highest quality needed",
        "examples": ["final report generation", "executive summaries"],
        "recommended_models": [
            "anthropic/claude-3-opus",      # Primary - highest quality
            "anthropic/claude-3.5-sonnet"  # Fallback
        ],
        "cost_priority": "lowest",
        "speed_priority": "lowest"
```

```
    }
}
```

3.2 Agent-Specific Model Recommendations

Document Processing Agent:

- **Primary**: Claude 3 Haiku (fast, cheap for parsing)
- **Fallback**: GPT-3.5 Turbo
- **Reasoning**: High volume, low complexity tasks

Web Research Agent:

- **Primary**: Claude 3 Haiku (quick source evaluation)
- **Secondary**: Claude 3.5 Sonnet (complex source analysis)
- **Reasoning**: Mix of simple and moderate complexity

RAG Query Agent:

- **Primary**: Claude 3.5 Sonnet (semantic understanding)
- **Fallback**: Claude 3 Haiku
- **Reasoning**: Requires good semantic reasoning

Analysis Agent:

- **Primary**: Claude 3.5 Sonnet (complex reasoning)
- **Critical Tasks**: Claude 3 Opus (deep analysis)
- **Reasoning**: High complexity analytical work

Report Writer Agent:

- **Primary**: Claude 3.5 Sonnet (excellent writing)
- **Premium**: Claude 3 Opus (executive reports)
- **Reasoning**: Quality writing and formatting crucial

QA Agent:

- **Primary**: Claude 3 Haiku (quick validation)
- **Complex**: Claude 3.5 Sonnet (detailed review)
- **Reasoning**: Mostly simple validation tasks

3.3 Cost Optimization Strategy

Budget Allocation (Monthly Target: $50):

- **70% - Analysis & Writing**: $35 (where quality matters most)
- **20% - Research & Processing**: $10 (high volume, moderate quality)
- **10% - Buffer & Testing**: $5 (development and unexpected usage)

Dynamic Cost Management:

```python
class CostOptimizer:
    def __init__(self):
        self.monthly_budget = 50.0
        self.current_spend = 0.0
        self.model_costs = {
            "anthropic/claude-3-haiku": 0.00025,      # per 1K tokens
            "anthropic/claude-3.5-sonnet": 0.003,     # per 1K tokens
            "anthropic/claude-3-opus": 0.015,         # per 1K tokens
        }

    def select_model(self, task_complexity: str, estimated_tokens: int):
        remaining_budget = self.monthly_budget - self.current_spend

        if remaining_budget < 5.0:  # Emergency mode
            return "anthropic/claude-3-haiku"
        elif task_complexity == "critical" and remaining_budget > 15.0:
            return "anthropic/claude-3-opus"
        else:
            return TASK_COMPLEXITY_MATRIX[task_complexity]["recommended_models"][0]
```

---

4. Risk Management

4.1 Technical Risks

High Risk:

- **API Rate Limits**: OpenRouter/external service limitations
- **Mitigation**: Implement robust retry logic, multiple API keys, fallback services

Medium Risk:

- **Vector Database Performance**: Large-scale embedding operations
- **Mitigation**: Implement database optimization, monitoring, scaling strategies

Low Risk:

- **Model Availability**: Specific models becoming unavailable
- **Mitigation**: Multiple model options per task type, graceful degradation

4.2 Timeline Risks

Dependencies:

- External API access and reliability
- Docker/Kubernetes learning curve for cloud deployment
- CrewAI framework stability and documentation

Mitigation Strategies:

- 20% buffer time built into each phase
- Weekly checkpoint reviews and scope adjustment
- Alternative technology options identified

---

5. Success Metrics

5.1 Version 1.0 Success Criteria

- [ ] Complete research workflow in <10 minutes
- [ ] Process documents up to 100 pages successfully
- [ ] Generate professional reports with proper citations
- [ ] Maintain cost under $2 per research report
- [ ] Achieve 90%+ uptime during demonstration period

5.2 Version 2.0 Success Criteria

- [ ] Support 5+ concurrent research jobs
- [ ] Deploy successfully to cloud infrastructure
- [ ] Implement proper authentication and security
- [ ] Demonstrate horizontal scaling capabilities
- [ ] Complete monitoring and alerting setup

5.3 Business Success Criteria

- [ ] Create compelling portfolio demonstration
- [ ] Generate 3+ detailed case studies
- [ ] Develop reusable components for future projects
- [ ] Document best practices and lessons learned
- [ ] Establish foundation for AI consulting business

---

6. Resource Requirements

6.1 Development Resources

- **Primary Developer**: Full-time equivalent (you)
- **Development Environment**: Cursor IDE, local development machine
- **Cloud Resources**: For Version 2.0 deployment and testing

6.2 External Services

- **OpenRouter API**: $50/month budget
- **Domain & Hosting**: $20/month for cloud deployment
- **Additional Tools**: GitHub, Docker Hub (free tiers)

6.3 Learning & Development

- **CrewAI Documentation**: Framework mastery
- **Kubernetes Basics**: For cloud deployment
- **Vector Database Optimization**: Milvus performance tuning

---

7. Quality Assurance

7.1 Testing Strategy

- **Unit Tests**: 90%+ code coverage target
- **Integration Tests**: End-to-end workflow validation
- **Performance Tests**: Load testing with multiple concurrent jobs
- **User Acceptance Tests**: Real-world research scenarios

7.2 Code Quality

- **Code Reviews**: Self-review checklists and best practices
- **Documentation**: Inline comments and comprehensive README

- **Version Control**: Proper Git workflow with feature branches
- **CI/CD**: Automated testing and deployment pipelines

---

8. Post-Launch Activities

8.1 Portfolio Development

- **Case Studies**: Document 3-5 detailed research examples
- **Demo Videos**: Create engaging demonstration content
- **Technical Blog Posts**: Share learnings and best practices
- **GitHub Repository**: Professional presentation with documentation

8.2 Business Development

- **LinkedIn Content**: Share project milestones and insights
- **Networking**: Engage with AI community and potential clients
- **Feedback Collection**: Gather input from beta users and peers
- **Iteration Planning**: Identify features for future versions

8.3 Maintenance & Evolution

- **Monthly Reviews**: Performance metrics and cost analysis
- **Feature Backlog**: Prioritize enhancements based on usage
- **Technology Updates**: Keep dependencies current and secure
- **Scaling Preparation**: Plan for increased usage and demand