Censys AI Summarization Agent - Project Specifications

Project Overview

Develop a stateless full-stack AI agent that can intelligently summarize Censys cybersecurity data, providing actionable insights from datasets through an intuitive web interface.

Technical Stack Recommendations

- **Backend**: Python (FastAPI/Flask) or Node.js (Express)
- **Frontend**: React/Vue.js or vanilla JavaScript
- **AI/ML**: LiteLLM for unified AI model access
- **Architecture**: Stateless, no database required

Epic 1: Project Setup & Infrastructure

Story 1.1: Initialize Project Structure

Tasks:

- [ ] Create GitHub repository with proper .gitignore
- [ ] Set up virtual environment/dependency management
- [ ] Create basic project structure (frontend/, backend/, docs/, tests/)
- [ ] Initialize README.md with project overview
- [ ] Set up basic development workflow

Acceptance Criteria:

- Repository is publicly accessible
- Project structure follows best practices
- Dependencies are properly managed
- Initial README provides project context

Story 1.2: Development Environment Setup

Tasks:

- [ ] Configure development environment for Windsurf
- [ ] Create environment variable management (.env files)
- [ ] Set up code formatting and linting tools
- [ ] Configure debugging setup

- [ ] Install and configure LiteLLM

Documentation References:

- @web https://docs.censys.com/docs/platform-datasets
- @web https://docs.litellm.ai/docs/

Epic 2: Data Processing & Analysis

Story 2.1: Censys Data Ingestion & Validation

Tasks:

- [ ] Create data models for Censys datasets (Host, Web Property, Certificate)
- [ ] Implement data validation and sanitization
- [ ] Create data parser for sample dataset formats
- [ ] Implement error handling for malformed data
- [ ] Add data preprocessing utilities for AI consumption

Acceptance Criteria:

- Can successfully parse provided sample dataset
- Validates data integrity
- Handles edge cases gracefully
- Supports multiple Censys data formats
- Prepares data in format suitable for AI processing

Documentation References:

- @web https://docs.censys.com/docs/platform-host-dataset#/
- @web https://docs.censys.com/docs/platform-web-property-dataset#/
- @web https://docs.censys.com/docs/platform-certificate-dataset#/

Epic 3: AI Summarization Service

Story 3.1: Comprehensive Summarization Service

Tasks:

- [ ] Set up LiteLLM client with multiple model support
- [ ] Create dataset-specific summarization prompts and strategies
- [ ] Implement individual dataset summarizers (Host, Web Property, Certificate)

- [ ] Create unified multi-dataset summarizer
- [ ] Add prompt engineering for Censys data context
- [ ] Implement error handling and fallback mechanisms
- [ ] Add summary quality validation and formatting

Acceptance Criteria:

- LiteLLM successfully processes Censys data
- Individual dataset summarizers work effectively
- Multi-dataset summarizer provides holistic insights
- Generates relevant, concise summaries
- Handles API failures gracefully
- Configurable model parameters and providers

Documentation References:

- @web https://docs.censys.com/docs/platform-host-dataset#/
- @web https://docs.censys.com/docs/platform-web-property-dataset#/
- @web https://docs.censys.com/docs/platform-certificate-dataset#/
- @web https://docs.litellm.ai/docs/

Questions to Clarify:

- Should we prioritize individual dataset summaries or unified multi-dataset analysis?
- What specific security insights should be highlighted in summaries?
- Are there compliance requirements for AI model usage?

Epic 4: Backend API Development

Story 4.1: Core API Endpoints

Tasks:

- [ ] Create REST API structure using FastAPI/Flask
- [ ] Implement dataset upload/input endpoint
- [ ] Create individual dataset summarization endpoints
- [ ] Add multi-dataset summarization endpoint
- [ ] Implement health check and status endpoints
- [ ] Add request/response validation

Acceptance Criteria:

- RESTful API design
- Proper HTTP status codes
- Request/response validation
- Support for different input formats
- Clear API documentation

Story 4.2: API Testing & Documentation

Tasks:

- [ ] Create unit tests for API endpoints
- [ ] Implement integration tests with mock data
- [ ] Add API documentation (OpenAPI/Swagger)
- [ ] Create test data sets for different scenarios
- [ ] Implement request validation and error handling

Acceptance Criteria:

- 80%+ test coverage
- All endpoints properly tested
- Comprehensive API documentation
- Robust error handling

Epic 5: Frontend User Interface

Story 5.1: Core UI Components

Tasks:

- [ ] Create responsive layout structure
- [ ] Implement data upload/input interface
- [ ] Create summary display components
- [ ] Add loading states and error handling
- [ ] Implement basic styling and UX

Acceptance Criteria:

- Intuitive user interface
- Responsive design

- Clear data visualization
- Proper error messaging
- Support for different input methods

Story 5.2: Enhanced User Experience

Tasks:

- [ ] Add progress indicators for processing
- [ ] Implement summary export functionality
- [ ] Create summary comparison features
- [ ] Add summary customization options (length, focus areas)
- [ ] Implement dataset type selection interface

Acceptance Criteria:

- Smooth user experience
- Clear visual feedback
- Export functionality works
- Summary customization available

Epic 6: Integration & Testing

Story 6.1: End-to-End Integration

Tasks:

- [ ] Integrate frontend with backend
- [ ] Test complete data flow from input to summary
- [ ] Implement comprehensive error handling
- [ ] Add logging and monitoring
- [ ] Create E2E test scenarios

Acceptance Criteria:

- Seamless frontend-backend integration
- Complete user journey testing
- Comprehensive error handling
- Proper logging implementation

Epic 7: Documentation & Finalization

Story 7.1: Comprehensive Documentation

Tasks:

- [ ] Complete README.md with setup instructions
- [ ] Document API endpoints and usage
- [ ] Create user guide with examples
- [ ] Document AI approach and techniques used
- [ ] Create future enhancements list
- [ ] Add testing instructions (manual and automated)

Acceptance Criteria:

- Clear setup instructions
- Complete API documentation
- User-friendly guide with examples
- Technical documentation for developers
- Comprehensive future enhancements list

Architecture Decisions

Summarization Strategy Options

Option A: Individual Dataset Summarizers

- Separate endpoints for each dataset type
- Specialized prompts for each data format
- Allows focused analysis per dataset

Option B: Unified Multi-Dataset Summarizer

- Single endpoint that processes all datasets together
- Holistic analysis across all data types
- Provides comprehensive security posture overview

Recommendation: Implement both approaches

- Individual summarizers for detailed analysis
- Multi-dataset summarizer for comprehensive insights
- Let users choose based on their needs

LiteLLM Integration Benefits

- Unified interface for multiple AI providers
- Built-in fallback mechanisms
- Cost optimization features
- Easy model switching and testing

Definition of Done

- [ ] All acceptance criteria met
- [ ] Code reviewed and follows best practices
- [ ] Tests passing (unit, integration, E2E)
- [ ] Documentation updated and comprehensive
- [ ] Security considerations addressed
- [ ] Performance requirements met

Questions Requiring Clarification

1. **Summarization Scope**: Should we prioritize individual dataset analysis or unified multi-dataset insights?
2. **Data Input Method**: File upload, API input, or both?
3. **Summary Format**: Technical reports, executive summaries, or configurable formats?
4. **AI Model Preferences**: Any specific models to prioritize through LiteLLM?
5. **Sample Dataset**: What format and size is the sample dataset?
6. **Security Focus**: What specific security insights should be highlighted?

Time Estimation

- **Epic 1**: 0.5 hours
- **Epic 2**: 0.5 hours
- **Epic 3**: 1.5 hours
- **Epic 4**: 0.5 hours
- **Epic 5**: 0.5 hours
- **Epic 6**: 0.5 hours
- **Epic 7**: 0.5 hours

Total Estimated Time: 4 hours

Success Metrics

- Functional summarization of Censys data (individual and multi-dataset)
- Clean, maintainable stateless architecture
- Comprehensive documentation
- Effective LiteLLM integration
- User-friendly interface
- Robust error handling and testing