GenAI Proficiency Test - Junior GenAI Engineer

Test Overview

Test Objectives and Deliverables

▼ Purpose, Duration, and Expected Outputs

Test Objective

- **Evaluate proficiency in using genAI tools (Cursor, Windsurf, Claude, etc.) for research, planning, and implementation**
- **Assess understanding of GenAI technologies, frameworks, and best practices**
- **Measure ability to translate GenAI concepts into clear, actionable documentation and implementation plans**
- **Test documentation skills following established style guidelines**

Duration and Deliverables

- **Expected time commitment**:
  - This exercise is designed for you (the candidate) to complete **in OUT-OF-OFFICE time in 5 days**
- **Primary deliverables**: `report_<task>.md` files for each selected task **(AT LEAST ONE MAIN FILE FOR EACH TASK)**
  - If you have multiple main files, name them: `report_<task>_part01_<part_name>.md`
  - You should save your main/long prompts for GenAI workflow illustration
    - Prompt file naming: `report_<task>_prompt.md` or `report_<task>_part01_prompt.md`
- **GenAI implementation focus**: Framework analysis, system design, technology evaluation
- **Report specifications**: Expected 1000-1500 lines per report file, 2-10 files per task
  - **Choose your approach** based on technical complexity and stakeholder communication needs

Documentation Viewing (Optional but Recommended; Skip if you have issues)

- **Install npm first**:
  - **Windows**: Download and install Node.js using the MSI installer from https://nodejs.org/en/download/
  - **Ubuntu**: `sudo apt update && sudo apt install nodejs npm`

    - **Mac**: `brew install node npm`
- **Use Docusaurus for better viewing**: Download `docusaurus_stub.zip` from this GoogleDrive link, unzip, add your markdown files to `/docs` folder
- **Setup**: Run `npm install` and `npm run start` at root project, fix any errors if needed
    - npm server should run at `http://localhost:3000/` after `npm run start`
- **Docusaurus view is superior to IDE view or git view** for reading and reviewing GenAI technical reports

Requirements and Assessment Criteria

Submission Requirement

▼ Deliverable Format and Submission Standards

- **Primary deliverable format**: `report_<task>.md` files with GenAI technology focus
- **File naming convention**:
    - Main files: `report_<task>.md` (AT LEAST ONE MAIN FILE FOR EACH TASK)
    - Multiple parts: `report_<task>_part01_<part_name>.md`
    - Prompt files: `report_<task>_prompt.md` or `report_<task>_part01_prompt.md`
- **Report specifications**: Expected 1000-1500 lines per report file, 2-10 files per task
- **GenAI workflow prompts**: Include `report_<task>_prompt.md` showing strategic genAI usage for technical research and planning
- **Style compliance**: Must follow `ctx_doc_style.md` formatting exactly
- **Multi-audience accessibility**: GenAI content understandable by both technical teams and business stakeholders
- **Task selection**: Minimum one from List A (mandatory), additional from List B (optional)
- **Supplementary materials**: Code snippets, framework comparisons, implementation guides **if applicable**
- **GenAI UTILIZATION**: DEMONSTRATE EFFECTIVE USE OF AI TOOLS to meet all technical analysis and documentation requirements through documented workflows

GenAI Implementation and Design Requirement

▼ GenAI Technology and Framework Standards

GenAI Architecture Requirements

- **System architecture documentation** - detailed GenAI component interactions, data flows, and integration patterns
- **Framework analysis** - comprehensive evaluation of GenAI tools, libraries, and platforms
- **Model integration design**:
  - LLM selection and configuration strategies
  - API integration and cost optimization
  - Performance monitoring and evaluation
- **Implementation planning** - deployment strategies and scaling considerations
- **Technology evaluation**:
  - Framework comparison and selection criteria
  - Cost-benefit analysis and trade-offs
  - Integration complexity and maintenance requirements

Technical Implementation Standards

- **GenAI workflow design**:
  - Detailed prompt engineering and model interaction patterns
  - Data processing and transformation pipelines
  - Agent coordination and communication strategies
- **Integration specifications**:
  - API design and external service connections
  - Real-time processing and batch operation patterns
  - Error handling and fallback mechanisms
- **Technical accountability**:
  - Comprehensive documentation for development teams
  - Implementation timelines and success metrics
  - Code examples for architecture clarity
  - Performance monitoring and optimization procedures

Documentation Requirement

▼ Documentation Standards and Technical Communication Requirements

Documentation and Standards

- **GenAI-first documentation** - comprehensive technology analysis accessible to both technical teams AND business stakeholders
- **Multi-audience technical documentation** - GenAI insights accessible to both engineering teams and business leaders
- **Terminology standardization** - create consistent GenAI terminology section and use throughout report
- **Documentation frameworks** - establish standards for team GenAI documentation and knowledge sharing
- **Technical cross-functional communication** - translates GenAI complexity for business stakeholders
- **Clear accountability** - demonstrates understanding of implementation commitments and project management
- **Stakeholder alignment** - shows approach for keeping all parties informed about GenAI progress and capabilities

Technical Visualization and Implementation

- **Architecture diagrams** - detailed GenAI system design, component relationships, data flow topology
- **Workflow visualization** - prompt engineering flows, model interaction patterns, processing sequences
- **Framework comparison charts** - technology evaluation matrices, feature comparisons, decision frameworks
- **Stakeholder materials** - GenAI capability overviews and implementation updates for business/executive audiences
- **Mermaid charts** - for GenAI workflows, system architectures, and integration diagrams
- **Technical focus** - diagrams facilitate development coordination and stakeholder understanding
- **Integration approach** - include visualizations directly in reports or as supplementary materials

List A - Mandatory Tasks - Choose AT LEAST one task

▼ GenAI Core Technology Tasks (Choose Minimum One)

A01 - LiteLLM and LangGraph Analysis

What You Need to Do

Create comprehensive tutorials for both LiteLLM and LangGraph, then provide detailed comparison analysis. These are critical tools in the GenAI ecosystem that teams need to understand for effective AI implementation.

Task Clarity

- **Dual tutorial creation** - separate detailed guides for LiteLLM and LangGraph
- **Comparative analysis** - detailed comparison between both tools
- **Learning approach** - structured for knowledge retention and team sharing
- **Your role** - technical educator helping team understand these essential GenAI tools
- **Think practical** - focus on real-world usage scenarios and implementation patterns

What You're Solving

Development teams need to understand when and how to use LiteLLM vs LangGraph.

Key Questions to Answer

- **What is LiteLLM?** - core functionality, use cases, advantages
- **What is LangGraph?** - workflow capabilities, agent patterns, execution models
- **When to use each?** - decision framework for tool selection
- **How do they integrate?** - combining both tools in production systems
- **What are the trade-offs?** - performance, complexity, maintenance considerations

Deliverable Requirements

- **LiteLLM tutorial** - comprehensive functionality and usage guide with Python examples
- **LangGraph tutorial** - functionalities, concepts, usage with Python examples
- **Comparison report** - strengths, weaknesses, use case analysis
- **Implementation examples** - practical code snippets and scenarios

- **Integration patterns** - how to use both tools together effectively

A02 - LLM Fine-tuning Guide

What You Need to Do

Create a comprehensive fine-tuning tutorial for large language models that covers multiple approaches, technical considerations, and practical implementation strategies.

Task Clarity

- **Comprehensive coverage** - multiple fine-tuning approaches and techniques
- **Technical depth** - quantization, data sources, optimization techniques
- **Step-by-step documentation** - detailed implementation procedures
- **Your role** - technical guide for teams implementing custom LLM solutions
- **Think production-ready** - focus on approaches that work in real business scenarios

What You're Solving

Teams need to customize LLMs for specific domains or tasks, but fine-tuning is complex with many approaches and technical considerations. Your guide should help teams choose the right approach and implement it effectively.

Key Approaches to Cover

- **Parameter-Efficient Fine-tuning** - LoRA, QLoRA, adapters
- **Full Fine-tuning** - when and how to do complete model retraining
- **Instruction Tuning** - training models to follow specific instructions
- **RLHF (Reinforcement Learning from Human Feedback)** - alignment techniques
- **Quantization strategies** - reducing model size while maintaining performance

Deliverable Requirements

- **Fine-tuning strategies comparison** - analysis of different approaches with use cases
- **Technical specifications** - quantization methods, data requirements, hardware considerations
- **Implementation steps** - chronological fine-tuning procedures with code examples
- **Performance optimization** - efficiency and quality improvement techniques
- **Troubleshooting guide** - common issues and solutions
- **Cost analysis** - computational requirements and budget considerations

A03 - RAG and Reasoning Frameworks Tutorial

What You Need to Do

Create a comprehensive tutorial on Retrieval Augmented Generation (RAG) and related reasoning frameworks that covers concepts, implementation, and advanced techniques for building intelligent information systems.

Task Clarity

- **RAG fundamentals** - core concepts, architecture, and implementation patterns
- **Reasoning frameworks** - advanced techniques for logical reasoning and chain-of-thought
- **Tool ecosystem** - popular frameworks, libraries, and platforms
- **Your role** - technical educator for teams building knowledge-based AI systems
- **Think end-to-end** - from basic RAG to advanced reasoning capabilities

What You're Solving

Teams need to build AI systems that can access and reason over large knowledge bases. RAG is the foundation, but advanced reasoning frameworks enable more sophisticated question-answering and decision-making capabilities.

Key Components to Cover

- **Basic RAG architecture** - retrieval, augmentation, generation pipeline
- **Vector databases** - storage and retrieval of embeddings
- **Chunking strategies** - how to split and organize documents
- **Reasoning frameworks** - chain-of-thought, tree-of-thought, reasoning chains
- **Advanced techniques** - multi-hop reasoning, fact verification, source attribution
- **Evaluation methods** - measuring RAG system performance

Required Tools and Frameworks

- **LangChain, LlamaIndex** - popular RAG frameworks
- **Vector databases** - Pinecone, Weaviate, Chroma comparison
- **Reasoning tools** - ReAct, self-consistency, planning frameworks
- **Evaluation frameworks** - RAGAS, custom evaluation approaches

Deliverable Requirements

- **RAG fundamentals** - architecture, components, and basic implementation
- **Reasoning framework analysis** - different approaches to logical reasoning
- **Tool comparison** - detailed analysis of popular RAG and reasoning frameworks
- **Implementation guide** - step-by-step building of RAG systems with code examples
- **Advanced techniques** - multi-hop reasoning, fact checking, source verification
- **Performance optimization** - retrieval accuracy, generation quality, latency improvements
- **Case study examples** - real-world RAG applications and their implementation

A04 - AI Web Data Extraction and Article Generation System

What You Need to Do

Design and plan a complete AI system that automatically extracts information from predefined web sources, performs aggregation and analysis, generates new articles, and serves content through a REST API.

Task Clarity

- **End-to-end system design** - from web scraping to content serving
- **AI-powered processing** - intelligent extraction, aggregation, and content generation
- **REST API design** - serving generated articles to various consumers
- **Your role** - system architect designing an automated content generation platform
- **Think scalable** - system should handle multiple sources and generate content regularly

What You're Solving

Organizations need automated ways to monitor multiple information sources, synthesize insights, and generate comprehensive articles. This system replaces manual research and writing with AI-powered automation.

System Architecture Components

- **Web scraping module** - automated data extraction from predefined sources
- **Content processing pipeline** - cleaning, parsing, and structuring extracted data
- **AI aggregation engine** - intelligent synthesis and analysis of collected information
- **Content generation system** - AI-powered article writing and formatting

- **Storage and organization** - structured storage of generated articles
- **REST API server** - content delivery and management endpoints

Technical Stack Considerations

- **Web scraping** - Scrapy, BeautifulSoup, Playwright for different site types
- **AI models** - LLMs for content generation, embeddings for similarity analysis
- **Processing frameworks** - workflow orchestration and scheduling
- **Storage design** - database schema for articles, metadata, and source tracking
- **API framework** - FastAPI, Flask, or similar for REST endpoints

Deliverable Requirements

- **System architecture diagram** - complete data flow from sources to API
- **Web scraping strategy** - handling different source types, rate limiting, error handling
- **AI processing pipeline** - content extraction, aggregation, and generation workflows
- **Storage schema design** - database structure for articles, metadata, and relationships
- **REST API specification** - endpoints, request/response formats, authentication
- **Content generation workflow** - AI prompts, templates, and quality control
- **Scheduling and monitoring** - automated execution and system health tracking
- **Implementation timeline** - step-by-step development and deployment plan

A05 - Multi-Source Knowledge Base with AI Agent

What You Need to Do

Design and plan a comprehensive knowledge base system that extracts content from 10-20 different data sources, organizes information with proper metadata and relationships, and includes an AI agent for intelligent querying.

Task Clarity

- **Multi-source extraction** - handle diverse data formats and sources (not just Wikipedia)
- **Knowledge organization** - entity definitions, metadata, content relationships, and linking
- **Storage design** - efficient storage of pages, metadata, and interconnections
- **AI agent integration** - intelligent query processing using the knowledge base
- **Your role** - architect of a domain-specific knowledge platform (e.g., ML/Statistics/AI domain)

What You're Solving

Teams need comprehensive, searchable knowledge bases for specific domains, but information is scattered across many sources in different formats. Your system should create a unified, intelligent knowledge platform that can answer complex questions.

System Components

- **Multi-source extraction engine** - handle websites, PDFs, APIs, databases, documentation sites
- **Content processing pipeline** - parsing, cleaning, entity extraction, relationship identification
- **Knowledge organization system** - entity definitions, categorization, tagging, cross-references
- **Storage architecture** - efficient storage of content, metadata, and relationships
- **AI agent framework** - query understanding, context retrieval, and response generation
- **Search and discovery** - semantic search, browsing, and exploration capabilities

Data Source Considerations

- **Academic sources** - arXiv, research papers, technical documentation
- **Educational platforms** - course materials, tutorials, reference guides
- **Industry sources** - blogs, whitepapers, technical specifications
- **Community sources** - forums, Q&A sites, GitHub repositories
- **Structured data** - APIs, databases, knowledge graphs

Metadata and Organization Design

- **Entity definitions** - concepts, methods, algorithms, tools
- **Taxonomies and tags** - hierarchical organization and categorization
- **Relationships** - prerequisites, related concepts, applications
- **Content versioning** - tracking updates and changes over time
- **Quality metrics** - source credibility, content freshness, accuracy indicators

Deliverable Requirements

- **Multi-source extraction strategy** - handling 10-20 different source types and formats
- **Content processing pipeline** - parsing, entity extraction, and relationship identification
- **Storage schema design** - database structure for entities, content, metadata, and relationships
- **Knowledge organization system** - taxonomies, tagging, and cross-reference strategies
- **AI agent architecture** - query processing, context retrieval, and response generation
- **Search and discovery features** - semantic search, browsing interfaces, and exploration tools
- **Data quality framework** - content validation, source verification, and update mechanisms

- **Implementation roadmap** - phased development approach with milestones and timelines

A06 - Dify Platform Analysis and Comparison

What You Need to Do

Create a comprehensive tutorial and comparison analysis for Dify, covering its usage, comparison with n8n, and comparison with traditional IDE + Cursor workflows. Include practical use cases and implementation guidance.

Task Clarity

- **Dify tutorial** - complete usage guide with practical examples
- **Platform comparisons** - detailed analysis of Dify vs n8n vs IDE+Cursor approaches
- **Use case analysis** - when to choose each platform for different scenarios
- **Your role** - platform evaluator helping teams choose the right GenAI development approach

What You're Solving

Teams need to understand different approaches to building GenAI workflows: visual platforms like Dify, automation tools like n8n, and traditional coding with AI assistance. Each has different strengths for different use cases.

Dify Platform Analysis

- **Core functionality** - workflow building, model integration, deployment capabilities
- **User interface** - visual workflow designer, no-code/low-code features
- **Integration capabilities** - APIs, data sources, external services
- **Deployment options** - cloud, on-premise, scaling considerations
- **Pricing and limitations** - cost structure and platform constraints

Comparison Framework

- **Dify vs n8n** - workflow automation, GenAI integration, ease of use
- **Dify vs IDE+Cursor** - development speed, flexibility, customization
- **Development experience** - learning curve, debugging, collaboration
- **Production readiness** - scalability, monitoring, maintenance
- **Cost considerations** - development time, licensing, operational costs

Deliverable Requirements

- **Dify comprehensive tutorial** - features, capabilities, and step-by-step usage guide
- **Platform comparison matrix** - detailed feature and capability comparison
- **Use case scenarios** - when to choose Dify, n8n, or traditional development
- **Implementation examples** - practical workflows built on each platform
- **Migration considerations** - moving between platforms and integration strategies
- **Best practices** - optimization, troubleshooting, and production deployment
- **Decision framework** - guide for teams to choose the right platform

A07 - ComfyUI Comprehensive Tutorial

What You Need to Do

Create a comprehensive tutorial for ComfyUI covering installation, usage, workflows, and advanced techniques for AI image generation and processing.

Task Clarity

- **Complete tutorial** - from basic setup to advanced workflow creation
- **Practical focus** - hands-on examples and real-world applications
- **Workflow examples** - diverse use cases and implementation patterns
- **Your role** - technical instructor for teams adopting ComfyUI for image generation

What You're Solving

Teams need to understand ComfyUI as a powerful alternative to other AI image generation tools, but it has a steep learning curve. Your tutorial should make ComfyUI accessible and demonstrate its unique advantages.

Tutorial Components

- **Installation and setup** - system requirements, dependencies, initial configuration
- **Interface overview** - node-based workflow editor, basic operations
- **Core concepts** - nodes, connections, data flow, model loading
- **Basic workflows** - text-to-image, image-to-image, inpainting
- **Advanced techniques** - custom nodes, complex workflows, batch processing
- **Model management** - downloading, organizing, and using different models

Key Features to Cover

- **Node-based workflow** - advantages over traditional interfaces
- **Model flexibility** - using different models and checkpoints
- **Custom workflows** - building complex generation pipelines
- **Batch processing** - automating repetitive tasks
- **Integration capabilities** - APIs, external tools, automation
- **Performance optimization** - memory usage, speed improvements

Deliverable Requirements

- **Installation guide** - step-by-step setup for different operating systems
- **Interface tutorial** - navigating the node-based editor and basic operations
- **Workflow examples** - practical tutorials for common use cases
- **Advanced techniques** - custom nodes, complex workflows, and optimization
- **Model management** - best practices for organizing and using models
- **Troubleshooting guide** - common issues and solutions
- **Use case scenarios** - when ComfyUI is the best choice vs alternatives
- **Integration strategies** - connecting ComfyUI with other tools and workflows

A08 - MCP, A2A, and OpenTools Analysis

What You Need to Do

Create comprehensive tutorials for Model Context Protocol (MCP) and Agent-to-Agent (A2A) communication, then compare these approaches with OpenTools. Include practical implementation guidance and use case analysis.

Task Clarity

- **MCP tutorial** - understanding and implementing Model Context Protocol
- **A2A tutorial** - agent-to-agent communication patterns and implementation
- **OpenTools comparison** - how these approaches compare with OpenTools framework
- **Your role** - technical guide for advanced GenAI system integration patterns

What You're Solving

Advanced GenAI systems need sophisticated communication and context-sharing mechanisms. Teams need to understand when to use MCP, A2A, or OpenTools for different integration scenarios.

MCP (Model Context Protocol) Analysis

- **Protocol fundamentals** - context sharing, memory management, persistence
- **Implementation patterns** - how to integrate MCP in applications
- **Use cases** - when MCP provides significant advantages
- **Technical requirements** - infrastructure and development considerations

A2A (Agent-to-Agent) Communication

- **Communication patterns** - direct messaging, event-driven, pub/sub
- **Coordination mechanisms** - task distribution, result aggregation
- **Implementation frameworks** - tools and libraries for A2A systems
- **Scalability considerations** - handling multiple agents and complex workflows

OpenTools Integration

- **OpenTools overview** - capabilities and integration patterns
- **Comparison framework** - when to choose MCP/A2A vs OpenTools
- **Hybrid approaches** - combining different integration methods
- **Migration strategies** - moving between different integration approaches

Deliverable Requirements

- **MCP tutorial** - protocol understanding, implementation, and practical examples
- **A2A communication guide** - patterns, frameworks, and implementation strategies
- **OpenTools analysis** - capabilities, use cases, and integration approaches
- **Comparative analysis** - detailed comparison of all three approaches
- **Implementation examples** - code snippets and practical scenarios
- **Decision framework** - guide for choosing the right integration approach
- **Best practices** - optimization, troubleshooting, and production considerations

A09 - Agent Frameworks Comprehensive Comparison

What You Need to Do

Create an in-depth analysis and comparison of all major agent frameworks, focusing not just on API differences but on practical concerns like boilerplate code, exit costs, framework reliability, and extensibility limitations.

Task Clarity

- **Comprehensive framework coverage** - all major agent frameworks in the market
- **Deep analysis beyond APIs** - focus on practical development and operational concerns
- **Risk assessment** - what happens when frameworks fail or become incompatible
- **Your role** - strategic advisor helping teams choose frameworks that won't trap them

What You're Solving

Teams often choose agent frameworks based on feature lists but later face problems with vendor lock-in, lack of flexibility, or framework abandonment. Your analysis should help teams make informed decisions about long-term viability.

Frameworks to Analyze

- **LangChain/LangGraph** - ecosystem, flexibility, community support
- **AutoGen** - multi-agent capabilities, Microsoft backing
- **CrewAI** - team-based agent coordination
- **Agency Swarm** - OpenAI-focused agent framework
- **Phidata** - data-centric agent framework
- **Haystack** - search and NLP focused agents
- **Others** - emerging and specialized frameworks

Critical Analysis Areas

- **Boilerplate code requirements** - how much setup and maintenance code needed
- **Exit cost analysis** - difficulty of migrating away from framework
- **Framework reliability** - what happens when APIs break or frameworks are abandoned
- **Extensibility limitations** - what you can't do within framework constraints
- **Vendor lock-in risks** - dependency on specific models, services, or platforms
- **Community and support** - long-term sustainability and help availability

Practical Considerations

- **Development velocity** - time to first working prototype vs production system
- **Debugging and troubleshooting** - visibility into agent behavior and failures
- **Performance characteristics** - latency, resource usage, scalability
- **Integration complexity** - connecting with existing systems and tools
- **Customization flexibility** - adapting frameworks to specific needs

Deliverable Requirements

- **Framework overview** - comprehensive list of major agent frameworks
- **Feature comparison matrix** - detailed capability comparison
- **Boilerplate analysis** - code complexity and maintenance requirements
- **Exit cost assessment** - migration difficulty and vendor lock-in risks
- **Reliability evaluation** - framework stability and support quality
- **Extensibility analysis** - limitations and workaround strategies
- **Decision framework** - guide for choosing frameworks based on project needs
- **Risk mitigation strategies** - protecting against framework failures and changes

A10 - Browser-Use Deep Investigation

What You Need to Do

Conduct an in-depth investigation of github.com/browser-use/browser-use, covering its functionalities, usage patterns, user workflows, and most importantly, the underlying mechanisms and methodologies that make it work.

Task Clarity

- **Comprehensive functionality analysis** - what browser-use can and cannot do
- **User workflow documentation** - how people actually use this tool
- **Deep technical investigation** - understand the underlying mechanisms and methodology
- **Your role** - technical investigator providing complete understanding of this solution

What You're Solving

Teams need to understand browser automation solutions for GenAI applications, but most analyses are superficial. Your investigation should provide deep technical understanding of how browser-use works internally.

Investigation Areas

Functionality Analysis

- **Core capabilities** - what browser-use does and its intended use cases
- **API and interface** - how developers interact with the tool
- **Supported browsers** - compatibility and integration options
- **Limitation analysis** - what it cannot do and workarounds

User Flow Documentation

- **Typical usage patterns** - common workflows and use cases
- **Setup and configuration** - initial setup and customization
- **Integration patterns** - how it fits into larger applications
- **Troubleshooting flows** - common issues and resolution patterns

Technical Deep Dive

- **Architecture analysis** - internal components and their relationships
- **Browser interaction mechanisms** - how it controls browser behavior
- **AI integration methods** - how it uses AI for browser automation
- **Communication protocols** - how different components communicate
- **Error handling strategies** - how it manages failures and edge cases

Methodology Investigation

- **Design principles** - underlying philosophy and approach
- **Implementation strategies** - technical choices and trade-offs
- **Performance optimization** - how it achieves efficiency and reliability
- **Security considerations** - safety measures and potential risks

Technical Analysis Focus

- **Browser control mechanisms** - WebDriver, CDP, or other approaches
- **AI decision-making** - how AI determines actions and handles ambiguity
- **State management** - tracking browser state and maintaining context
- **Scalability approach** - handling multiple sessions and concurrent usage
- **Extension and customization** - how to modify or extend functionality

Deliverable Requirements

- **Functionality overview** - comprehensive capability analysis with examples
- **User workflow documentation** - detailed usage patterns and integration examples
- **Technical architecture analysis** - internal design and component relationships
- **Mechanism deep dive** - detailed explanation of how browser automation works
- **Methodology investigation** - design principles and implementation strategies
- **Performance analysis** - efficiency, reliability, and scalability characteristics
- **Comparison with alternatives** - how it differs from other browser automation solutions
- **Implementation recommendations** - best practices for using browser-use effectively

List B - Optional Enhancement Tasks

▼ Learning and Documentation Tasks (Additional Credit)

B01 - Vector Database Tutorial

Task Description

- **Comprehensive tutorial** creation for vector database technology
- **Learning focus** - suitable for self-study and team knowledge sharing
- **Content scope**: Definitions, common tools, detailed tool analysis
- **Writing style** - simple, direct, plain language approach

Deliverable Requirements

- **Concept introduction** - vector database definitions and use cases
- **Tool comparison** - popular vector database options
- **Deep dive analysis** - detailed examination of one selected tool
- **Implementation guidance** - practical usage examples
- **Best practices** - optimization and performance considerations

B02 - Crypto Exchange Products Guide

Task Description

- **User-focused tutorial** on Centralized Exchange products
- **Product coverage**: Spot, Convert, Futures, Margin, Options
- **Perspective** - detailed user experience and functionality analysis
- **Fee analysis** - comprehensive fee structure documentation

Deliverable Requirements

- **Product explanations** - functionality from user perspective
- **Fee structure analysis** - detailed cost breakdown by product type
- **User journey mapping** - typical workflows and processes
- **Risk considerations** - user awareness and safety guidelines

B03 - Market Making Analysis

Task Description

- **Comprehensive Market Making guide** covering all aspects
- **Scope**: Functionalities, top MM services, tools, strategies
  - **Include examples of these strategies**
- **Industry focus** - current market making landscape analysis

Deliverable Requirements

- **Market making fundamentals** - core concepts and mechanisms
- **Service provider analysis** - top market making services comparison
- **Tool evaluation** - market making software and platforms
- **Strategy documentation** - common approaches and methodologies
- **Industry insights** - current trends and best practices

B04 - Crypto Custody Tutorial

Task Description

- **Complete custody solution guide** for cryptocurrency assets
- **Storage types**: Hot, warm, cold wallet strategies
- **Operational focus** - daily operations and security procedures
    - Include examples. You need to understand the purposes of these operations
- **Service evaluation** - top custody service providers

Deliverable Requirements

- **Custody fundamentals** - security models and risk management
- **Wallet strategies** - hot/warm/cold storage implementations
- **Operational procedures** - daily custody management workflows
- **Service comparison** - top custody providers analysis
- **Security best practices** - comprehensive protection strategies

B05 - Technical Analysis Tutorial (Trading)

Task Description

- **Comprehensive technical analysis guide** for cryptocurrency and traditional trading
- **Chart analysis focus** - price patterns, indicators, and trading signals
- **Practical application** - how to use technical analysis for trading decisions
- **Tool coverage** - popular charting platforms and technical analysis software

Deliverable Requirements

- **Technical analysis fundamentals** - core concepts, chart types, timeframes
- **Indicator analysis** - moving averages, RSI, MACD, volume indicators, and other key tools
- **Pattern recognition** - support/resistance, trend lines, chart patterns (head and shoulders, triangles, etc.)
- **Trading signal interpretation** - how to identify entry/exit points using technical analysis
- **Platform comparison** - TradingView, charting tools, and other technical analysis platforms
- **Risk management** - position sizing and stop-loss strategies using technical analysis
- **Case study examples** - real trading scenarios with technical analysis application

B06 - Fundamental Analysis Tutorial (Trading)

Task Description

- **Comprehensive fundamental analysis guide** for evaluating investment opportunities
- **Financial analysis focus** - company financials, economic indicators, market valuation
- **Crypto-specific fundamentals** - tokenomics, protocol analysis, adoption metrics
- **Decision-making framework** - how to use fundamental analysis for investment decisions

Deliverable Requirements

- **Fundamental analysis basics** - core principles and methodology
- **Traditional asset analysis** - P/E ratios, revenue growth, balance sheet analysis, industry comparison
- **Cryptocurrency fundamentals** - tokenomics, protocol revenue, developer activity, adoption metrics
- **Economic indicator analysis** - inflation, interest rates, GDP impact on markets
- **Valuation methods** - different approaches to determining fair value
- **Information sources** - where to find reliable fundamental data and analysis
- **Integration with technical analysis** - combining both approaches for better decisions
- **Case study examples** - fundamental analysis applied to real investment scenarios

B07 - On-Chain Analysis Tutorial

Task Description

- **Comprehensive on-chain analysis guide** for cryptocurrency markets
- **Blockchain data focus** - transaction analysis, wallet behavior, network metrics
- **Tools and platforms** - on-chain analytics platforms and data interpretation
- **Trading applications** - how to use on-chain data for investment decisions

Deliverable Requirements

- **On-chain analysis fundamentals** - what blockchain data reveals about market behavior
- **Key metrics analysis** - active addresses, transaction volume, network hash rate, whale movements

- **Wallet behavior analysis** - identifying smart money, retail vs institutional patterns
- **Network health indicators** - congestion, fees, validator/miner behavior
- **Platform comparison** - Glassnode, Nansen, Dune Analytics, and other on-chain tools
- **Trading signal identification** - how to spot market trends using on-chain data
- **DeFi-specific analysis** - TVL, yield farming patterns, protocol token flows
- **Case study examples** - real market events explained through on-chain data

B08 - Real World Assets (RWA) Tutorial

Task Description

- **Comprehensive Real World Assets guide** covering tokenization of physical assets
- **Asset tokenization focus** - how physical assets become digital tokens
- **Market analysis** - current RWA projects, platforms, and market opportunities
- **Technical implementation** - blockchain infrastructure for RWA tokenization

Deliverable Requirements

- **RWA fundamentals** - definition, benefits, and challenges of asset tokenization
- **Asset categories** - real estate, commodities, debt instruments, equity, art, and other physical assets
- **Tokenization process** - technical steps to convert physical assets to blockchain tokens
- **Platform analysis** - major RWA platforms, protocols, and infrastructure providers
- **Regulatory considerations** - legal framework, compliance requirements, and jurisdictional differences
- **Market opportunities** - current trends, investment potential, and growth areas
- **Technical architecture** - smart contracts, oracles, and blockchain infrastructure for RWA
- **Case study examples** - successful RWA tokenization projects and their implementation details

B09 - Product-UIUX-Designer Team Analysis

Task Description

- **Comprehensive team structure analysis** for Product-UIUX-Designer organizations
- **Role definition focus** - detailed breakdown of responsibilities and sub-roles

- **Cross-team interaction** - how this team collaborates with engineering, business, and other departments
- **Operational workflow** - daily activities, project lifecycle, and deliverable processes

Deliverable Requirements

- **Team structure breakdown** - Product Manager, UI Designer, UX Designer, UX Researcher roles and sub-specializations
- **Role responsibilities** - detailed daily activities, key deliverables, and success metrics for each role
- **Sub-role analysis** - specialized positions like Service Designer, Interaction Designer, Visual Designer
- **Cross-functional collaboration** - interaction patterns with Engineering, Data Analytics, Business Development, Marketing
- **Workflow documentation** - project lifecycle from concept to launch, including design sprints and iteration cycles
- **Tool ecosystem** - Figma, Adobe Creative Suite, prototyping tools, user research platforms
- **Communication protocols** - how design decisions are communicated and implemented across teams
- **Success measurement** - KPIs, user feedback integration, and design impact assessment

B10 - Product Management Office (PMO) Analysis

Task Description

- **Comprehensive Product Management Office guide** covering organizational structure and operations
- **PMO functions focus** - strategic planning, resource allocation, project coordination
- **Cross-departmental role** - how PMO interfaces with all business units
- **Operational excellence** - processes, methodologies, and best practices

Deliverable Requirements

- **PMO fundamentals** - definition, purpose, and organizational positioning
- **Core functions** - portfolio management, resource planning, process standardization, performance tracking
- **Role structure** - PMO Director, Program Managers, Project Coordinators, Business Analysts

- **Strategic planning** - roadmap development, priority setting, resource allocation strategies
- **Cross-functional coordination** - interaction with Engineering, Sales, Marketing, Finance, Operations
- **Process management** - standardized workflows, documentation requirements, quality assurance
- **Performance measurement** - KPI tracking, project success metrics, team productivity analysis
- **Tools and systems** - project management software, collaboration platforms, reporting dashboards
- **Best practices** - successful PMO implementation strategies and common pitfalls to avoid