GenAI Proficiency Test - Data Engineer + DevOps

Test Overview

Test Objectives and Deliverables

▼ Purpose, Duration, and Expected Outputs

Test Objective

- **Evaluate proficiency in using genAI tools (Cursor, Windsurf, Claude, etc.) for technical documentation and planning**
- **Assess understanding of complex infrastructure and data engineering concepts**
- **Measure ability to translate technical requirements into clear, actionable reports**
- **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 technical workflow illustration
    - Prompt file naming: `report_<task>_prompt.md` or `report_<task>_part01_prompt.md`
- **Technical architecture focus**: Infrastructure design, deployment planning, system integration
- **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 technical reports

Requirements and Assessment Criteria

Submission Requirement

▼ Deliverable Format and Submission Standards

- **Primary deliverable format**: `report_<task>.md` files with technical architecture 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
- **Technical planning prompts**: Include `report_<task>_prompt.md` showing strategic genAI usage for technical planning
- **Style compliance**: Must follow `ctx_doc_style.md` formatting exactly
- **Multi-audience accessibility**: Technical content understandable by both engineering teams and business stakeholders
- **Task selection**: Minimum one from List A (mandatory), additional from List B (optional)
- **Supplementary materials**: Code snippets, infrastructure plans, technical diagrams **if applicable**
- **GenAI UTILIZATION**: DEMONSTRATE EFFECTIVE USE OF AI TOOLS to meet all technical, architectural and documentation requirements through documented workflows

Infrastructure and Deployment Requirement

▼ Technical Architecture and System Design Standards

Infrastructure Architecture Requirements

- **System architecture documentation** - detailed component interactions, dependencies, and technical integration patterns
- **Infrastructure components** - VMs, services, modules, component interactions, access control systems
- **Network and storage design**:
  - Network architecture with security groups and load balancing
  - Storage architecture for shared access and data persistence
  - Access control systems and permission structures
- **Infrastructure as Code** - Terraform configurations and Ansible automation strategy
- **Performance considerations**:
  - Scaling approach and resource allocation
  - Performance specifications and optimization
  - Load testing and monitoring requirements

Technical Implementation Standards

- **Deployment procedures**:
  - Detailed installation steps and configuration
  - Chronological deployment timeline
  - Technical maintenance workflows
- **Integration specifications**:
  - API design and database schema
  - Real-time processing architecture
  - Component communication patterns
- **Technical accountability**:
  - Comprehensive documentation for engineering teams
  - Infrastructure delivery timelines
  - Technical snippets for architecture clarity
  - System monitoring and troubleshooting procedures

Documentation Requirement

▼ Documentation Standards and Technical Communication Requirements

Documentation and Standards

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

Technical Visualization and Implementation

- **Architecture diagrams** - detailed system integration, component relationships, infrastructure topology
- **Timeline visualization** - Gantt charts with milestones, dependencies, deployment sequences
- **Technical coordination charts** - implementation workflows, system dependencies, integration patterns
- **Stakeholder materials** - infrastructure overviews and progress updates for business/executive audiences
- **Mermaid charts** - for architecture, flows, and system integration diagrams
- **Technical focus** - diagrams facilitate engineering coordination and stakeholder understanding
- **Integration approach** - include visualizations directly in reports or as supplementary materials

List A - Mandatory Tasks - Choose AT LEAST one task

▼ Infrastructure and Data Engineering Core Tasks (Choose Minimum One)

A01 - AWS Data Platform Foundation

What You Need to Do

Design and plan the deployment of a comprehensive AWS Data Platform that can serve as the foundation for a data engineering team. You are creating the blueprint and deployment strategy for a complete data infrastructure.

Task Clarity

- **You are the architect** - create the master plan for building an AWS-based data platform from scratch
- **Target outcome** - a production-ready platform that multiple team members can use for data engineering work
- **Your role** - design the system architecture and create deployment documentation that DevOps engineers can follow
- **Think enterprise-level** - this platform should support 20-30 users with proper security and access controls

Required Platform Components

- **User Linux systems** - EC2 instances for data engineers to work on
- **User AWS access** - proper IAM roles and policies for team members
- **NFS storage** - shared network file system for team collaboration
- **FreeIPA integration** - centralized user authentication and management
- **Infrastructure as Code** - everything must be deployable via Terraform and Ansible (no manual clicking in AWS console)

What Success Looks Like

Your deliverable should answer: "If I give this plan to a DevOps engineer, can they build this entire platform step-by-step without confusion?"

Deliverable Requirements

- **Infrastructure architecture diagram** - show how all components connect and interact
- **Access control design** - how users log in, what permissions they have, how security is managed
- **Terraform configuration plans** - infrastructure provisioning strategy and module structure
- **Ansible playbook strategy** - automation approach for configuration management
- **Step-by-step deployment chronology** - exact order of implementation with timeline estimates
- **Operational procedures** - how to maintain, update, and troubleshoot the platform

A02 - Dask Cluster for Data Processing

What You Need to Do

Design and plan a Dask distributed computing cluster that will be added to the AWS Data Platform from A01. This cluster needs to handle concurrent data processing workloads for 20-30 team members.

Task Clarity

- **Build on A01 foundation** - assume the basic AWS Data Platform (A01) is already operational
- **Dask cluster purpose** - enable distributed Python computing for large datasets
- **Scalability focus** - system must handle multiple users running concurrent data processing jobs
- **Integration requirement** - must work seamlessly with existing platform infrastructure
- **Your role** - design the cluster architecture and create deployment strategy

What You're Solving

Data engineers need to process large datasets that don't fit on single machines. Dask allows Python code to run across multiple machines, but setting up a multi-user cluster requires careful planning.

Required Components

- **Dask scheduler** - coordinates work across cluster nodes
- **Dask workers** - machines that execute the actual computations
- **User access integration** - connect with FreeIPA authentication from A01
- **Resource management** - prevent users from overwhelming the cluster
- **Monitoring setup** - track cluster performance and user activity

Deliverable Requirements

- **Cluster architecture design** - node configuration, networking, and resource allocation strategy
- **Terraform and Ansible integration** - how to deploy cluster using existing IaC approach
- **User access management** - how users connect to cluster and submit jobs
- **Performance optimization** - configuration for handling 20-30 concurrent users efficiently
- **Monitoring and alerting setup** - track cluster health and performance metrics
- **Step-by-step deployment chronology** - implementation timeline with dependencies

A03 - Metaflow ML Pipeline Platform

What You Need to Do

Design and plan a Metaflow cluster deployment that enables the team to build, run, and manage machine learning pipelines. This will be added to the AWS Data Platform from A01 to support ML workflow orchestration.

Task Clarity

- **Build on A01 foundation** - leverage existing AWS Data Platform infrastructure
- **Metaflow purpose** - Netflix's ML platform for creating reproducible, scalable ML pipelines
- **Team scale** - support 20-30 data scientists and ML engineers
- **Pipeline focus** - enable users to build ML workflows that can run locally or scale to cloud
- **Your role** - architect the Metaflow deployment and integration strategy

What You're Solving

ML teams need to manage complex workflows: data ingestion → feature engineering → model training → model deployment. Metaflow provides infrastructure for versioning, scaling, and monitoring these pipelines, but requires proper cluster setup.

Required Components

- **Metaflow service** - central orchestration and metadata service
- **AWS integration** - S3 for artifacts, EC2/Batch for compute scaling
- **User workspace setup** - how team members develop and run pipelines
- **Pipeline templates** - standardized starting points for common ML workflows
- **Integration with A01** - leverage existing authentication and storage systems

Deliverable Requirements

- **Architecture design** - Metaflow components and their relationships with AWS services
- **Infrastructure automation** - Terraform and Ansible configurations for deployment
- **User workflow setup** - how team members create, test, and deploy ML pipelines
- **Integration strategy** - connections with existing data sources and A01 platform
- **Operational procedures** - maintenance, troubleshooting, and scaling strategies
- **Step-by-step deployment chronology** - implementation timeline with milestones

A04 - Web/App Tracking Analysis

What You Need to Do

Create comprehensive analysis of web and mobile app tracking systems, comparing custom-built solutions versus third-party services like AppsFlyer. You need to understand and document how businesses track user behavior and measure marketing effectiveness.

Task Breakdown

Task A04a - Custom Tracking Service Analysis

Research and document how to build your own tracking system

- **Focus areas**: App install tracking, conversion tracking, user event flow tracking
- **Technical understanding**: How tracking links work, what data gets collected, how attribution works
- **Architecture assumption**: You're designing a tracking service from scratch for your company

Task A04b - AppsFlyer Integration Assessment

Analyze AppsFlyer as an alternative to custom tracking

- **Service evaluation**: What AppsFlyer does and how their system works
- **Problem-solution mapping**: What challenges from A04a does AppsFlyer solve?
- **Implementation comparison**: Benefits of using AppsFlyer vs building custom solution

What You're Solving

Marketing teams need to know: "Which marketing campaigns are bringing users?" and "What do users do after they install our app?" This requires complex tracking infrastructure that most

companies struggle to build effectively.

Key Questions to Answer

- **How do tracking links work?** - technical mechanism behind attribution
- **What data gets collected?** - user actions, device info, marketing source data
- **How is attribution determined?** - connecting user actions back to marketing campaigns
- **What are the challenges?** - why is custom tracking difficult to implement correctly
- **Why use AppsFlyer?** - specific problems it solves better than custom solutions

Deliverable Requirements

- **Technical comparison** - detailed analysis of custom vs AppsFlyer approaches
- **Event flow diagrams** - visualize how tracking data moves through systems
- **Challenge documentation** - specific problems with custom tracking and how AppsFlyer addresses them
- **Implementation planning** - considerations for choosing between approaches

A05 - Real-Time Streaming Data Pipeline

What You Need to Do

Design a complete streaming data pipeline that ingests data from AppsFlyer in real-time, processes it through multiple aggregation layers, and delivers insights to business dashboards.

Task Clarity

- **Build on A04** - assume AppsFlyer integration is implemented (from A04b)
- **Real-time focus** - data flows continuously, not in batches
- **Processing layers**: Raw events → Hourly aggregations → Daily aggregations
- **End goal** - business teams get up-to-date analytics dashboards
- **Your role** - architect the entire data flow from source to dashboard

What You're Solving

Business teams need real-time insights: "How many users installed our app today?" "Which marketing campaigns are performing now?" But raw AppsFlyer data is too detailed and frequent for direct dashboard use - it needs processing and aggregation.

Pipeline Architecture

- **Data ingestion** - continuous streaming from AppsFlyer APIs
- **Stream processing** - real-time data transformation and cleaning
- **Aggregation layers**:
  - **Hourly aggregations** - group events by hour for trend analysis
  - **Daily aggregations** - summarize daily performance metrics
- **Dashboard integration** - processed data feeds business analytics tools

Required Components

- **Streaming infrastructure** - Kafka, Kinesis, or similar for data ingestion
- **Processing engine** - Apache Flink, Spark Streaming, or similar for real-time processing
- **Storage layer** - where aggregated data gets stored for dashboard access
- **Dashboard connection** - how business tools consume the processed data

Deliverable Requirements

- **End-to-end pipeline architecture** - complete data flow design from AppsFlyer to dashboards
- **Infrastructure deployment plan** - required AWS services and configurations
- **Processing logic documentation** - detailed aggregation layer implementations
- **Dashboard integration strategy** - tool selection and database design
- **Code specifications** - processing step implementations and data transformations
- **Monitoring and alerting** - pipeline health tracking and performance metrics

A06 - Rapid Analytics Solution

What You Need to Do

Design a quick-to-deploy analytics solution that can handle immediate business needs while a full data pipeline is being built. This is a temporary but functional system that prioritizes speed and flexibility over perfect architecture.

Task Clarity

- **Speed over perfection** - deploy in days/weeks, not months
- **Temporary nature** - designed to be replaced by a full pipeline later (like A01+A02+A03, or A05)

- **Business priority** - satisfy immediate analytics requests while buying time for proper infrastructure
- **Flexibility focus** - easily accommodate various business team requests
- **Your role** - design a pragmatic solution that balances speed with functionality

What You're Solving

Business teams need analytics NOW, but building proper data infrastructure (like A05) takes months. You need a solution that can provide valuable insights quickly while the full pipeline is under development.

Solution Characteristics

- **Multi-source integration** - handle both streaming data and batch data sources
- **Rapid deployment** - can be set up and running quickly
- **Business-friendly interface** - dashboards that business teams can actually use
- **Accommodation of changes** - easy to modify when business requirements change
- **Migration path** - clear plan for transitioning to full pipeline later

Possible Approaches to Consider

- **Low-code/no-code tools** - faster setup than custom development
- **Cloud-managed services** - reduce infrastructure management overhead
- **Simplified data models** - basic but useful analytics instead of complex modeling
- **Incremental enhancement** - start simple, add features as needed

Deliverable Requirements

- **Solution architecture** - lightweight, flexible design that prioritizes deployment speed
- **Data source integration strategy** - how to handle multiple input types efficiently
- **Dashboard implementation plan** - business-friendly analytics interface design
- **Rapid deployment procedures** - step-by-step setup and configuration guide
- **Migration planning** - clear transition strategy to full pipeline solution

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 - LiteLLM and LangGraph Analysis

Task Description

- **Dual tutorial creation** - separate reports for LiteLLM and LangGraph
- **Comparative analysis** - detailed comparison between both tools
- **Learning approach** - structured for knowledge retention and sharing

Deliverable Requirements

- **LiteLLM tutorial** - comprehensive functionality and usage guide
  - Include Python examples
- **LangGraph tutorial** - Functionalities, concepts, usage
  - Python examples
- **Comparison report** - strengths, weaknesses, use case analysis

- **Implementation examples** - practical code snippets and scenarios

B03 - LLM Fine-tuning Guide

Task Description

- **Comprehensive fine-tuning tutorial** for large language models
- **Option evaluation** - analysis of multiple fine-tuning approaches
- **Technical depth** - quantization, data sources, optimization techniques
- **Step-by-step documentation** - detailed implementation procedures

Deliverable Requirements

- **Fine-tuning strategies** - comparison of different approaches
- **Technical specifications** - quantization methods and data requirements
- **Implementation steps** - chronological fine-tuning procedures
- **Performance optimization** - efficiency and quality improvement techniques
- **Troubleshooting guide** - common issues and solutions

B04 - 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

B05 - 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

B06 - 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

B07 - 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

B08 - 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

B09 - 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

B10 - 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

B11 - 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

B12 - 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