Certified Agentic AI & Robotics Engineer (CAARE): Program Guide

Version: 3 (September, 2025)

Introduction

Artificial intelligence and robotics are reshaping every industry. Organizations require professionals who can design, build, and deploy autonomous software agents and physical robots that operate reliably, securely, and at scale. The Certified Agentic AI & Robotics Engineer (CAARE) program provides a comprehensive, multi-level curriculum designed to equip engineers with these skills.

The AI Revolution is Here: The Future Belongs to the Architects of Intelligence

The demand for elite AI talent has skyrocketed, creating unprecedented opportunities for skilled engineers. Companies are investing billions not just in technology, but in the human minds that can harness its power. We are very lucky to be living in the age of AI. The following real-world examples illustrate the immense value and potential of a career in Artificial Intelligence.

The New Reality of AI Talent & Innovation

- **The $250 Million Offer:** Meta successfully recruited a 24-year-old Ph.D. dropout, Matt Deitke, with a staggering $250 million package after an initial $125 million offer was declined.
- **The $1.5 Billion Rejection:** Andrew Tulloch, co-founder of Thinking Machines Lab, rejected a six-year, $1.5 billion personal compensation package from Meta, shortly after the company's failed $1 billion bid for his startup.
- **The $250 Million Startup Acquisition:** The two-year-old Generative AI cybersecurity startup, Prompt Security, was acquired by SentinelOne for $250 million in cash after having raised only $23 million.
- **The $3.1 Billion Valuation:** AI sales-automation platform Clay raised $100 million in a funding round led by Alphabet's CapitalG, rocketing its valuation to $3.1 billion—more than double its worth just three months prior.
- **The $2.4 Billion Talent Acquisition:** Google licensed technology from Windsurf for $2.4 billion, bringing its CEO and R&D team to Google DeepMind to advance its AI ambitions.
- **The $500 Billion Valuation:** OpenAI is negotiating a share-sale round that would value the company at $500 billion, surpassing SpaceX and demonstrating the immense financial race to secure and retain top AI talent.
- **The Rise of the 20-Something Founders:** A new wave of young entrepreneurs is flocking to San Francisco to launch AI startups, with figures like Alexandr Wang (Scale AI) becoming multi-billionaires and Chief AI Officers before the age of 30.

These figures surpass the earnings of most elite athletes and Fortune 500 CEOs. The message is clear: the race for AI dominance is more intense than ever, and the future belongs to those who can build and master intelligent systems.

Your Path to Becoming an AI Leader

Our **Certified Agentic AI & Robotics Engineering (CAARE) Program** is an intensive, hands-on journey designed to forge the next generation of AI pioneers. We focus on the next wave of AI—**Agentic AI and Robotics**—moving beyond foundational knowledge to build practical, real-world expertise.

This is not just a certification; it's an interactive career pathway with continuous feedback on your progress. The program is structured into four distinct levels, with exams conducted online via a fixed, proctored schedule.

Agentic AI Strategy for Pakistan

Premise: Pakistan must place smart, early bets on agentic AI as we train millions of developers and launch new ventures. Our strategy rests on four working hypotheses:

1. **Agentic AI is the trajectory.**
   AI is moving from chat to outcome-oriented agents that plan, use tools, and take actions. This guides our curriculum, tooling, and venture pipeline.

2. **Cloud-native foundations win.**
   Kubernetes plus Dapr (Actors, Workflows, Agents) and Ray provides the scalable, observable, resilient base for distributed agent systems.

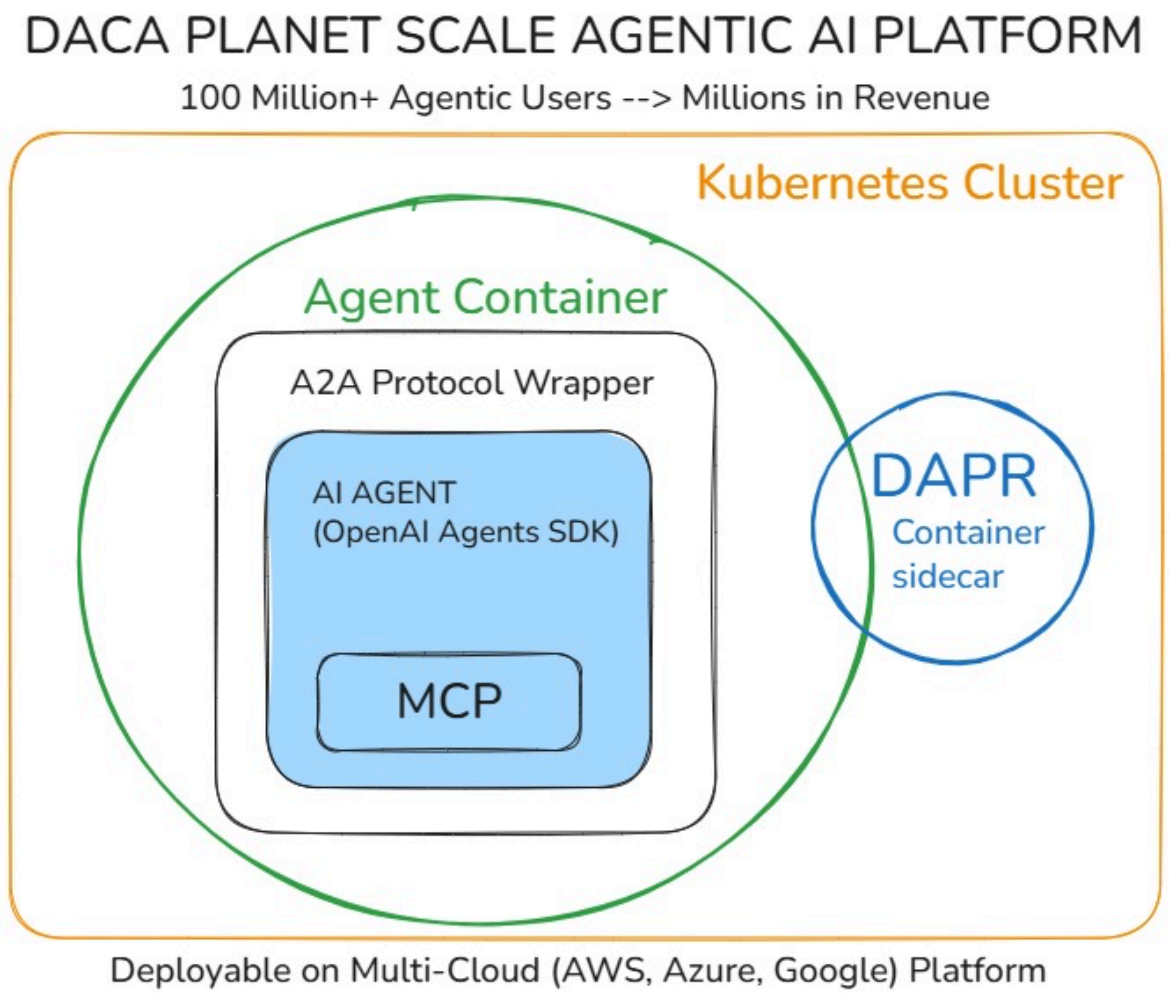
3. **The learning gap is the bottleneck.**
   Most failures stem from weak workflow design, integration, and governance—not model capability. We will close this gap with hands-on training, playbooks, guardrails, and ROI-first delivery.

4. **The web is becoming agentic and interoperable.**
   Open protocols—MCP (capability discovery/context), A2A (authenticated agent-to-agent), and NANDA (identity/authorization/audit)—enable composable automation across apps, devices, and clouds. The browser shifts from “tabs” to an outcome orchestrator.

Execution Pillars:

- **Talent engine**: workforce development in agentic patterns, workflow design, and safety.
- **Reference stack**: production blueprints on K8s + Dapr + Ray with observability and cost controls.
- **Standards readiness**: MCP/A2A/NANDA-aware architectures.
- **Measure & adapt**: publish results, iterate, and reallocate based on evidence.

Certification Overview



In our program we use these building blocks to develop Planet Scale AI Agents.

The certification program consists of four levels, each with specific exams tailored to different expertise levels, exams are conducted online, on a fixed schedule, and are proctored:

Level 1: Agentic AI Foundations

Focuses on foundational knowledge in Prompt and Context Engineering, n8n, Python, Markdown, Agentic AI, and related concepts.

1. Exam L1:P0-PTE: Prompt and Context Engineering: Effective AI Communication Level 1 Certification Exam
Duration: 2 Hours and 45 Minutes | Questions: 150

The exam is divided into thematic sections, with questions building from basics to advanced applications.

a. Fundamentals

- Primary goals and differences between prompt engineering (crafting instructions for desired outputs) and context engineering (providing relevant facts/documents for grounding).
- How LLMs work (as prediction engines guessing next tokens, not human-like understanding).
- Common failure modes (e.g., hallucinations from missing info, messy outputs from vague instructions).
- Recommended workflows (prompt first, then ground with context; feeding tool outputs back in agentic flows).
- Viewing LLMs as sophisticated autocomplete systems.

b. Configuration

- Settings like temperature (controls randomness/creativity; low for consistency, high for diversity), top-K (limits to top likely tokens), top-P (nucleus sampling until probability threshold), output/token limits (affects length and cost).
- Starting points for configs (conservative: low temp/top-P/K; balanced; creative: higher values).
- Appropriate uses (e.g., temp 0 for math, higher for writing).

c. Prompting Techniques

- Basic: zero-shot (direct question), few-shot (3–5 examples for patterns), one-shot.
- Advanced: role prompting (assign persona), system prompting (behavior guidelines), Chain of Thought (CoT; "think step by step"), Self-Consistency (multiple paths, pick common answer), Step-Back (general question first), ReAct (reasoning + actions/tools), Tree of Thoughts (ToT; branching for complex decisions).
- Tips like using "Let's think step by step" with temp=0 for consistency.

d. Best Practices & Pitfalls

- Best practices: specific/action verbs, positive instructions, structured formats (e.g., JSON), variables for reusability, break complex tasks into chains, provide context from prior interactions, optimize token use.
- Pitfalls: vague/ambiguous instructions (unpredictable outputs), overloading with constraints, negative constraints over positives, over-relying on tools without reasoning, excessive details/deadlines.
- Examples of strong vs. weak prompts for analysis, coding, essays.

e. Testing and Evaluation

- Frameworks: record prompt versions/goals/settings/quality notes.

- A/B testing (compare versions), metrics (accuracy, relevance, completeness, style, format, consistency across runs).

f. Advanced Techniques and Tips

- Context management in long conversations (summarize past, break tasks).
- Prompt chaining (sequential steps), structured outputs (JSON for analysis).
- Multi-modal prompting (explicit about image details).

g. Mixture-of-Experts (MoE) & Prompting

- MoE architecture: gating network/router selects experts (sub-networks), sparse activation for efficiency/scalability.
- Impact on prompting: domain-specific signals upfront, separate mixed tasks, front-load cues, reduce temp for consistency.
- Benefits/drawbacks: efficiency/specialization vs. routing instability/memory overhead.
- Expert-aware prompting: explicit language/style, avoid vagueness.

h. Practical Application

- Elements for effective prompts in content creation (topic, audience, tone, format), code generation (language, requirements, examples), customer feedback analysis (sentiment, themes, recommendations).

i. Context Engineering Integration

- RAG (Retrieval-Augmented Generation): prioritize relevant/newest passages, optimize chunking/ranking/deduping/token budgets.
- Combining with prompts: prompts for behavior, context for knowledge.

j. Comprehensive/Applied Scenarios

- Consolidated examples: optimized prompts for reports, Q&A bots, stories; addressing pitfalls in MoE (e.g., front-loading, low temp).

k. Context Engineering

- Basics: LLM as CPU, context window as RAM; for building AI agents/apps.
- Differences from prompt engineering (more comprehensive, code-like for agents).
- Agent components: model, tools (external interaction), memory (dynamic history), knowledge (static info), orchestration, guardrails (behavior/safety), audio/speech.
- Multi-agent systems: splitting tasks (e.g., search + summarize), sharing context.
- Strategies: writing/selecting/compressing/isolating context, actions as decisions.
- Advanced: architecture (state management, compression like hierarchical summarization), optimization (accuracy/latency/cost/recovery), security

(isolation), memory hierarchies, RAG integration (semantic chunking), temporal reasoning (graph-based), enterprise trade-offs (modular components).

I. 6-Step Framework

- Steps: strong command verbs (e.g., "analyze"), rule of three for context (who/what/when? or past/present/future), logic (reasoning/output format), roleplay (specify expertise), questions (iterative refinement until repetition), voice memos (natural follow-ups).
- When to use extensive/minimal context (complex vs. simple tasks).
- Great vs. weak prompters (use questions to refine).
- Common mistakes: vague commands, generic info.
- Benefits: transforms interactions into expert consultations.

J. AI Image Generation Prompting (Nano Banana)

- Core principles: specificity over generality, visual hierarchy (subject → environment → lighting → technical details), professional photography language (camera terminology, lighting setups, photography styles).
- Anatomy of effective image prompts: subject description, pose/action, environment, lighting, style, technical specifications, mood/atmosphere.
- Professional photography terminology: lens specifications (85mm, 50mm, 35mm), aperture settings (f/1.4, f/2.8, f/8), depth of field effects (shallow for subject isolation, deep for environmental context), lighting patterns (Rembrandt, studio, natural window light).
- Best practices: anchor subject with "of the uploaded photo" for consistency, control scene with environment cues, specify style and mood (cinematic, corporate, fine art), borrow photography language for realism, add branding elements (text, logos, graphics).
- Style categories: corporate/professional (executive portraits, headshots, LinkedIn profiles), creative/artistic (fine art, black and white, documentary), fashion/editorial (magazine style, high fashion, editorial shoots), lifestyle/personal branding (environmental context, authentic moments).
- Common mistakes: being too vague, conflicting styles, overcomplicating prompts, missing key elements (lighting, pose, environment).
- Quality control checklist: subject clearly described, pose/positioning specific, lighting type and direction specified, environment/background detailed, camera/technical specs included, mood and style communicated, professional photography language used.

---

2. Exam L1:P1-LCF Low-Code Full-Stack Agentic AI Development
Duration: 3 Hour | **Questions:** 155

- Fundamentals of AI chatbots vs. agents, core components, and architecture
- No-code, low-code, and full-code development spectrum and n8n's role

- Benefits and vendor lock-in risks of low-code platforms, migration strategies
- Integration, strengths, and use cases of n8n with OpenAI Agents SDK
- Target audience, advantages, disadvantages, and scalability of AI agents
- Model Context Protocol (MCP) role, benefits, architecture, and future
- Balancing speed, scalability, and architectural control in development
- Career implications and division of labor between n8n and OpenAI Agents SDK
- "Perceive, plan, and act" paradigm, planner/reasoner functions, and agentic scenarios
- n8n Editor UI, canvas, nodes, workflow templates, and execution behavior
- Trigger Nodes, Action roles, node controls, connections, and adding nodes
- Data flow, sticky notes, search functionality, and connecting nodes in n8n
- n8n node processing model, item structure, binary data, and mapping techniques
- Risks and purposes of "Always Output Data," data pinning, and partial execution
- Splitting, merging, looping, waiting, sub-workflows, and error handling in workflows
- Conditional branching, branch combination risks, and credentials management
- n8n expressions purpose, placeholders, greeting/nested field access, and computation
- Ternary forms, multi-item/timestamp access, missing data handling, and usage locations
- Template literals, conditional logic, and expression syntax rationale
- Definition, differences, and requirements of AI agents vs. LLMs
- Nodes for agent communication, memory impact, and AI Agent Tool functionality
- Cluster nodes, RAG meaning/enhancement, and vector stores with search mechanisms
- Custom knowledge access, chunk size, LangChain, and tool selection in agents
- LangSmith role, "hallucination," "completions," and tools in agent loops
- Agent validation, "agent node" role, and multi-agent benefits/challenges
- MCP interaction sequence, protocols, server triggers, and security practices
- UXPilot and Lovable philosophy, data flow, consistency steps, and prompt structure
- Export formats, Lovable code generation, and post-generation developer focus
- Webhook and AI Agent node functions, response configuration, and Supabase suitability
- Row-Level Security (RLS), vector similarity search, and RAG workflow sequence
- File processing triggers, long-term memory, and semantic search queries
- Handling AI-generated code and SQL-first RAG with pgvector advantages
- Real-time triggered action patterns and first skills in low-code AI development
- Workflow orchestration platform and Supabase capabilities/features

---

3. Exam L1:P2-FMP Fundamentals of Modern AI Python
Duration: 2 Hours | **Questions:** 50

- Basics & Typing – Python syntax, type annotations, input(), bytecode, .pyc files, and platform independence.
- Data Types & Immutability – Strings, tuples, lists, sets, dictionary keys, boolean evaluation, and isinstance().
- Operators & Expressions – Operator precedence, identity vs equality, short-circuiting, bitwise operators, floor division.
- Strings & Slicing – String indexing, raw strings, strip(), replace(), slicing, and escape sequences.
- Lists & Tuples – List references, slicing, unpacking, tuple mutability quirks.
- Sets & Dictionaries – Set operations, dictionary keys, overwriting keys, comprehensions, frozenset, and len() on dicts.
- Control Flow & Loops – if-elif-else, while, for loops, else on loops, continue, comprehensions with conditions, and try-except-else.
- Functions & Scope – Variable scope, dictionary comprehensions, type conversion, None return, import caching, package structure, generators, and assignment expressions (:=).

---

4. Exam L1:P3-OOP Object-Oriented Programming in Modern AI Python
Duration: 2 Hours 30 Minutes | **Questions:** 70

- Class Fundamentals: Covers defining classes, using the __init__ constructor, understanding the self parameter, and differentiating between class and instance attributes.
- Inheritance and Polymorphism: Includes the syntax for inheritance, how method overriding works, using the super() function to access parent methods, and the concept of polymorphism.
- Encapsulation: Tests knowledge of naming conventions for public, protected (single underscore _), and private (double underscore __) members.
- Magic Methods (Dunder Methods): Focuses on methods like __str__, __repr__, __add__, __len__, __eq__, and __call__ to enable built-in functionality for objects.
- OOP Decorators: Examines the use of @property (with setters), @classmethod, and @staticmethod.
- Abstract Base Classes (ABCs): Involves the abc module and the @abstractmethod decorator to create interface-like classes.
- Core OOP Principles: Includes understanding concepts like "composition over inheritance" and SOLID principles such as the Single Responsibility Principle (SRP) and Dependency Inversion Principle (DIP).

- Design Patterns: Covers the purpose of the Singleton and Factory design patterns.
- Python's Data Model: Tests understanding of the "everything is an object" concept and the relationship between type and object.
- Modules and Error Handling: Includes the role of the __init__.py file in packages and basic try...except blocks for handling errors.

---

5. Exam L1:P4-FAI Fundamentals of Agentic AI
Duration: 2 Hours | **Questions:** 50

Prompt Engineering

- Temperature, top_k, and top_p effects
- Safe system messages for sensitive data
- Chain of Thought prompting
- Tree of Thoughts prompting

Markdown

- Clickable images with tooltips
- Numbered and bulleted list formatting

Pydantic

- @pydantic.dataclasses.dataclass vs BaseModel
- Type hints for validation and schema definition
- Using dataclasses as output_type in agents

OpenAI Agents SDK

- General concepts & defaults
- Handoffs (concept, usage, parameters, callbacks)
- Tool calls & error handling during execution
- Dynamic instructions & context objects
- Guardrails (purpose, timing, tripwires)
- Tracing (traces vs spans, multi-run traces)
- Hooks (RunHooks, AgentHooks)
- Exception handling (MaxTurnsExceeded, ModelBehaviorError, etc.)
- Runner methods (run, run_sync, run_streamed) and use cases
- ModelSettings and resolve() method
- output_type behavior and schema strictness

Level 2: Professional Agentic AI Development

Targets advanced proficiency in Python, Agentic AI, AI protocols, Agentic Web, and Building Effective Agents.

1. **Exam L2:P1-PAI Professional Agentic AI Development**
   Duration: 2 Hours 30 Minutes | **Questions:** 60
   - OpenAI Agents SDK core principles and architecture
   - Python-first orchestration and control flow in agent development
   - Agents, Tools, and Handoffs – roles, usage, and design patterns
   - Runner methods (run_sync, run, run_streamed) – behaviors and blocking
   - Pydantic models for input/output validation and default handling
   - @function_tool decorator – usage and schema generation
   - Dynamic instructions using callable functions with context
   - AgentHooks and lifecycle event handling
   - Synchronous vs asynchronous tools integration
   - Input and output guardrails for validation and compliance
   - Error handling during tool execution and error propagation
   - Tool choice and tool use behavior (auto, required, stop_on_first_tool, etc.)
   - Agent cloning vs creating new agents
   - Context management with RunContextWrapper
   - Conversation history handling in multi-agent handoffs
   - Output type parsing and handling invalid or extra fields
   - Prompt engineering techniques (system prompts, dynamic context, CoT)
   - Planning reminders and reasoning in prompts
   - Sensitive data handling and persona-based instruction design
   - Multi-agent workflows and sequential orchestration
   - Access control and permission checks for tools and handoffs
   - Regional compliance filtering in handoffs (e.g., GDPR, US regulations)
   - Feature gating and subscription-based tool access
   - Custom runners with orchestration hooks and context engineering
   - Keyword-based routing limitations in handoffs
   - Audit logging and tool call tracking with tool_call_id
   - Markdown basics (links and images)

---

2. **Exam L2:P2-MCP Model Context Protocol (MCP)**
   Duration: 1 Hour 30 Minutes | **Questions:** 100
   - HTTP – Concepts, request–response cycle, message structure, methods, status codes, and statelessness.
   - REST – Principles (statelessness, client-server separation, cacheability), URI design, best practices, and status code usage.
   - JSON-RPC 2.0 – Requests vs. Notifications, Response/Error objects, reserved method names, parameters, transport-agnostic nature.
   - MCP Fundamental Primitives – MCP architecture, MCP Client & Server roles, message flow (ListTools, CallTool), transport agnosticism.

- Advanced MCP Topics – Sampling, logging, progress notifications, security via roots, transport configurations (e.g., stateless HTTP).
- MCP & OpenAI Agents SDK Integration – Using MCP within OpenAI Agents, tool discovery, prompts, stateless connections, and workflows.

Level 3: Agentic Startups, Agent-Native Cloud, and Advanced Topics

1. **Exam Exam L3:P1-ASF Agentic AI Startup Founder**
   Duration: 4 Hours | **Questions:** 162
   This list outlines the core concepts, strategies, and terminologies you will be tested on. The topics are organized into five main pillars, reflecting the lifecycle of building and scaling an agentic AI startup.

1. Foundations of Innovation in Agentic AI

This section focuses on the integrated methodologies required to build innovative and user-centric AI products.

- **Core Innovation Methodologies**
  - **Design Thinking**: Its role in ensuring desirability by solving real user problems and building trust. The 5 stages (Empathize, Define, Ideate, Prototype, Test) as applied to AI agents.
  - **Lean Startup**: Its role in ensuring viability by validating assumptions through the Build-Measure-Learn loop.
  - **Agile Development**: Its role in ensuring feasibility by providing a framework for iterative and adaptive development of complex AI systems.
- **Integrated Framework**
  - How the three methodologies (Lean, Design Thinking, Agile) combine to reduce uncertainty across desirability, viability, and feasibility.
  - The concept of a **Minimum Viable Agent (MVA)** versus a traditional MVP, focusing on doing one capability extremely well.
- **AI-Specific Development Practices**
  - The expanded **"Definition of Done"** for AI features, including model performance, user testing, and monitoring.
  - Understanding and managing **Technical Debt** in AI, including data debt and model debt.

- The importance of **Red Teaming** to adversarially test for bias, safety, and brittleness.

2. Go-to-Market & Strategic Positioning

This pillar covers how to enter markets, compete effectively, and create defensible advantages.

- **The Piggyback Protocol Pivot (PPP) Strategy**
  - The core problem it solves: overcoming high CAC, domain knowledge barriers, integration complexity, and trust gaps in mature, fragmented markets.
  - **Phase 1 (Piggyback)**: Leveraging existing vendor ecosystems for low-cost market entry and domain learning. The role of the **"Expert-in-the-Middle"** AI agent.
  - **Phase 2 (Pivot)**: Transitioning to an independent, AI-native platform after achieving product-market fit.
  - Core components: **Model Context Protocol (MCP)** for standardization and the **Agentic Layer**.
- **Sales & Product-Led Growth (PLG)**
  - Understanding different GTM models: PLG, Enterprise (Top-Down) Sales, and Open-Source.
  - Key metrics for PLG success, such as **Daily/Monthly Active Users (DAU/MAU)** engagement.
- **Pricing Strategies for AI**
  - Different models: Consumption-Based, Tiered SaaS, and **Value-Based Pricing** tied to ROI.
  - Impact of **inference costs** on pricing and gross margins.
- **Competitive Strategy**
  - How startups can compete with tech giants: focusing on **vertical specialization**, user experience, and agile innovation.
  - Creating defensible moats through **Data Flywheels** and **Network Effects**.

3. Funding & Financial Management

This section covers the financial aspects of launching and scaling an AI startup, from bootstrapping to venture capital.

- **Funding Strategies**
  - **Bootstrapping**: Pros and cons, especially in the context of high compute costs for AI.
  - **Venture Capital (VC)**: Understanding the high-risk, high-reward model.
  - **Alternatives to VC**: Revenue-Based Financing (RBF), Angel Investors, and Grants.
- **The VC Funding Process**
  - **Funding Rounds**: Stages from Pre-Seed to Series A, B, C+ and the typical use of funds at each stage.
  - VC evaluation criteria for agentic AI startups: Market (TAM), Team, Technology, and Traction.
- **Financial Management & Key Metrics**
  - **Unit Economics**: The importance of the **LTV:CAC ratio**, with a healthy target of **3:1 or better**.
  - **Core Financial Terms**: Burn Rate , Runway , COGS (including inference costs), and Gross Margin.
  - **SaaS Metrics**: MRR/ARR , Churn Rate , Net Revenue Retention (NRR) , and the **Rule of 40**.
- **Equity & Legal Instruments**
  - **SAFE (Simple Agreement for Future Equity)**: Its benefits for early-stage startups, such as avoiding immediate valuation.
  - **Equity Terms**: Dilution , Vesting (standard 4-year with 1-year cliff) , Option Pools, Pro Rata Rights , and Protective Provisions.

4. Building the Company: Operations, Talent & Governance

This pillar covers the essential business functions required to build a durable and trustworthy organization.

- **Legal & Corporate Structure**
  - The recommended structure for VC-backed startups: **Delaware C-Corporation**.
  - **Intellectual Property (IP)** strategy: protecting proprietary data, workflows, and models.

- **AI Product & Technology Management**
  - The unique role of an **AI Product Manager**.
  - Operationalizing AI with **Human-in-the-Loop (HITL)** workflows for quality and safety.
  - Understanding and monitoring for **Model Drift**.
- **Talent, Team & Culture**
  - Hiring and retaining scarce AI talent in a competitive market. Compensation ranges for AI researchers and engineers.
  - Non-monetary benefits crucial for retention: research time, conference attendance, and compute resources.
  - High-performance AI team structure: **Research, Engineering, and Product** tracks.
  - Measuring team health with the **Employee Net Promoter Score (eNPS)**.
- **Ethics & Social Responsibility**
  - The importance of conducting ethics reviews at multiple stages: project initiation, development milestones, and pre-launch.
  - Assessing the full **carbon footprint** of AI, including both training and inference.

5. Acronyms and Core Terminology

A significant portion of the exam tests fluency in the language of the startup and AI ecosystem.

- **Business & Finance**: MVP , LTV , CAC , VC , TAM/SAM/SOM , ROI , COGS , MRR/ARR , NRR , SAFE , TCO (Total Cost of Ownership) , IP , RBF.
- **Go-to-Market**: PLG (Product-Led Growth) , SaaS (Software as a Service) , SLA (Service Level Agreement).
- **Product & Operations**: DAU/MAU , MVA (Minimum Viable Agent).

---

2. **Exam L3:P2-DOK Containerizing AI Agents with Docker**
   Duration: 2 Hours | **Questions:** 60
   - Containerization and Docker Fundamentals
   - Building Docker Images for AI Agents
   - Containerizing MCP Servers
   - Managing AI Agent Dependencies
   - Docker Compose for AI Agent Orchestration
   - Deploying AI Agents with Docker
   - Security Best Practices for Agentic Containers
   - Performance Optimization for Agentic Containers
   - CI/CD with Docker
   - Managing Data in AI Containers
   - Monitoring and Debugging Agentic Containers
   - Advanced Docker Concepts for AI
   - Case Studies and Practical Applications
   - Testing and Validation
   - Troubleshooting Common Issues

---

3. **Exam L3:P3-KUB Agent-Native Cloud with Kubernetes**
   Duration: 3 Hours | **Questions:** 100

   - **Kubernetes Fundamentals**
     a. Kubernetes architecture and components (e.g., pods, nodes, clusters, control plane)
     b. Kubernetes objects (e.g., Deployments, Services, ConfigMaps, Secrets)
     c. Container orchestration concepts
   - **Cloud-Native Concepts**
     a. Cloud-native principles and CNCF landscape
     b. Microservices architecture
     c. CI/CD pipelines for cloud-native applications
     d. Observability (monitoring, logging, tracing)
   - **Agent-Native Development**
     a. Role of agents in cloud-native ecosystems
     b. Agent-based automation and management
     c. Integration of agents with Kubernetes (e.g., service meshes, sidecars)
     d. Security considerations for agent-native applications
   - **Container Management**
     a. Container runtimes (e.g., Docker, containerd)
     b. Building and managing container images
     c. Container registries and image security
   - **Kubernetes Networking**
     a. Service discovery and load balancing
     b. Ingress controllers and networking policies
     c. Cluster DNS and CoreDNS
     d. Network plugins (e.g., CNI, Flannel, Calico)

- **Storage in Kubernetes**
    a. Persistent Volumes (PV) and Persistent Volume Claims (PVC)
    b. Storage Classes and dynamic provisioning
    c. State management for stateful applications
- **Kubernetes Security**
    a. Role-Based Access Control (RBAC)
    b. Pod security policies and standards
    c. Secrets management
    d. Network security and policies
- **Deploying and Managing Applications**
    a. Deployment strategies (e.g., rolling updates, blue-green, canary)
    b. Scaling applications (manual and auto-scaling)
    c. Managing application configurations
- **Helm Charts**
    a. Creating and managing Helm charts
    b. Helm repositories and versioning
    c. Customizing deployments with Helm values
    d. Helm lifecycle management (install, upgrade, rollback)
    e. Best practices for Helm chart development
- **Monitoring and Logging**
    a. Kubernetes monitoring tools (e.g., Prometheus, Grafana)
    b. Centralized logging (e.g., Fluentd, Elasticsearch)
    c. Application performance monitoring
- **Troubleshooting and Maintenance**
    a. Debugging Kubernetes clusters and applications
    b. Common failure scenarios and resolutions
    c. Cluster upgrades and maintenance
- **Cloud-Native Tools and Ecosystem**
    a. Service mesh (e.g., Istio, Linkerd)
    b. Serverless frameworks on Kubernetes (e.g., Knative)
    c. Integration with cloud providers (e.g., AWS EKS, Google GKE, Azure AKS)
    d. GitOps and infrastructure-as-code tools (e.g., ArgoCD, Flux)
- **Agent-Native Use Cases**
    a. Real-world applications of agent-native architectures
    b. Case studies on agent-driven automation
    c. Performance optimization with agents in Kubernetes

---

4. Exam L3:P4-DPR Distributed AI Agents with Dapr, Dapr Actors, and Dapr Workflows
Duration: 2 Hours | **Questions:** 60

- **Dapr Fundamentals**
    - Overview of Dapr and its microservices runtime
    - Core building blocks: state management, pub/sub, and service invocation

- **Dapr Actors**
  - Virtual Actor pattern and its implementation in Dapr
  - Actor model: state, behavior, and messaging
  - Actor lifecycle, activation, and deactivation
  - Use cases for Dapr Actors in AI agent systems
- **Dapr Agents**
  - Dapr Agents framework for AI agent development
  - Agent patterns for reasoning, acting, and adapting
  - Integration with Large Language Models (LLMs) (e.g., OpenAI, AWS Bedrock, Anthropic)
- **Dapr Workflows**
  - Workflow concepts and orchestration for AI agents
  - Building resilient, stateful workflows
  - Managing multi-agent coordination and task execution
- **Agent-to-Agent (A2A) Communication**
  - A2A protocols for secure, context-aware collaboration
  - Model Context Protocol (MCP) for standardized tool use
  - Implementing inter-agent communication with Dapr
- **Kubernetes Integration**
  - Deploying Dapr-based AI agents on Kubernetes
  - Kubernetes concepts for AI workloads (e.g., pods, services, scaling)
  - Certified Kubernetes Application Developer (CKAD) preparation basics
- **Scalability and Resilience**
  - Designing scalable AI agent systems with Dapr
  - Handling failures and retries in distributed environments
  - Observability and monitoring with Dapr
- **Self-Hosted Large Language Models (LLMs)**
  - Setting up and managing self-hosted LLMs
  - Scalability and performance considerations for LLMs
  - Fine-tuning LLMs for specific use cases
- **Voice-Enabled AI Agents**
  - Fundamentals of voice interfaces for AI agents
  - Integration and optimization of voice-enabled agents
- **Data-Driven Agents**
  - Connecting agents to databases and unstructured data
  - Data integration strategies for agentic systems
- **DACA Design Pattern**
  - Overview of Dapr Agentic Cloud Ascent (DACA) framework
  - AI-first and cloud-first development principles
  - Stateless containerization for flexible deployment
- **OpenAI Agents SDK**
  - Core features and integration with Dapr
  - Building agent logic with OpenAI SDK
- **Security and Authentication**
  - Implementing Role-Based Access Control (RBAC) in Dapr
  - Secure agent communication and access scopes
- **Deployment and Optimization**

    - Deploying Dapr-based AI agents on cloud platforms (e.g., Azure Container Apps)
    - Optimizing for performance and cost-efficiency
    - Best practices for production-grade deployments

---

5. Exam L3:P5-RAY Distributed AI Computing with Ray
Duration: 3 Hours | **Questions:** 100

- **Introduction to Ray Framework**
    - Overview of Ray as an AI compute engine for scaling Python applications.
    - Key concepts: Distributed runtime, tasks, actors, and objects.
    - Benefits for distributed AI: Scalability, fault tolerance, and integration with ML ecosystems.
- **Ray Core Primitives**
    - Remote functions (tasks): Defining, invoking, and parallelizing with @ray.remote.
    - Actors: Stateful workers, lifecycle management, and use cases in distributed systems.
    - Object store: Sharing immutable data across the cluster, references, and serialization.
- **Cluster Management and Deployment**
    - Setting up and launching Ray clusters (local, cloud: AWS/GCP/Azure, Kubernetes).
    - Resource allocation: CPUs, GPUs, TPUs, and heterogeneous compute.
    - Monitoring and debugging: Ray Dashboard, logs, and distributed debugger.
- **Distributed Data Processing with Ray Datasets**
    - Creating and loading datasets from various sources (e.g., NumPy, files, cloud storage).
    - Transformations: Mapping, filtering, and batch processing at scale.
    - Parallel inference: Applying models over datasets with actors or predictors.
- **Distributed Model Training with Ray Train**
    - Scaling training for frameworks like PyTorch, TensorFlow, XGBoost, and LightGBM.
    - Strategies: Data parallelism, model parallelism, and fault-tolerant training.
    - GPU/CPU configurations and multi-node setups.
- **Hyperparameter Optimization with Ray Tune**
    - Defining search spaces, trials, and schedulers (e.g., ASHA, HyperBand).
    - Distributed tuning: Running multiple trials in parallel.
    - Integration with training libraries and result analysis.
- **Model Serving and Deployment with Ray Serve**

    - Building deployments: Defining replicas, handling requests with FastAPI.
    - Autoscaling and resource management (e.g., GPU support).
    - Serving LLMs and batch inference workflows.
- **Advanced Topics and Best Practices**
    - Fault tolerance: Handling failures in tasks, actors, and clusters.
    - Performance optimization: Scheduling, profiling, and avoiding common pitfalls.
    - Integrations: With tools like DVC, Kubernetes, and cloud services for end-to-end AI pipelines.

---

6. Exam L3:P6-BEA Building Effective Agents
Duration: 3 Hours | **Questions:** 120

- Workflows and Agents – differences, trade-offs, and use cases.
- LLM Augmentation – retrieval, tools, and memory.
- Agentic Design Patterns – routing, prompt chaining, parallelization, orchestrator-worker, evaluator-optimizer (reflection).
- Agentic Memory – semantic, episodic, procedural, and short-term memory; knowledge graphs, vector databases, temporal knowledge graphs.
- Neo4j AuraDB & Knowledge Graph – Cypher basics, MERGE, MATCH, SET, relationships, and querying.
- Graphiti – temporal knowledge graphs, episodes, fact triples, CRUD operations, hybrid search, communities, namespacing.
- Augmentation Retrieval – vector embeddings, vector databases, traditional vs. agentic RAG, function calling, data freshness.
- Agentic Payments & Economy – agentic economy concepts, APIs as tools, Stripe MCP integration, security/trust challenges.

---

7. Exam L3:P7-AGW Agentic Web
Duration: 2 Hours | **Questions:** 100

- Agentic Web Concepts
- Agent Attention Economy
- Agent-to-Agent (A2A) protocol

---

8. Exam L3:P8-AMP Advanced Modern AI Python
Duration: 2 Hours 30 Minutes | **Questions:** 50

- **Advanced Static Typing**
    - Type hints, generics, variance (invariance, covariance)
    - Structural subtyping with typing.Protocol
    - Mypy and Pyright usage

- **Abstract Base Classes (ABCs)**
  - Defining and implementing abstract methods
  - Interface conformance checks
- **Asynchronous Programming (asyncio)**
  - Coroutines and event loops
  - Concurrency patterns (asyncio.gather, asyncio.wait_for)
  - I/O-bound vs CPU-bound considerations
- **Object-Oriented Programming (OOP)**
  - Multiple inheritance and Method Resolution Order (MRO)
  - Composition vs aggregation
  - Duck typing
- **Modern Python Libraries**
  - Pydantic v2 (data validation, type coercion, field aliases)
  - Dataclasses (immutable classes, auto-generated methods)
- **CPython and the Global Interpreter Lock (GIL)**
  - Implications for concurrency and threading
- **Core Python Features**
  - *args and **kwargs usage and type hints
  - .pyc files and bytecode
  - Protocols vs duck typing in runtime and static checks
- **Miscellaneous**
  - Method overriding rules and return type consistency
  - Type checker behavior with collections (list, tuple)
  - Forward references in type hints
  - Static typing pitfalls in reassignment and mutability

Level 4: Professional Physical AI Development and Hardware Integration

Focuses on physical AI and robotics, integrating AI with hardware systems using cutting-edge NVIDIA technologies. (Under Development)

1. NVIDIA Isaac ROS for robotic operating systems
2. NVIDIA Isaac GR00T for general-purpose robotic intelligence
3. NVIDIA Isaac Sim for robotic simulation
4. Real-world applications of physical AI in robotics

Agentic AI Course Catalog

These courses are designed to prepare students and professionals for the certification exams.

100-Level Courses

AI-101: Low Code n8n Agentic AI Development & Modern Python Programming
AI-101 serves as a comprehensive gateway to Python programming for Artificial Intelligence and Low Code Agentic AI Development and prepares you for **L1:P1-N8N** and **L1:P2-FMP** certifications. This foundational course emphasizes modern Python programming skills with static typing—a cornerstone of robust, scalable AI projects. You'll design robust n8n workflows, and orchestrate plan-act-observe agent loops that call tools, keep memory, and respect guardrails. The curriculum spans foundational agentic concepts through advanced techniques including Model Context Protocol (MCP) and Retrieval-Augmented Generation (RAG), and deployment (self-hosted or cloud).

200-Level Courses

AI-201: Fundamentals of Agentic AI

AI-201 introduces students to the core principles and practices of Agentic AI development. The course explores foundational theories underlying intelligent agent behaviour and provides extensive hands-on experience developing context-aware AI agents using the OpenAI Agents SDK. Through a balanced approach combining theoretical grounding with practical implementation projects, students will develop the expertise necessary to design and deploy functional multi-agent systems. This course prepares you for **L1:P3-OOP, L1:P4-FAI** and **L2:P1-PAI** certifications.

AI-210: MCP and Building Effective Agents

Building upon the foundation established in AI-201, this course introduces advanced Agentic AI concepts with particular focus on Model Context Protocol (MCP), Agentic Memory, and Agentic RAG. Students will study established agentic design patterns for building effective agents.

AI-220: Agentic Web

AI-220 focuses on Agentic Web theory and the practical implementation of Agent-to-Agent (A2A) Protocol systems. Students will explore how intelligent agents interact across web-based environments, share contextual information, and collaborate effectively on distributed computational tasks. The course examines core design principles for building web-native, cooperative AI systems and demonstrates

how A2A protocols enable real-time communication between agents operating in diverse environments.

300-Level Courses

AI-301: Agent Native Cloud Development

This advanced course focuses on cloud-first development methodologies for Agentic AI systems. Students will explore scalable, distributed AI architectures utilizing industry-standard containerization and orchestration technologies including Docker and Kubernetes for deploying AI Agents and MCP Servers. Through comprehensive hands-on projects, students will design and deploy production-ready AI applications optimized for cloud environments.

AI-310: Planet-Scale Distributed AI Agents

The capstone course in distributed AI systems covers enterprise-grade distributed application runtime environments, focusing on Distributed Application Runtime (Dapr) implementation alongside managed database systems and messaging architectures. Students will design and implement planet-scale AI agent networks capable of operating across global distributed infrastructure.

400-Level Courses

AI-451: Physical and Humanoid Robotics AI

Artificial intelligence (AI) has experienced remarkable advancements in recent years. However, the future of AI extends beyond the digital space into the physical world, driven by robotics. This new frontier, known as “Physical AI,” involves AI systems that can function in the real world and comprehend physical laws. This marks a notable transition from AI models confined to digital environments. Humanoid robots are poised to excel in our human-centred world because they share our physical form and can be trained with abundant data from interacting in human environments.

This course provides an in-depth exploration of humanoid robotics, focusing on the integration of ROS 2 (Robot Operating System), Gazebo Robot Simulator, and NVIDIA Isaac™ AI robot development platform. Students will learn to design, simulate, and deploy advanced humanoid robots capable of natural interactions. The curriculum covers essential topics such as ROS 2 for robotic control, simulations with Gazebo and Unity, and using OpenAI’s GPT models for conversational AI. Through practical projects and real-world applications, students will develop the skills needed to drive innovation in humanoid robotics.

Appendix:

The Reality of the AI-powered Era: The Future Belongs to Those Who Master AI

Companies are willing to spend fortunes for elite AI minds.
One exceptional individual can create billions in value. Read these stories to understand the potential of AI-powered careers.

1. https://nypost.com/2025/08/01/business/meta-pays-250m-to-lure-24-year-old-ai-whiz-kid-we-have-reached-the-climax-of-revenge-of-the-nerds/
   ↳ Matt Deitke, a PhD dropout
   ↳ Initial offer: $125M (declined as too low)
   ↳ Zuckerberg counters: $250M
   ↳ $100M in the first year alone
2. https://gulfnews.com/technology/who-is-andrew-tulloch-meet-the-man-who-rejected-marck-zuckerbergs-15b-job-offer-1.500223824
   ↳ Andrew Tulloch, co-founder of Thinking Machines Lab
   ↳ Meta dangled a six-year, **$1.5 B** pay package
   ↳ Tulloch flat-out **rejected** Zuckerberg's offer
   ↳ Came right after Meta's failed **$1 B** bid for his startup
3. https://www.calcalistech.com/ctechnews/article/im5ma59bu
   ↳ **Prompt Security**, a two-year-old GenAI-cyber startup
   ↳ Snapped up by SentinelOne for **$250 M** cash deal
   ↳ Had raised only **$23 M** before the buyout
   ↳ Founded by 8200-unit vets Itamar Golan & Lior Drihem
4. https://www.reuters.com/technology/clay-valued-31-billion-latest-fundraise-ai-continues-run-hot-2025-08-05/
   ↳ **Clay**, an AI sales-automation platformReuters
   ↳ Raises **$100 M** led by Alphabet's CapitalGReuters
   ↳ Valuation rockets to **$3.1 B** (up from $1.5 B three months ago)Reuters
   ↳ Customer roster already includes Google & RedditReuters
5. https://www.reuters.com/business/google-hires-windsurf-ceo-researchers-advance-ai-ambitions-2025-07-11/
   ↳ Google pays a **$2.4 B** license fee to Windsurf
   ↳ CEO Varun Mohan & R&D team join Google DeepMind
   ↳ Move follows OpenAI's abortive **$3 B** takeover talks
   ↳ Most of Windsurf's 250 staff stay; investors keep stakes
6. https://www.theguardian.com/technology/2025/aug/06/openai-chatgpt-talks-share-sale-price-more-than-musk-spacex
   ↳ OpenAI negotiating an internal **share-sale** round
   ↳ Would value the firm at **$500 B**, topping SpaceX
   ↳ Jump of **~67 %** from its prior $300 B mark
   ↳ Seen as lever to retain talent amid Meta's **$100 M** bonuses

7. https://www.nytimes.com/2025/08/04/technology/ai-young-ceos-san-francisco.html
   ↳ Wave of 20-something AI founders is flocking to San Francisco, saying they can't "afford to wait" before launching A.I. start-ups
   ↳ Headliners include Scott Wu (Cognition AI), Michael Truell (Cursor), Roy Lee (Cluely) and Alexandr Wang (Scale AI)
   ↳ Wu's Cognition is so intense it offers nine-month buyouts to staff unwilling to adopt an 80-hour-week "extreme performance culture"
   ↳ Truell's Cursor has hit $300 M ARR in <3 years, Lee's Cluely just raised $15 M led by a16z, and 28-year-old Wang is already Meta's chief AI officer and multi-billionaire