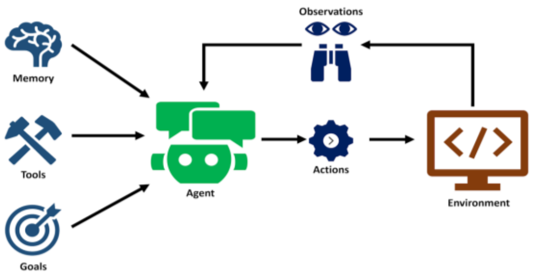
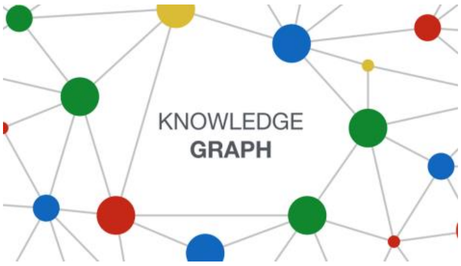
Certified Agentic and Robotic AI Engineer

Master the Future: Getting You Ready For The $100 Trillion AI Industrial Revolution

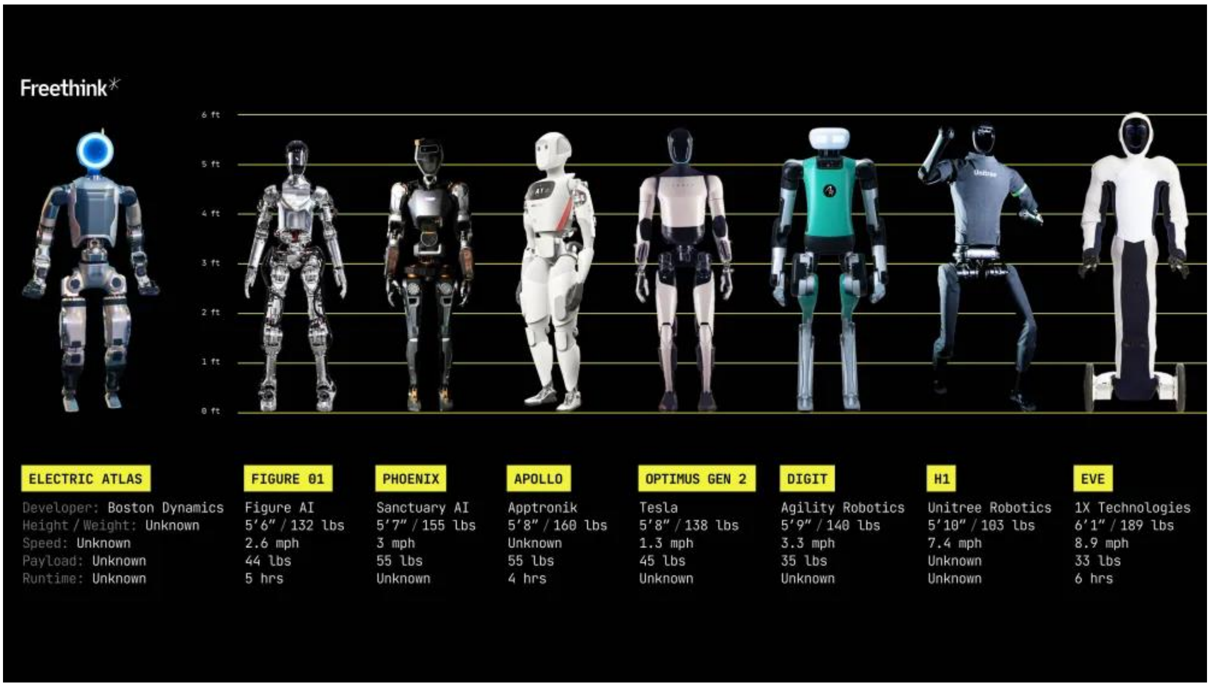


Agentic AI



Graph Databases/GQL

Humanoids



ELECTRIC ATLAS	FIGURE 01	PHOENIX	APOLLO	OPTIMUS GEN 2	DIGIT	H1	EVE
Developer: Boston Dynamics	Figure AI	Sanctuary AI	Apptronik	Tesla	Agility Robotics	Unitree Robotics	1X Technologies
Height / Weight: Unknown	5’6” / 132 lbs	5’7” / 155 lbs	5’8” / 160 lbs	5’8” / 138 lbs	5’9” / 140 lbs	5’10” / 103 lbs	6’1” / 189 lbs
Speed: Unknown	2.6 mph	3 mph	Unknown	1.3 mph	3.3 mph	7.4 mph	8.9 mph
Payload: Unknown	44 lbs	55 lbs	55 lbs	45 lbs	35 lbs	Unknown	33 lbs
Runtime: Unknown	5 hrs	Unknown	4 hrs	Unknown	Unknown	Unknown	6 hrs

Learn to Build Autonomous AI Agents, Humanoids, and Fine-Tune LLMs

Version: 16.1 (Implementation and adoption starting from January, 2025)

Must Watch Video To Get Started: Customer Experience Trends for 2025: The Rise of AI Agents and Agentic AI
Watch the Rise of Agentic AI Video Presentation:
https://www.facebook.com/share/v/1Q5ZmFBx7u/
Note: If you have difficulty in watching this video install Opera Browser after you have activated its builtin VPN
The Rise of the Agentic AI Slides: https://bit.ly/4hTqT4G
The Certification Discussion Podcast: https://youtu.be/ViRWA4wLI8k
Detailed Class Schedule for this Quarter: https://bit.ly/piaic-dec-sch
Certification Program Review by ChatGPT:
https://chatgpt.com/share/6732a6f1-a3c4-8001-99cb-1b272c3b3881

The leading technological trends today include Agentic AI, Physical AI, Knowledge Graphs, and Cloud Native and distributed computing technologies. Agentic AI refers to AI systems designed to autonomously perceive, reason, and act to achieve specific objectives, often through iterative decision-making and learning. Agents can be either software-based or robotic. Our initial focus is on software agents, such as Vertical LLM Agents and the development and deployment of SaaS applications. Eventually, we'll turn our attention to humanoid robots and Physical AI, aiming to bridge the gap between digital intelligence and physical capabilities by creating systems that understand and interact with the world in a human-like manner. Cloud Native technology provides a scalable and reliable platform for running applications, while AI imbues these applications with intelligent, human-like features. A knowledge graph is a structured representation of real-world entities and their relationships, organised as nodes (entities) and edges (relationships) in a graph format. This enables both humans and machines to understand, integrate, and reason about complex, interconnected data from various sources. Our goal is to train you to become an exceptional global developer in Cloud Native, Distributed Computing, Knowledge Graphs, Agentic AI, and Physical AI.

Material to Understand the Coming Agentic AI Age:

- Agentic AI Explained
- AI Agents Explained Like You're 5
- AI Is About To FLIP Your Life Upside Down
- The Future Is Agentic
- The agent economy
- Why Vertical LLM Agents Are The New $1 Billion SaaS Opportunities
- Vertical AI Agents Could Be 10X Bigger Than SaaS
- OpenAI's Path to AGI | Five Levels of Intelligence
- AI Agents: Are We Ready For Machines That Make Decisions?
- Function calling

- Generative AI's Act o1
- Watch AGI could Double GDP
- The INSANE Race for AI Humanoid Robots
- **The AI agents stack**

This core program duration is one and a half years, if you take one course at a time and equips you with the skills to thrive in the age of Generative, Agentic, and Physical AI, and cloud native distributed computing. However, **you can reduce the duration of the program if you take multiple courses in a quarter**. You will become an expert Custom GPT, AI Agent, and Humanoid Robotics Developer. The program is divided into two levels: core level and professional level. **Students will be able to start working after completing the core level. They will continue their professional level studies while working.**

Why This Program?

- **Cutting-Edge Skills:** Develop in-demand skills to build intelligent, scalable cloud applications using Generative AI and Cloud Native technologies.
- **Industry-Ready:** Prepare for global certifications, startup and freelance opportunities after just six months.
- **Future-Proof Your Career:** Stay ahead of the curve in a rapidly evolving tech landscape.

What You'll Learn:

- **Multi AI Agent Systems and Custom GPTs:** Learn to fine-tuning foundational AI models, and market them in GPT stores. Learn key principles of designing effective AI agents, and organising a team of AI agents to perform complex, multi-step tasks. Build Knowledge Graphs. Apply these concepts to automate common business processes.
- **Physical AI and Humanoid Robotics**: We will learn to design, simulate, and deploy advanced humanoid robots capable of natural interactions.
- **Develop AI Powered Microservices:** Master Python, build APIs using FastAPI, SQLModel, Postgres, Kafka, Kong, and leverage cutting-edge GenAI APIs like OpenAI, and Open Source AI LLMs.
- **Cloud Native Expertise:** Design and deploy cloud-native pipelines, and microservices using Docker, DevContainers, TestContainers, Kubernetes, Terraform, and GitHub Actions.
- **Distributed System Design**: Designing systems that run on multiple computers (or nodes) simultaneously, interacting and coordinating their actions using Ray.
- **Designing AI Solutions using Design Thinking and Behaviour Driven Development (BDD)**: We will learn to leverage these methodologies to create AI solutions that are not only technically sound but also highly user-centric and aligned with real-world needs.

- **Fine-Tuning Open-Source Large Language Models using PyTorch, and Fast AI**: We will learn to fine-tuning of open-source Large Language Models (LLMs) like Meta LLaMA 3 using PyTorch, Ray, and Fast AI, with a focus on cloud-native training and deployment. We will set up development environments, preprocess data, fine-tune models, and deploy them using cloud native platforms.

Flexible Learning:

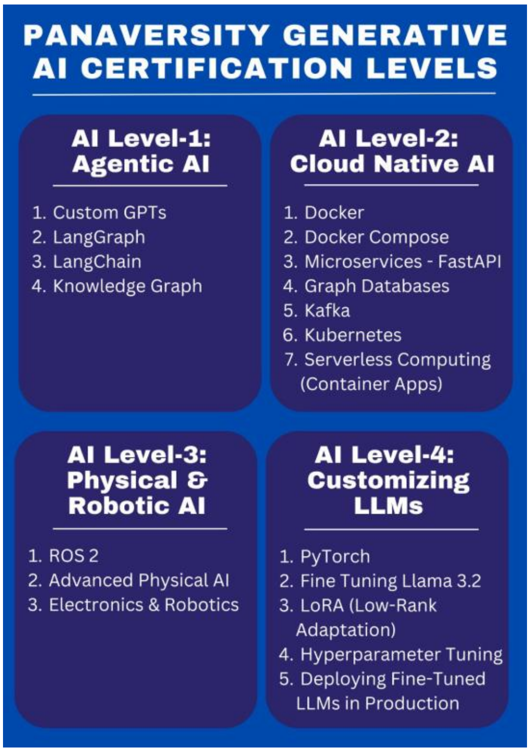
- **Earn While You Learn:** Start freelancing or contributing to projects after the third quarter.

Also Focus on Communication Skills:

- **Technical + Communication:** You can negotiate any reality you want

The Level-Based Structure of the Panaversity AI Program::

Our program's level structure transforms the learning process into a progressive journey. The levels are designed to ensure that participants thoroughly grasp and demonstrate proficiency in each stage before advancing to the next. The process starts by learning Python AI and progressively moves from Agentic AI, to Cloud Native AI, then to Physical and Robotic AI, culminating in Customizing LLMs. Each level builds on the knowledge and skills from the previous one, increasing in complexity and depth.



PANAVERSITY GENERATIVE AI CERTIFICATION LEVELS

AI Level-1: Agentic AI

1. Custom GPTs
2. LangGraph
3. LangChain
4. Knowledge Graph

AI Level-2: Cloud Native AI

1. Docker
2. Docker Compose
3. Microservices - FastAPI
4. Graph Databases
5. Kafka
6. Kubernetes
7. Serverless Computing (Container Apps)

AI Level-3: Physical & Robotic AI

1. ROS 2
2. Advanced Physical AI
3. Electronics & Robotics

AI Level-4: Customizing LLMs

1. PyTorch
2. Fine Tuning Llama 3.2
3. LoRA (Low-Rank Adaptation)
4. Hyperparameter Tuning
5. Deploying Fine-Tuned LLMs in Production

I. Autonomous Agentic and Robotic AI Core Level

- **AI-101: Modern AI Python Programming**

  The main focus in this course will be on mastering the fundamentals of Modern Python with Typing, the go-to language for AI and using AI to write Python Programs. We will then move to understanding the basics of GenAI and Prompt Engineering. In the end of the program we will understand the basics of Linux, Docker, VSCode, Devcontainer, and GitHub.

  - **Certification:**
    - Certified Professional Python Programmer (CPPP1)

  Learning Repo: https://github.com/panaversity/learn-cloud-native-modern-python

  Prerequisite: None

- **AI-201: Fundamentals of Agentic AI**

  In this quarter, students will embark on a comprehensive journey into the realms of Generative AI and Agentic AI. The curriculum begins with an introduction to these foundational concepts, establishing a solid understanding of their principles and distinctions. Students will then engage in hands-on development of custom GPTs, facilitating a deeper grasp of AI functionalities and applications. A significant portion of the quarter is dedicated to mastering Prompt Engineering, emphasising the creation of effective prompts to optimise AI outputs. Leveraging the user-friendly CrewAI framework, students will develop AI agents, applying their theoretical knowledge to practical scenarios. The course also delves into the construction of Knowledge Graphs using Graph Databases and GQL, equipping students with the skills to organise and query complex data structures. Additionally, the semester covers Agentic Payments, exploring the integration of AI agents in financial transactions and payment systems.

  Learning Repo:

  https://github.com/panaversity/learn-agentic-ai/

  Covers Chapters from -01 to 11 or the Repo

  Prerequisite: AI-101

- **AI-202: Advanced Agentic AI Engineering**

Building upon the foundational knowledge acquired in AI-201, this quarter focuses on advanced Agentic AI engineering using more sophisticated frameworks. Students will delve into LangGraph and LangChain, gaining proficiency in these powerful tools to create complex AI agents. The curriculum emphasises the development of sophisticated AI agents capable of performing intricate tasks and decision-making processes. Through a combination of theoretical instruction and practical projects, students will enhance their capabilities in designing, implementing, and deploying advanced AI solutions, preparing them for challenges in the evolving field of Agentic AI.

The course will then transition to frontend development, where learners will be introduced to Next.js and its powerful features for building dynamic web applications. By incorporating TypeScript, participants will enhance their development process with strong typing and advanced features. This part concludes with a project, allowing learners to develop a complete AI agent frontend, combining their Knowledge Graphs with modern web technologies for a seamless user experience.

Learning Repo:

https://github.com/panaversity/learn-agentic-ai/

Covers Chapters 12 and onwards

Prerequisite: AI-101, AI-201

- **AI-301: Cloud Native AI Microservices**

  Build scalable AI Powered APIs using FastAPI, GQL, Neo4j, Kafka, Kong, GenAI APIs like OpenAI Chat Completion APIs, Assistant APIs, LangChain and Open Source AI LLMs, develop them using Containers and Dev Containers, and deploy them using Docker Compose locally and Kubernetes Powered Serverless Container Services on the cloud.

  We will also learn to integrate design thinking and Behavior-Driven Development (BDD) in developing AI systems. We will learn to create AI solutions that are deeply aligned with user needs and expectations. Design thinking ensures a thorough understanding of the user and problem space, while BDD provides a structured approach to defining and validating the desired behaviours of the AI system. Together, these methodologies lead to the development of AI solutions that are not only technically robust but also highly user-centric and effective in solving real-world problems.

  - **Certifications:**
    - Neo4j Certified Professional

        - [Confluent Certified Developer for Apache Kafka (CCDAK)]
        - [Design Thinking Professional Certificate (DTPC)]
        - [Test and Behavior Driven Development (TDD/BDD)]

  Learning Repo:

  https://github.com/panaversity/learn-cloud-native-ai-powered-microservices/

  We Will Be Using Microsoft Azure as our Default Cloud Platform
  Amazon is still the cloud king based on market share. But many analysts agree: In the battle for the cloud, AI is now a game-changer — and Amazon's main competitors, particularly Microsoft, have the momentum. In our program we will be using Azure as our default provider for teaching and deployment.

  Prerequisite: AI-101

- **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.

  Learning Repo: https://github.com/panaversity/learn-physical-ai-humanoid-robotics

  Prerequisite: AI-101

- **AI-461: Distributed AI Computing**

Ray is the AI Compute Engine. Ray manages, executes, and optimises compute needs across AI workloads. It unifies infrastructure via a single, flexible framework—enabling any AI workload from data processing to model training to model serving and beyond. This course provides an in-depth exploration of distributed computing using Ray, a framework for building and scaling distributed Python applications. Students will learn to develop, deploy, and optimise distributed systems using Ray, with applications in machine learning, data processing, and reinforcement learning. Ray is an open-source distributed computing framework designed to simplify the development and scaling of machine learning (ML) and Python applications.

The following examples highlight Ray's widespread adoption and its effectiveness in enhancing scalability, performance, and cost-efficiency in AI and machine learning workloads.

- OpenAI: Utilises Ray to train large models, including ChatGPT, enabling faster iteration at scale.
- Amazon Web Services (AWS): Employs Ray to enhance scalability, reduce latency by over 90%, and improve cost efficiency by over 90% in specific applications.
- Ant Group: Deployed Ray Serve on 240,000 cores for model serving, achieving a peak throughput of 1.37 million transactions per second during high-demand periods.
- Uber: Leverages Ray to rapidly pretrain, fine-tune, and evaluate large language models (LLMs).
- Instacart: Uses Ray to run deep learning workloads 12 times faster, reduce costs by 8 times, and train models on 100 times more data. (Ray)
- Samsara: Implemented Ray to scale the training of deep learning models to hundreds of millions of inputs, accelerating deployment and cutting inference costs by 50%.
- Cohere: Utilises Ray to simplify the development of scalable distributed programs for large language model pipelines. (Ray)

Prerequisite: AI-101

II. Professional Level

- **AI-500: AI Ethics and Governance: Principles and Practices**

In the rapidly evolving field of artificial intelligence, understanding the ethical implications and governance frameworks is crucial. This course delves into the core principles of AI ethics, including fairness, transparency, accountability, and privacy. Participants will explore the societal impacts of AI across various sectors, examine international and national governance structures, and learn strategies for integrating ethical considerations into AI design and deployment. Through case studies and a capstone project, students will gain practical skills to navigate and address ethical challenges in AI, preparing them to lead responsible AI initiatives in their organisations.

- **AI-501: Distributed Machine Learning**

  Generative AI tools like ChatGPT, Gemini, and DALL-E have revolutionised our professional landscape. This hands-on course guides you through the exciting distributed process of building and training AI models using Python and the versatile, open-source PyTorch and Ray frameworks. You'll delve into the core concepts of Generative Adversarial Networks (GANs), Transformers, Large Language Models (LLMs), variational autoencoders, diffusion models, and more. Along the way, you'll gain practical experience and a deep understanding of these cutting-edge technologies.

  Learning Repo: https://github.com/panaversity/genai-with-pytorch

  Prerequisite: AI-101, AI-461

- **AI-502: Customising Open Source LLMs**

  This comprehensive course is designed to guide learners through the process of fine-tuning open-source Large Language Models (LLMs) such as Meta LLaMA 3 using PyTorch, with a particular emphasis on cloud-native training and deployment. The course covers everything from the fundamentals to advanced concepts, ensuring students acquire both theoretical knowledge and practical skills.

  The journey begins with an introduction to LLMs, focusing on their architecture, capabilities, and the specific features of Meta LLaMA 3. Next, the course dives into PyTorch fundamentals, teaching students how to perform basic operations with tensors and build simple neural networks. This foundation is crucial for understanding the mechanics behind LLMs. Data preparation is a crucial aspect of training models. The course covers comprehensive data collection and preprocessing techniques, such as tokenization and text normalisation. These steps are essential for preparing datasets suitable for fine-tuning LLMs like Meta LLaMA 3. Through practical exercises, students learn how to handle and preprocess various types of text data, ensuring they can prepare their datasets for optimal model performance.

Fine-tuning Meta LLaMA 3.2 with PyTorch forms a significant part of the course. Students will delve into the architecture of Meta LLaMA 3, learn how to load pre-trained models, and apply fine-tuning techniques. The course covers advanced topics such as regularisation and optimization strategies to enhance model performance. Practical sessions guide students through the entire fine-tuning process on custom datasets, emphasising best practices and troubleshooting techniques.

A critical aspect of this course is its focus on cloud-native training and deployment using Nvidia NIM. Furthermore, students learn how to deploy models using Docker and Kubernetes, set up monitoring and maintenance tools, and ensure their models are scalable and efficient.

To round off the learning experience, the course includes an in-depth segment on exporting models for inference and building robust inference pipelines. Students will deploy models on cloud platforms, focusing on practical aspects of setting up monitoring tools to maintain model performance and reliability.

The course culminates in a capstone project, where students apply all the skills they have learned to fine-tune and deploy Meta LLaMA 3 on a chosen platform. This project allows students to demonstrate their understanding and proficiency in the entire process, from data preparation to cloud-native deployment.

Learning Repo:

https://github.com/panaversity/learn-fine-tuning-llms

Prerequisite: AI-101, AI-461, AI-501

- **AI-651: Advanced Cloud Native and Distributed AI Computing**

  Master Kubernetes, Ray, Terraform, and GitHub Actions to deploy your AI pipelines, APIs, microservices, and open source models in the cloud. We will cover distributed system design involving creating AI systems that are distributed across multiple nodes, focusing on scalability, fault tolerance, consistency, availability, and partition tolerance.

  Certifications:

  - Certified Kubernetes Application Developer (CKAD)
  - HashiCorp Certified: Terraform Associate

  Learning Repo: https://github.com/panaversity/learn-kubernetes

  Prerequisite: AI-101, AI-301, AI-461

III. Vertical Specialization Level (Optional)

Students will have the option of selecting one of the following specialisations, details available at the end of FAQs:

1. **AI-701: Healthcare and Medical Agentic AI**
2. **AI-702: Web3, Blockchain, and Agentic AIIntegration**
3. **AI-703: Metaverse, 3D, and Agentic AI Integration**
4. **AI-704: Agentic AI for Accounting, Finance, and Banking**
5. **AI-705: Agentic AI for Engineers**
6. **AI-707: Agentic AI for Sales and Marketing Specialization**
7. **AI-708: Agentic AI for Automation and Internet of Things (IoT)**
8. **AI-709: Agentic AI for Cyber Security**

Zia Khan, CEO Panaversity
MSE, MBA, MAC, MA, CPA, CMA
https://www.linkedin.com/in/ziaukhan/

Common Questions (FAQs) with Detailed Answers

1. **What is a Certified Agentic and Robotic AI Engineer?**
   A Certified Agentic and Robotic AI Engineer is a professional skilled in the development of autonomous, AI-driven systems that can act and make decisions independently (agentic AI) and physical systems that interact with the physical world (robotic or physical AI). This certification program, offered by Panaversity, equips participants with expertise in building and deploying autonomous software agents, humanoid robots, and fine-tuning large language models (LLMs) for specific applications.

   The program covers skills across several domains:

   - **Agentic AI**: Focus on AI systems that can autonomously learn, perceive, reason, and act, including multi-agent AI systems, AI-powered SaaS solutions, and knowledge graphs.
   - **Humanoid Robotics and Physical AI**: Training on designing, simulating, and deploying robots capable of interacting with humans naturally using platforms like ROS 2 and NVIDIA Isaac.
   - **Cloud Native and Distributed Computing**: Includes building scalable AI-powered microservices, leveraging Docker, Kubernetes, and Ray for cloud-native and distributed applications.
   - **Custom GPT and LLM Fine-Tuning**: Students learn to customize LLMs, using tools like PyTorch and FastAI, for specific applications.

- **AI Ethics and Governance**: To prepare participants for real-world deployment, the program includes principles and practices for responsible AI use.

This comprehensive curriculum enables students to become leaders in the $100 trillion AI-driven industrial revolution, blending technical prowess with ethical awareness and a future-focused approach to the development and application of AI in both virtual and physical domains￼.

2. **How valuable can the Certified Agentic and Robotic AI Engineer be in the new age of AI?**
The Certified Agentic and Robotic AI Engineer certification positions professionals at the forefront of the rapidly expanding AI industry, bridging critical skills in autonomous agents, humanoid robotics, and applied generative AI. Here's why it's especially valuable in the new AI age:

a. High Demand for AI Talent in Emerging Fields

With the rise of agentic AI (systems that can autonomously act and make decisions) and physical AI (robots capable of interacting in human environments), industries from healthcare to manufacturing, finance, and more require specialised professionals who can design, develop, and deploy these solutions. The certification addresses these needs by training engineers to implement cutting-edge AI in real-world applications.

b. Market Need for Multi-Disciplinary AI Engineers

This program combines expertise in cloud-native distributed computing, AI ethics, and generative AI models with practical robotic engineering. Such a multidisciplinary skill set is rare and in high demand, especially as industries increasingly integrate AI into complex systems that require intelligent automation and decision-making capabilities.

c. Potential for Lucrative Careers and Entrepreneurship

The certification prepares professionals to either secure high-paying roles or start their own ventures, given the growing demand for AI-driven applications and the ability to deploy them effectively. With cloud-native and agentic AI skills, certified engineers can create specialised applications, SaaS products, or even bespoke AI agents tailored for specific sectors, all of which are lucrative markets. The annual salary for AI professionals, especially those with expertise in cloud-native and generative AI, can range from $150,000 to over $200,000, depending on experience and region ￼.

d. Future-Proofing with Robotics and Physical AI

Robotics, especially humanoid robots powered by AI, are expected to become more prevalent in the workforce, from automated service roles to advanced manufacturing and healthcare. This program’s focus on humanoid robotics, ROS 2, and NVIDIA Isaac platforms enables graduates to contribute to this pivotal transition, making them invaluable as more businesses adopt physical AI solutions.

e. Aligning with the Fourth Industrial Revolution

AI, especially generative and agentic AI, is part of what some are calling the Fourth Industrial Revolution—a shift expected to transform industries at a $100 trillion scale by 2030. Certified engineers from this program are positioned to drive innovations that align with this transformation, making their skills and knowledge a vital component of the global economy’s future.

In short, the Certified Agentic and Robotic AI Engineer is a credential that prepares professionals not only for immediate high-value opportunities but also for leadership roles in AI’s next evolution. This makes it a highly valuable certification as AI expands its impact across virtually every industry ￼.

3. What is the potential for Certified Agentic and Robotic AI Engineers to start their own companies and become successful startup founders?

The Certified Agentic and Robotic AI Engineer credential equips professionals with unique skills that position them as prime candidates for launching successful startups in the AI-driven market. Here’s why certified professionals have a strong potential for entrepreneurial success:

a. Emerging Market Demand for Autonomous Solutions

With AI agents, robotic systems, and generative AI applications rapidly becoming integral across industries, there is substantial demand for startups that can deliver these technologies in tailored, high-impact solutions. Certified engineers have the knowledge to develop autonomous agents and AI systems for niche markets, such as healthcare, finance, logistics, or personal assistance, where custom solutions can command premium pricing.

b. Leverage Agentic AI and Physical AI Skills for Differentiation

The certification program’s dual focus on agentic AI (autonomous software agents) and physical AI (robots capable of real-world interactions) provides a distinct edge in the market. Founders with these skills can differentiate themselves by creating products that bridge digital and physical experiences—such as customer service bots that interact seamlessly with customers or humanoid robots designed for human environments.

c. Versatile Skills for Lean Startup Operations

Certified engineers are trained across the full stack of AI-powered applications, including cloud-native architectures, distributed computing, custom GPTs, and robotics, which allows them to operate lean, cost-effective startups. These technical skills mean they can handle a significant portion of the development themselves, reducing reliance on external resources and allowing for faster prototyping and iterative product development.

d. Access to Growing Funding Opportunities for AI and Robotics

Venture capital and other funding sources are increasingly directed toward startups that leverage AI and robotics, as these areas are viewed as high-growth, high-impact fields. Certified engineers have expertise in trending fields such as multi-agent AI systems, AI ethics, and humanoid robotics, making their startups attractive to investors looking to fund cutting-edge AI applications and transformative robotics technologies.

e. Built-In Flexibility to Adapt to Market Needs

The certification’s breadth—including cloud-native deployment, knowledge graphs, API-as-a-Product, and custom AI agent creation—enables entrepreneurs to pivot based on market feedback. For instance, if there’s a demand for SaaS AI agents in customer service, founders can adapt their technical skills to meet this need, quickly creating and deploying new solutions that align with market trends.

f. Scalability and Innovation Using AI-Driven Models

With training in scalable cloud-native development and advanced knowledge of distributed computing tools like Ray and Kubernetes, certified engineers can build and grow their startups to serve a global customer base. They are equipped to create robust AI systems that can be easily scaled, attracting a larger client base and enhancing profitability.

g. Lowered Barriers for Entry in GenAI and Robotics

Traditionally, robotics and AI development were resource-intensive fields. However, cloud platforms, generative AI APIs, and containerized deployment options (skills certified engineers possess) significantly lower these barriers. This means startup founders can deploy cutting-edge AI without needing the massive infrastructure once required, allowing for quicker entry and competitive advantages in the market.

h. Focus on Long-Term Industry Transformations

With a solid grounding in industry applications, certified engineers understand how to leverage AI technologies to drive significant efficiencies, improve user experiences, and automate complex processes. This is particularly valuable in verticals like finance, healthcare, IoT, and cyber security, where AI-driven startups can introduce transformative solutions and redefine industry standards.

In conclusion, Certified Agentic and Robotic AI Engineers have a rare combination of skills, technical flexibility, and industry insight that makes them well-suited for successful entrepreneurship. By leveraging their expertise in agentic and physical AI, cloud-native computing, and multi-agent systems, they are poised to create startups that meet the modern market’s demand for autonomous, intelligent, and scalable solutions. Their potential to succeed as founders is bolstered by both the technical depth of their skills and the strong demand for innovative AI applications ￼.

4. **Is the core program not too long, one and half years is a long time?**
   The length of the program is 18 months which is broken down into six quarters of three months each. However, if students want to shorten the duration of the program they can take two or more courses simultaneously. The program covers a wide range of topics including Python, GenAI, Microservices, Database, Cloud Development, Fine-tuning, DevOps, GPTs, AI Agents, and Humanoids. The program is designed to give students a comprehensive understanding of Agentic AI and prepare them for careers in this field.

5. **Why don't we use TypeScript (Node.js) to develop APIs instead of using Python?**
   We will not use Typescript in AI powered API development because Python is a priority with the AI community when working with AI and if any updates come in libraries they will first come for Python. Python is always a better choice when dealing with AI and API.

   - **Python is the de facto standard for AI Development**.

- TypeScript is a more modern language that is gaining popularity for Web Development, but Python is more widely used and has a larger ecosystem of libraries and frameworks available, especially for AI.
- TypeScript is used for web user interfaces, while Python is used for APIs.
- Python is a more commonly used language for AI and API development, and it has a larger ecosystem of libraries and frameworks available for these purposes.
- TypeScript is a more modern language that is becoming increasingly popular for API development also, but it is still not as widely used as Python, especially for AI applications and development.

6. **What do we use LangGraph and LangChain for developing AI Agents as compared to other frameworks like CrewAI, AutoGen, or Litta?**
   Let's break down the key features and differences of LangGraph, LangChain, CrewAI, and AutoGen:

LangGraph

Purpose: LangGraph is designed for building stateful, multi-actor applications with large language models (LLMs).

Key Features:

- Cycles and Branching: Allows for loops and conditionals in applications.
- Persistence: Automatically saves state after each step, supporting error recovery and human-in-the-loop workflows.
- Integration Seamlessly integrates with LangChain and LangSmith, but can be used independently.
- Streaming Support: Outputs can be streamed as they are produced by each node.
- Human-in-the-Loop: Supports interrupting graph execution for approvals or edits.

LangChain

Purpose: LangChain is a framework for developing applications powered by LLMs.

Key Features:

- Tool Integration: Allows chaining together LLM tasks.
- Custom Tools: Developers can define custom tools for agents to use.
- Memory Management: Supports conversational memory for multi-turn interactions.

- ReAct Framework: Provides a framework for chain-of-thought reasoning.

CrewAI
Purpose: CrewAI focuses on creating collaborative AI agents that can work together on tasks.

Key Features:

- Collaboration: Designed for multi-agent collaboration and teamwork.
- Task Management: Efficiently manages tasks and workflows among agents.
- Scalability: Supports scaling up to handle complex, large-scale projects.

AutoGen
Purpose: AutoGen is aimed at automating the generation of code and other content.

Key Features:

- Code Generation: Specialises in generating code based on prompts.
- Content Creation: Can automate the creation of various types of content.
- Customization: Allows for customization of the generated output.

Letta
Purpose: Letta focuses on adding memory to LLMs to enhance their reasoning capabilities and provide transparent long-term memory.

Key Features:

- Memory Management: Supports advanced reasoning with long-term memory.
- Model Agnostic: Can work with any LLM, allowing developers to choose the best model for their use case.
- White Box Systems: Provides full visibility into the inner workings of LLMs and agents.
- Deployment: Offers both hosted cloud services and local deployment options.
- Use Cases: Ideal for personalised chatbots, agents connected to external data sources, and automated AI workflows.

Each of these frameworks has its strengths and is suited for different types of AI agent development. LangGraph and LangChain are particularly strong in handling stateful, multi-actor applications and integrating with other tools, while CrewAI, AutoGen and Litta focus more on collaboration, automation, and long term memory respectively.

7. **Why don't we use Flask or Django for API development instead of FastAPI?**
    - **FastAPI is a newer and more modern framework than Flask or Django.** It is designed to be fast, efficient, and easy to use. FastAPI is also more scalable than Flask or Django, making it a better choice for large-scale projects.
    - **FastAPI is also more feature-rich than Flask or Django.** It includes several built-in features that make it easy to develop APIs, such as routing, validation, and documentation.
    - **Overall, FastAPI is a better choice for API development than Flask or Django.** It is faster, more scalable, and more feature-rich.

8. **Why do we need to learn Cloud technologies in a Generative AI program?**
   Cloud technologies are essential for developing and deploying generative AI applications because they provide a scalable and reliable platform for hosting and managing complex workloads.

    - Cloud computing offers a vast pool of resources that can be provisioned on demand, which is ideal for generative AI applications that can be computationally intensive.
    - Cloud providers offer a wide range of services that can be used to support generative AI applications, including storage, computing, networking, and machine learning.
    - Cloud services are typically more cost-effective than on-premises infrastructure, which can be a significant advantage for generative AI applications that are often used for large-scale projects.

   The Certified Agentic and Robotic AI Engineering Program teaches you how to use cloud native services, including containers and Kubernetes, to deploy your applications to the cloud. You will also learn how to use **Docker containers** to package and deploy your applications, and how to use Terraform to manage your cloud infrastructure.

   By the end of the program, you will be able to:
    - Use Docker containers to package and deploy your applications

- Develop and deploy generative AI applications to the cloud
- Manage your cloud infrastructure using Terraform

9. **What is the purpose of Docker Containers and what are the benefits of deploying them with Docker Compose, and Kubernetes?**
    - **Docker Containers** are a way to package software into a single unit that can be run on any machine, regardless of its operating system. It is used to create a Dockerfile, which is a text file that describes how to build a Docker image. The image is then used to create a container, which is a running instance of the image. This makes them ideal for deploying applications on a variety of platforms, including cloud-based services.
    - **Docker Compose** is a tool provided by Docker that allows you to define and manage multi-container Docker applications locally. It enables you to use a YAML file to configure the services, networks, and volumes needed for your application's setup. With Docker Compose, you can describe the services your application requires, their configurations, dependencies, and how they should interact with each other, all in a single file. This makes it easier to orchestrate complex applications locally composed of multiple interconnected containers.
    - **Kubernetes** is a container orchestration system that automates the deployment, scaling, and management of containerized applications. It allows you to run multiple containers on a single machine or across multiple machines. It is an open source and can be deployed in your data centre or the cloud.

10. **What is the purpose of learning to develop APIs in a Generative AI program?**
    APIs (Application Programming Interfaces) are used to connect different software applications and services together. They are the building blocks of the internet and are essential for the exchange of data between different systems.

    In the Certified Agentic and Robotic AI Engineering Program, students will learn to develop APIs not just as a backend but also as a **product** itself. In this model, the API is at the core of the business's value.

    - APIs are used to make it possible for different software applications to communicate with each other.
    - APIs are used to access data from a remote server.

- APIs are used to create new services or applications that are integrated with existing systems.
- APIs are used to improve the security of applications by providing a way to control access to data.
- By learning to develop APIs, students will gain the skills necessary to create powerful and efficient software applications that can be used to solve a variety of business problems.

11. **What is the purpose of using Python-based FastAPI and related technologies?**
    In this Program, students will learn how to use Python-based FastAPI as a core library for API development.

    - FastAPI is a high-performance, lightweight, and easy-to-use framework for building APIs.
    - It is designed to be fast, scalable, and secure.
    - FastAPI is compatible with a wide range of programming languages and frameworks, making it a good choice for developers with different skill sets.
    - Students will also learn about the following related technologies:
    - **Pydantic:** Pydantic is a Python library that helps to improve the quality of your code by checking for errors and potential problems.
    - **GQL:** The Graph Query Language (GQL) is an international standard published by the International Organization for Standardization (ISO) as ISO/IEC 39075:2024. GQL is designed for querying property graphs and is the first database query language ISO has published since SQL in 1987. It defines data structures and operations for creating, accessing, querying, maintaining, and controlling property graphs, providing a standardised way to manage graph data across different implementations.
    - **Neo4j:** Neo4j is a powerful, open-source graph database designed to store and manage highly connected data. Unlike traditional relational databases, Neo4j uses a graph model, where data is represented as nodes, relationships, and properties. This structure is ideal for applications where relationships between data points are as important as the data itself.

    By the end of the program, students will be able to use Python-based FastAPI to develop APIs that are fast, scalable, and secure.

12. **What does the API-as-a-Product model entail?**
API-as-a-Product is a type of Software-as-a-Service that monetizes niche functionality, typically served over HTTP. In this model, the API is at the core of the business's value. The API-as-a-Product model is different from the traditional API model, where APIs are used as a means to access data or functionality from another application. In the API-as-a-Product model, the API itself is the product that is being sold.

    The benefits of the API-as-a-Product model include:

    - **Increased flexibility:** APIs can be used to access data or functionality from any application, regardless of the underlying platform or technology. This gives businesses greater flexibility in how they integrate APIs into their applications.
    - **Reduced development costs:** APIs can be reused by multiple applications, which can save businesses the time and expense of developing their custom APIs.
    - **Improved scalability:** APIs can be scaled up or down as needed, which makes them well-suited for businesses with fluctuating or unpredictable traffic demands.
    - **Enhanced security:** APIs can be more secure than traditional methods of data exchange, as they can be protected by a variety of security measures, such as encryption and access control.

13. **What are the benefits of using Docker Containers for development, testing, and deployment?**
Docker Containers are a fundamental building block for development, testing, and deployment because they provide a consistent environment that can be used across different systems. This eliminates the need to worry about dependencies or compatibility issues, and it can help to improve the efficiency of the development process. Additionally, Docker Containers can be used to isolate applications, which can help to improve security and make it easier to manage deployments.

14. **What is the advantage of using open Docker, Kubernetes, and Terraform technologies instead of using AWS, Azure, or Google Cloud technologies?**
Using open-source technologies like Docker, Kubernetes, and Terraform offers several advantages over relying solely on proprietary cloud services from AWS, Azure, or Google Cloud. Here’s a detailed comparison:

Advantages of Using Docker, Kubernetes, and Terraform (Open Technologies)

1. Portability and Flexibility:
   - Vendor Agnostic: These tools are cloud-agnostic, meaning you can run your applications on any cloud provider or on-premises infrastructure without being locked into a specific vendor.
   - Ease of Migration: Applications packaged in Docker containers can easily be moved across different environments, and Kubernetes provides a consistent orchestration layer, ensuring seamless transitions.

2. Cost Efficiency:
   - Avoid Vendor Lock-In: Being locked into a single cloud provider can lead to higher costs over time. Using open technologies allows you to leverage competitive pricing from multiple providers or even use on-premises resources.
   - Optimised Resource Utilisation: Kubernetes helps in efficiently managing resources through automated scaling and load balancing, potentially reducing costs.

3. Community and Ecosystem:
   - Open Source: These tools are backed by strong, active open-source communities that continuously improve the software, provide support, and share best practices.
   - Ecosystem: A rich ecosystem of tools and integrations is available, providing flexibility to choose the best components that fit your specific needs.

4. Standardisation and Consistency:
   - Unified Platform: Using Docker for containerization, Kubernetes for orchestration, and Terraform for infrastructure as code (IaC) provides a standardised way to deploy, manage, and scale applications across different environments.
   - Consistency Across Environments: These tools ensure that your development, staging, and production environments are consistent, reducing bugs and deployment issues.

5. Customization and Control:
   - Full Control: Open-source tools give you complete control over your infrastructure and deployment pipelines. You can customise and extend the functionality to suit specific requirements.
   - Transparency: Access to the source code means you can audit and modify the software to meet your security and compliance needs.

Advantages of Using AWS, Azure, or Google Cloud Technologies

1. Managed Services:
   - **Ease of Use:** Cloud providers offer a wide range of managed services that abstract away the complexity of setting up and managing infrastructure. This can save time and reduce operational overhead.
   - Integrated Solutions: These platforms provide integrated services and tools, such as databases, machine learning, analytics, and monitoring, which can be easily combined to build complex applications.

2. Scalability and Reliability:
   - Global Infrastructure: Cloud providers have extensive global infrastructure, ensuring high availability, redundancy, and low latency.
   - Auto-Scaling: Advanced auto-scaling capabilities can dynamically adjust resources to meet changing demands, ensuring optimal performance.

3. Security and Compliance:
   - Built-In Security: Cloud providers offer robust security features, including identity and access management, encryption, and compliance certifications, helping to protect your data and meet regulatory requirements.
   - Automatic Updates: Managed services often include automatic updates and patches, reducing the risk of security vulnerabilities.

4. Innovation and Support:
   - Cutting-Edge Technology: Major cloud providers continuously innovate and introduce new services and features, allowing you to leverage the latest technologies without significant investment.
   - Support and SLA: Comprehensive support services and Service Level Agreements (SLAs) ensure that you have access to expert help and guaranteed uptime.

Conclusion

Choosing between open-source technologies like Docker, Kubernetes, and Terraform versus proprietary cloud services from AWS, Azure, or Google Cloud depends on your specific needs and priorities.

- Open Technologies: Offer portability, cost efficiency, customization, and control, making them ideal for multi-cloud strategies, avoiding vendor lock-in, and having more control over your infrastructure.
- Cloud Providers: Provide ease of use, managed services, scalability, security, and access to cutting-edge technology, which can be advantageous for rapid development, scaling, and leveraging advanced services.

In many cases, a hybrid approach that combines the strengths of both open-source tools and cloud provider services can provide the best of both worlds, allowing you to optimise for cost, flexibility, and innovation.

15. **Why in this program are we not learning to build LLMs ourselves? How difficult is it to develop an LLM like ChatGPT 4 or Google's Gemini?**
Developing an LLM like ChatGPT 4 or Google Gemini is extremely difficult and requires a complex combination of resources, expertise, and infrastructure. Here's a breakdown of the key challenges:

Technical hurdles:

Massive data requirements: Training these models requires an immense amount of high-quality data, often exceeding petabytes. Compiling, cleaning, and structuring this data is a monumental task.
Computational power: Training LLMs demands incredible computational resources, like high-performance GPUs and specialised AI hardware. Access to these resources and the ability to optimise training processes are crucial.
Model architecture: Designing the LLM's architecture involves complex decisions about parameters, layers, and attention mechanisms. Optimising this architecture for performance and efficiency is critical.
Evaluation and bias: Evaluating the performance of LLMs involves diverse benchmarks and careful monitoring for biases and harmful outputs. Mitigating these biases is an ongoing research challenge.

Resource and expertise:

Team effort: Developing an LLM like ChatGPT 4 or Google Gemini requires a large team of experts across various disciplines, including AI researchers, machine learning engineers, data scientists, and software developers.
Financial investment: The financial resources needed are substantial, covering costs for data acquisition, hardware, software, and talent. Access to sustained funding is critical.

Additionally:

Ethical considerations: LLMs raise ethical concerns like potential misuse, misinformation, and societal impacts. Responsible development and deployment are crucial.
Rapidly evolving field: The LLM landscape is constantly evolving, with new research, models, and benchmarks emerging. Staying abreast of these advancements is essential.

Therefore, while ChatGPT 4 and Google Gemini have made impressive strides, developing similar LLMs remains a daunting task accessible only to a handful of organisations with the necessary resources and expertise.

In simpler terms, it's like building a skyscraper of knowledge and intelligence. You need the right materials (data), the right tools (hardware and software), the right architects (experts), and a lot of hard work and attention to detail to make it stand tall and function flawlessly.

Developing similar models would be a daunting task for individual developers or smaller teams due to the enormous scale of resources and expertise needed. However, as technology progresses and research findings become more accessible, it might become incrementally more feasible for a broader range of organisations or researchers to work on similar models, albeit at a smaller scale or with fewer resources. At that time we might also start to focus on developing LLMs ourselves.

To sum up, the focus of the program is not on LLM model development but on applied Cloud GenAI Engineering (GenEng), application development, and fine-tuning of foundational models. The program covers a wide range of topics including Python, GenAI, Microserices, API, Database, Cloud Development, and DevOps, which will give students a comprehensive understanding of generative AI and prepare them for careers in this field.

16. **Business wise does it make more sense to develop LLMs ourselves from scratch or use LLMs developed by others and build applications using these tools by using APIs and/or fine-tuning them?**
Whether it makes more business sense to develop LLMs from scratch or leverage existing ones through APIs and fine-tuning depends on several factors specific to your situation. Here's a breakdown of the pros and cons to help you decide:

Developing LLMs from scratch:

Pros:

Customization: You can tailor the LLM to your specific needs and data, potentially achieving higher performance on relevant tasks.
Intellectual property: Owning the LLM allows you to claim intellectual property rights and potentially monetize it through licensing or other means.
Control: You have full control over the training data, algorithms, and biases, ensuring alignment with your ethical and business values.

Cons:

High cost: Building and training LLMs require significant technical expertise, computational resources, and data, translating to high financial investment.
Time commitment: Developing an LLM is a time-consuming process, potentially delaying your go-to-market with your application.
Technical expertise: You need a team of highly skilled AI specialists to design, train, and maintain the LLM.

Using existing LLMs:

Pros:

Lower cost: Leveraging existing LLMs through APIs or fine-tuning is significantly cheaper than building them from scratch.
Faster time to market: You can quickly integrate existing LLMs into your applications, accelerating your launch timeline.
Reduced technical burden: You don't need a large team of AI specialists to maintain the LLM itself

Cons:

Less customization: Existing LLMs are not specifically designed for your needs, potentially leading to lower performance on some tasks.
Limited control: You rely on the data and biases of the existing LLM, which might not align with your specific requirements.
Dependency on external parties: You are dependent on the availability and maintenance of the LLM by its developers.

Here are some additional factors to consider:

The complexity of your application: Simpler applications might benefit more from existing LLMs, while highly complex ones might require the customization of a dedicated LLM.
Your available resources: If you have the financial and technical resources, developing your own LLM might be feasible. Otherwise, existing options might be more practical.
Your competitive landscape: If your competitors are using LLMs, you might need to follow suit to remain competitive.
Ultimately, the best decision depends on your specific needs, resources, and business goals. Carefully evaluating the pros and cons of each approach will help you choose the strategy that best aligns with your success.

17. What are Custom GPTs?

"Custom GPTs" refers to specialised versions of the Generative Pre-trained Transformer (GPT) models that are tailored for specific tasks, industries, or data types. These custom models are adapted from the base GPT architecture, which is a type of language model developed by OpenAI. Custom GPT models are trained or fine-tuned on specific datasets or for particular applications, allowing them to perform better in those contexts compared to the general-purpose models.

Here are some examples of what custom GPT models might be used for:

1. **Industry-Specific Needs**: A custom GPT for legal, medical, or financial industries could be trained on domain-specific texts to understand and generate industry-specific language more accurately.

2. **Language and Localization**: Models can be customised for different languages or dialects that might not be well-represented in the training data of the base model.

3. **Company-Specific Applications**: Organisations might develop a custom GPT model trained on their own documents and communications to assist with internal tasks like drafting emails, generating reports, or providing customer support.

4. **Educational Purposes**: Educational institutions might develop custom GPTs trained on educational material to assist in creating teaching materials or providing tutoring in specific subjects.

5. **Creative Writing and Entertainment**: Custom models could be trained on specific genres of literature or scripts to assist in creative writing or content creation.

6. **Technical and Scientific Research**: A custom GPT model could be trained on scientific literature to assist researchers in summarising papers, generating hypotheses, or even drafting new research.

These custom models are created through a process of fine-tuning, where the base GPT model is further trained (or 'fine-tuned') on a specific dataset. This process allows the model to become more adept at understanding and generating text that is relevant to the specific use case. Fine-tuning requires expertise in machine learning and natural language processing, as well as access to relevant training data.

18. What are Actions in GPTs?

Actions are a way to connect custom GPTs to external APIs, allowing them to access data or interact with the real-world. For example, you can use actions to create a GPT that can book flights, send emails, or order pizza. **Actions are defined using the OpenAPI specification**, which is a standard for describing APIs. You can import an existing OpenAPI specification or create a new one using the GPT editor.

19. **What are AI Agents and how do they differ from Custom GPTs?**
AI Agents and Custom GPTs are both tools that utilise artificial intelligence to perform tasks, but they have distinct functionalities and use cases. Here's a breakdown of their differences:

AI Agents
AI Agents are autonomous programs that can perceive their environment, make decisions, and act upon them to achieve specific goals. They often interact with other systems or users, continuously learning and adapting based on their experiences.

Key Characteristics:

- 1. Autonomy: AI Agents operate independently without continuous human intervention.
- 2. Learning: They often employ machine learning algorithms to improve performance over time.
- 3. Interactivity: AI Agents can interact with their environment, other systems, and users.
- 4. Goal-Oriented: They are designed to achieve specific objectives and can adapt their actions to optimise towards these goals.
- 5. Multi-Modal Capabilities: AI Agents can incorporate various forms of AI, such as computer vision, natural language processing, and decision-making algorithms.

Examples:

- - Robotics: Autonomous robots that navigate and perform tasks.
- - Virtual Assistants: Programs like Siri or Alexa that interact with users and perform tasks based on voice commands.
- - Game AI: Non-player characters (NPCs) that adapt and react to player actions.

Custom GPTs
Custom GPTs are tailored instances of OpenAI's ChatGPT, launched in late 2022. They are designed for specific purposes and enhanced with context. Each custom GPT can have a unique "personality," including tone of voice, language complexity, and responsiveness to specific topics. For example, a financial institution's custom GPT could be trained on financial reports and

industry-specific terminology, while a healthcare provider’s version might focus on medical literature and health policy documents

Key Differences

1. Autonomy:
   - AI Agents: Operate autonomously and continuously interact with their environment.
   - Custom GPTs: Typically respond to specific inputs and generate outputs accordingly, but don’t operate autonomously beyond text generation tasks.

2. Learning and Adaptation:
   - AI Agents: Often incorporate continuous learning and adaptation mechanisms.
   - Custom GPTs: Rely on pre-training and fine-tuning phases, with limited continuous learning capabilities.

4. Interactivity:
   - AI Agents: Can interact with both digital and physical environments.
   - Custom GPTs: Primarily interact through text-based inputs and outputs.

In summary, while both AI Agents and Custom GPTs utilise AI, AI Agents are designed for autonomous, goal-oriented actions in diverse environments, and Custom GPTs are specialised in generating and understanding human-like text for specific applications.

20. Do we need to use Design Thinking and BDD for designing custom GPTs and AI Agents?

Design Thinking and Behavior-Driven Development (BDD) are methodologies that can greatly enhance the process of designing custom GPTs and AI Agents, though they are not strictly necessary. Here’s how each can be beneficial:

Design Thinking

Design Thinking is a user-centred approach to innovation and problem-solving that involves understanding the user, challenging assumptions, redefining problems, and creating innovative solutions through iterative prototyping and testing.

Benefits for Custom GPTs and AI Agents:
1. User-Centric Focus: Ensures that the AI solutions are tailored to the actual needs and pain points of users.

2. Empathy: Helps in understanding the context and environment in which the AI will be used, leading to more relevant and effective solutions.
3. Iterative Development: Encourages continuous testing and refinement of ideas, leading to more robust and user-friendly AI models.
4. Collaboration: Promotes cross-disciplinary collaboration, which can bring diverse perspectives and expertise to the design process.

Behaviour-Driven Development (BDD)

BDD is a software development methodology that encourages collaboration between developers, QA, and non-technical stakeholders through the use of natural language descriptions of the desired behaviour of the software.

Benefits for Custom GPTs and AI Agents:
1. Clear Requirements: Ensures that the requirements are clearly understood and agreed upon by all stakeholders.
2. Testable Scenarios: Facilitates the creation of testable scenarios that can validate the AI's behaviour against the expected outcomes.
3. Documentation: Provides clear and comprehensive documentation of the AI's intended behaviour, which is useful for future maintenance and enhancements.
4. Alignment: Ensures that the development stays aligned with business goals and user expectations.

Application in Designing Custom GPTs and AI Agents

For Custom GPTs:

- Design Thinking:
  - Understand the specific use cases and user interactions where the GPT will be applied.
  - Iterate on the model's performance by gathering user feedback and refining the fine-tuning process.
  - Prototype different conversation flows and evaluate their effectiveness with real users.

- BDD:
  - Define the expected behaviours of the GPT in natural language scenarios.
  - Create automated tests that validate the GPT's responses against these scenarios.
  - Ensure that the GPT's behaviour aligns with user stories and business requirements.

For AI Agents:

- Design Thinking:

- Map out the user journey and identify critical interaction points where the AI Agent will provide value.
- Prototype and test the agent's interactions in various environments to ensure robustness and usability.
- Use empathy maps and personas to better understand and anticipate user needs and behaviours.

- BDD:
  - Write behaviour scenarios that describe how the AI Agent should react in different situations.
  - Develop tests that simulate these scenarios to verify the agent's decision-making and learning processes.
  - Continuously refine the agent's behaviour based on test results and user feedback.

While not strictly necessary, Design Thinking and BDD can significantly enhance the design and development process of custom GPTs and AI Agents by ensuring a user-centred approach, clear requirements, and continuous improvement through iterative testing and feedback. These methodologies help in creating more effective, reliable, and user-friendly AI solutions.

21. What is Ray, how do we use it in developing AI?

Ray is an open-source distributed computing framework specifically designed to simplify the development and scaling of AI applications and other data-intensive tasks. It provides a flexible platform for building distributed applications that can scale from a single machine to a large cluster, making it ideal for AI workloads that demand substantial computational power, such as machine learning training, reinforcement learning, data processing, and model serving.

Here's how Ray is used in AI development and why it's advantageous:

a. Distributed Computing for AI Workloads

Ray's primary feature is its ability to distribute computation across multiple CPUs and GPUs, allowing developers to run tasks in parallel and manage complex workflows efficiently. This is particularly beneficial in AI development, where training large models or processing vast datasets can be time-consuming and resource-intensive.

- Parallelization: Ray enables developers to parallelize tasks, such as data preprocessing or model training, across multiple nodes, reducing runtime and maximising resource usage.

- Scaling: AI tasks can be distributed across clusters, allowing them to scale up to cloud-based environments or large on-premises setups with minimal code changes.

b. Simplified Development with Python API

Ray is designed to work seamlessly with Python, which is the most popular language in AI and machine learning. Its intuitive API makes it easy for developers to turn existing Python functions into distributed tasks, which reduces the complexity of parallel programming.

- Ease of Use: Developers can use familiar Python code without deep knowledge of distributed systems, thanks to Ray’s ability to automatically handle task scheduling, data sharing, and fault tolerance.
- Flexible Abstractions: Ray provides high-level abstractions, such as tasks and actors, to manage distributed computations efficiently. This flexibility allows developers to easily implement complex parallel workflows.

c. Frameworks Built on Ray for Specialized AI Tasks

Ray includes several libraries tailored for specific AI use cases, simplifying complex tasks and streamlining workflow integration:

- Ray Tune: Used for hyperparameter tuning, which is crucial in optimizing AI models. Ray Tune distributes the tuning process, allowing multiple hyperparameter configurations to be evaluated in parallel, which speeds up model optimization significantly.
- Ray RLlib: A library for reinforcement learning (RL) that simplifies the implementation and scaling of RL algorithms. It supports a range of algorithms out of the box and can run on single machines or distributed clusters, making it suitable for large-scale RL projects.
- Ray Serve: Designed for model serving, it provides a scalable, flexible way to deploy trained AI models as services that can handle multiple requests in real-time, making it ideal for production applications.

d. Use Cases in AI Development

Ray is applicable across various stages of AI development, including:

- Data Processing and Preprocessing: Ray can distribute data loading, transformation, and cleaning tasks, which are often bottlenecks in AI workflows, across a cluster. This accelerates the process and allows for more efficient handling of large datasets.
- Model Training: With Ray, model training can be distributed across multiple GPUs or nodes, reducing training time and enabling the handling of larger models or datasets.
- Hyperparameter Optimization: Ray Tune simplifies running multiple experiments to find the best model configurations, automating the tuning process and improving model performance.

- Reinforcement Learning: Ray RLlib supports scaling RL algorithms, which are often computationally intensive and require significant parallelization to achieve optimal performance.
- Model Deployment: Ray Serve allows developers to deploy models for real-time predictions, handling requests with high throughput and low latency.

5. Advantages of Using Ray in AI

- Scalability: Ray's distributed architecture makes it easy to scale AI workloads, whether they are running on a laptop, a multi-GPU setup, or a cloud cluster.
- Efficiency: By enabling distributed processing and parallelism, Ray reduces the computational time needed for data processing, model training, and serving, leading to faster iteration cycles.
- Unified Framework: Ray integrates various stages of the AI pipeline (training, tuning, deployment) under a single platform, making it easier for teams to manage and scale AI workflows consistently.
- Cost Optimization: By optimising resource usage across clusters, Ray helps minimise infrastructure costs, especially in cloud environments where resources are billed based on usage.

22. When Fine-Tuning Open-Source Large Language Models, how is Ray and PyTorch important and what role do these libraries play?

Fine-tuning open-source Large Language Models (LLMs) like Meta's LLaMA or other foundational models is a complex and resource-intensive task that involves adapting a pre-trained model to perform better on specific tasks or domains. Ray and PyTorch play essential roles in this process by providing tools to efficiently handle the heavy computational requirements of fine-tuning, making it scalable, flexible, and streamlined.

Here's how each of these libraries contributes to fine-tuning LLMs:

a. Role of PyTorch in Fine-Tuning LLMs

PyTorch is a deep learning framework widely used for developing and fine-tuning neural networks, including LLMs. It provides the foundational tools for setting up, training, and optimising neural network models.

- **Flexible Model Building**: PyTorch's dynamic computation graph, also known as define-by-run, allows researchers and developers to easily customise the LLM architecture or adjust training configurations to fit specific tasks. This is especially useful when experimenting with different model architectures or training techniques.

- **Pre-Trained Model Integration**: PyTorch integrates seamlessly with the Hugging Face Transformers library, which provides access to many pre-trained LLMs. This integration simplifies the fine-tuning process, allowing developers to load pre-trained models, prepare them for task-specific data, and modify them as needed.
- **GPU Acceleration**: PyTorch supports GPU and distributed training, crucial for handling the immense computational requirements of fine-tuning LLMs. Large language models have millions to billions of parameters, so GPU acceleration dramatically reduces training time.
- **Tools for Optimization**: PyTorch offers numerous built-in optimizers and training utilities, such as AdamW (an optimizer widely used in NLP), learning rate schedulers, and gradient clipping. These tools help ensure that fine-tuning is efficient, effective, and stable.

b. Role of Ray in Scaling and Distributing the Fine-Tuning Process
Ray is critical for scaling the fine-tuning process across multiple GPUs or even across distributed clusters, which is essential for handling the large-scale computations required for LLMs.

- **Distributed Training**: Ray makes it easy to distribute PyTorch workloads across multiple CPUs or GPUs by turning Python functions into remote functions that can be executed concurrently. This is useful for model parallelism, where different parts of the model are processed on different devices, or for data parallelism, where the same model is trained on different data batches across multiple devices.
- **Hyperparameter Tuning with Ray Tune**: Fine-tuning often involves adjusting hyperparameters like learning rate, batch size, or dropout rates. Ray Tune, a library within Ray, automates this process by running multiple configurations simultaneously, speeding up the search for the optimal settings.
- **Efficient Resource Utilisation**: Ray helps manage computational resources efficiently by automatically handling task scheduling, load balancing, and fault tolerance. This allows developers to focus on model training without manually managing cluster infrastructure.
- **Integration with PyTorch Distributed**: Ray seamlessly integrates with PyTorch's native distributed framework (such as torch.distributed). This allows developers to leverage Ray's parallelism tools along with PyTorch's model training capabilities to create efficient distributed training pipelines.

c. Combined Workflow Using PyTorch and Ray for Fine-Tuning
The ideal workflow for fine-tuning LLMs combines the strengths of both PyTorch and Ray:

- **Model Loading and Preparation**: Load the pre-trained model in PyTorch and set up training parameters.
- **Data Parallelism with Ray**: Distribute data across multiple GPUs or nodes using Ray to increase training speed.
- **Hyperparameter Tuning with Ray Tune**: Use Ray Tune to test various configurations, automatically tracking results and selecting the optimal combination.
- **Training Management and Optimization**: Use PyTorch for efficient training with distributed backpropagation and GPU utilisation, while Ray manages the orchestration across clusters.

d. Advantages of Using Ray and PyTorch Together in Fine-Tuning

- **Scalability**: Ray allows PyTorch models to scale beyond a single device or server, which is crucial for handling large-scale models and datasets.
- **Speed and Efficiency**: Distributing data processing and training across multiple devices can significantly reduce fine-tuning time, making iterative experimentation faster.
- **Flexibility**: Ray's integration with Python and PyTorch allows developers to build complex distributed training pipelines without needing to learn complex distributed systems programming.
- **Cost Efficiency**: By efficiently managing hardware resources, Ray helps reduce infrastructure costs, particularly in cloud environments where resources are billed based on usage.

Summary

Ray and PyTorch together provide a robust platform for fine-tuning large language models. PyTorch offers the deep learning capabilities required for LLM training, while Ray adds scalability and resource management, making distributed fine-tuning accessible, efficient, and effective. This combination enables developers to fine-tune powerful models on large datasets across distributed environments, paving the way for faster, scalable AI development.

23. In this course PyTorch plays a crucial role in the fine-tuning of open-source LLMs, why don't we use TensorFlow instead?

While TensorFlow is a powerful and widely-used deep learning framework, PyTorch offers several advantages that make them particularly well-suited for fine-tuning open-source Large Language Models (LLMs). Here's a detailed comparison highlighting why PyTorch might be preferred over TensorFlow for this specific task:

PyTorch vs. TensorFlow

a. Dynamic Computation Graphs:

- PyTorch: Uses dynamic computation graphs (define-by-run), which allow for greater flexibility and ease of debugging. This is especially useful when experimenting with new models and training strategies.
- TensorFlow: Initially used static computation graphs (define-and-run). Although TensorFlow 2.0 introduced eager execution to support dynamic graphs, PyTorch's implementation is often considered more intuitive and easier to work with for dynamic tasks.

b. Ease of Use:

- PyTorch: Known for its simplicity and clear, Pythonic code, which makes it easier to learn and use, especially for research and prototyping.
- TensorFlow: While TensorFlow 2.0 improved usability, it is still considered more complex compared to PyTorch, particularly for newcomers.

c. Community and Ecosystem:

- PyTorch: Has seen rapid adoption in the research community, leading to a rich ecosystem of tools, libraries, and community support. Libraries like Hugging Face’s Transformers are built primarily for PyTorch, offering extensive support for LLMs.
- TensorFlow: Has a strong industrial presence and is widely used in production environments. However, the research community has increasingly favoured PyTorch.

d. Integration with Hugging Face:

- PyTorch: Hugging Face’s Transformers library, which is a go-to for working with LLMs, is deeply integrated with PyTorch. This library provides pre-trained models, tokenizers, and utilities that simplify the process of fine-tuning LLMs.
- TensorFlow: Although Hugging Face provides TensorFlow support, the integration is not as seamless or feature-rich as it is with PyTorch.

While TensorFlow remains a powerful framework, particularly in production environments, PyTorch and provides a combination of flexibility, ease of use, and community support that make them particularly well-suited for the fine-tuning of open-source LLMs.

24. What is Physical AI?

Physical AI refers to the integration of artificial intelligence with physical entities, such as robots, that can operate and interact in the real world. This concept involves AI systems that not only process data and make decisions but also perform physical actions and understand the laws of physics.

Key Characteristics:

1. Real-World Interaction:
   - Physical AI systems can perceive their environment through sensors, process this information, and take appropriate actions using actuators.

2. Embodiment:
   - Unlike purely digital AI, Physical AI involves AI embedded in physical bodies, like humanoid robots, which can navigate and manipulate the physical world.

3. Understanding Physics:
   - These AI systems are designed to comprehend and adhere to the physical laws that govern real-world interactions, such as gravity, friction, and object dynamics.

4. Human-like Functionality:
   - Humanoid robots are a prime example of Physical AI, as they are built to perform tasks in environments designed for humans, utilising a form factor that mirrors human anatomy.

5. Data-Driven Training:
   - Physical AI leverages vast amounts of real-world data to train AI models, enabling robots to improve their performance through machine learning and interaction experiences.

Applications:

- **Healthcare**:
  - Assistive robots that help with patient care, rehabilitation, and surgery.
- **Service Industry**:
  - Robots that perform tasks such as cleaning, delivery, and customer service.
- **Manufacturing**:
  - Industrial robots that assemble products, manage inventory, and ensure quality control.
- **Exploration**:
  - Robots designed for exploration in environments like space, underwater, or disaster zones.

Physical AI represents a significant shift from traditional AI applications confined to virtual environments. It aims to bridge the gap between digital intelligence and physical capability, creating systems that can understand and interact with the world in a human-like manner. This evolution has the potential to revolutionise various industries by enhancing automation, improving efficiency, and enabling new forms of human-machine collaboration.

25. **What are the different specialisations offered at the end of the program and what are their benefits?**

At the end of the certification program we offer eight specialisations in different fields:

AI-701: Healthcare and Medical Agentic AI: This specialisation will teach students how to use agentic and generative AI to improve healthcare and medical research. This is relevant to fields such as drug discovery, personalised medicine, and surgery planning.

Benefits:

- Learn how to use generative and agentic AI to identify diseases, develop new drugs, and personalise treatment plans.
- Gain a deeper understanding of the ethical implications of using generative AI in healthcare.
- Prepare for a career in a growing field with high demand for skilled professionals.

AI-702: Web3, Blockchain, and Agentic AI Integration

Integrating Web3, blockchain, and agentic AI technologies creates a powerful ecosystem where autonomy, security, transparency, and user control are at the forefront. These technologies combined enable:

- Secure, decentralised AI applications that protect user privacy,
- Transparent AI interactions that foster trust,
- Autonomous, scalable AI agents that operate reliably in peer-to-peer environments,
- Tokenized incentives that drive community participation, and
- Innovative ownership models for digital assets, enabling new economic opportunities.

AI-703: Metaverse, 3D, and Agentic AI Integration: This specialisation will teach students how to create and use 3D models and other immersive content manually and with generative AI. This is relevant to fields such as gaming, marketing, and architecture.

Benefits:

- Learn how to use generative AI to create realistic and immersive 3D models.
- Develop the skills necessary to work in the growing field of virtual reality (VR) and augmented reality (AR).
- Apply generative AI to solve real-world problems in areas such as product design, marketing, and education.

AI-704: Agentic AI for Accounting, Finance, and Banking: This specialisation will teach students how to integrate generative AI with Web3 and blockchain technologies. This is relevant to fields such as finance, healthcare, and supply chain management.

Benefits:

- Learn how to create smart contracts and decentralised applications (dApps).
- Gain a deeper understanding of the potential of blockchain technology and how it can be used to improve business processes.
- Develop the skills necessary to work in a rapidly growing field with high demand for skilled professionals.

AI-705: Agentic AI for Engineers: This specialisation will teach students how to use generative AI to improve engineering design and problem-solving. This is relevant to fields such as manufacturing, construction, and product development.

Benefits:

- Learn how to use generative AI to create simulations, optimize designs, and predict failures.
- Gain a deeper understanding of the engineering design process and how generative AI can be used to improve it.
- Prepare for a career in a growing field with high demand for skilled professionals.

AI-706: Agentic AI for Sales and Marketing: This specialisation will teach students how to use generative AI to improve sales and marketing campaigns. This is relevant to fields such as advertising, public relations, and customer service.

Benefits:

- Learn how to use generative AI to create personalised marketing messages, generate leads, and track campaign performance.

- Gain a deeper understanding of the latest marketing trends and how generative AI can be used to improve them.
- Prepare for a career in a growing field with high demand for skilled professionals.

AI-707: Agentic AI for Automation and Internet of Things (IoT) :

- **Provide Multi-Modal User Interface for the IoT systems:** Multimodal interaction exploits the synergic use of different modalities to optimise the interactive tasks accomplished by the users. This allows a user to use several input modes such as speech, touch, and visual to interact with IoT systems.
- **Improve efficiency and accuracy of industrial processes:** By implementing GenAI in automation and IoT systems, industries can optimise their processes, reduce manual labour, and increase productivity while ensuring higher accuracy and consistency.
- **Enhance decision-making:** GenAI can analyse vast amounts of data collected by IoT sensors to derive valuable insights, enabling businesses to make informed decisions regarding operations, maintenance, and resource allocation.
- **Personalise user experiences:** GenAI can leverage IoT data to understand user preferences and behaviours, enabling the creation of personalised experiences across smart devices and IoT-enabled systems.

AI-708: Agentic AI for Cyber Security:

- **Strengthen threat detection and response:** GenAI can be used to rapidly detect and respond to cyber threats by analysing large volumes of security data in real time, identifying anomalies, and suggesting appropriate countermeasures.
- **Enhance security monitoring and analysis:** GenAI can assist security analysts in monitoring and analysing security logs, automating threat detection, and providing insights into security risks and vulnerabilities.
- **Improve threat intelligence:** GenAI can be used to gather and analyse threat intelligence from various sources, enabling organisations to stay informed about the latest threats and trends and proactively strengthen their security posture.