Agentic AI Course for Developers

Introduction

This course is designed to help developers build intelligent, autonomous systems—AI agents—that can reason, plan, and take action using modern frameworks like LangChain, LangGraph, and no-code tools such as n8n. You'll learn how to design and deploy agents that use tools, integrate APIs, manage memory, and collaborate with other agents to solve complex tasks. The course combines theory, hands-on labs, and real-world projects, guiding you from simple workflows to fully deployed multi-agent applications with monitoring, safety, and scalability built in.

CURRICULUM:

Sr. No.	Contents

1	**Anatomy of Agents + LangChain Single-Agent Basics**
	• Theory / Concepts
	  ◦ What is an “agent” in AI context: perception, reasoning, action, memory.
	  ◦ Different types of agents (tool-using, planner, reactive, etc.).
	  ◦ When to use an agent vs simpler chain / pipeline.
	• Hands-On with LangChain
	  ◦ Install LangChain, set up environment.
	  ◦ Build a basic LangChain agent: define some tools (e.g. search, calculator), set up a prompt, run agent on simple inputs.
	  ◦ Explore LangChain’s “Build an Agent” tutorial: agent with search tool and memory.
	• Deliverable
	  ◦ A working single-agent script / small app that can do queries, use tools, maintain simple conversational memory.

2	**Improving Single Agents – Memory, Tool Chains, Error Handling**
	• Theory / Concepts
	  ◦ Memory types (short term, long term), retrieval augmented generation (RAG).
	  ◦ Tool chaining (having an agent invoke multiple tools/steps in order).
	  ◦ Error handling and fallback strategies.
	• Practical Work
	  ◦ Extend your single agent: add vector store memory for past interactions.
	  ◦ Add more tools; build chains (e.g. search ? summarization ? answer).
	  ◦ Handle errors: what if tool fails, or retrieval returns bad content.
	• Deliverable
	  ◦ Enhanced agent with memory storage + fallback logic; tests with failing tool & recovery.

3	**Introduction to LangGraph – Graphs, Nodes, State, Control Flow**
	• Theory / Concepts
	  ◦ What is LangGraph: graph-based orchestration on top of LangChain. Difference vs simple agents.
	  ◦ Components: nodes (agents/tools), edges (control flow), state (carried along), commands (goto, update).
	  ◦ When to use LangGraph: for workflows, branching, multi-step agents, multi-agent etc.
	• Hands-On
	  ◦ Install LangGraph.
	  ◦ Create simple LangGraph workflow: maybe two nodes: node A gets user input; node B uses tool; control flow linear.
	  ◦ Visualize or log the state transitions.
	• Deliverable
	  ◦ Working graph workflow, showing state transitions; simple linear graph.

4	**Branching, Conditional Logic & Memory in LangGraph**
	• Theory / Concepts
	  ◦ Branching: how to decide which node to call next (conditional edges).
	  ◦ Updating state: how nodes can update state and influence future nodes.
	  ◦ Memory management inside graph: persisting results, caching, summarization, etc.
	• Lab
	  ◦ Build a graph that branches: e.g. if query needs real-time info ? call search tool; else, use memory or stored context.
	  ◦ Use state to remember past interactions and influence branch decisions.
	• Deliverable
	  ◦ Graph workflow with branching (at least one conditional), memory usage, state updates.

5	**Multi-Agent Patterns: Supervisors, Role Agents, Parallelism**
	• Theory / Concepts
	  ◦ What multi-agent means in LangGraph: independent agent nodes, how they are connected (edges).
	  ◦ Supervisor/ Orchestrator agent pattern.
	  ◦ Parallel node execution vs sequential; trade-offs.
	• Practical
	  ◦ Build a system with multiple agents: e.g. one agent researches, another formats, another verifies. Use supervisor to route tasks.
	  ◦ Demonstrate parallel vs sequential execution.
	• Deliverable
	  ◦ Multi-agent graph: at least 3 agents cooperating under supervision; also document latency or cost differences between parallel & sequential.

6	**Hierarchical & Team Architectures in LangGraph**
	• Theory / Concepts
	  ◦ Hierarchical agents: teams under supervisors, nested graphs.
	  ◦ Edge cases: how top-level supervisors manage teams, how state is partitioned or shared.
	  ◦ Handling complex workflows: how to keep them maintainable.
	• Lab
	  ◦ Build a team-based agent structure: e.g. Team A handles research + data fetching; Team B handles summarization + formatting; top-level supervisor coordinates between them.
	  ◦ Include team agents running possibly in parallel, merging outputs.
	• Deliverable
	  ◦ Hierarchical multi-agent project prototype. Share architecture diagram + code.

7	**Example Revisions / Reflect Agents + Feedback Loops**
	• Theory / Concepts
	  ◦ Agents that “reflect” or critique: improve output via cycles.
	  ◦ Revision loops: generate ? reflect ? revise.
	  ◦ How to build critique agents and integrate them.
	• Practical
	  ◦ Use an existing open-source multi-agent example: e.g., “report writer” with multiple agents (plan, generate, reflect, critique) using LangGraph. (Repo: botextractai / ai-langgraph-multi-agent)
	  ◦ Modify or extend: add more revisions or change criteria.
	• Deliverable
	  ◦ Report producer with reflection loops; evaluate improved output vs first draft.

8	**Tool / API Integration & Security**
	• Theory / Concepts
	  ◦ How to integrate external tools / APIs safely.
	  ◦ Permissions & scope: limiting what each agent can do.
	  ◦ Input sanitization, output validation, policy checks.
	• Lab
	  ◦ Add agents that call external APIs (e.g. weather, search, data APIs).
	  ◦ Include guard nodes: checks to ensure outputs meet safety / policy criteria.
	  ◦ Possibly sandbox tool execution if coding tools are used.
	• Deliverable
	  ◦ Graph with agents invoking external services; safety checks implemented; example malicious / malformed input handled gracefully.

9	**Observability, Testing, and Metrics**
	• Theory / Concepts
	  ◦ Logging per node, state trace, time/latency, errors.
	  ◦ Defining metrics: accuracy, cost (API/LLM tokens), time, user satisfaction.
	  ◦ Testing workflows: unit tests for nodes, integration tests for the graph.
	• Practical
	  ◦ Instrument one of your existing graph workflows with logs/traces.
	  ◦ Create tests: feed in known inputs, assert outputs, simulate failures.
	  ◦ Create basic dashboard (could be local, or simple web UI) summarizing node performance.
	• Deliverable
	  ◦ Graph with tests; logging; a simple metrics dashboard; report of analysis.

10	**Deployment & Scaling**
	• Theory / Concepts
	  ◦ Containerization (Docker), REST or other service wrappers.
	  ◦ Scaling workflows: concurrent execution, queueing, rate-limit, cost.
	  ◦ State persistence: vector DB, state DB, checkpointing.
	• Lab
	  ◦ Wrap your multi-agent graph system in a service (FastAPI or Flask).
	  ◦ Containerize it. Possibly deploy locally or to cloud (AWS/GCP/Azure).
	  ◦ Use persistent storage for state / memory (vector DB or Postgres + embeddings).
	• Deliverable
	  ◦ Deployed prototype (could be simple staging); documentation on scale, environment; performance observations.

11	**Multi-Agent Capstone Kick-Off**
	• Theory / Preparation
	  ◦ Define capstone project: scope, agents, roles, workflows, tools needed.
	  ◦ Architecture design: graph structure, supervisor/hierarchical setup, memory, safety, observability.
	  ◦ Team roles & division of labor.
	• Project Starts
	  ◦ Students / teams pick their project. Examples:
	  ▪ A multi-agent assistant that gathers data, verifies, writes, and formats outputs (e.g. report).
	  ▪ Multi-agent system for customer service: routing, knowledge fetching, response generation.
	  ▪ Workflow for content generation + review + scheduling.
	  ◦ Build initial nodes & graphs; set up basic interactions.

12	**Capstone Finalization & Presentation**
	• Practical Completion
	  ◦ Complete all agent nodes, integrate tools, reflection or feedback loops.
	  ◦ Add testing, safety checks, logging & monitoring.
	  ◦ Deploy or simulate deployment; possibly wrap with UI or API.
	• Presentation & Review
	  ◦ Each team presents: architecture diagram, demo, issues faced, trade-offs made, future improvements.
	  ◦ Compare different projects; peer review.

Learning Outcomes:

By the end of this course, participants will be able to:

- Create and execute effective digital marketing campaigns independently.
- Use SEO and keyword research to increase website visibility.
- Manage Google Ads and Meta Ads with data-driven targeting.
- Analyze performance through analytics and reporting dashboards.
- Develop social media and content strategies that drive engagement.
- Apply email marketing and conversion optimization techniques effectively.
- Demonstrate the ability to work as a digital marketing expert or freelancer.
- Contribute to organizational growth through strategic online branding.

Course Benefits:

- Learn directly from industry professionals with real-world experience.
- Gain hands-on exposure through live projects and campaign simulations.
- Improve career and freelancing opportunities in Pakistan and abroad.
- Master advanced advertising techniques across Google, Facebook, Instagram, and LinkedIn.
- Build confidence in content creation, marketing strategy, and analytics reporting.
- Understand how to use AI tools and automation to improve performance.
- Earn a recognized certification that enhances professional credibility.

Skill-Wise Earnings:

Skill Level	Avg Monthly Salary
Junior	75k-100k
Mid-Level	100k - 170k

Advanced	250k- 450k
Freelancer	Earn in millions

Affiliation & Collaboarations

PBTE

PSDA