Here's a 40-hour course syllabus for an AI QA Engineer, drawing upon the provided "AI QA Engineer – Master Syllabus.pdf" and structured for practical learning.

Course Title: AI QA Engineer: Testing AI & LLM Applications

Course Duration: 40 Hours

Target Audience: Experienced QA engineers, ML engineers, and AI product testers looking to specialize in testing AI/ML systems, particularly LLM and RAG applications.

Course Goals:

- Understand fundamental AI/ML concepts relevant to QA.
- Master strategies and techniques for testing AI models, data pipelines, and LLM/RAG systems.
- Gain hands-on experience with key tools and frameworks for AI/ML testing, including DeepEval, RAGAS, and Ollama.
- Learn to integrate AI QA into CI/CD pipelines and monitor AI system performance.
- Develop skills in responsible AI testing, including bias, fairness, and explainability.

Module 1: Foundations of AI/ML for QA (8 Hours)

- **Overview of AI, Machine Learning, and Deep Learning:**
  - Definitions and key differences.
  - Supervised, Unsupervised, and Reinforcement Learning paradigms.
  - Common ML algorithms (Linear Regression, Decision Trees, Neural Networks) and their implications for testing.
- **The AI/ML Lifecycle from a QA Perspective:**
  - Data lifecycle and preprocessing (ingestion, transformation, validation).
  - Model training, validation, testing, and understanding overfitting/underfitting.
  - Introduction to AI/ML pipelines and MLOps basics.
- **Traditional QA vs. AI QA Mindset:**
  - Deterministic vs. probabilistic systems and their impact on testing approaches.
  - Challenges unique to AI systems (data quality, model drift, bias).
- **Hands-on:**
  - Simple classification model with Scikit-learn (Python).
  - Explore data preprocessing steps using Pandas.

Module 2: Data Quality & Validation for AI Systems (8 Hours)

- **Testing Data Pipelines:**

- Ingestion, transformation, and model input/output validation.
  - Schema validation.
  - Identifying and handling nulls, outliers, and duplicates.
  - Record-level vs. aggregate-level validation.
  - Testing joins, aggregations, and transformation logic.
- **Tools for Data Quality Automation:**
  - Introduction to Great Expectations and Pandera for schema and data validation.
  - SQL validation for data integrity.
- **Synthetic Data Generation & Test Data Augmentation:**
  - Understanding their role in AI testing.
- **Hands-on:**
  - Validate a dataset using Great Expectations and SQL.
  - Implement data quality checks for a sample dataset.

Module 3: Testing AI Models & Strategies (8 Hours)

- **Evaluating AI Models:**
  - Key metrics: Accuracy, Precision, Recall, F1 Score.
  - Confusion matrix analysis.
  - Understanding drift detection (data drift, concept drift) and its importance.
  - Bias and fairness testing.
- **Black-box & White-box Testing for ML Models:**
  - Techniques and considerations for each approach.
- **Hands-on:**
  - Calculate and interpret model evaluation metrics (accuracy, precision, recall, F1) manually in Python.
  - Analyze a confusion matrix for a classification model.
  - Perform a basic bias/fairness test on a sample model.

Module 4: Testing LLM & RAG Applications (8 Hours)

- **Foundations of LLM Testing:**
  - Evaluation challenges in LLMs and RAG systems.
  - What to test: accuracy, hallucination, grounding, relevance.
  - Anatomy of LLM apps (Prompt → LLM → Output).
  - Types of LLM testing: prompt evaluation, response evaluation, factuality.
  - Hallucinations vs. grounded responses.
  - Overview of RAG (Retrieval-Augmented Generation).
  - Metrics for LLMs: BLEU, ROUGE, BERTScore, faithfulness, toxicity, helpfulness.
- **Setting Up Local LLMs with Ollama:**

    - Installing and configuring Ollama.
    - Pulling and running local LLMs (e.g., LLaMA2, Mistral, Phi).
    - Testing latency, output length, and resource usage with Ollama.
    - Fine-tuning/testing prompt templates for RAG apps locally.
- **Evaluating with DeepEval:**
    - Installation and test suite structure for DeepEval.
    - Creating evaluation test cases using StringMatchEvaluator, ContextualEval (faithfulness), AnswerRelevancyEvaluator, ToxicityEval.
    - Writing tests for different tasks: summarization, QA, chatbot responses.
- **Validating RAG Pipelines with RAGAS:**
    - Overview of RAGAS metrics: Context Precision, Context Recall, Faithfulness, Answer Correctness.
    - Testing chunking strategy, retrieval accuracy, hallucination risks.
    - Connecting RAGAS to LangChain or LlamaIndex.
- **Hands-on:**
    - Set up Ollama and run a local LLM.
    - Write DeepEval test cases for LLM outputs (e.g., answer relevancy).
    - Explore RAGAS for evaluating a simple RAG system (conceptual walkthrough/demonstration).

Module 5: Automation, MLOps, and Responsible AI (8 Hours)

- **Test Automation for Data-Driven Systems:**
    - Python + PyTest/Robot Framework + Pandas for automation.
    - Testing ML APIs (REST, GraphQL) using tools like Postman.
- **CI/CD Integration for AI Pipelines:**
    - Integrating test automation into CI/CD using GitHub Actions or Jenkins.
    - Model versioning and deployment tracking (MLflow).
    - Testing model deployment workflows (batch vs. real-time).
- **Cloud & MLOps Integration:**
    - QA in cloud-based ML workflows (AWS Sagemaker, Azure ML, GCP Vertex AI).
    - Infrastructure-as-Code (Terraform, CloudFormation) for managing environments.
- **Monitoring & Logging AI Systems:**
    - CloudWatch, ELK stack, Prometheus, Grafana.
    - Model serving logs and error tracking.
    - Canary releases and A/B testing of models.
    - Tracking performance drifts and degradation in model versions.
- **Responsible AI Testing:**
    - Explainability (SHAP, LIME).
    - Ethical testing: bias, fairness, and transparency.
    - Testing for adversarial robustness.
    - Data privacy & security validations (GDPR, HIPAA).
    - Human-in-the-loop systems and QA implications.

- **Hands-on:**
  - Set up a basic CI pipeline (e.g., GitHub Actions) to run model code tests.
  - Explore SHAP/LIME for model explainability (demonstration/conceptual understanding).
  - Discuss and analyze a case study on bias detection in an AI system.