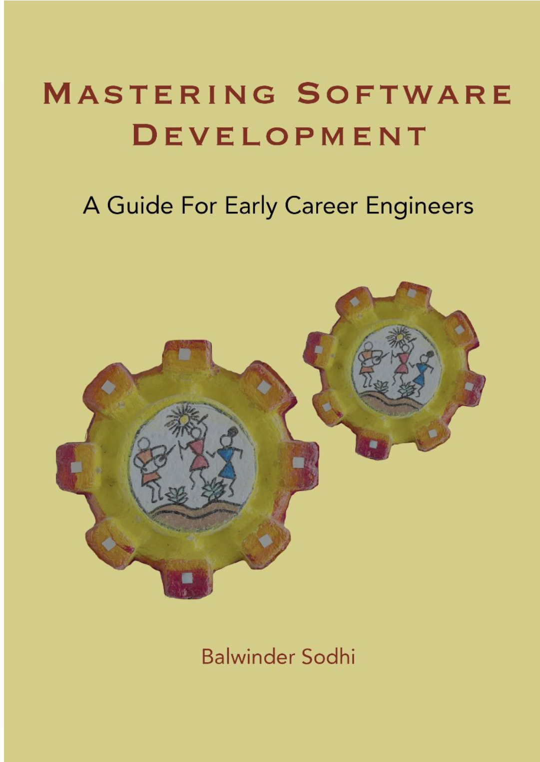


Architecting Software in the Era of AI/ML

Balwinder Sodhi

Why AI/ML Changes Architecture

- Software is shifting from deterministic logic → systems that learn from data.
- Architecting AI/ML systems requires:
    - Handling probabilistic behaviour.
    - Managing continuous learning loops.
    - Designing around data, models, pipelines, and feedback flows.
- Opportunities include personalization, automation, intelligent decision-making, and large-scale pattern recognition.

Opportunity — Intelligent Automation

- Replace manual workflows with predictive or generative components.

Examples:

- Automated document classification.
- Forecasting models in supply chain.
- LLM-based assistants embedded into enterprise workflows.

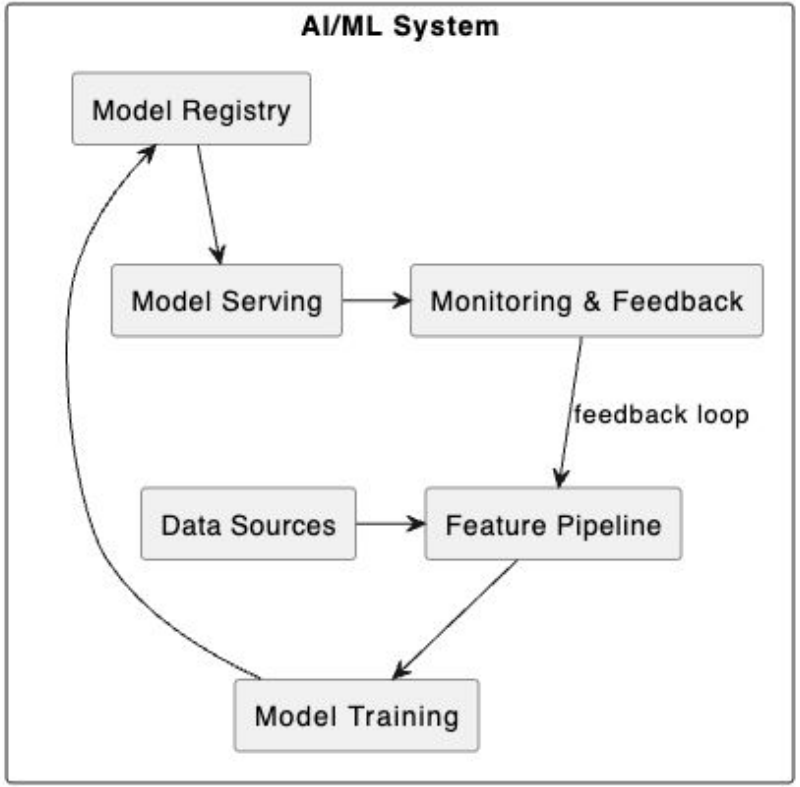
Opportunity — Data-Driven Decisions

- Shift from rule-based engines to model-driven predictions.
- Models adapt as new data arrives instead of manually updating rules.
- Enables analytics → prescriptive insights → autonomous actions.

Opportunity — Large-Scale Personalization

- Recommendations, dynamic workflows, and content ranking.
- Systems evolve with user behaviour.
- Requires architecting around:
  - Real-time event flows.
  - High-throughput feature pipelines.
  - Fast inference pathways.

General Architecture Pattern (High-Level)



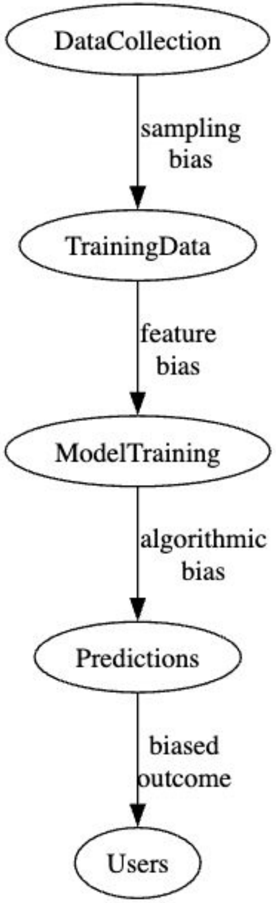
Important Issues and Threats in AI/ML Software Architecture

Ethical and Bias Issues

- What they are & why relevant
  - Models amplify patterns in data—including harmful or discriminatory ones.
  - Bias leaks into hiring systems, loan approval, fraud detection, healthcare triage.
  - Regulatory expectations: auditability, fairness, transparency.
- Architectural Implications
  - Need data lineage and traceability.
  - Model explainability components must be built-in.
  - Human oversight and ethical gates in MLOps workflow.
  - Bias metrics integrated into CI/CD pipeline for ML.
- Mitigation
  - Bias detection tools (Fairlearn, AIF360).
  - Diverse and representative datasets.
  - Ethical review boards and “responsible AI checks” before deployment.
  - Shadow deployments with human monitoring.

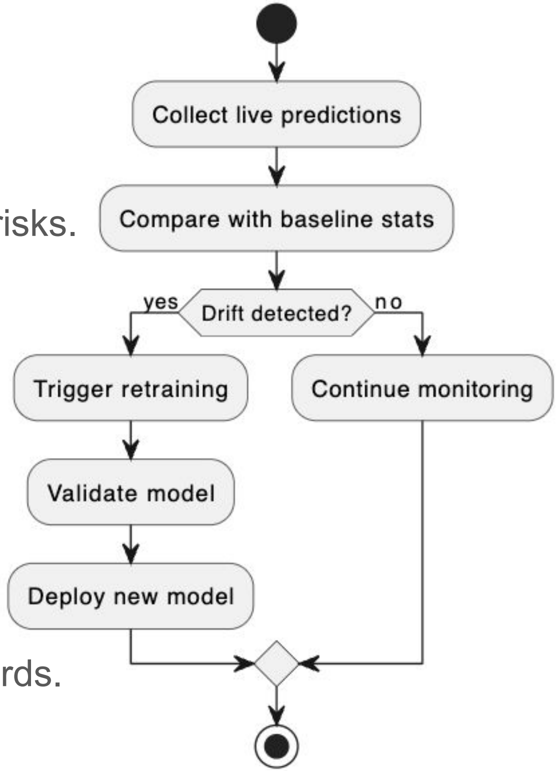
Bias in AI Systems

- Bias originates from:
  - Historical data skew.
  - Poor sampling.
  - Feature selection mistakes.
  - Model architecture biases.
- Architectural responsibility:
  - Build fairness checks.
  - Enable dataset auditing.
  - Track lineage for every model and dataset.



Model Drift and Data Quality

- What they are & why relevant
  - Data drift: Input feature distribution changes.
  - Concept drift: Target variable relationship changes.
  - Leads to degrading accuracy and financial or operational risks.
- Architectural Implications
  - Requires continuous monitoring pipelines.
  - Real-time tracking of feature stats.
  - Versioned datasets and models.
  - Canary deployments and performance baselines.
- Mitigation
  - Automated retraining triggers.
  - Human review workflows when drift crosses thresholds.
  - Better observability: data quality scorecards, drift dashboards.



Data Privacy and Security

- What they are & why relevant
  - AI pipelines often involve personal or sensitive data.
  - Violations can lead to legal penalties and user distrust.
  - Attack surfaces expand due to data stores, model APIs, and training artefacts.
- Architectural Implications
  - Need access controls, encryption, secure data transmission.
  - Differential privacy for training.
  - Federated learning where raw data should stay on-device.
  - Model extraction and poisoning threats must be considered.
- Mitigation
  - Data minimisation and anonymisation.
  - Zero-trust data pipelines.
  - Model-level defences (e.g., adversarial training).
  - API-level throttling, auth, WAF.

Data Privacy and Security: Key Threats

- Data leakage during:
    - Collection
    - Processing
    - Transfer
    - Model training
- Attacks:
    - Membership inference (finding if someone is in training data)
    - Model extraction
    - Prompt injection (LLM-specific)
    - Data poisoning

Data Privacy and Security: Architectural Mitigations

- Differential privacy.
- Secure enclaves for training/inference.
- Encryption at rest and in transit.
- RBAC for data and features.
- Governance boundaries between raw data, feature store, and models.

Model Interpretability and Explainability

- What they are & why relevant
  - Engineers and business stakeholders need visibility into how AI makes decisions.
  - Required for trust, debugging, and regulatory compliance.
- Architectural Implications
  - Explanation layer as a service.
  - Capture model inputs/outputs for inspectability.
  - Integrate LIME/SHAP or model-specific explanation tools.
  - Logs and reason traces must be stored.
- Mitigation
  - Use interpretable models where possible.
  - Provide global and local explanations.
  - Embed explanation APIs alongside prediction APIs.

Infrastructure and Scalability

- What they are & why relevant
  - AI/ML workloads require GPUs/TPUs, rapid scale-out, and efficient data movement.
  - Training and inference have very different needs.
- Architectural Implications
  - Separate training and inference clusters.
  - Autoscaling for inference load.
  - Feature store for low-latency, consistent datapoints.
  - Batch vs. streaming pipelines.
- Mitigation
  - Use managed ML services or containerised GPU workloads.
  - Caching and model compression (quantisation, distillation).
  - On-demand scaling of compute nodes.

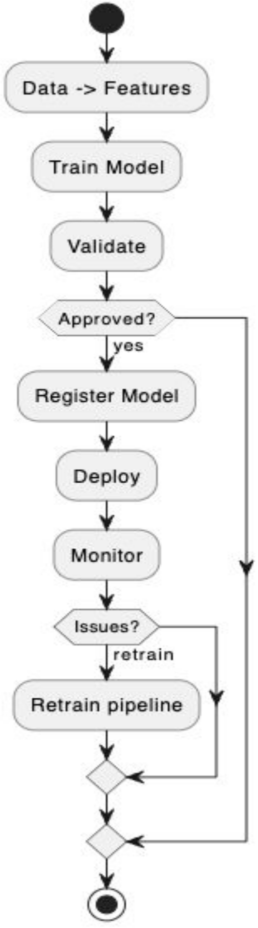
How Is Architecting Software Different with AI/ML in the Mix?

Data as the Primary Asset

- Data quality > model complexity.
- Architect for:
  - Data lineage tracking.
  - Versioned datasets.
  - Feature stores as first-class components.

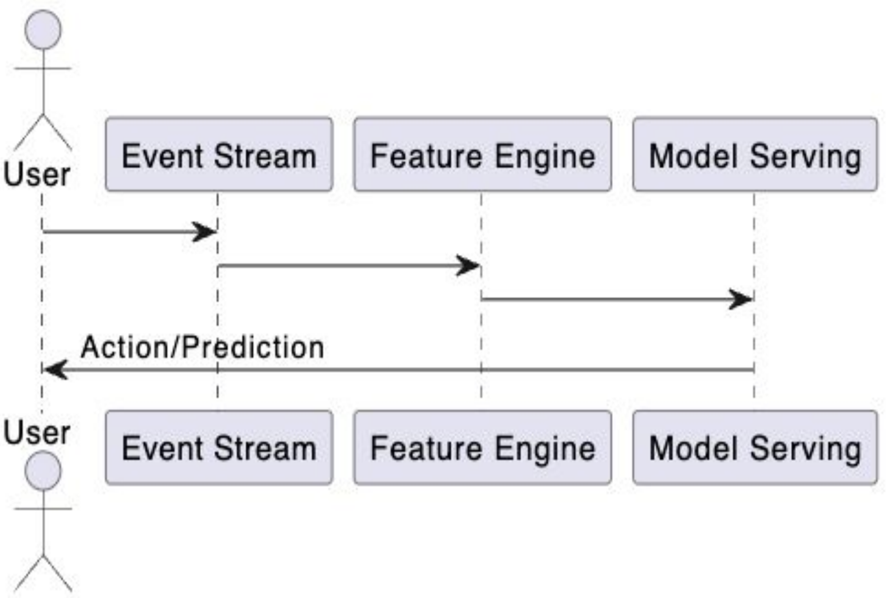
Model Lifecycle Management

- Continuous iteration:
    - Data ingestion
    - Feature engineering
    - Training
    - Validation
    - Deployment
    - Monitoring
    - Feedback-driven retraining



Real-Time Processing and Decision-Making

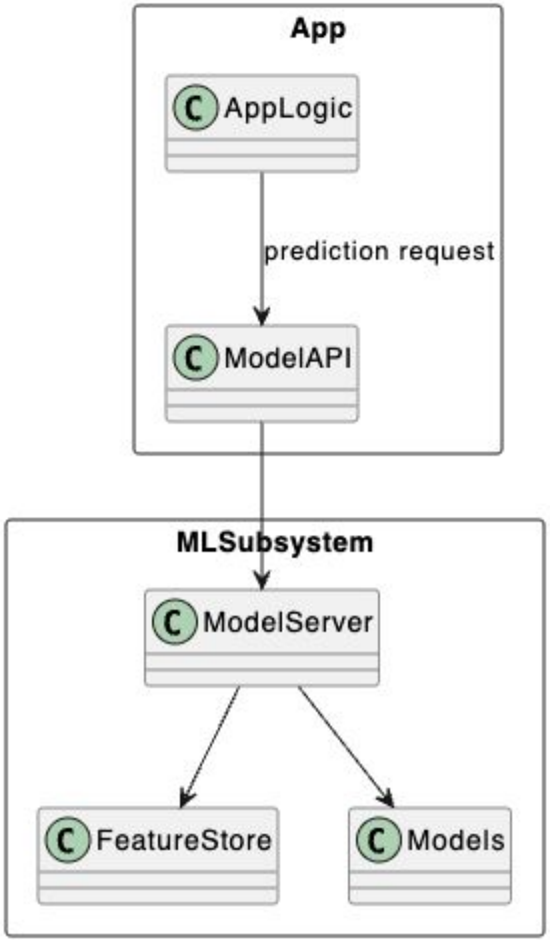
- Event-driven flows.
- Stream processors (Flink, Kafka Streams).
- Low-latency inference (tens of milliseconds).
- Useful in fraud detection, recommendations, IoT.



Decoupling AI/ML Components

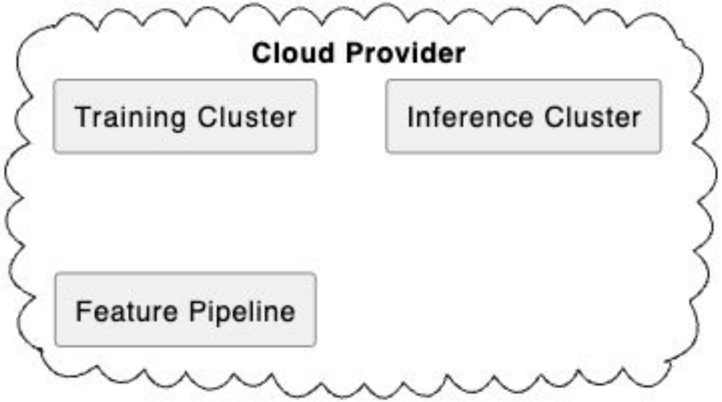
Why Decouple?

- Models evolve independently from app logic.
- Multiple models coexist (AB tests, shadow deployments).
- Avoid entangling business logic with ML pipelines.



Infrastructure for AI/ML: Types of Compute

- Training:
  - GPU/TPU clusters
  - Distributed training frameworks
- Inference:
  - Latency-sensitive → CPU/GPU autoscaling
  - LLM inference → vLLM / TensorRT-LLM / spec decoding



Explainability and Transparency

Making AI Behaviour Inspectable

- Provide APIs for:
  - Model confidence
  - Reason tokens for LLMs
  - SHAP explanations
- Integrate explanations into monitoring dashboards.

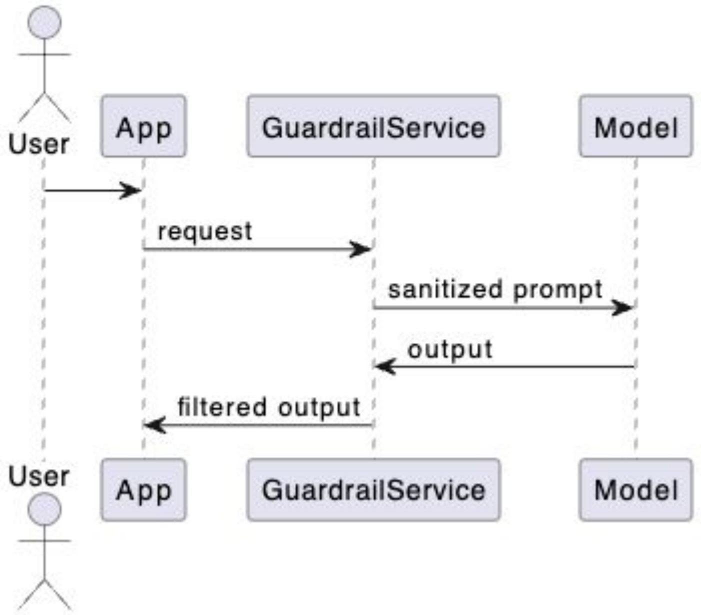
Ethical AI and Bias Management

Architectural Must-Haves

- Bias testing pipeline.
- Dataset audit trails.
- Transparency logs.
- Human-in-the-loop approval for high-impact decisions.

AI-Specific Security Requirements

- Guardrails for prompt injection.
- Response filtering pipelines for LLMs.
- Training data isolation.
- Identity-aware feature access.



Governance and Compliance

- Model lineage.
- Dataset versioning with metadata.
- Deployment audit logs.
- Access control around models and features.
- Alignment with:
  - GDPR
  - HIPAA
  - EU AI Act (high-risk systems)

Key Takeaways

- AI/ML systems demand new architectural thinking:
  - Data-first mindset
  - Model lifecycle as a continuous loop
  - Decoupled, observable, governable components
- Ethical, security, and operational concerns are integral—not optional.
- Real-world AI/ML architectures must balance:
  - Performance, Interpretability, Stability, Safety, Compliance
- The architect's role evolves to include:
  - MLOps integration
  - Data governance
  - Responsible AI