Zaalima Development - Data Science & AI Division: Q4 Production AI Project Assignments

To: Machine Learning Engineering Track (Cohort Sigma)

From: Principal Data Scientist & AI Architect

Date: November 30, 2025

Subject: Q4 Production AI Project Assignments: Building Production-Grade MLOps Systems-----**A Call to Production**

Team, let this be explicitly clear: **Stop training models on the Titanic dataset.** That era is over. The goal of this program is not to produce highly-paid hobbyists. In the real world, data is messy, models drift in production, resource latency matters, and business-critical decisions rely on your code. You are here to build **Production-Grade AI Systems**, not just Jupyter Notebook experiments.

The next four weeks are your final, intensive crucible. We are adopting a strict, non-negotiable **MLOps-first mindset**: if you can't containerize it, monitor it, deploy it, and automatically re-train it, it doesn't exist. Your success will be measured by system resilience and latency, not just the F1-score on a static test set.

The following four projects are designed to cover the entire spectrum of modern Data Science and Machine Learning engineering:

1. **Tabular ML & Time-Series:** Predictive Maintenance (Industrial IoT).
2. **Deep Learning (NLP):** Intelligent Document Processing.
3. **Deep Learning (Computer Vision):** Automated Visual Quality Control.
4. **Big Data Processing:** Scalable Churn Prediction (Apache Spark).

-----Contents - Project Index

-----1. Tech Stack - The Zaalima Production AI Architecture

We utilize a modern, hybrid stack that surgically balances rapid, local experimentation with scalable, distributed deployment. Mastery of these tools is non-negotiable.

Category	Primary Tools	Purpose & Production Context
Data Foundation	**Pandas & NumPy**	**Vectorization & Local ETL.** Used for smaller datasets, high-performance feature vectorization, and data manipulation up to a few GB. Efficient serialization via `joblib` or `pickle` is mandatory for moving features to the model.
	Apache Spark (PySpark)	**Big Data Processing & Distributed ETL.** Essential for Project 4. When data volumes exceed RAM (Terabytes), Spark handles distributed processing, data aggregation, and massive joins across a cluster.
Modeling Engines	**Scikit-Learn**	**Baseline & Feature Pipelines.** The robust industry standard for classical ML (Logistic Regression, SVM, Random Forest). Used for establishing initial baselines and building reusable pre-processing pipelines.
	XGBoost/LightGBM	**High-Performance Tabular ML.** The "Kaggle Winners." Use these gradient boosting frameworks for state-of-the-art results on structured, tabular datasets.
	TensorFlow/Keras	**Deep Learning Framework.** The workhorse for all complex Neural Network architectures in NLP (LSTMs,

		Transformers) and Computer Vision (CNNs, ResNets).
Development & MLOps	**Jupyter Notebooks**	**Strictly for EDA & Prototyping.** Must be refactored into production-ready `.py` scripts before deployment. A notebook is an exploratory tool, not a deployable artifact.
	PyCharm/VS Code	**Production Code IDEs.** Used for writing modular, tested, production-ready Python modules and functions.
	MLflow (Bonus)	**Experiment & Model Versioning.** Highly recommended for tracking all experiments, logging parameters, metrics, and model artifacts for seamless reproducibility.
	Docker	**Containerization.** Mandatory for Project 2 and Project 4 (simulating a cluster). All API endpoints must be containerized to ensure environment parity between development and production.

-----2. Project 1: Manufacturing - IoT Predictive Maintenance Engine

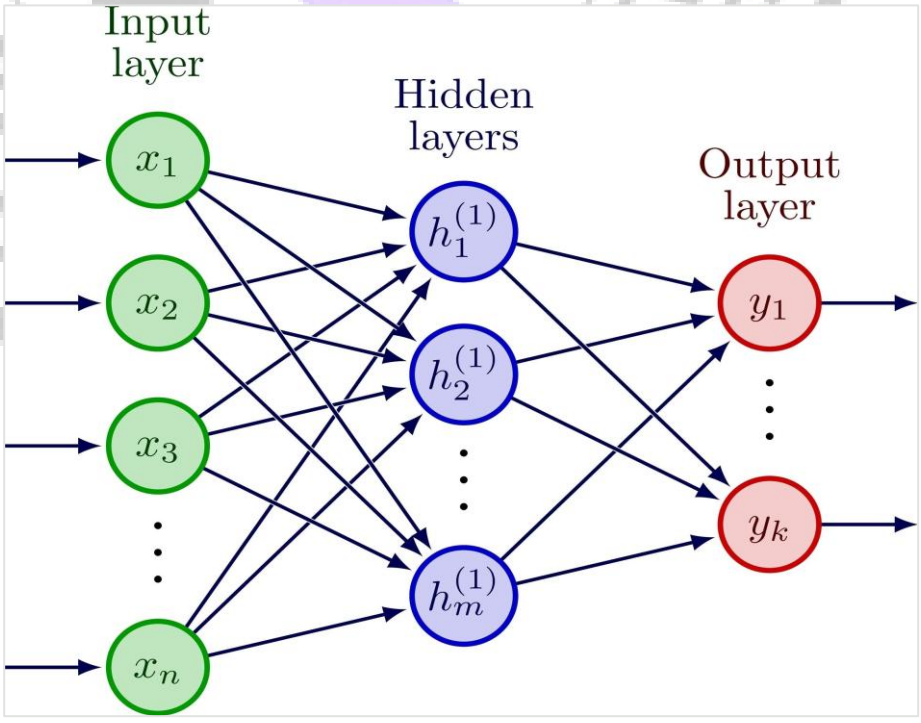
Project Title: IoT Predictive Maintenance Engine (Time-Series Classification)
Product Brand Name: "FactoryGuard AI"
Use Case (Production): A critical manufacturing plant floor contains 500 robotic arms with vibration, temperature, and pressure sensors. The objective is to predict a catastrophic failure **24 hours before it occurs** to allow for scheduled, preemptive maintenance, avoiding millions in unscheduled downtime.

Area	Production Requirements & Features	Implementation Details
Feature Engineering	**Advanced Rolling Window Statistics:** Creating time-series features such as Rolling Mean, Exponential Moving Average, Standard Deviation of sensor readings over the last 1, 6, and 12 hours. **Lag Features** (t-1, t-2) are essential.	Focus on Pandas for feature creation; efficient serialization with `joblib`.
Modeling	**Model Selection:** Start with a simple Logistic Regression or Scikit-Learn Random Forest as a baseline. The production model must be **XGBoost** or **LightGBM** for maximum predictive power.	**Week 2:** Hyperparameter Tuning via `GridSearchCV` or the advanced `Optuna` library.
Imbalance Handling	**Class Imbalance is Extreme:** Failures are rare (typically <1% of the data). **Do NOT use Accuracy.** Must use **Precision-Recall Area Under Curve (PR-AUC)**. Handle imbalance using **SMOTE** or, preferably, by adjusting **Class Weights** in the model.	Use the `imbalanced-learn` library. Prioritize High Precision (avoiding false alarms).
Explainability	**Model Trust & Adoption:** Use **SHAP (SHapley Additive exPlanations)** values to generate local explanations. The maintenance engineer needs to know *why* the model predicted a failure (e.g., "The rolling mean of temperature	**Week 3:** Integrate SHAP visualization into your report.

	exceeded 80°C and the variance of vibration tripled").	
Deployment	**Real-Time API:** Save the final model and pre-processing pipeline using `joblib`. Build a simple **Flask API endpoint** that accepts a JSON payload of current sensor readings and returns a failure probability (`<50ms response time`).	**Week 4:** Latency check is a critical review point.

-----3. Project 2: Fintech - Intelligent Document Processing (NLP)

Project Title: Legal Contract Entity Extractor (NER)
Product Brand Name: "LexiScan Auto"
Use Case (Production): A large financial law firm manages millions of PDF contracts. Manual review is slow and error-prone. The system must automatically extract key, structured entities: **Dates, Party Names, Dollar Amounts, and Termination Clauses** to automatically index and search the documents.



Area	Production Requirements & Features	Implementation Details
Data Acquisition	**OCR Integration:** The system must handle both native digital and **scanned PDF documents**. Integrate the **Tesseract OCR pipeline** to convert pixels to raw, processable text before NLP begins.	**Week 1:** Focus on text quality and noise reduction from OCR.
Core Modeling	**Named Entity Recognition (NER):** This is a custom NER task. Training a deep learning model to recognize legal-specific entities.	Use a **Bi-Directional LSTM** (TensorFlow/Keras) or fine-tune an industrial-grade library like **Spacy** on your custom annotated data.
Performance & Context	**Transfer Learning:** Fine-tuning a pre-trained contextual embedding (e.g., GloVe, or a small **BERT variant**) for superior context awareness, significantly boosting F1-scores.	**Week 2:** The F1-Score on entity extraction is the key metric. Annotation using tools like **Doccano** is required for training data.
Post-Processing	**Rule-Based Validation:** Raw NER output is often imperfect. Implement logic to validate extracted entities (e.g., "Date" must conform to `YYYY-MM-DD` format; "Amount" must contain currency symbols).	**Week 3:** Robust handling of edge cases and noise is paramount for lawyer trust.
Deployment	**Containerized Microservice:** The entire OCR + NLP pipeline (which can be memory-intensive) must be wrapped in a **Docker container**. Expose the functionality via a resilient **REST API** endpoint that accepts the PDF/text and	**Week 4:** The end-to-end test must pass: Upload PDF -> Receive structured JSON.

	returns a JSON payload of extracted entities.	

-----4. Project 3: Retail - Automated Visual Quality Control (Computer Vision)

Project Title: Automated Quality Control (Defect Detection)
Product Brand Name: "VisionSpec QC"
Use Case (Production): A high-speed manufacturing line for PCBs (Printed Circuit Boards) requires real-time quality inspection. The AI must process an image captured by a camera and immediately classify it as **"Pass" or "Defect"** (and provide localization of the defect) with high throughput.

Area	Production Requirements & Features	Implementation Details
Data Preparation	**Robust Data Augmentation:** Real-time data augmentation (rotation, zooming, flipping, brightness shifts) using the Keras `ImageDataGenerator` during training to ensure the model generalizes across lighting and camera variability.	**Week 1:** Visualize augmented batches to confirm they are realistic but varied.
Core Modeling	**Transfer Learning for Accuracy:** Building a CNN from scratch is inefficient. Use a pre-trained, powerful architecture like **ResNet50** or a mobile-optimized one like **MobileNetV2**. Freeze the base layers and fine-tune only the custom classification head.	**Week 2:** Focus on achieving a low validation loss without overfitting (checked via learning curves).

Interpretability	**Visual Verification (Grad-CAM):** To build trust, the system must show *where* it is looking. Implement **Grad-CAM** (Gradient-weighted Class Activation Mapping) to generate a visual heatmap overlaid on the image, highlighting the specific area of the defect.	**Week 3:** Verify that heatmaps highlight the actual soldering joint or component, not background noise.
Inference Optimization	**High-Speed Inference:** The model needs to run at production line speed (e.g., >10 frames per second). Optimize the model artifact (e.g., saving as an optimized `.h5` or `SavedModel`).	**Week 4:** Write a Python script using **OpenCV** to simulate a live webcam feed and perform frame-by-frame prediction. **Live demo is mandatory.**

-----5. Project 4: Telecom - Big Data Churn Prediction (Apache Spark)

Project Title: Scalable User Churn Prediction (Big Data ML)
Product Brand Name: "SparkScale Churn"
Use Case (Production): A major telecom company generates **2 Terabytes** of user log data (call duration, data usage, complaints, billing). This volume cannot be loaded into a single machine's RAM. The objective is to build a distributed ML pipeline to predict which high-value users will cancel their subscription next month.

Area	Production Requirements & Features	Implementation Details
Data Processing	**Distributed ETL & Storage:** Using **PySpark RDDs and DataFrames** to ingest and process massive, distributed CSV or Parquet files across a simulated cluster environment.	**Week 1:** Setup a local Spark Cluster (Master + 2 Workers) using **Docker**. Focus on PySpark schema validation and row counts.

Feature Engineering	**Spark SQL for Aggregation:** Feature engineering must happen at scale. Utilize **Spark SQL** to calculate complex user-level aggregates (e.g., "Average data usage in the last 90 days," "Total complaint count"). Use **VectorAssembler** to combine features.	**Week 2:** Analyze the Spark Execution Plan (DAG visualization) to identify bottlenecks.
Modeling	**Spark MLlib Native:** Scikit-Learn cannot be used. We must leverage Spark's native, distributed ML library (**pyspark.ml**). Train models such as Distributed Logistic Regression and **Distributed Random Forest**.	**Week 3:** Setup the `BinaryClassificationEvaluator`. The focus is on the *training time* and *scalability* of the model, not just accuracy.
Deployment	**Persisted Pipeline:** The entire sequence of transformations (Indexers, Aggregators, Model) must be saved as a single, reusable **Spark Pipeline** object. This is the production artifact.	**Week 4:** Write a reusable batch script that loads the saved pipeline, takes in new monthly user data, and generates the final churn probability predictions, simulating a "Daily Batch Job" run.

-----Submission Requirements:

All project assets, including source code, documented pipelines, and the final deployable artifact (Flask API, Docker image, or Spark script), must be available in a dedicated Git repository. The repo link must be submitted to the Zaalima LMS by **Friday 17:00 of Week 4.** Ensure a comprehensive `requirements.txt` file is present to guarantee a fully reproducible environment.

Zaalima Development

Algorithms define reality. Code responsibly.