Instructions And Sample Tasks

Guidelines for Participants:

1. You initiate the hackathon by forking the Git repository given by **Python Guru**, which contains instructions and guidelines for the event.
2. Your initial change must involve updating the README.md file to include a comprehensive problem statement and solution description.
3. You are required to exhibit the implementation of solutions in the domains of Generative AI (ChatBots, RAG, Agents, and Agentic AI) and Machine Learning technologies utilizing Python, JavaScript, or TypeScript.
4. You must possess foundational knowledge of Machine Learning principles, LLM parameters, embeddings, prompt engineering, context engineering, RAG, agents, agentic AI, MCP, LangChain, ChromaDB, and token generation.
5. **It is beneficial to possess understanding regarding Guardrails and Evaluations which has additional weightage for evaluation**.
6. If you are utilizing local LLMS, ensure they are downloaded and prepared prior to the hackathon, since they require significant internet bandwidth. If employing external APIs such as OpenAI, Gemini, or Anthropic, you must provide your API key, as **Python Guru** does not supply one.
7. You are permitted to utilize Generative AI tools such as ChatGPT, Gemini, Perplexity, and coding tools like Copilot and Cursor; however, you must retain conversation history and safeguard it from deletion.
8. Select a problem statement from any domain, preferably: a) Education b) Finance c) Healthcare d) Telecom e) Productivity f) Technological Innovation. Sample problems are included in the attached document for your comprehension; you may select and adapt the ideas presented.

9. It's nice to have the user interface(UI/Frontend), but it doesn't help with review much because managing time during a hackathon is very important.

Assessment Standards

1. 25% Innovation
2. 25% Technical Implementation
3. 25% Utilization of Artificial Intelligence
4. 15% Impact and Expandability
5. 10% Presentation

Submission Checklist

- Updated README.md (problem, data link, design, assumptions).
- Reproducible Notebook(s) and/or minimal FastAPI service (no UI required).
- requirements.txt / environment.yml and run commands.
- Evaluation notes (metrics, tests, guardrails, limitations).
- Commit history & AI chat logs (attach/export or link).
- **A 10min demonstration video has to be recorded should be shared through YouTube link**

☐ **Data requirement:** Every assignment below includes a **sample dataset link** (Kaggle or public web). If a link becomes inaccessible, choose another assignment.

FINANCE — 10 Assignments

- F1)Automated MD&A Draft from Financials (RAG + Summarization)
- Problem: Generate first-draft MD&A style narrative from tabular financial statement extracts.
  Sample data: Financial Statement Extracts (SEC) – https://www.kaggle.com/datasets/securities-exchange-commission/financial-statement-extracts
  Outcome (24h): Notebook + script that: (a) loads statements, computes YoY/QoQ deltas & KPIs; (b) chunks filings; (c) produces sectioned markdown draft (trends, revenue drivers, risks) via LLM with citations to source chunks.
  Suggested stack: Python, Pandas, LangChain, embeddings (e.g.,

text-embedding-3-small), a chat model (OpenAI/Gemini/Claude or local), ChromaDB/FAISS, pydantic for schema.

- F2) Portfolio Recommender Agent (Risk-profile → Allocation)
- **Problem**: Build an agent that proposes ETF/stock allocations given risk tolerance & horizon.
  Sample data: Stock Portfolio Data with Prices & Indices – https://www.kaggle.com/datasets/nikitamanaenkov/stock-portfolio-data-with-prices-and-indices
  Outcome: Notebook that computes mean/variance, runs basic MPT or Black-Litterman variant, then explains the allocation in plain English (LLM). Export JSON of weights.
  Stack: Python, numpy/pandas, cvxpy or PyPortfolioOpt, LangChain agent with tool-calls, FastAPI (optional) to expose /recommend.

F3) Financial News Sentiment Classifier

Problem: Classify news headlines/sentences as positive/negative/neutral for market context.
Sample data: Sentiment Analysis for Financial News – https://www.kaggle.com/datasets/ankurzing/sentiment-analysis-for-financial-news
Outcome: Baseline classical model + LLM zero-shot comparison; confusion matrix + short error analysis.
Stack: Python, scikit-learn, HuggingFace transformers (optional), LangChain eval prompts.

F4) Transaction Fraud Detection (Imbalanced Classification)

Problem: Detect fraudulent payments in a skewed dataset; report precision/recall@k.
Sample data: PaySim (Synthetic Financial Fraud) – **https://www.kaggle.com/datasets/ealaxi/paysim1**
Outcome: Notebook with feature engineering, stratified split, class-weighting/SMOTE experiment, and threshold tuning; export fraud_score CSV.
Stack: Python, scikit-learn/lightgbm, imbalanced-learn, MLflow (optional).

F5) Loan Approval Prediction

Problem: Predict loan approval outcome from applicant features; provide explainability.
Sample data: Loan Approval Classification – https://www.kaggle.com/datasets/taweilo/loan-approval-classification-data
Outcome: Model + SHAP (top 5 drivers) + fairness quick-check on selected attributes.
Stack: Python, scikit-learn, SHAP, FastAPI endpoint /score.

F6) Credit Score Bucketizer

Problem: Predict credit score category and generate a human-readable rationale. **Sample data**: **Credit Score** – https://www.kaggle.com/datasets/conorsully1/credit-score
Outcome: Model for class prediction + LLM template that translates SHAP into actionable

advice.
Stack: Python, scikit-learn, SHAP, LangChain.

F7) SEC Filing Summarizer & Q&A (RAG)

Problem: Query 10-K/10-Q filings and answer investor questions with source citations.
Sample data: SEC Filings – https://www.kaggle.com/datasets/kharanshuvalangar/sec-filings
Outcome: Index a small subset; implement ask(question) returning answer + chunk URLs.
Stack: Python, unstructured, LangChain, embeddings + vector store, FastAPI /ask (optional).

F8) Market Data Forecaster & Alert Agent

Problem: Forecast next-day or next-week close for a small set of symbols; trigger rule-based alerts.
Sample data: Stock Market Dataset for Financial Analysis –
https://www.kaggle.com/datasets/s3programmer/stock-market-dataset-for-financial-analysis
Outcome: Baseline ARIMA/Prophet vs. simple ML; CSV of predictions; text alert rationale via LLM.
Stack: Python, Prophet or statsmodels, Pandas, LangChain for explanations.

F9) Personal Expense Categorization Assistant

Problem: Auto-tag transactions into categories; produce a monthly summary.
Sample data: Personal Budget Transactions –
https://www.kaggle.com/datasets/ismetsemedov/personal-budget-transactions-dataset
Outcome: Rules + model hybrid; confusion matrix; exported category and monthly rollup.
Stack: Python, scikit-learn, regex heuristics, LangChain for ambiguous descriptions.

F10) Contract Clause Extractor (NLP)

Problem: Extract confidentiality/termination/liability clauses from contracts and summarize risk.
Sample data: Contracts Clauses Dataset –
https://www.kaggle.com/datasets/mohammedalrashidan/contracts-clauses-datasets
Outcome: Notebook that tags clause spans, then LLM produces risk-focused summaries per clause.
Stack: Python, spaCy/HF token-classification, LangChain summarization.

HEALTHCARE — 10 Assignments

Image data tasks should use small subsamples; no PHI; prefer openly accessible datasets.

H1) Doctor's Note Transcription & Structuring (Speech → SOAP)

Problem: Convert short doctor dictations to text and structure into SOAP fields.
Sample data: Audio Recording (Whisper) – https://www.kaggle.com/datasets/najamahmed97/audio-recording-whisper
Outcome: Batch transcribe 20–50 clips; JSON per note with subjective/objective/assessment/plan.
Stack: Python, OpenAI Whisper (or faster-whisper), LangChain, pydantic.

H2) Medication Reminder Chatbot (Label-Aware)

Problem: Q&A over drug labels + schedule reminders (no phone/SMS integration needed).
Sample data: openFDA Drug Label – https://open.fda.gov/apis/drug/label/download/
Outcome: RAG ask_drug(question) + sample reminder JSON plan per drug & dosage.
Stack: Python, LangChain, retrieval (ChromaDB), FastAPI (optional).

H3) Clinical Trial Matcher (Text Retrieval)

Problem: Match a sample patient profile to relevant trials.
Sample data: ClinicalTrials.gov (COVID-19 subset) – https://www.kaggle.com/datasets/parulpandey/covid19-clinical-trials-dataset
Outcome: Top-k trial IDs + justification bullets from inclusion/exclusion text.
Stack: Python, LangChain, sentence-transformers, vector store.

H4) Symptom Checker (Tabular → Preliminary Dx)

Problem: Suggest top-3 likely conditions from symptom inputs (not diagnostic).
Sample data: Disease & Symptoms – https://www.kaggle.com/datasets/choongqianzheng/disease-and-symptoms-dataset
Outcome: Multilabel or top-k ranking model + LLM explanation template.
Stack: Python, scikit-learn, LangChain.

H5) Mental Health Triage from Text

Problem: Classify depression severity / flag risk from user-provided text (ethics note & disclaimers).
Sample data: DAIC-WOZ / Depression – https://www.kaggle.com/datasets/arashnic/the-depression-dataset
Outcome: Baseline classifier + sensitivity analysis; save triage CSV with risk scores.
Stack: Python, scikit-learn or transformers, eval guards.

H6) Health Insurance Claim Fraud Detection

Problem: Identify likely fraudulent healthcare claims.
Sample data: NHIS Healthcare Claims & Fraud – https://www.kaggle.com/datasets/bonifacechosen/nhis-healthcare-claims-and-fraud-dataset

Outcome: Imbalanced classification with feature importance + brief policy rules.
Stack: Python, lightgbm/xgboost, imbalanced-learn, SHAP.

H7) Drug–Drug Interaction Checker (Graph + RAG)

Problem: Given a list of meds, flag risky interaction pairs and explain the mechanism.
Sample data: DDInter (Drug–Drug Interactions) – https://www.kaggle.com/datasets/montassarba/drug-drug-interactions-database-ddinter
Outcome: Build a small graph (NetworkX/Neo4j optional); return flagged pairs + rationale snippet.
Stack: Python, NetworkX/Neo4j (optional), LangChain.

H8) Lifestyle Coach from NHANES (Recommendation Summaries)

Problem: Generate lifestyle suggestions from basic vitals/labs.
Sample data: NHANES – https://www.kaggle.com/datasets/thedevastator/national-health-and-nutrition-examination-survey
Outcome: Simple rules/ML + templated explanations; CSV of recommendations.
Stack: Python, Pandas, LangChain.

H9) Hospital Bed Utilization Forecaster

Problem: Forecast ward-level bed occupancy 1–7 days out.
Sample data: **Hospital Beds Management** – https://www.kaggle.com/datasets/jaderz/hospital-beds-management
Outcome: Prophet/ARIMA + MAE/MAPE; markdown ops notes from LLM.
Stack: Python, Prophet/statsmodels, Pandas, LangChain.

H10) Medical Coding Assistant (ICD-10 from Notes)

Problem: Map short de-identified notes to ICD-10 codes (top-k retrieval + rerank).
Sample data: MIMIC-III Clinical Notes (open access gated) – https://physionet.org/content/mimiciii/1.4/
Outcome: Prototype retrieval over a small sample; output code candidates + justification spans.
Stack: Python, LangChain, embeddings, BM25, reranker (optional).
Note: Access requires quick registration on PhysioNet; if unavailable, swap to H1/H2/H4/H9.

TELECOM — 10 Assignments

T1) AI Customer Service Agent (RAG over Tickets & Dialogues)

Problem: Build a knowledge assistant that answers common telco support queries.
Sample data: **Telecom Agent–Customer Interaction Text** –
https://www.kaggle.com/datasets/avinashok/telecomagentcustomerinteractiontext
Outcome: /ask returns answer + source IDs; add simple escalation rules.
Stack: Python, LangChain, ChromaDB/FAISS; (optional) add Customer Support Ticket Dataset –
https://www.kaggle.com/datasets/suraj520/customer-support-ticket-dataset

T2) Cellular Signal KPI Monitor (Anomaly Spots)

Problem: Detect abnormal cells based on KPI time series.
Sample data: Cellular Network Performance Data –
https://www.kaggle.com/datasets/suraj520/cellular-network-performance-data
Outcome: Notebook that flags top-N anomalous cells per hour/day with brief LLM summary.
Stack: Python, scikit-learn/isolation-forest or Twitter ADVec, LangChain for incident text.

T3) Telco Customer Churn Prediction

Problem: Predict churn and generate "why" reasons per customer.
Sample data: Telco Customer Churn (IBM sample) –
https://www.kaggle.com/datasets/blastchar/telco-customer-churn
Outcome: Model + SHAP waterfall for 3 example customers; CSV of churn scores.
Stack: Python, lightgbm, SHAP; optional FastAPI /score.

T4) Marketing Content Generator (Data-aware)

Problem: Produce personalized copy for retention/upsell based on campaign KPIs.
Sample data: Marketing Campaign Performance –
https://www.kaggle.com/datasets/manishabhatt22/marketing-campaign-performance-dataset
Outcome: Given a segment, LLM drafts 3 variants + A/B hypothesis; store to JSON.
Stack: Python/TS, LangChain/TypeScript SDKs.

T5) IVR Voice Unit Builder (TTS)

Problem: Synthesize short IVR prompts in a consistent voice from scripts.
Sample data: LJ Speech – https://keithito.com/LJ-Speech-Dataset/
Outcome: Generate WAVs for 10 prompts; log MOS-like heuristic scores (duration, silence).
Stack: Python, Coqui-TTS or other TTS, ffmpeg; prompt crafting guardrails.

T6) Network Intrusion / Anomaly Detection

Problem: Classify traffic as normal vs. attacks; report per-class metrics.
Sample data: NSL-KDD – https://www.kaggle.com/datasets/hassan06/nslkdd (or KDDCUP'99 alternative)

Outcome: Train/test pipeline + confusion matrix; top features.
Stack: Python, scikit-learn/xgboost, imbalanced-learn.

T7) Predictive Maintenance for Base-Station Gear

Problem: Predict failure likelihood of equipment and rank top at-risk assets.
Sample data: **Predictive Maintenance** – https://www.kaggle.com/datasets/hiimanshuagarwal/predictive-maintenance-dataset
Outcome: Binary model + feature importances; maintenance short-notes via LLM.
Stack: Python, lightgbm/xgboost, LangChain.

T8) SMS Spam Filter (Telco Messaging)

Problem: Classify SMS as spam/ham for a telco messaging gateway.
Sample data: **SMS Spam Collection** – https://www.kaggle.com/uciml/sms-spam-collection-dataset
Outcome: Tokenization + baseline NB/SVM + simple profanity/URL rules; export predictions.
Stack: Python, scikit-learn, regex; optional HF pretrained text-classifier.

T9) Plan/Device Recommendation Assistant

Problem: Recommend a plan/device based on constraints (budget, 5G, RAM, etc.).
Sample data: Mobile Price Classification – https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification
Outcome: Rule + ML hybrid selection with LLM explanation; return JSON recommendation.
Stack: Python, scikit-learn, LangChain, small retrieval store for plan text.

T10) Complaint Topic Routing (Telecom subset)

Problem: Auto-route consumer complaints to teams (billing, coverage, robocalls, etc.).
Sample data: FCC Consumer Complaints (telecom categories) – https://opendata.fcc.gov/api/views/3xyp-aqkj/rows.csv?accessType=DOWNLOAD
Outcome: Text classifier + confidence; label mapping YAML; sample routing table CSV.
Stack: Python, scikit-learn/transformers, Light text cleaning; LangChain for label descriptions.

PHARMA — 10 Assignments

P1) AI-Driven Molecule Generator (Generative Chemistry)

Problem: Generate novel drug-like molecules optimized for properties such as LogP, toxicity, solubility, and binding likelihood using generative diffusion models or transformer-based SMILES generators.

Sample data: MOSES Molecule Dataset – https://github.com/molecularsets/moses OR ZINC-250k SMILES – https://www.kaggle.com/datasets/saivarunk/zinc-molecular-dataset

Outcome: Model that generates 500+ candidate molecules, filters them by basic ADMET rules, and outputs top-20 high-scoring structures in SMILES + PNG.

Stack: Python, RDKit, diffusion models / GPT-based SMILES generator, SPICE/RAPIDS (optional), property predictor (QSAR), LangChain for explanation.

P2) Automated Clinical Trial Protocol Draft (RAG over Regulatory Docs)

Problem: Generate first-draft clinical trial protocols (Objectives, Endpoints, Study Design, Visit Schedule) using RAG over ICH-GCP, past trial protocols, and disease guidelines.

Sample data: ClinicalTrial Protocol Samples – https://www.kaggle.com/datasets/parulpandey/covid19-clinical-trials-dataset AND NIH / ICH guidelines (public PDFs).

Outcome: Given study drug + phase + indication → Generate a structured protocol draft (Phase I/II/III), with citations to retrieved documents.

Stack: Python, LangChain, embeddings (OpenAI / Sentence-Transformers), ChromaDB, pydantic schema enforcement.

P3) Pharmacovigilance Signal Detection Agent

Problem: Detect early safety signals by analyzing adverse event reports, literature, social media, and safety bulletins using LLM agents + clustering.

Sample data: FAERS Adverse Event Reports (public FDA dataset) – https://fis.fda.gov/extensions/FPD-QDE-FAERS/FPD-QDE-FAERS.html OR WHO-UMC sample AE datasets (public subsets).

Outcome: LLM summarizes top signals, identifies emerging AE clusters (e.g., hepatotoxicity), and generates a weekly signal-detection dashboard (JSON + text).

Stack: Python, Pandas, scikit-learn clustering, LangChain agents, embeddings, FAISS/Chroma, evaluation prompts for risk classification.

P4) Drug Label Consistency Checker (LLM Safety Rules)

Problem: Identify inconsistencies and contradictions across sections of a drug label (Dosage, Contraindications, Warnings, ADRs).

Sample data: openFDA Drug Label – https://open.fda.gov/apis/drug/label/download

Outcome: /validate returns JSON with detected discrepancies (e.g., dosage mismatch), plus LLM-generated corrective suggestions with citations.

Stack: Python, LangChain, ChromaDB, rule-based label parsing, structured LLM outputs with guardrails (pydantic)

P5) Manufacturing Batch Deviation Analyzer (Multimodal RAG)

Problem: Analyze manufacturing batch records, sensor logs, and deviation reports to diagnose root causes of batch failures.

Sample data: Pharma Manufacturing Sensor Data (synthetic) – https://www.kaggle.com/datasets/benhamner/pharma-manufacturing-data AND Deviation text samples (public synthetic datasets).

Outcome: Multimodal RAG that takes CSV logs + PDF batch records → identifies anomalies → outputs root cause hypotheses + recommended CAPA steps.

Stack: Python, Pandas, sensor anomaly detection (Isolation Forest), LangChain Multimodal RAG, FAISS/Chroma, PDF parsing with Unstructured.

P6) Medical Affairs Response Generator (MI Chatbot with Guardrails)

Problem: Generate medically accurate, compliant responses to HCP queries using curated medical documents + safety guardrails.

Sample data: FDA drug label text, PubMed abstracts (open access samples), Synthetic Medical Information Response Dataset.

Outcome: MI chatbot producing structured, compliant responses (Key Evidence, Study Citations, Safety Points) with restricted hallucination guardrails.

Stack: Python/TS, LangChain, Guardrails.AI, Retrieval over approved docs, LLM finetuning (optional).

P7) Market Access Value Dossier Assistant (HEOR + RAG)

Problem: Draft sections of AMCP/Value Dossiers (clinical evidence, budget impact, unmet need) using HEOR guidelines + literature.

Sample data: AMCP sample dossiers (public examples) AND Drug performance datasets from Kaggle (e.g., sales + outcomes data).

Outcome: Structured dossier sections produced in Markdown/PDF, including value story, evidence summaries, and payer messaging (with citations).

Stack: Python, LangChain, embeddings, RAG, Pandas for cost modeling, LLM summarizers.

P8) Regulatory Submission Assistant (eCTD-ready content)

Problem: Generate structured summaries for eCTD modules (2.5, 2.7, 2.3) from clinical study reports and quality documents using RAG + structured prompts.

Sample data: Sample CSR documents (public excerpts) AND FDA eCTD guideline PDFs.

Outcome: eCTD-aligned outputs with sections like Clinical Overview, Clinical Summary, Quality Summary; auto-validated for structure and terminology.

Stack: Python, LangChain, PDF parsing (Unstructured), embeddings, pydantic structured templates.

P9) Real-Time Adverse Event Triage Agent

Problem: Classify AE reports into Serious / Non-Serious, Expected / Unexpected, and route them to safety teams.

Sample data: SIDER Side Effects Dataset – https://www.kaggle.com/datasets/mettab/sider2 AND AE narrative samples (synthetic).

Outcome: Notebook + API that ingests AE text and returns triage decisions, seriousness level, MedDRA term suggestions, and routing group.

Stack: Python, scikit-learn or transformer classifier, LangChain LLM explanations, MedDRA dictionary lookup (open subset).

P10) Sales Rep Call Intelligence Analyzer

Problem: Analyze call transcripts between medical reps and HCPs to extract insights, objections, competitor mentions, and recommended next best actions.

Sample data: Call Transcription Dataset (synthetic) – https://www.kaggle.com/datasets/willirath/synthetic-call-center-data AND Clinic conversation text samples.

Outcome: LLLM-based insight extraction system producing: Top topics discussed, Objections raised, Product sentiment, Suggested next action plan (NBAs).

Stack: Python, ASR (Whisper) for audio (optional), LangChain, embeddings, topic modeling (BERTopic/HDBSCAN), JSON insight output.