VPBank

powered by aws

VPBANK TECHNOLOGY HACKATHON

CHINH PHỤC CÔNG NGHỆ, KIẾN TẠO TƯƠNG LAI CÙNG VPBANK

Learn Ideate Develop Compete

Hack2Hire

OVERVIEW OF CHALLENGE STATEMENT 2025

VPBank VPBANK TECHNOLOGY HACKATHON Hack2Hire

36

Challenge Statement
IT & EDA

33 IT

- IT Security Department (1)
- Technology Platform Department (2)
- IT Strategy & Enterprise Architect Department (2)
- IT Core Banking Applications Center (4)
- Infrastructure Operations Center (5)
- IT Digital Factory Center(8)
- IT Services for Coporate Banking Center (11)

03 EDA

- Data Platform Center (1)
- Artificial Intelligence Center (1)
- Data Governance Department (1)

VPBank

Technology Hackathon 2025 – IT Challenge Statement #1

VPBank VPBANK TECHNOLOGY HACKATHON Hack2Hire powered by aws

DETECTION AND RESPONSE BY AUTOMATION FLOW MANAGEMENT AND AI

1. Challenge Overview

When a security problem arises, every second matters. However, enterprise IT systems consist of many ever-growing and interconnected components, making manual checking and prevention an unfeasible task. An Automation and Orchestration tool is needed to reduce reaction time when incidents occur.

When analyzing an incident, a security analyst needs knowledge of system operations, logs, and experience to devise a solution, followed by a runbook to handle similar incidents in the future. With the help of AI, this process can be significantly shortened.
Your challenge is to build an Automation and Orchestration solution that leverages the power of AI to reduce incident response time and assist in the creation of runbooks for future use.

Target: Respond to events occurring across more than 200 AWS accounts.

VPBank

DETECTION AND RESPONSE BY AUTOMATION FLOW MANAGEMENT AND AI

2. Technical Requirements

- Programming Languages: Any (open choice)
- AI Model: Any LLM
- Workflow Automation: Any tool (recommended: n8n)
- Database: Any (recommended: PostgreSQL)
- Deployment: Docker, Kubernetes, or hosted-based application

3. Evaluation and Measurement

- Response time: < 10 sec
- Accuracy: > 99,9%

4. Deliverables

- Presentation
- Prototype Demo
- Source code

METRIC ANOMALY DETECTOR WITH ROOT CAUSE AI

1. Challenge Overview

Monitoring metrics like CPU usage, memory, response codes, and request counts is essential in DevSecOps. However, identifying anomalies and determining the root cause can be time-consuming. This challenge focuses on building a smart anomaly detection system that not only flags issues but also explains why they might have happened using AI for example: on trending/on spark/sale event.

2. Technical Requirements

The prototype should:

- Programming Languages: Python (preferred), or any language suitable for data processing
- Data Tools: Pandas, Matplotlib, etc.
- AI Integration: Any LLM API (e.g., OpenAI, Claude)
- Data Source: Simulated or real metric logs (CSV, JSON, Prometheus format

METRIC ANOMALY DETECTOR WITH ROOT CAUSE AI

3. Evaluation and Measurement

3.1 Anomaly Detection Accuracy
- Flags correct anomalies in metric data
- Uses a defined method (e.g., z-score, thresholds)

3.2 Root Cause Explanations
- Outputs useful hypotheses with context
- Based on time-aligned patterns or logs

3.3 Outputs useful hypotheses with context
- Based on time-aligned patterns or logs

3.4 Visualization & Result Clarity
- Graphs clearly show trends and anomaly points
- Legends/labels make data easy to interpret

3.5 Documentation
- Explains anomaly method, usage, setup using markdown
- Includes input/output examples

3.6 Observation system alert Simulation base on detected event

4. Deliverables
- Metric Analyzer Script or Notebook develop description
- Anomaly Detection Logic explanation
- Root Cause Analysis via AI Module explanation
- Sample Input and Output Files
- Documentation on Deploy.
- Monitoring Dashboard Simulation presentation
-CI/CD Trigger Simulation presentation

LLM-BASED IAC DRIFT ANALYZER

1. Challenge Overview

Infrastructure drift happens when the deployed infrastructure no longer matches the version defined in Infrastructure-as-Code (IaC). This can lead to unexpected behavior, failed pipelines, or security risks. The goal is to build a tool that helps teams identify and understand infrastructure drift automatically, using AI to explain what happened and why

2. Technical Requirements

- Programming Languages: Python, Bash, or similar scripting language
- Infrastructure Tooling: Terraform, Terragrunt or any IaC scripting language (even Pulumi or Crossplane)
- AI Integration: Any LLM Model
- CI/CD scripting or automation knowledge
- Infrastructure platform: AWS

LLM-BASED IAC DRIFT ANALYZER

3. Evaluation and Measurement

3.1 Drift Detection Accuracy with Cloud event and Pipeline event
- Correctly identifies drifted resources (new, deleted, changed)
- Integrate with cloud event to detect drifted resource
- Integrate with CICD pipeline to aware to user that they might change resource which are out-scope with current Merge request.

3.2 Compare the output of Plan event and the Merge Request Intend (based on code changes, description etc)
- Periodically execute drift detection with resource that created from IaC

3.3 Schedule with timely basis, traverse by each IaC code repo

3.4 AI-Powered Explanations
- Uses a language model to generate plain-language descriptions
- Provides relevant root cause and actionable recommendations

3.5 Sample Scenarios Quality
- At least two well-crafted example inputs and outputs
- Covers multiple drift situations, including edge cases

3.6 Documentation (README.md)
- Basic setup and usage instructions
- Clear explanation of classification logic
- Detail explanation on System architecture

4. Deliverables
- Executable Drift Detection setup with cloud event and schedule job
- Drift Classification System architecture presentation
- AI-Powered Explanation Module description
- Sample Input and Output Files
- CI/CD Integration Example explanation

AI/ML ENHANCED CREDIT SCORING

1. Challenge Overview

Traditional credit scoring systems rely heavily on structured financial data, often overlooking alternative data sources that could better reflect the creditworthiness of new-to-bank or thin-file customers. This challenge invites participants to develop an AI-enhanced credit scoring prototype that leverages Generative AI and/or ML models to improve scoring accuracy, transparency, and inclusiveness.

The system should simulate scoring based on both traditional and non-traditional data inputs (e.g., transaction behavior, social data, utility bills, e-commerce activity), and provide interpretable outputs for business or credit risk staff.

AI/ML ENHANCED CREDIT SCORING

2. Technical Requirements

The prototype should:

- Programming Languages: Python, Node.js, TypeScript...
- Cloud Services: AWS
- Model AI: OpenAI, Claude, or any ML/LLM-based models (e.g., XGBoost, LightGBM, LangChain...)
- Others: Docker, Kubernetes

3. Evaluation and Measurement

- Data Handling: Ability to receive structured and unstructured mock input data (e.g., income, transactions, utility usage)
- Scoring Logic: Applies AI/ML model to generate a credit score or risk classification
- Explainability: Outputs a simple explanation (in natural language or charts) showing why the score was assigned
- Adaptability: Supports scoring for different types of customers (e.g., salaried, freelancers...)
- Relevance of Output: Output must be logical, explainable, and reflect input characteristics
- Functional Demo: Prototype must run stably and simulate end-to-end scoring on sample cases

AI/ML ENHANCED CREDIT SCORING

4. Deliverables

- Working prototype of the credit scoring system
- Mock input data samples and AI-generated scores
- Explainable output interface or report
- Demo flow showing customer types and scoring results

MICROSERVICE APPLICATION DESIGN EVALUATION

1. Challenge Overview

In VPBank IT's strategy for the coming years, IT units will increasingly take ownership in developing applications. The implementation trend will be based on a standard microservice framework provided by the Tech Platform. However, when developing applications, some units have not yet adhered to the design standards and best practices established by IT Enterprise Architecture (EA).

When performing design reviews, EA currently needs to manually review and comment based on its own analysis.

Objective: Build a tool to evaluate whether a microservice design complies with EA best practices according to a defined set of standards.

Inputs: The detailed design document, including:

Requirements

- Use case view
- Business capability
- Business process
- High-level application functions
- Non-functional requirements

(e.g., transaction volume – TPS, etc.)

- Application architecture (C4 standard – Level 2 & 3)
- **Data architecture:**
  - ✓ Database design
  - ✓ File/object storage
  - ✓ Queue/cache data model
- Integration architecture: API inventory

MICROSERVICE APPLICATION DESIGN EVALUATION

2. Technical Requirements

The prototype should:

- Propose the best practice for Microservice Application design with weight for scoring.
- Capture required design material in any standard format for analytic.
- Capture the required information via application form related to the design material.
- Provide the result as check list (pass or false) of standards and the overall scoring (0-100).

With note/comment is an advance.

3. Evaluation and Measurement

- Microservice Application design standard proposal: Best match the best practice defined by EA or better
- Standard format for design material that help the process of analytic: Popular standard
- Clearly explain the approach of analytic and scoring.
- Clear solution design document to implement the approach.
- Functional demo for 2 use cases: good design (score >90), bad design(score in 50 – 80).

4. Deliverables

Develop working prototype solution

THE FUTURE OF DIGITAL CHARITY

1. Challenge Overview

Context & Opportunity

In a society increasingly concerned with transparency, Corporate Social Responsibility (CSR), and Environmental, Social, and Governance (ESG), traditional charitable activities in Vietnam still face significant limitations:

- Lack of Transparency: Donors find it difficult to track the flow of funds and the verification process for fund usage is slow.
- Lack of Trust: Concerns about fund leakage, misuse, or disreputable organizations.
- Lack of Engagement: Donors rarely see the direct impact of their contributions.
- Lack of Modern Digital Tools: The community is fragmented, lacking a large-scale, reliable digital platform to foster a sustainable charity culture.

This presents an opportunity for VPBank to pioneer the use of breakthrough technology to solve these core issues, reinforcing its image as a community-oriented bank and a leader in the ESG trend.

THE FUTURE OF DIGITAL CHARITY

Challenge Objective

Build a Digital Charity & Social Impact Platform integrated into VPBank NEO, leveraging the power of AI and Blockchain to:

- Blockchain: Ensure 100% of donation transactions are transparent, immutable, and publicly verifiable in real-time.
- AI: Personalize the user experience, recommend suitable campaigns, analyze social impact, and detect fraud.
- Gamification & Social: Create an engaging digital charity ecosystem that spreads the culture of giving throughout the community, from retail customers to corporations.

2. Technical Requirements

The prototype should include the following features and technologies:

- Languages & Stack: Python, Node.js, TypeScript.
- Blockchain: Ethereum / Polygon / Hyperledger (optional).
- AI Services: AWS Bedrock (LLM, RAG), Amazon Comprehend (NLP), Amazon Fraud Detector.
- Storage & Data: Amazon S3, DynamoDB / PostgreSQL.
- Dashboard & UI: Amazon QuickSight / ReactJS

THE FUTURE OF DIGITAL CHARITY

3. Evaluation and Measurement

- Innovation & Technical Depth (25%): The effectiveness of combining AI and Blockchain.
- Social Impact & Problem Solving (25%): How well the solution addresses transparency, trust, and delivers clear social impact.
- User Experience & Engagement (20%): An intuitive, engaging interface that encourages community participation.
- Technical Feasibility & Scalability (15%): The potential for real-world implementation and integration into VPBank NEO.
- Presentation & Storytelling (15%): A clear presentation, a persuasive demo, and a long-term vision

4. Evaluation and Measurement

- Prototype Demo:

A web portal or mobile-integrated dashboard.

oFeatures: Campaign uploads, on-chain donation flow, AI-powered recommendations, impact dashboard.

- Presentation (5–10 minutes): Problem context, technical solution, illustrative demo, and integration roadmap for VPBank NEO.
- Source Code: Repo on GitHub/GitLab.
- Impact Scorecard: A report describing social metrics and contribution transparency

MODULAR PAYMENT SERVICE WITH AI-POWERED INSIGHTS & FRAUD DETECTION

1. Challenge Overview

Digital payments are now at the core of banking transformation. However, scaling payment services to meet real-time, secure, and intelligent processing while extracting meaningful customer insights and detecting anomalies requires sophisticated system design and seamless operations.

This challenge aims to develop an enterprise-grade modular payment system that not only routes and processes transactions (IBFT, QR, POS, eWallet) in real-time, but also incorporates AI/ML to:

- Analyze customer behavior
- Detect fraudulent patterns or anomalies
- Generate internal market insights

Beyond building intelligent capabilities, teams are expected to demonstrate ZeroOps practices—systems that operate with minimal manual intervention, using fully automated deployment, monitoring, and self-healing features. Participants are encouraged to integrate AI Agents or GenAI to generate personalized recommendations or contextual explanations based on transaction data

MODULAR PAYMENT SERVICE WITH AI-POWERED INSIGHTS & FRAUD DETECTION

2. Technical Requirements

The prototype should support:

- Programming Languages: Python, NodeJS, Java, etc.
- Cloud Services: AWS (EC2, API Gateway, Lambda, S3, DynamoDB, etc.)
- AI/ML Capabilities:
  - ✓ Transaction anomaly detection (via ML or GenAI)
  - ✓ Personalized recommendations
  - ✓ Market behavior clustering
- Architecture: Microservices with REST/Async APIs
- DevOps/ZeroOps:
  - ✓ GitOps, CI/CD pipelines
  - ✓ Infrastructure-as-Code (Terraform/CDK)
  - ✓ Auto-scaling and self-healing mechanisms
- Security: OAuth2/JWT, input validation, PII masking
- Observability: Centralized logging, tracing, metrics dashboard (Elastic, Grafana, CloudWatch, etc.)

MODULAR PAYMENT SERVICE WITH AI-POWERED INSIGHTS & FRAUD DETECTION

3. Evaluation and Measurement

- Functional Correctness: Transaction validation, routing logic, response accuracy
- Modularity: Service separation, API boundaries, container isolation
- AI/ML Integration: Quality and relevance of behavior insights and anomaly detection
- ZeroOps Maturity: Degree of automation in CI/CD, self-healing, infra provisioning
- Resilience: System response under failure, retry mechanisms, fallbacks
- Security Compliance: Secure APIs, data protection, compliance with best practices
- Observability: Monitoring, logging, tracing, alerting coverage
- Documentation: Architecture diagram, README, API spec
- Bonus: AWS Free Tier deployment, use of GenAI in user interaction

MODULAR PAYMENT SERVICE WITH AI-POWERED INSIGHTS & FRAUD DETECTION

4. Deliverables

- Working prototype with modular services
- AI/ML component for fraud detection or behavior insight
- Dashboard or API with transaction analytics
- Full source code and automated infrastructure setup
- CI/CD pipeline script and automation logic
- Demo video or AWS-hosted environment
- Technical documentation:
  - ✓ Architecture diagram, API spec
  - ✓ Instructions (README)
  - ✓ Design decisions summary
- Presentation slides (7–10 minutes

AI-DRIVEN FRAUD PREVENTION & DISPUTE AUTOMATION FOR CUSTOMER-CENTRIC CARD SERVICES

1. Challenge Overview

Card payment systems face two major pain points: sophisticated fraud and a manual, costly chargeback process. This leads to financial losses and erodes customer trust.
The challenge is to build a comprehensive, automated, and intelligent system using AI to solve both issues. Your solution should not only detect fraud in real-time but also automate the entire dispute workflow, from evidence collection to customer interaction. The goal is to minimize risk, reduce costs, and improve the customer experience.

AI-DRIVEN FRAUD PREVENTION & DISPUTE AUTOMATION FOR CUSTOMER-CENTRIC CARD SERVICES

2. Technical Requirements

2.1 System Architecture

- Design a scalable, fault-tolerant architecture (e.g., microservices on Kubernetes) capable of handling high-volume, real-time transaction processing.

- Support modular deployment for fraud detection, dispute management, explainability, and personalization services.

2.2 Explainable AI/ML

- Apply machine learning models (XGBoost, Autoencoder, LSTM, GNN) to assign a risk score to each transaction.

- Ensure explainability (XAI): when a transaction is flagged, the system must provide a clear, human-readable reason (e.g., “High risk due to unusual location and abnormal transaction amount”).

- Continuously retrain and update models to adapt to evolving fraud patterns.

2.2 API Design & Automation

Expose a robust set of RESTful APIs for integration with a separate front-end, covering:

- Dispute Submission: Accept detailed customer complaints with attached evidence.

- Smart Code Recommendation: Use AI/NLP to suggest the most suitable chargeback code based on the customer’s description, along with its estimated success probability.

-Transparent Tracking: Allow customers to track their dispute status in real time, including progress milestones and expected resolution timelines.

AI-DRIVEN FRAUD PREVENTION & DISPUTE AUTOMATION FOR CUSTOMER-CENTRIC CARD SERVICES

2. Technical Requirements

2.4 Data & Security

- Efficiently process large-scale financial data while maintaining low-latency responses.
- Ensure compliance with financial security standards (e.g., PCI DSS), including encryption, tokenization, and secure audit trails.
- Support data governance: evidence storage, retention policies, and explainability logs.

2.5 Prototype Technology Stack

- Programming Languages: Python, SQL, Java, JavaScript, Node.js.
- Cloud Services (AWS):
  - ✓ SageMaker (model training/hosting),
  - ✓ Fraud Detector (fraud pattern recognition),
  - ✓ Personalize (customer-centric recommendations),
  - ✓ Kinesis (real-time streaming),
  - ✓ Neptune (graph-based fraud analysis).
- AI Models: XGBoost, Autoencoder, LSTM, Graph Neural Networks (GNN), Collaborative Filtering.
- Others: Docker, Kubernetes (container orchestration), diagram generation tools (Mermaid.js, draw.io API).

Technology Hackathon 2025 – IT Challenge Statement #8

VPBank VPBANK TECHNOLOGY HACKATHON Hack2Hire powered by aws

AI-DRIVEN FRAUD PREVENTION & DISPUTE AUTOMATION FOR CUSTOMER-CENTRIC CARD SERVICES

3. Evaluation and Measurement

3.1 AI Model Performance:

- Fraud Detection: Achieve Precision > 90% and Recall > 85%. Maintain a False Positive Rate < 0.5%.
- Explainability: The clarity and accuracy of the AI's reasoning will be a key factor.
- Chargeback Success: Increase the win rate by at least 15% compared to manual processes

3.2 Automation & Efficiency:

- Manual Effort: Automate > 80% of the initial dispute process.
- Latency: API response time for chargeback code recommendation must be < 500ms.

3.3 Scalability & Security:

- Throughput: Handle 10 mil transactions/day without performance degradation.
- Availability: Target 99.9% uptime.

3.4 Fraud Detection Performance

- Accuracy of identifying fraudulent transactions.
- Metrics: Precision, Recall, F1-score, False Positive Rate.

3.5 Personalization Effectiveness

- Relevance and usefulness of promotions or card recommendations.
- Measurement: Customer engagement rate (e.g., click-through, acceptance rate).

3.6 System Efficiency

- Processing speed and scalability for large transaction volumes.
- Ability to operate in real-time or near real-time.

VPBank

Technology Hackathon 2025 – IT Challenge Statement #8

VPBank
VPBANK TECHNOLOGY HACKATHON
Hack2Hire
powered by aws

AI-DRIVEN FRAUD PREVENTION & DISPUTE AUTOMATION FOR CUSTOMER-CENTRIC CARD SERVICES

3.7 Business Impact

- Potential reduction in fraud-related losses.
- Value creation for customers (improved satisfaction, loyalty).

3.8 Compliance & Security

- Data privacy, anonymization, and regulatory alignment.
- Explainability of AI-driven decisions.

3.9 Presentation & Usability

- Clarity of solution design, dashboards, and reporting.
- Professionalism and persuasiveness of the pitch/demo

VPBank

AI-DRIVEN FRAUD PREVENTION & DISPUTE AUTOMATION

4. Deliverables

Each team must submit:

- Source Code: The complete backend source code on GitHub/GitLab.

- API Documentation: A detailed document describing all API endpoints.

- Technical Documentation: A presentation or document outlining the system architecture and explaining the AI models, with a focus on their explainability.

- Product Demo: A video or live presentation showcasing the system's end-to-end flow, including the AI's transparent reasoning and smart chargeback code recommendations.

FAIL & FORWARD: EMBRACING FAILURE WITH AUTOMATED RECOVERY

1. Challenge Overview

Enterprise systems often struggle to identify failures early, especially when relying on static, pre-defined thresholds for anomaly detection. Unexpected issues, such as sudden spikes in rare exceptions or unusual error patterns, remain invisible until they escalate into major incidents.

This challenge focuses on building a smart agent that delivers two core capabilities:

Dynamic Detection of Abnormal Application Behavior: continuously analyzing logs and metrics to surface anomalies in real time, without depending solely on static rules.

Automated Recovery Flow in Failure Scenarios: – taking corrective actions such as restarting pods, scaling services, or notifying operators to ensure service continuity.

Participants are expected to demonstrate these two aspects in a simulated failure scenario, showing how the agent detects the anomaly and executes recovery automatically.

FAIL & FORWARD: EMBRACING FAILURE WITH AUTOMATED RECOVERY

2. Technical Requirements

2.1 Application Behavior Detection:

- Collect logs and metrics from diverse application platforms (JBoss, Tomcat, Spring Boot on EKS).
- Apply anomaly detection (ML-based or statistical) to identify abnormal patterns such as rare exceptions or sudden error surges.

2.2 Automated Recovery Flow:

- Integrate with Kubernetes/EKS at cluster-admin level.
- Trigger automated recovery actions (restart pods, scale deployments, failover) when anomalies are confirmed.

2.3 Notification & Visibility:

- Send proactive alerts to operators (via Email/Slack/Teams).
- Provide a simple dashboard/CLI for anomaly status and recovery history.

2.4 Tools & Stack (suggested): Python/Go, Prometheus, FluentBit/Loki/ELK, Docker, Kubernetes (EKS), AI/ML libraries (Scikit-learn, PyTorch, OpenAI).

FAIL & FORWARD: EMBRACING FAILURE WITH AUTOMATED RECOVERY

3. Evaluation and Measurement

- Abnormal Application Detection: Accuracy and timeliness in detecting unusual errors, spikes, or unexpected patterns.
- Automated Recovery Flow: Effectiveness of recovery actions (restart, scale, alert) in a simulated failure scenario.
- End-to-End Demonstration: Clear demo of "anomaly detected → recovery executed → system stabilized."
- Proactive Notification: Ability to alert operators with relevant, actionable insights.
- Reliability: Stability of the prototype during the simulated run.

FAIL & FORWARD: EMBRACING FAILURE WITH AUTOMATED RECOVERY

4. Deliverables

4.1 A working prototype agent with:

- Real-time anomaly detection of application logs/metrics.
- Automated recovery actions on EKS resources (restart, scale, heal).

Demonstration of a failure scenario where:

- An anomaly is detected (e.g., spike in rare exception).
- Automated recovery is triggered.

Operators are notified with meaningful alerts.

4.2 A dashboard or CLI interface displaying:

- Detected anomalies.
- Recovery actions taken.

Current service health.

4.3 Documentation describing:

- Detection approach.
- Recovery logic.
- Deployment steps for EKS integration.

SYSTEM CENTRALIZED DC–DR SYNCHRONIZATION MANAGEMENT PLATFORM

1. Challenge Overview

Currently, most banks maintain DC–DR synchronization across multiple layers separately (OS, Database, Application configuration, File system, Container). This fragmented approach increases operational complexity and the risk of inconsistency.

Your challenge is to build a centralized platform to monitor and manage synchronization status for these layers, which provide unified dashboard, alerting & notifications, centralized control actions (resume/restart the synchronization if there are errors). The platform must support integration with common virtualization platform (VM, Oracle, SQL server, Kubernetes)

2. Technical Requirements

The platform should base on common programming language and AWS platform:

- Programming Languages: Python, Nodejs, Typescript...
- Cloud platform: AWS

SYSTEM CENTRALIZED DC–DR SYNCHRONIZATION MANAGEMENT PLATFORM

3. Evaluation and Measurement

- The system diagram should be able to show system scalability and automated deployment on AWS
- Unified dashboard: Real time visualization of synchronization status across all layers
  - ✓ Database: Oracle Data Guard, SQL AlwaysOn, MySQL/Mongo replication
  - ✓ VM: rsync, robocopy
  - ✓ Application configuration: configuration replication
  - ✓ Container: Kubernetes cluster, PV sync

Alerting & notifications: Integrate with monitoring system and being able to set up threshold for replication lag, config drift, sync job failed

Centralized control actions: resume/restart failed sync job from the central console and being able to set up automation rules (such as automatically retry sync jobs and escalate if fail many times)

4. Deliverables

- A brief architecture diagram and explanation (e.g., using draw.io). The document should explain the choice of AWS services, data flow, and design trade-offs (e.g., cost vs. performance, serverless vs. container-based), as well as the mechanism to secure control actions, credentials management and system scalability. (20%)
- Show a single console that displays synchronization status across at least 3 layers (Database, File/Storage, Container). (30%)
- Simulate a synchronization error (e.g., database replication lag, file sync failure) and show how the platform detects and alerts. Design automated workflow in case there is synchronization error (30%)
- From the same console, demonstrate resuming or restarting a failed synchronization job. (20%)

SYSTEM AUTOMATION AND IAC FOR SCALABLE AND RELIABLE VIRTUAL INFRASTRUCTURE

1. Challenge Overview

The current infrastructure runs Virtualization layer with storage, but administration and monitoring are still done manually, leading to high effort and frequent errors.

Your task is to design an automation and Infrastructure-as-Code (IaC) model that minimizes manual work, standardizes infrastructure, and ensures stability, security, and scalability. You need to propose an automation deployment architecture, define the end-to-end workflow from request to delivery, and implement scripts/Ansible playbooks/Terraform modules to automatically provision virtual machines using predefined templates.

The solution must be able to create VMs, assign IPs from a given range, perform basic setup (OS, user, SSH key, monitoring agent), and flexibly change the number of VMs with a single variable. Finally, you will deliver documentation of the architecture and process, along with a demo showcasing automated VM provisioning

SYSTEM AUTOMATION AND IAC FOR SCALABLE AND RELIABLE VIRTUAL INFRASTRUCTURE

2. Technical Requirements

- Programming Languages: Terraform, Ansible, Python, TypeScript...
- Cloud Services: AWS
- Model AI: Any LLM (OpenAI, Claude, Gemini...)
- Others: Docker, Kubernetes, diagram generation tools (Mermaid.js, draw.io API, etc.)

3. Evaluation and Measurement

- System Architecture: Understanding of system architecture, system components, and the ability to integrate and ensure compatibility with VMware and OpenStack solutions.
- Workflow/process: Understanding of the process, execution steps, and mandatory requirements necessary for the process to be successfully carried out.
- Function Demo: Demonstration of product features, stability, and accuracy

4. Deliverables

- Architecture document
- Workflow/process document
- Script/Play book: automated server creation

NOC: INTELLIGENT RCA ASSISTANT: LEVERAGING LLMS AND MCP FOR NOISY MICROSERVICES OBSERVABILITY

1. Challenge Overview

Modern microservices produce huge volumes of metrics, logs, and traces, along with frequent change events such as releases, configuration updates, scaling, or incident notes. During outages, Site Reliability Engineers (SREs) must quickly pinpoint the root cause hidden in this noisy data.

Your challenge is to build an intelligent assistant using Large Language Models (LLM) and the Model Context Protocol (MCP) to perform automatic root cause analysis (RCA). The assistant should reason like an SRE: form hypotheses, correlate with recent changes, query observability data, validate with evidence, and explain the most likely cause.

NOC: INTELLIGENT RCA ASSISTANT: LEVERAGING LLMS AND MCP FOR NOISY MICROSERVICES OBSERVABILITY

2. Technical Requirements

- Integration with Observability Stack: The assistant must connect to at least one source of metrics (e.g., Prometheus, Mimir, CloudWatch), logs (e.g., Elasticsearch, ClickHouse, CloudWatch Log), and traces (e.g., Jaeger, Tempo, X-Ray).
- Change Events Correlation: Support ingesting information from change events (release notes, config updates, scaling activities, incident annotations).
- Reasoning Engine with LLM + MCP:
- Use LLMs to emulate SRE reasoning (hypothesis → correlation → validation).
- Leverage MCP to manage multi-source context and enable structured queries.
- Evidence-based RCA: Analysis output must point to concrete evidence (log snippet, metric anomaly, trace segment).
- Explainability: The assistant should clearly explain why it suggests a particular root cause.
- User Interaction: Provide a chat interface or dashboard for SREs to interact and review RCA results.

Technology Hackathon 2025 – IT Challenge Statement #12

VPBank VPBANK TECHNOLOGY HACKATHON Hack2Hire powered by aws

NOC: INTELLIGENT RCA ASSISTANT: LEVERAGING LLMS AND MCP FOR NOISY MICROSERVICES OBSERVABILITY

3. Evaluation and Measurement

- Teams will be evaluated based on:
- Accuracy (40%): Ability to correctly identify the root cause in simulated incident scenarios.
-Explainability (20%): Logical and well-supported explanations backed by observability data.
- Integration (20%): Successful integration with diverse observability data sources and change events.
- Usability (10%): Ease of use, intuitive UI/UX for interaction.
- Innovation (10%): Novel or creative application of LLM + MCP beyond the baseline expectation.

4. Deliverables

- Source Code & Deployment Guide: Repository with runnable demo (e.g., Docker Compose / Kubernetes manifests).
- Demo Application: A chatbot or dashboard that ingests observability data and produces RCA.
- Test Scenarios: A set of simulated incidents (e.g., misconfig, service bottleneck, memory leak).
- Documentation:
- System architecture diagram and description.
- Explanation of reasoning workflow (hypothesis → evidence → conclusion).
- Pitch/Demo Video (5–7 minutes): Presentation of problem, solution, and live demo.

VPBank

Technology Hackathon 2025 – IT Challenge Statement #13

VPBank VPBANK TECHNOLOGY HACKATHON Hack2Hire powered by aws

NETWORK: IT INFRASTRUCTURE AUTOMATION FOR REDUCING HUMAN ERRORS AND STANDARDIZING OPERATIONS

1. Challenge Overview

Modern IT operations still involve many repetitive manual tasks such as user provisioning, log collection, VM creation, backup, or network configuration updates, which are error-prone, time-consuming, and difficult to standardize.

Your challenge is to build an automation platform that reduces manual effort, codifies and standardizes operational processes, and introduces a doer/checker mechanism to mitigate risks. You need to analyze at least five common IT operations (system, network, security, servers, or applications), evaluate which is most error-prone and which consumes the most time, then select at least three operations to automate.

Propose how to codify them using tools such as Python, Ansible, Terraform, or PowerShell, and design an automation architecture that covers multiple infrastructure layers. Define a complete workflow from request submission to approval, automation execution, and verification, clearly showing roles, steps, and expected outputs. Finally, deliver documentation of your architecture and process, along with a real demo of automated operations that demonstrate reliability, safety, and scalability.

VPBank

Technology Hackathon 2025 – IT Challenge Statement #13

VPBank VPBANK TECHNOLOGY HACKATHON Hack2Hire powered by aws

NETWORK: IT INFRASTRUCTURE AUTOMATION FOR REDUCING HUMAN ERRORS AND STANDARDIZING OPERATIONS

2. Technical Requirements

- Programming Languages: Python, Node.js, TypeScript...
- Tool: Ansible, Puppet, Saltstacks, Vault, ArgoCD...
- Others: Teraform, CloudFormation AWS, Jenkins, Gitlab CI/CD...

3. Evaluation and Measurement

- Accuracy (40%): The solution works exactly according to the designed workflow.
- Explainability (20%): Logical and well-supported explanations
- Integration (20%): Successful integration with systems including network, security, servers
- Usability (10%): Ease of use, intuitive UI/UX for interaction.
- Innovation (10%): The issues raised and resolved are real pain points in operations.

4. Deliverables

- Source Code & Deployment Guide: Repository with runnable demo
- Demo Application: at least 3 activities. Describe the architecture/model of automation implementation (can draw a block diagram or explain in words). Applicable objects: including systems, networks, security, servers (legacy /container) or applications
- Documentation:
  - ✓ System architecture diagram and description.
  - ✓ Explanation of reasoning workflow (hypothesis → evidence → conclusion).
- Pitch/Demo Video (5–7 minutes): Presentation of problem, solution, and live demo.

VPBank

IT COLLABORATION: BUILDING A SYSTEM PORTFOLIO

1. Challenge Overview

We want to build CMDB (Central Infrastructure Device Management) application manages a catalog of devices such as PCs, laptops, UPS, and servers. This application is designed with .NET framework platform the large number of device records (approximately 19,000).

The CMDB should has at least these information: Name of device, Serial number, type of device, date of maintenance support start (MA), date of MA end, cost of MA

The challenge:

- Make CMDB with 19k records that allow to input manually or by uploading from csv form
- Each device should record any change
- Making the list of MA for those are end of support (MA) with relevant cost
- Allow upload csv or manual to update new date of MA and new cost

IT COLLABORATION: BUILDING A SYSTEM PORTFOLIO

2. Technical Requirements

2.1 Front-End Expertise:

- HTML5, CSS3, and modern JavaScript (ES6+).
- React, Angular, or Vue.js.

2.2 Microservices & API Expertise:

- RESTful APIs.
- Node.js, Python, or Go.
- Database technologies (SQL and NoSQL).

2.3 DevOps Expertise:

- CI/CD tools such as Jenkins, GitLab CI, or GitHub Actions.
- Infrastructure as Code (IaC) using Terraform, CloudFormation, or Ansible.
- Docker and Kubernetes.
- Cloud platforms like AWS, Azure, or Google Cloud Platform (GCP).
- Bash or Python.

3. Evaluation and Measurement

Build the core of the CMDB application.

- Develop the solution using a microservices model to improve performance and ease of use.
- Use a SQL Server farm for the database to easily convert existing CMDB data.
- Build APIs so other departments can share the CMDB data.
- Automatically pull data from various sources (you can simulate other system providing information of device, at least 2 external systems) to collect data for the CMDB

IT COLLABORATION: BUILDING A SYSTEM PORTFOLIO

4. Deliverables

-A CMDB application that performs well with large datasets.
-A data update process that can be performed via a workflow.
- Logs for user actions and data changes.
- Data processing to generate business-related reports.
- A user-friendly UI/UX interface

BUSINESS LOG MONITORING

1. Challenge Overview

At VPBank, there are many services running connecting to many partners with different business logics, making monitoring very difficult. This challenge aims to build a system to:

- Alert the operations team of service abnormalities according to configured business criteria, system.
- Automatically classify errors by system errors, business errors.
- Push notify for ops member.
- Allow add warning configurations

2. Technical Requirements

The prototype should include the following components:

- Backend(Java Spring Boot) for:
- Dashboard management.
- REST APIs for web and mobile.
- Integration with AI services (via HTTP/gRPC).
- AI Services (Python):
- NLP/LLM (OpenAI, Gemini, Claude, etc.) for classifying service requests.
- ML model for business logic.
- Script generator to push ops member.
- Frontend(ReactJS Web & Flutter Mobile App):
- Manager dashboard.
- Database: PostgreSQL (appointments, customers, services, queue, logs).
- Cloud & Deployment:
- AWS (EKS/EC2).
- CI/CD pipeline (GitLab CI).
- Apply any workflow engines

BUSINESS LOG MONITORING

3. Evaluation and Measurement

- Business logic Handling: System supports declare business logic check.
- Ops member Script Support: Ops receive AI-generated notify for system checking.
- Functional Demo: Prototype operates reliably, demonstrating the full flow: business rule → monitor log → ops notify.

4. Deliverables

- Working Prototype including:
- ReactJS web (admin dashboard).
- Java Spring Boot backend (APIs, business logic management).
- Python AI services (NLP, matching, prediction, script generator).
- AI-powered Service Matching Engine integrated end-to-end.
- Demo Flow showcasing: business rule → monitor log → ops notify.

AI AGENTS FOR THE CRM

1. Challenge Overview

VPBank's CRM platform is the digital backbone that empowers our sales, customer service, and marketing teams to deliver personalized experiences to millions of customers. Built on a scalable and modular architecture, the CRM system integrates various customer touchpoints, providing a unified 360-degree view of each customer.

Key functions include:

- Lead & opportunity management
- Customer interaction history
- Campaign management
- Automated workflows and alerts
- Integration with core banking and digital channels

The system is API-first, enabling external services to plug in and enhance its capabilities.

Imagine a smart AI assistant embedded within the CRM that:

- Reminds a Relationship Manager (RM) of a customer's key milestones (e.g., birthday, expiring deposit)
- Automatically drafts follow-up emails or call scripts
- Suggests next-best actions based on historical interactions
- Helps marketers generate campaign content based on customer profiles

You are challenged to build this AI Agent, powered by the Multiple Context Protocol (MCP) – an architecture that supports seamless, context-aware conversation with AI agents across various CRM modules.

AI AGENTS FOR THE CRM

The AI Agent will act as a co-pilot for CRM users, transforming manual processes into intelligent, context-driven interactions.

Develop an AI Agent that can:

- Communicate through a friendly UI
- Retrieve and understand multiple CRM contexts (customer profile, product info, interaction history, etc.)
- Respond smartly using the MCP protocol
- Demonstrate tangible value for CRM users (sales, service, or marketing)

2. Technical Requirements

2.1 The AI Agent should:

- Parse multi-turn conversations with CRM users
- Retrieve context (from simulated or static CRM datasets)
- Respond in natural language with actionable insight or data
- Show the ability to switch between different CRM contexts

2.2 Tech-stack:

- AI Agent Core: Python / Node.js + OpenAI API / LLM (ChatGPT, Claude, etc.)
- Natural Language Layer: LangChain / LlamaIndex / Haystack (context routing)
- UI/UX Interface: React.js or Vue.js
- MCP Server: REST API or WebSocket-based server (Python FastAPI / Node Express)
- Data Store (Optional): MongoDB / MySQL/ Postgre

AI AGENTS FOR THE CRM

3. Evaluation and Measurement

Criteria	Description
Functionality	Does the AI Agent respond correctly and contextually?
MCP Protocol Implementation	Is the MCP protocol properly implemented to manage context switching?
UI/UX Design	Is the interface intuitive and user-friendly?
Technical Architecture	Is the system scalable, modular, and well-documented?
Integration Strategy	Is there a clear plan to integrate with the CRM system?
Innovation & Usefulness	Does the solution bring meaningful value to CRM users?

AI AGENTS FOR THE CRM

4. Deliverables

Deliverable	Description
1. AI Agent Frontend	Interactive web-based UI for CRM users to talk with the agent
2. MCP Server	Backend service that handles context flow across modules
3. CRM Context Simulations	Static or mock data representing CRM entities (customer, product, etc.)
4. Architecture Documentation	Technical diagram and explanation of components and data flow
5. Integration Proposal	Strategy on how the solution can be plugged into VPBank CRM system
6. Source Code	Clean, well-documented repository with instructions to run locally
7. Demo Video (Optional but encouraged)	Short walkthrough video of your solution in action

MODERNIZED DATA PLATFORM FOR PROMOTION CAMPAIGNS

1. Challenge Overview

Traditional promotion systems in banks depend heavily on existing customer data to design cross-selling campaigns, sales contests, or challenge/ promotional programs. Typically, the data is prepared by business units in batch mode with a delay of weeks or even months.

This results in:

- Low effectiveness of promotional campaigns.
- Lack of flexibility in execution.
- High time and resource costs for deployment.

Objective:

Develop a modern, flexible, and configurable promotion data platform that allows business users to design and execute campaigns with minimal technical dependency. The system should support both batch data (large volumes) and event/transaction data (real-time or near real-time) to enable timely and personalized promotions.

2. Technical Requirements

The prototype should:

- Programming Languages: Python, Node.js, TypeScript...
- Cloud Services: AWS
- Database: AWS Redshift, Redis
- Others: Docker, Kubernetes, Kafka, ...

MODERNIZED DATA PLATFORM FOR PROMOTION CAMPAIGNS

3. Evaluation and Measurement

- Processing Architecture
- Provide an overview of the proposed architecture, explaining the function of each component.
- Must support batch, near real-time, and real-time data pipelines.
- Data Handling
- Ingest structured and unstructured input data (customer profiles, transactions, events).
- Support data cleaning, enrichment, and transformation for promotional use cases.
- Promotion Configurability: Allow business staff (non-technical users) to configure rules for promotions or campaigns (e.g., spending thresholds, merchant categories, time periods).
- Explainability & Transparency: Outputs should provide clear explanations of why a customer qualifies for a promotion (e.g., “Customer spent
- 5M VND in Dining category this week, reaching Gold tier”).
- Adaptability
- Support diverse types of promotions (e.g., cashback, tiered rewards, sales contests, challenges).
- Enable easy modification of promotion rules without code changes.
- Functional Demo
- Prototype must simulate end-to-end execution of a promotional use case:
- Data ingestion (batch or streaming).
- Rule evaluation.
- Customer eligibility output.
- Performance: Demonstrate with metrics how the solution can handle different data input volumes (e.g., 1M+ daily transactions).

MODERNIZED DATA PLATFORM FOR PROMOTION CAMPAIGNS

4. Deliverables

- Prototype solution (not full production, but functional demo).
- Documentation: architecture overview, data flow, key assumptions.
- Demo presentation: simulate at least 2 promotional scenarios (e.g., batch-driven cashback program, real-time sales contest).

QUALITY UNDER COMPLEXITY

1. Challenge Overview

In the context of modern banking, information systems play a central role in business operations. These systems are characterized **by** high complexity**,** integration with many other systems, and frequent, ongoing changes. This situation creates major challenges for quality control in software development, where ensuring stability, security, and consistency becomes critical.

The goal of this challenge is:

- To design and propose a concept for processes, practices, and tools that can effectively support quality control throughout the development lifecycle.
- Participants are encouraged to explore innovative approaches that improve reliability, streamline testing, and enable organizations to sustain high-quality software in such a demanding environment.

2. Technical Requirements

The concept should:

- Quality assurance principles: Test Strategy & Process, Quality Metrics & Monitoring, Security & Compliance, Scalability & Maintainability
- Integration testing: Ability to handle complex multi-system integration scenarios (APIs, services, data exchange)
- Testing tools: test management tools and automation test tools (selenium, playwright, Jmeter, ...)
- Programming languages: Java, Python, Javascript, ...

QUALITY UNDER COMPLEXITY

3. Evaluation and Measurement

Submissions will be evaluated based on the following criteria:

3.1 Completeness of Quality Assurance Approach (25%)
- Coverage of QA principles (test strategy, metrics, security, scalability).
- Logical structure, practicality, and relevance to the banking environment.

3.2 Use of Tools and Technologies (20%)
- Appropriateness of test management and automation tools (e.g., Selenium, Playwright, JMeter).
- Effective use of automation and CI/CD pipelines.

3.3 Integration and Complexity Handling (20%)
- Approach to testing in multi-system integration environments.
- Ability to handle frequent changes and interdependencies between systems.

3.4 Innovation and Practicality (15%)
- Creativity in processes, tools, or proposed solutions.
- Realistic potential for improving quality assurance practices.

3.5 Measurable Quality Metrics (10%)
- Definition of quality metrics (e.g., defect density, code coverage, system availability).
- Methods for collecting, monitoring, and reporting metrics.

3.6 Feasibility and Applicability (10%)
- Practical feasibility of the proposed concept.
- Potential for adoption and application in real-world projects.

QUALITY UNDER COMPLEXITY

4. Deliverables

- **Concept Document** – A written report describing the proposed QA strategy, processes, tools, and metrics.
- **Process/Architecture Diagram(s)** – Visual representation of the testing framework, integration flows, or quality control process.
- **Prototype or Demo (Optional but encouraged)** – A proof-of-concept implementation (e.g., automated test scripts, CI/CD pipeline setup, dashboards).
- **Presentation Slides** – A concise summary of the solution for final.
- **Demonstration of Tools** – A prototype or example showing how automation tools and test management tools can be applied in the proposed solution. (Optional but will earn higher scores in evaluation)

SMART SERVING

1. Challenge Overview

At VPB branches, customers often face long waiting times and may be routed to the wrong counter, causing frustration and lowering satisfaction. Tellers also struggle with handling a wide variety of service requests efficiently. This challenge aims to build a system to:

- Allow customers to book appointments remotely or check in at the branch.
- Automatically classify customer service needs from natural language input (text/voice).
- Place customers into a smart queue that assigns them to the most suitable teller based on skill and workload.
- Predict and display estimated waiting times transparently to customers.
- Provide tellers with AI-generated support scripts and document checklists to handle cases more efficiently.

2. Technical Requirements

The prototype should include the following components:

2.1 Backend(Java Spring Boot) for:

- Queue, appointment, customer, and teller management.
- REST APIs for web and mobile.
- Integration with AI services (via HTTP/gRPC).

2.2 AI Services (Python):

- NLP/LLM (OpenAI, Gemini, Claude, etc.) for classifying service requests.
- ML model for queue prediction (waiting time estimation).
- Script generator to support tellers (checklists, advisory scripts).

2.3 Frontend(ReactJS Web & Flutter Mobile App):

- Teller dashboard.
- Manager dashboard.
- Queue display (TV screen).
- Remote appointment booking.
- Queue status updates and notifications.
- QR-based check-in at the branch.
- Database: PostgreSQL (appointments, customers, services, queue, logs).

2.4 Cloud & Deployment:

- AWS (EKS/EC2).
- CI/CD pipeline (GitLab CI).

SMART SERVING

3. Evaluation and Measurement

- Appointment Handling: System supports both remote booking and on-site check-in successfully.
- Service Categorization: AI classifies customer requests from natural language input with >80% accuracy.
- Smart Matching: Customers are routed to the most suitable teller based on skills/workload.
- Wait Time Prediction: Estimated wait time is accurate within an acceptable margin.
- Teller Script Support: Tellers receive AI-generated scripts and checklists relevant to the customer's service.
- Functional Demo: Prototype operates reliably, demonstrating the full flow: booking/check-in → queue → teller support.

4. Deliverables

Working Prototype including:

- Flutter mobile app (remote booking, QR check-in, queue updates).
- ReactJS web (teller dashboard, manager dashboard, TV queue display).
- Java Spring Boot backend (APIs, queue management).
- Python AI services (NLP, matching, prediction, script generator).
- AI-powered Service Matching Engine integrated end-to-end.
- Demo Flow showcasing: appointment → check-in → smart queue → teller assisted by AI script.

SPEAK TO INPUT

1. Challenge Overview

Banking operations often involve complex, multi-step service and product workflows that require processing a large amount of information. The corresponding information systems for these workflows typically contain input screens with numerous data fields. When performed manually via keyboard typing, data entry can become time-consuming and error-prone, ultimately reducing work efficiency.
Reducing processing time, minimizing errors, and enhancing the user experience in data entry are critical requirements. This challenge aims to leverage the power of Generative AI (GenAI) to significantly improve and streamline the data entry process.
The objective is to develop a simple application prototype to demonstrate the use of GenAI in enabling an alternative data entry method via voice commands:

- Users interact with AI to command actions such as inputting, editing, or deleting content in a field/input screen.
- Users can instruct AI to trigger functional buttons on screens or menus.
- The AI must be capable of automatically correcting common spelling mistakes and understanding regional accents and intonations.

SPEAK TO INPUT

2. Technical Requirements

- The Application should:
- Back-end : Java, Sprintboot
- Front-end : react-js

AI Service :

- Programming Languages: Python, Node.js, Typescript...
- Cloud Services: AWS
- AI Model: Any LLM (OpenAI, Gemini, Claude...)
- Others: Docker, Kubernetes

3. Evaluation and Measurement

- Interface: Simple, user-friendly, and easy to interact with.
- Features: Support essential features such as AI voice interaction, accurate speech recognition, correct execution of data entry commands, triggering UI functions, understanding regional accents, and auto-correction of common spelling mistakes.

4. Deliverables

- A simple demo application showcasing the core features to prove solution feasibility.
- Input screens where users can interact with AI to perform data entry and functional operations.A clean, intuitive user interface.

AI ASSET VALUATION

1. Challenge Overview

In banking and finance, asset valuation (such as houses and cars) plays a vital role in loan approvals, risk management, and credit portfolio optimization. Currently, valuation processes are often manual, dependent on experts, leading to delays and lack of consistency.
This challenge aims to apply AI to build an automated valuation tool, enabling the system to analyze input data (asset characteristics, location, usage condition, market data, etc.) and provide a transparent, reasonable estimated value.
Participants are expected to develop a prototype application capable of automatic valuation and presenting results in a user-friendly interface.

2. Technical Requirements

- Back-end: Java Spring Boot / Node.js / Python Flask or FastAPI
- Front-end: React.js or Angular
- AI Service:
  - ✓ Machine Learning / Deep Learning models (Regression / Gradient Boosting / Neural Networks)
  - ✓ Optional: Integrate AI APIs (OpenAI, Gemini, Claude, etc.) for unstructured data processing (asset descriptions, images)
- Cloud Services: AWS / GCP / Azure
- Supporting Tools: Docker, Kubernetes, GitHub

AI ASSET VALUATION

3. Evaluation and Measurement

- Accuracy of Valuation (30%): Results close to market price or sample dataset
- Idea & Innovation (25%): Novel approaches, effective AI utilization
- Scalability & Practical Application (20%): Potential integration into banking systems
- UI/UX & Demo (15%): Clear, user-friendly interface
- Pitching & Teamwork (10%): Presentation skills and collaboration

4. Deliverables

- Demo Application: Input forms for assets (houses, cars) and estimated valuation display
- Presentation Slides: Describe solution, architecture, AI models used
- Source Code: GitHub/GitLab repository link
- Demo Video (3–5 minutes): Illustrating tool usage

FRAUD DETECTION FOR DIGITAL BANKING PLATFORM

1. Challenge Overview

Leverage machine learning and behavioral analytics to identify and prevent fraudulent activities in digital banking transactions. Participants will design AI models that analyze real-time transaction data (such as login patterns, device fingerprints, transfer amounts, and customer behavior) to generate risk scores and flag suspicious activity. The goal is to create solutions that protect customers from fraud while ensuring a seamless and secure banking experience.

2. Technical Requirements

The prototype should:

- Programming Languages: Python, Nodejs, Typescript...
- Cloud services: AWS
- Model AI: any LLM (OpenAI, Gemini, Claude...)
- Others: Docker, Kubernetes

FRAUD DETECTION FOR DIGITAL BANKING PLATFORM

3. Evaluation and Measurement

3.1 Accuracy & Effectiveness (30%)
- How well does the AI detect fraudulent transactions?
- Metrics: Precision, Recall, F1-score, AUC-ROC.
- Balanced performance (not too many false positives that frustrate users).

3.2 Innovation & Approach (20%)
- Novelty of the AI/ML techniques used (e.g., behavioral biometrics, graph-based detection, anomaly detection).
- Creative use of data sources and features.

3.3 Real-time Performance & Scalability (20%)
- Can the solution evaluate transactions in milliseconds?
- Will it scale to millions of daily banking transactions without bottlenecks?

3.4 Customer Experience Impact (15%)
- Minimizing friction for genuine users.
- Smart use of risk-based authentication instead of blanket OTPs.

3.5 Explainability & Interpretability (10%)
- Can the system explain why a transaction was flagged?
- Useful for compliance and regulator trust.

3.6 Presentation & Business Value (5%)
- Clarity of demo, storytelling, and practical alignment with real banking use cases.
- Demonstrated potential ROI for banks.

4. Deliverables
- Develop working prototype solution.
- Generated process workflow design/representation (visual or structured) using prompts.
- Interface for demo interaction.

MODULAR CREDIT ASSESSMENT ARCHITECTURE FOR SMES

1. Challenge Overview

The credit assessment process for granting credit limits to SMEs is extremely complex. Each product (e.g., overdraft, term loan, trade finance) may share similar rules but apply different conditions, while requiring integration with multiple external and internal systems (e.g., RRT, ECMBPM, CIC, internal scoring engines).
Currently, when a new SME product is developed, the development team often needs to re-code existing rules, re-integrate with multiple systems, and re-test end-to-end workflows, which leads to high effort, duplicated work, and significantly slower time-to-market. This challenge is to propose and develop an architectural solution that allows:

- Reusability of rules across different SME lending products.
- Configurable conditions without heavy code changes.
- Seamless connectivity with external systems (RRT, ECMBPM, CIC...)
- Faster deployment of new SME lending products while maintaining consistency, compliance, and stability.

MODULAR CREDIT ASSESSMENT ARCHITECTURE FOR SMES

2. Technical Requirements

The prototype should use:

- Programming Languages: Java Spring boot, Reactjs
- Cloud Services: AWS EKS/ECS, S3
- Data Sources: SME personas, market studies, anonymized SME transaction/usage data
- Database: MySql, MongoDb
- Others: Kafka, Redis, Docker, Kubernete, GitLab CI + ArgoCD, OpenAPI/Swagger...

3. Evaluation and Measurement

- Rule Reusability: Same rules applied across SME products without re-coding or with minimum re-coding.
- Configurability: Ability to adjust conditions via configuration (not code) with minimal developer intervention.
- Time-to-Market Impact: Demonstrated reduction in development/testing cycle for a new product vs current baseline.
- Functional Demo: End-to-end flow for one product using the new architecture.

4. Deliverables

- Architecture design document
- Demo interface for defining/configuring product-specific conditions and executing credit assessment workflow.

INTEREST RATE ENGINE FOR SMES

1. Challenge Overview

SMEs have highly diverse needs and financial behaviors. Today, interest rates for SME lending products are often offered based on generic pricing models, which fail to fully reflect customer value, loyalty, transaction behaviors, or risk profile. This leads to missed opportunities for rewarding loyal customers, optimizing risk-based pricing, and improving competitiveness.

The challenge is to design and prototype an AI-powered Personalized Interest Rate Engine that dynamically calculates and offers SME-specific pricing. The solution should consider multiple factors such as:

- Customer loyalty & tenure with the bank.
- Transaction volumes and product usage across banking services.
- Credit/risk profile (CIC score, repayment history, internal risk models).
- Market competitiveness (benchmarking against industry or peer group).

By providing personalized, transparent, and fair interest rates, the bank can enhance SME satisfaction, drive product adoption, and optimize portfolio risk-return.

INTEREST RATE ENGINE FOR SMES

2. Technical Requirements

The prototype should use:

- Leverage ML/AI models for risk-based pricing and customer segmentation.
- Integrate data from core banking, transaction systems, and risk engines.
- Support configurable pricing rules for policy compliance.
- Allow simulation of different scenarios (e.g., impact of loyalty score on rate).
- Technologies: Python, Node.js, Java Spring boot, Reactjs...

3. Evaluation and Measurement

- Accuracy: Interest rates reflect customer behavior and risk profile.
- Fairness: Transparent, explainable logic behind rate recommendations.
- Flexibility: Ability to adjust weightings of factors (loyalty, transactions, risk).
- Business Impact: Demonstrated improvement in SME product uptake or risk-adjusted returns in simulation.
- Functional Demo: Prototype shows how different SMEs receive different personalized rates for the same product.

4. Deliverables

- A working prototype of personalized SME interest rate engine.
- Pricing simulation dashboard showing impact of different input factors.
- Demo interface where product managers input SME data and receive a personalized interest rate recommendation.

AI POWERED SYNTHETIC USER RESEARCH FOR SME DIGITAL BANKING

1. Challenge Overview

SMEs are a vital but diverse customer segment for banks, with varied industries, digital maturity levels, and financial needs. Traditional research methods (interviews, focus groups, surveys) are slow, costly, and often fail to capture the full spectrum of SME behaviors. This limits banks' ability to design, test, and refine digital banking products that truly meet SME expectations.

AI-powered Synthetic User Research introduces a scalable alternative. By leveraging Generative AI trained on SME personas, market research, and anonymized usage data, banks can simulate realistic SME feedback at scale. This enables product teams to test new features (e.g., lending, payments, cashflow dashboards, FX services, business networking), evaluate UX flows, validate pricing strategies, and predict adoption trends — all before engaging real SMEs.
The challenge is to build a prototype solution that generates SME-specific synthetic feedback, simulates digital banking journeys, and produces actionable insights. This empowers product teams to design more relevant, user-centric SME digital services, while reducing research time and cost.

AI POWERED SYNTHETIC USER RESEARCH FOR SME DIGITAL BANKING

2. Technical Requirements

The prototype should:

- Programming Languages: Java Spring boot, Reactjs
- Cloud Services: AWS EKS/ECS, S3
- AI Models: LLMs (OpenAI, Gemini, Claude...), fine-tuned SME persona models
- Database: MySql, MongoDb

Others: Kafka, Redis, Docker, Kubernete, GitLab CI + ArgoCD, OpenAPI/Swagger...

3. Evaluation and Measurement

- Persona Simulation Quality: AI-generated SME personas should reflect diverse industries, sizes, and digital maturity levels.
- Scenario Testing: Ability to simulate SME reactions to digital banking features (e.g., lending, onboarding, payments, FX, analytics, networking).
- Insight Accuracy: AI insights align with known SME pain points (e.g., liquidity, digital adoption, user experience).

Agility: Reduced time-to-insight compared to traditional SME research.

- Relevance: Outputs are actionable for SME product, design, and strategy teams.

-Functional Demo: Prototype demonstrates generating synthetic SME feedback for a defined digital banking scenario.

AI POWERED SYNTHETIC USER RESEARCH FOR SME DIGITAL BANKING

4. Deliverables

- A working prototype of SME-focused synthetic user research platform.
- AI-generated SME personas, customer journey maps, and feedback simulations.
- Demo interface where product teams input hypotheses or feature designs and receive synthetic SME insights.

FLOW ANALYSIS & NOTIFICATIONS

1. Challenge Overview

Currently, during development and operations, multiple product streams are deployed simultaneously with frequent go-lives. The quality assessment of each product stream after go-live still heavily relies on manual processes, which are time-consuming and resource-intensive.

The goal of the project is to leverage the power of Artificial Intelligence (AI) to automate the analysis and synthesis of product operation streams, including key metrics such as Turnaround Time (TAT), exceptions, as well as log and related data analysis. This solution will help users detect issues early, proactively address them, and provide timely alerts, thereby enhancing operational efficiency, minimizing risks, and improving product quality.

2. Technical Requirements

The prototype should:

- Programming Languages: Python, Nodejs, Java,JavaScript...
- Cloud services: AWS
- Model AI: any LLM (OpenAI, Gemini, Claude...)
- Others: Docker, Kubernetes

FLOW ANALYSIS & NOTIFICATIONS

3. Evaluation and Measurement

- Flow Simulation: Ability to recognize and simulate the product flow path.
- Alerts: Automatic alerts when the flow deviates or abnormal logs are detected.
- AI-powered Log Fix Suggestions: Use AI to process exception logs and recommend possible resolutions.
- AI-based Flow Improvement Suggestions: Provide at least one improvement proposal for the flow using AI insights.
- Function Demo: Product flow visualization, alert rules, and alert notifications.

4. Deliverables

Working Prototype including:

- ReactJS web (admin dashboard).
- Java Spring Boot backend (APIs, business logic management).
- Python AI services (NLP, matching, prediction, script generator).
- AI-powered Service Matching Engine integrated end-to-end.
- Demo Flow showcasing

MINI WORKFLOW

1. Challenge Overview

During development, the team is simultaneously implementing multiple product flows. These flows often reuse predefined APIs, screens, and components to optimize development efficiency. However, when creating new APIs, the team tends to reuse existing functions without a clear method to identify the internal flow of each API. As a result, new flow development is still carried out manually, lacking explicit configuration. This leads to difficulties in assessing the impact when modifying an API or adding new flows.

Project Objective:

Dynamic Workflow Configuration: Automatically configure workflows based on existing or newly developed APIs and components and visualize them through auto-generated workflow diagrams.

API Flow Configuration: Define and manage the internal flows within each API.

Impact Assessment: Evaluate and forecast the potential impact when introducing or modifying a flow in the system.

Audit & Testing: Support auditing and testing of flows to ensure system quality and stability

MINI WORKFLOW

2. Technical Requirements

- The prototype should:
- Programming Languages: Java, Python, Node.js, TypeScript, ...
- Cloud Services: AWS
- Others: Docker, Kubernetes, diagram generation tools (Mermaid.js, draw.io API, etc.)

3. **Evaluation and Measurement**

- Product Flow Configuration: Configure and review workflows directly on the interface.
- API-Level Workflow Configuration: Define and manage workflows within each API.
- Audit & Flow Testing: Run audits and test flows to ensure quality.
- Alerts on Shared API/Function Changes: Trigger alerts when shared APIs or functions are modified.

4. **Deliverables**

Working Prototype including:

- ReactJS web (admin dashboard).
- Java Spring Boot backend (APIs, business logic management).
- AI-powered Service Matching Engine integrated end-to-end.
- Demo Flow showcasing

Technology Hackathon 2025 – IT Challenge Statement #28

VPBank TECHNOLOGY HACKATHON Hack2Hire

AI AND DATA MINING FOR FRAUD PREVENTION AND CUSTOMER-CENTRIC CARD SERVICES

1. Challenge Overview

With the rise of advanced technologies, fraud and scams in banking, especially in card transactions, are becoming more sophisticated, threatening both customers and the bank's reputation. By applying data mining and AI, we can analyze spending habits, identify suspicious transactions, and deliver real-time alerts to prevent fraud. At the same time, the system leverages customer card usage patterns and the bank's criteria to recommend tailored promotions and suggest the most suitable card types, creating both security and personalized value for customers

2. Technical Requirements

The prototype should:

- Programming Languages: Python, SQL, Java, Javascript, Node.js,...
- Cloud services: AWS services (SageMaker, Fraud Detector, Personalize, Kinesis, Neptune)
- Model AI: XGBoost, Autoencoder, LSTM, GNN, Collaborative Filtering.
- Others: Docker, Kubernetes, diagram generation tools (Mermaid.js, draw.io API, etc.)

VPBank

AI AND DATA MINING FOR FRAUD PREVENTION AND CUSTOMER-CENTRIC CARD SERVICES

3. Evaluation and Measurement

The proposed solutions will be evaluated using the following dimensions:

3.1 Fraud Detection Performance

- Accuracy of identifying fraudulent transactions.
- Metrics: Precision, Recall, F1-score, False Positive Rate.

Personalization Effectiveness

- Relevance and usefulness of promotions or card recommendations.
- Measurement: Customer engagement rate (e.g., click-through, acceptance rate).

System Efficiency

- Processing speed and scalability for large transaction volumes.
- Ability to operate in real-time or near real-time.

3.2 Business Impact

- Potential reduction in fraud-related losses.
- Value creation for customers (improved satisfaction, loyalty).

3.3 Compliance & Security

- Data privacy, anonymization, and regulatory alignment.
- Explainability of AI-driven decisions.

Presentation & Usability

- Clarity of solution design, dashboards, and reporting.
- Professionalism and persuasiveness of the pitch/demo.

AI AND DATA MINING FOR FRAUD PREVENTION AND CUSTOMER-CENTRIC CARD SERVICES

4. Deliverables

- AI Model Artifacts: trained fraud detection model + recommendation model.
- System Architecture: AWS-based architecture diagram.
- Demo / Prototype: real-time fraud alert + customer offer suggestion.
- Documentation: approach, technical stack, evaluation metrics, business case

Technology Hackathon 2025 – IT Challenge Statement #29

VPBank VPBANK TECHNOLOGY HACKATHON Hack2Hire powered by aws

OCR HANDWRITTEN DOCUMENTS

1. Challenge Overview

To approve opening ekyc for corporation customers, DVTK (Dịch vụ tài khoản) is currently required to manually verify the Chief Accountant Appointment Decision document. This manual process is time-consuming and prone to errors.

Moreover, there is no standardized format for this document across businesses. Samples can be easily found online, and they often contain both printed and handwritten text, which further complicates the verification process

2. Technical Requirements

- Programming language: Java/Python
- AI/ML: Using Large Language Models (LLM)
- Natural Language Processing (NLP): Text Classification
- Deployment infrastructure: AWS
- Frontend (if any): ReactJS

3. Evaluation and Measurement

- OCR speed: Processing time < 15 seconds per case
- Convincing demo & logical presentation

4. Deliverables

- Demo version: web or app
- Solution presentation slides: including architecture and algorithms.

VPBank

USER ROLE MATRIX SYSTEM

1. Challenge Overview

Currently, the management of the User Role Matrix (URM) is primarily handled manually and through email-based approvals, which leads to various risks such as errors, delays, and limited control. Furthermore, there is no centralized system in place to manage, approve, and update the URM in real-time, making role and access management complex and lacking transparency.

The proposed solution aims to enhance role and permission management efficiency, standardize the approval workflow, and ensure transparency throughout the entire lifecycle of access management. With this system, managing the URM becomes more proactive, accurate, and controllable, reducing potential risks and improving overall operational efficiency

2. Technical Requirements

The system should ensure:

- Programming Languages: TypeScript, Node.js, or Python
- Frontend: React / Next.js (supporting rapid UI prototyping)
- Cloud Infrastructure: AWS / GCP / Azure
- Database: PostgreSQL or MongoDB (for storing role matrices and approval histories)
- File Management: AWS S3 or equivalent (for storing uploaded documents)
- AI Integration (optional): Automated role and permission suggestions based on user roles or historical usage
- Containerization & Deployment: Docker, Kubernetes

USER ROLE MATRIX SYSTEM

3. Evaluation and Measurement

- Role Data Management: Ability to receive, store, and manage role and permission data in a centralized, structured, and easily searchable manner.
- Approval Workflow: Supports multi-level approval flows tailored to specific functional groups or departments.
- Request Handling: Capable of receiving, tracking, and managing role and permission requests, including creation, updates, and revocations.
- Automation: Automatically grants or revokes roles and permissions according to the defined approval workflow.
- Traceability & Transparency: Maintains a complete audit trail and provides easy access for tracking and auditing purposes.
- Integration Capability: Ready to connect with other internal systems when required.
-System Stability: Ensures smooth performance and scalability to handle large volumes of requests

4. Deliverables

- A fully functional prototype of the User Role Matrix (URM) Management System.
- A sample process flow demonstrating steps from request creation to approval and role assignment.
- An interactive demo interface to manage, monitor, and review request statuses and outcomes.
- Reports and audit logs for monitoring, evaluation, and traceability purposes.

Technology Hackathon 2025 – IT Challenge Statement #31

VPBank powered by aws
VPBANK TECHNOLOGY HACKATHON
Hack2Hire

SMARTQUERY SME

1. Challenge Overview

Currently, the SME Portal provides access to business documents through a predefined category-based selection. However, this approach is limited in flexibility and efficiency, as users often spend time browsing multiple categories before finding the desired information.

The proposed solution is to integrate a Natural Language Query (NLQ) module into the SME Portal. Instead of navigating by categories, users can simply enter questions or requests in natural language. The system will process the query and return relevant, accurate, and timely documents, while also recording query history to enhance user experience and support access management.

2. Technical Requirements

The system should ensure:

- Programming Languages: Python, Node.js, or TypeScript.
- Frontend: React / Next.js with seamless integration into the existing SME Portal.
- Database & Search: Optimized solutions for storing and querying business documents.
- File Management: Centralized storage and secure access to original documents.
- API Integration: Tight connectivity with SME Portal and related internal systems.
- Deployment: Containerized architecture to ensure scalability, flexibility, and stable operation

VPBank

SMARTQUERY SME

3.Evaluation and Measurement

- Search Effectiveness: Ability to process and return relevant results quickly.
- Accuracy: Documents retrieved match user intent and search requirements.
- User Experience: Intuitive, seamless, and fully integrated with SME Portal.
- Performance Improvement: Enhanced search capability compared to traditional category-based browsing.
- Integration Capability: Flexibility to connect with other systems via APIs.
- Stability & Performance: High system reliability under large-scale query loads

4. Deliverables

- Natural Language Query Module fully integrated into the SME Portal.
- User Interface for entering queries, viewing results, and managing search history.
- Document Processing Engine to analyze and return the most relevant documents.
- Feature Demo showcasing advanced search capabilities compared to the existing method.
- Comprehensive Reports on query history and overall system performance

Technology Hackathon 2025 – IT Challenge Statement #32

VPBank VPBANK TECHNOLOGY HACKATHON Hack2Hire

powered by aws

GUARANTEE ISSUANCE PROCESS ON INTERNET BANKING

1. Challenge Overview

VPBank is currently aiming to develop its Online Bank Guarantee service via Internet Banking. Customers can submit guarantee issuance requests on the system and upload all required documents as per the bank's regulations. Credit officers will receive the requests and process them. If all conditions are met, the bank will issue the guarantee to the customer.

Requirements: Develop both the Internet Banking interface and a backend system for bank officers, with the following capabilities:

Automatically extract key data fields from uploaded documents and forms to populate the Internet Banking interface, minimizing manual data entry for customers.

Allow customers to apply for multiple types of guarantees using the same set of documents (e.g., Performance Guarantee, Advance Payment Guarantee, etc.).

Support the generation of Guarantee Letters in customer-specific templates upon request. The system must be able to populate these custom templates with accurate guarantee data issued by the bank.

VPBank

GUARANTEE ISSUANCE PROCESS ON INTERNET BANKING

2. Technical Requirements

The system must meet the following specifications:

- Programming language: .NET
- Cloud infrastructure: AWS / GCP / Azure
- AI model: Any LLM (e.g., OpenAI, Gemini, Claude...)
- Database: SQL Server
- File management: Support for storing uploaded documents.
- Containerization & deployment: Docker, Kubernetes

3. Evaluation and Measurement

- Document understanding and data extraction:

Ability to process various document types uploaded by customers (PDF, Word, image files such as .jpeg, .png) and accurately extract the required fields defined by the bank. The system should also allow users to review and edit extracted information.

- Support for multiple guaranteed types per request:

System capability to handle applications for multiple types of guarantees using the same document set. Ability to identify and map shared and specific fields between guarantee types. The system should provide a clear and simplified workflow for handling multiple guarantee requests in one submission.

GUARANTEE ISSUANCE PROCESS ON INTERNET BANKING

- Custom Guarantee Letter generation:

Ability to accept customer-provided guarantee letter templates (PDF format) and populate them with issuance data while preserving the original layout and formatting. Must accurately place the required fields in the correct positions and support multiple templates for different customers or transactions.

- System architecture and integration readiness:

Clear, scalable, and secure proposed architecture design. Deployment model should support scalability (preferably cloud-native or microservices-based). Security measures such as authentication, access control, and document protection should be clearly defined.

- Scalability and future improvement readiness: Flexibility to support future enhancements such as additional guarantee types or new document formats.

- Integration capability: Readiness to integrate with other internal systems when required.

- System stability: Reliable performance to handle a high volume of requests smoothly.

GUARANTEE ISSUANCE PROCESS ON INTERNET BANKING

4. Deliverables

- A fully functional prototype of the Online Bank Guarantee system, including Internet Banking interface for customers and backend interface for bank officers.
  - ✓ A sample process flow demonstrating the full cycle from request submission and document upload to review, approval, and guarantee issuance.
  - ✓ An interactive demo interface that enables:
- Uploading customer documents
- Automatic data extraction
- Registering multiple guarantee types in one submission
- Generating Guarantee Letters using customer-specific templates
- Tracking request status and issuance results
- Logging and reporting features to support internal control, performance monitoring, and audit trail capabilities.
- System design documentation, including architecture diagrams, API descriptions, data models, and proposed security configurations

AI REAL ESTATE COLLATERAL DRAFTING SYSTEM

1. Challenge Overview

Currently, the drafting and management of real estate collateral documents (including legal certificates, business registration, and ID/Passport) are still handled manually: staff must read paper documents, extract information, and manually input it into contract templates. This process is time-consuming, error-prone, difficult to control, and causes delays in loan processing.

Proposed solution:

- Apply AI/OCR to automatically read and extract data from legal documents such as real estate ownership certificates, business registration certificates, and ID/Passports.

- AI will automatically draft the collateral contract with data accurately populated based on the original documents.

- The system helps standardize processes, reduce errors, save time, and ensure transparency with full traceability.

AI REAL ESTATE COLLATERAL DRAFTING SYSTEM

2. Technical Requirements

The system must ensure:

- Programming languages: Python, Node.js (AI/ML + backend)
- Frontend: React / Next.js (data input and validation UI)
- AI/OCR Engine: Integration with Google Vision API, AWS Textract, or OpenAI Vision model
- Database: PostgreSQL / MongoDB (store documents, extracted data, drafted contracts)
- File management: AWS S3 or equivalent
- Data security: Encryption of personal and legal data
- Deployment: Docker, Kubernetes (Cloud-ready: AWS / GCP / Azure)

3. Evaluation and Measurement

- Data extraction accuracy: AI must recognize and read ≥ 90% of data fields from PDF/Image documents.
- Contract accuracy: Generated contracts must match ≥ 95% with the original source documents.
- Approval workflow: Support for multi-level approval (legal, credit, customer).
- Processing time: Reduce by ≥ 60% compared to manual data entry.
-Automation: End-to-end automation from OCR → validation → contract generation → storage.
- Traceability: Log all activities (reader, editor, approver).
- Integration capability: Ready to connect with Core Banking or BLOS (Business Loan Origination System).

AI REAL ESTATE COLLATERAL DRAFTING SYSTEM

4. Deliverables

- A fully functional prototype of AI OCR + automated collateral contract system.
- Sample workflow:
  - ✓ Upload legal documents →
  - ✓ AI reads & extracts data → Staff validates data →
  - ✓ System generates collateral contract →
  - ✓ Store & submit for approval/signing.
- Demo interface: Allows upload, review extracted data, edit, and track processing progress.
- Reports & activity logs: For monitoring, evaluation, and traceability.

PAPER TO PROTOCOL: REGCHECK AGENT FOR BANKING COMPLIANCE AUTOMATION

1. Challenge Overview

From Paper to Protocol: RegCheck Agent for Banking Compliance Automation invites participants to build a Generative AI Agent that can automate compliance checking in banking workflows. Whether it's validating trade finance documents or verifying car loan disbursement packages, the agent should understand regulatory frameworks (e.g., UCP600, internal policies), interpret document content, and assist staff in identifying non-compliance. The goal is to create a scalable solution that can be adapted across multiple banking domains, reducing manual effort and increasing accuracy.

PAPER TO PROTOCOL: REGCHECK AGENT FOR BANKING COMPLIANCE AUTOMATION

2. Technical Requirements

- Participants are expected to design and prototype a Generative AI Agent (or Agentic AI system) capable of:
- Parsing and understanding semi-structured documents (e.g., invoices, contracts, transportation orders).
- Interpreting regulatory texts and internal compliance rules.
- Comparing document content with system data and regulations to flag inconsistencies.
- Providing human-understandable explanations or suggestions for compliance issues. Solutions can be built using any modern GenAI frameworks (e.g., LangChain, LlamaIndex, OpenAI APIs, etc.) and should be modular enough to support future domain expansion

PAPER TO PROTOCOL: REGCHECK AGENT FOR BANKING COMPLIANCE AUTOMATION

3. Evaluation and Measurement

- Submissions will be evaluated based on five key criteria:
- Functionality (30%): Accuracy and reliability of compliance checking.
- Scalability (20%): Ability to adapt the solution to other banking domains.
- Innovation (20%): Novelty in approach, architecture, or user interaction.
- Usability (15%): Ease of use for VPBank staff and clarity of outputs.
- Presentation (15%): Clarity, completeness, and persuasiveness of the final pitch

4. Deliverables

By the end of the hackathon, each team must submit:

- Source code of the prototype (with documentation).
- A working demo or video walkthrough of the RegCheck Agent.
- A design document outlining architecture, domain adaptability, and regulatory handling.
- A final presentation (10–15 minutes) to showcase the solution to the judging panel.

GRAPH-BASED DATA LINEAGE AND DATA MODELING

1. Challenge Overview

This challenge is designed for data engineers or architects skilled in data modeling and familiar with graph databases. We aim to identify a candidate who can design and implement a graph-based solution for tracking data lineage using our technology ecosystem (see Technical Requirements). As our data pipelines grow in complexity, understanding how data flows from sources through transformations to final outputs is critical for governance, impact analysis, and troubleshooting. Currently, lineage information may be scattered across tools (e.g., ETL jobs, dbt models) or documented manually. This challenge focuses on using a graph model to capture end-to-end data lineage and inter-relationships between data entities, providing an interactive and visual way to explore how data moves through our system.

Demand of Graph-based Lineage Solution

We need a graph-centric data lineage tool that will help:

- Model Data Assets as a Graph: Define a graph schema where nodes represent data assets (such as tables, columns, datasets, or data processing jobs) and edges represent relationships or data flow. Key nodes and edges might include source and target tables and columns, transformation and joining conditions. The input data include SQL scripts or jobs or stored procedures, dbt model catalog and manifest files, Airflow DAGs, analytics mode

GRAPH-BASED DATA LINEAGE AND DATA MODELING

- **Metadata from Multiple Sources:** Automatically gather data lineage-relevant metadata at column level from multiple tool:

- From dbt: capture dependencies between models.

- From Airflow: capture task dependencies that involve data movement or transformation (for instance, an Airflow DAG that extracts data from source and loads into Redshift

-Other sources could include data catalog exports or even parsing SQL scripts if applicable.

The solution should provide an automated or semi-automated way to ingest these metadata and populate the graph, rather than hard-coding the lineage.

- Visualize and Query Lineage: Provide the ability to explore the lineage through visual or query-based means. This could involve using a graph database's query language (for example, Cypher in Neo4j) to answer questions like "What are all the upstream sources for this data table?" or "Which reports or tables will be impacted if this source file is delayed?". Ideally, participants should demonstrate a simple visualization of at least part of the lineage graph. The focus is on clarity: it should be easy to follow the path from a source node through transformations to an output and to column level.

- Highlight Data Relationships: In addition to lineage (which is typically directional data flow), the graph approach can model other relationships. For example, indicate if certain datasets share common dimensions or if a column in one table is derived from a column in another. The challenge can include data modeling in the graph such that the graph doubles as a data catalog: nodes could have properties (like column data types, owners, etc.) and edges could also represent conceptual links. This is an extra aspect to showcase how a graph model might unify data lineage with general data modeling (hence the challenge title).

GRAPH-BASED DATA LINEAGE AND DATA MODELING

2. Technical Requirements

- The prototype should be implementable using AWS resources and integrate with our current tech stack:

(1) Graph Database: Use Neo4j (preferred graph DB for this challenge) to store the lineage graph. Participants can use Neo4j Aura (cloud) or run Neo4j on AWS (EC2 or Docker). Knowledge of Cypher query language will be useful for creating and querying the graph.

(2) Data Integration Tools: The solution may ingest metadata from dbt, Airflow, and Redshift DDL:

- For dbt: The prototype can use dbt's artifact files (like manifest.json and catalog.json). These contain the DAG of models and sources and their relationships, as well as schema info.

- For Airflow: If direct integration is complex, participants can assume Airflow DAG definitions are available (e.g., as code or a simple representation of which tasks load which tables). Using open-source lineage collectors (like OpenLineage) is acceptable but not mandatory. Even a configuration file listing some known pipelines and their inputs/outputs could be used to simulate Airflow metadata.

GRAPH-BASED DATA LINEAGE AND DATA MODELING

- For Redshift: Use system information (for instance, information_schema or STL tables) for lineage clues, or rely on dbt (since dbt already knows which source tables feed which models, which create which tables in Redshift). At minimum, the lineage graph should include Redshift tables a or notebooks that read these sources and upsert nodes/edges into Neo4j.

s nodes (from staging and final models).

- The ingestion process can be done with Python scripts (3) AWS Environment & Other Tech: The entire solution should be workable in an AWS context. Using S3 to store any intermediate metadata (for example, placing dbt manifest files or Airflow log exports there) is fine. If using Airflow, you might use an Airflow DAG to run the lineage extraction job itself. Ensure any components used (e.g., a small web app for visualization) can be deployed on AWS (for instance, a Streamlit app or a simple React front-end could be hosted accordingly).

(4) Languages: Python is recommended for extracting and loading metadata (thanks to its APIs and libraries for AWS, dbt, etc.). Cypher will be used within Neo4j for querying. SQL may be used to query Redshift system tables if needed

GRAPH-BASED DATA LINEAGE AND DATA MODELING

3. Evaluation and Measurement

No	Measurement	Detail
1	Performance	Ingestion Speed: How efficiently the solution can load metadata and build the graph. For instance, processing time for ingesting a given set of dbt/Airflow metadata into Neo4j.
		Query Efficiency: The responsiveness of lineage queries. For example, retrieving all upstream sources for a table should be quick and not time out or hang. (This also reflects good graph design and indexing.)
2	Completeness	Lineage Coverage: The extent to which the prototype captures the full lineage. High marks if the solution covers multiple stages (source, staging, model, final) and multiple tools (e.g., both dbt and an Airflow ETL step). We will provide sample metadata; an ideal solution incorporates all of it into the graph.
		Accuracy: The correctness of relationships in the graph. Edges should correctly reflect actual data dependencies (no incorrect links). For example, if Table A doesn't actually feed Table B, it should not appear as related. Conversely, important dependencies from the input metadata should all be represented in the graph.
3	Visualization Clarity	The lineage is presented in a clear and insightful way. If a visual graph is provided, it should be organized and legible (not a jumbled hairball of nodes). If the interface is query-only (no GUI), the textual output of queries should still be easy to understand. Essentially, the solution should help a viewer quickly grasp the lineage relationships. Consideration will be given to any extra features like filtering the graph or focusing on a particular sub-tree of the lineage.

GRAPH-BASED DATA LINEAGE AND DATA MODELING

4. Deliverables

- Graph Schema & Design Documentation: A document explaining the graph data model (what node and relationship types were used, and what properties they carry), plus a brief on design decisions (e.g., how you integrated each metadata source, any assumptions or simplifications made). This should also cover any setup needed to run the solution (for example, Neo4j configuration).
- Prototype Demo (Code & Visualization): The working prototype which includes:
- Metadata ingestion code: e.g., Python scripts or a Jupyter Notebook that reads metadata and populates the Neo4j graph.
- Demo of lineage queries/visualization: This could be a recorded demo or live demo. If a UI was built, provide it (or a link). If not, provide a series of example Cypher queries and their results that demonstrate how a user can explore the lineage (for instance, a query to find all downstream dependencies of a source table, and the returned list of tables/jobs).
- The Neo4j database (or a dump of it) with the sample data loaded is a plus, so judges can explore if desired. Ensure to include any instructions or configuration required to run the Neo4j instance and the ingestion process (like dependencies, install guides, etc.).

AUTO BUSINESS DATA DICTIONARY

1. Challenge Overview

The organization has business terminology stored in documents (e.g BRDs and SRSs) , but this information is static, fragmented, and hard to access. As a result, teams face difficulties in understanding business procedures and definitions. The goal is to build an automatic tool that extracts and structures business terms from these documents into a centralized data dictionary, improving clarity and cross-team alignment.

2. Technical Requirements

The prototype should:

- Programming Languages: Nodejs, Python, Typescript...
- Cloud services: AWS
- Model AI: any LLM

AUTO BUSINESS DATA DICTIONARY

3. Evaluation and Measurement

The prototype will be evaluated based on its ability to extract and manage business terms and procedures from documents (BRD/SRS) . Key measurable criteria include:

- Extraction Accuracy: ≥ 90% for business terms, ≥ 85% for procedures
- Deduplication Effectiveness: ≥ 90% of similar terms normalized
- Usability Score: ≥ 80% user satisfaction from feedback survey
- Search Performance: < 2 seconds to retrieve a term
- Version Control: Support for tracking ≥ 3 versions of each term
- Export Capability: Export dictionary to Excel or JSON
- Business Coverage: ≥ 90% of relevant business areas captured

Evaluation will include functional testing and user validation against a curated benchmark.

4. Deliverables

- Presentation
- Prototype Demo

THANK YOU