HackX 4.0

Hackathon Problem Statements

Problem Statement 1: The Vibe-Audit

Challenge

Build an automated “Production-Ready” gatekeeper for applications developed through Vibe Coding (natural language generation).

The Problem

AI-generated applications may appear well-designed but often overlook critical aspects such as security, PII compliance, and cost efficiency.

The Goal

Develop a tool that scans a “vibe-coded” repository and generates a structured Go/No-Go production readiness report covering:

- Hardcoded secrets and prompt injection risks
- Compliance gaps (GDPR / SOC2)
- Hallucinated or insecure dependencies

Outcome

A comprehensive “Vibe-to-Value” score along with actionable auto-remediation prompts.

Problem Statement 2: The SME-Plug

Challenge

Develop a “hot-swappable” Subject Matter Expert (SME) plugin for AI agents.

The Problem

General-purpose AI agents often lack deep domain expertise and may hallucinate when handling highly technical or niche tasks.

The Goal

Design a universal skill plugin that injects specialized knowledge and structured reasoning logic into any AI agent.

The Deliverable

A modular plugin that:

- Enforces “Source of Truth” citations
- Injects domain-specific decision trees (e.g., “Think like a Structural Engineer”)
- Works across multiple agent frameworks (LangChain, AutoGPT, etc.)

Outcome

A scalable, plug-and-play expert module adaptable to complex industry use cases.

Problem Statement 3: Banquet Management System

1. Background

Banquet halls operating across multiple branches often depend on manual registers, spreadsheets, and disconnected tools to manage leads, bookings, billing, and event execution. This fragmented system leads to operational inefficiencies, revenue leakage, missed follow-ups, booking conflicts, and limited visibility for owners.

There is a need for a centralized, scalable, and easy-to-use banquet management system that integrates lead handling, booking management, event coordination, vendor management, and financial tracking into a single platform.

2. Problem Definition

Multi-branch banquet businesses face the following challenges:

- Inconsistent lead tracking and missed follow-ups
- Double bookings due to lack of real-time availability tracking
- Manual coordination between sales, kitchen, operations, and vendors
- Inaccurate or delayed billing and payment tracking
- No centralized branch-wise reporting or performance visibility
- Lack of menu planning

These issues directly impact revenue, operational efficiency, and overall customer experience.

3. Objective

To develop a comprehensive banquet management software that:

- Is simple and user-friendly for non-technical managers
- Supports centralized multi-branch operations with branch-level control
- Provides admin-based role access (Sales, Inventory, Kitchen, Property Manager, etc.)
- Manages leads through a structured end-to-end pipeline
- Prevents booking conflicts using real-time availability tracking (including nearest available branch suggestion)
- Streamlines event planning and vendor coordination workflows
- Automates billing, tax calculation, and payment tracking
- Provides role-based dashboards and performance analytics

Mandatory Features (MVP)

Multi-Branch Management

- Centralized owner dashboard
- Branch-level access control
- Branch-wise performance tracking

Lead Management

- Lead capture and manual entry
- Complete lifecycle tracking:
  Call → Property Visit → Food Tasting (optional) → Advance Payment → Menu Finalization → Decoration & Event Finalization → Full Payment → Post-Event Settlement (extra/refund) → Feedback
- Lead status pipeline (New → Converted / Lost)
- Follow-up reminders and assignment

Booking & Calendar Management

- Real-time availability calendar
- Conflict prevention mechanism
- Advance and balance payment tracking

Event Management

- Guest count, menu, and add-on selection
- Vendor allocation (internal/third-party)
- Event checklist and execution tracking

Billing & Payments

- Invoice generation
- Automated tax calculation
- Online and cash payment tracking
- Outstanding reminders

Reports & Analytics

- Revenue reports (overall & branch-wise)
- Conversion rate tracking
- Occupancy and outstanding summaries

Third-Party & Panel Vendor Management

- Vendor onboarding and panel management
- Tracking of external food/decor vendors
- Settlement and performance monitoring

Raw Material Management

- Real-time tracking of raw materials with low-stock alerts across branches
- Automatic stock deduction based on finalized menu and guest count
- Purchase order management with supplier tracking and cost analysis

---

Bonus Features

- AI-based lead scoring
- WhatsApp automation for follow-ups and reminders
- Dynamic pricing engine

- Vendor and inventory management
- Customer self-service portal
- Staff mobile application

Problem Statement 4: Review Management System

1. Background

Organizations receive customer reviews from multiple sources including internal feedback forms, staff interactions, and third-party platforms such as Google and Zomato. These reviews are scattered across different systems with no centralized visibility.

Due to this fragmented structure, management struggles to monitor customer sentiment, respond on time, and analyze performance effectively. Additionally, there is no structured mechanism to actively collect reviews directly from customers after service or events.

2. Problem Definition

Multi-branch businesses face the following challenges:

- No centralized platform to view all reviews in one place
- Reviews distributed across internal systems and third-party platforms
- Lack of structured review collection from customers
- Difficulty in categorizing reviews (food, service, staff, ambience, etc.)
- Delayed or inconsistent responses to feedback
- No branch-wise or staff-wise performance insights

These issues negatively impact brand reputation, service quality monitoring, and customer retention.

3. Objective

To develop a centralized Review Management System that:

- Aggregates reviews from internal and third-party platforms
- Enables structured review collection through a dedicated mobile application
- Categorizes reviews automatically based on service parameters

- Provides automated and customizable reply options
- Offers branch-wise and staff-wise performance insights
- Improves customer engagement and reputation management

Mandatory Features (MVP)

Centralized Review Dashboard

- Unified view of all reviews (internal + external platforms)
- Branch-level filtering and access control
- Real-time review updates

Review Collection Mobile App

- Simple mobile app for collecting customer feedback post-event
- QR code / link-based review submission
- Rating + category-based feedback form
- Staff tagging option
- Direct sync to central dashboard

Review Categorization

- Auto-categorization (Food, Service, Staff Behavior, Cleanliness, etc.)
- Manual override option
- Tag-based classification

Response Management

- Predefined automated reply templates
- AI-assisted response suggestions
- Response tracking and history

Analytics & Reporting

- Sentiment analysis (Positive / Neutral / Negative)
- Branch-wise rating comparison
- Staff performance insights
- Monthly trend reports

Bonus Features

- AI-based deep sentiment insights
- Escalation workflow for negative reviews
- WhatsApp/SMS alerts for critical feedback
- Competitor review benchmarking
- Customer satisfaction score (CSAT) tracking

Problem Statement 5: COSMEON FS-LITE (Orbital File System Simulation)

Problem Summary

COSMEON requires a lightweight, distributed file-system simulation to demonstrate how data may be stored across multiple orbital nodes. The goal is to create a minimal distributed file system (FS-Lite) that splits files into chunks, distributes them across nodes, and reconstructs them on request.

Expectations

Participants must design a working prototype of a file system capable of basic upload, chunk storage, metadata tracking, and download reconstruction in a multi-node environment.

Minimum Requirements

- Implement file upload functionality where a file is divided into multiple chunks
- Simulate multiple “satellite nodes” (folders, processes, containers, etc.)
- Assign each file chunk to different nodes using a simple distribution strategy
- Maintain metadata describing chunk locations
- Implement file download functionality that fetches all chunks and reconstructs the original file
- Provide a simple mechanism to simulate node failure and demonstrate system behavior
- Include a basic integrity check (checksums or hash validation)

Optional Enhancements

- Automatic rebalancing of chunks when nodes fail or return

- A small dashboard or CLI logs showing chunk distribution and node status
- Caching mechanisms for faster retrieval

Problem Statement 6: Satellite Data to Insight Engine for Climate Risk

Problem Summary

Earth observation satellites continuously generate massive volumes of imagery and environmental data. However, most of this data remains underutilized because it requires advanced processing to convert raw imagery into actionable intelligence. Governments, insurers, urban planners, and agricultural stakeholders need automated systems that can detect flood events, infrastructure exposure, and environmental risk in near real time.

COSMEON requires a software-based intelligence engine that transforms open satellite data into structured, decision-ready climate risk insights.

Expectations

Participants must build a software pipeline that ingests open satellite datasets, processes imagery using machine learning or geospatial techniques, and outputs structured climate risk insights. The solution must demonstrate automated detection and reporting rather than simple visualization.

Minimum Requirements

- Ingest publicly available satellite imagery such as Sentinel 1, Sentinel 2, or Landsat datasets
- Perform automated detection of flood-affected regions or environmental risk zones using image processing or ML models
- Implement change detection comparing historical and recent satellite data
- Generate structured output such as affected area statistics, district-level summaries, or risk classification labels
- Maintain a state table or structured data store showing timestamps, geographic regions, and detected risk status
- Provide detailed logs demonstrating data ingestion, processing steps, and detection outputs

Optional Enhancements

- Integrate external datasets such as rainfall, elevation, or population density for enhanced risk modeling
- Implement predictive modeling to forecast potential flood risk based on historical trends
- Provide an API endpoint for retrieving processed insights programmatically
- Build an interactive dashboard visualizing affected zones over time
- Implement confidence scoring for detected risk areas