Since 1983
Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

IGNISIA AI HACKATHON

Dr. Vishwanath Karad
MIT World Peace University

PRESENTS

Ignisia

24 HOUR NATIONAL LEVEL AI HACKATHON

Problem Statements

SME **Tools for Small and Medium Enterprise**	**FT** FinTech	**HC** Healthcare & Accessibility
ED Education & Skill	**EN** Environment & Sustainability	**SC** Smart Cities

24 Hours · 6 Domains

Think. Build. Evolve.

Problem Statements — Index

PS ID	Title	Domain
SME01	Multi-Format Knowledge Retrieval Agent for SME Operations	Tools for SME
SME02	Autonomous RFP Response & Competitive Quotation Orchestrator	Tools for SME
SME03	Few-Shot Visual Quality Inspection System for Micro-Manufacturing	Tools for SME
FT01	Webhook Reconciliation Engine with Autonomous Ledger Healing	FinTech
FT02	Real-Time MSME Credit Scoring via Alternative Business Signals	FinTech
HC01	Agentic Diagnostic Risk Assistant for ICU Complication Detection	Healthcare & Accessibility
HC02	Contact-Free Cardiac Stress Monitoring via Facial Video Analysis (rPPG)	Healthcare & Accessibility
HC03	Golden-Hour Emergency Triage & Constraint-Based Hospital Routing System	Healthcare & Accessibility
ED01	AI-Assisted Answer Clustering & Grading Acceleration Engine	Education & Skill
ED02	Real-Time Procedural Skill Coach Using Hand-Tracking & Spatial Analysis	Education & Skill
EN01	Satellite-Based Urban Tree-Equity Auditor & Planting Prioritization Engine	Environment & Sustainability
EN02	Hyperspectral Methane Super-Emitter Detection, Quantification & Source Attribution	Environment & Sustainability
SC01	Predictive Last-Mile Transit Synchronizer with Dynamic Fleet Positioning	Smart Cities
SC02	Drone Image-Based 3D Infrastructure Defect Detection & Severity Scoring	Smart Cities

Tools For SME

Domain Code: SME

SME01 Multi-Format Knowledge Retrieval Agent for SME Operations

THE PROBLEM

Employees in SMEs spend enormous amounts of time hunting for information scattered across incompatible formats — old email threads, pricing spreadsheets, and policy PDFs all stored in different places. When a client query comes in, an employee may need to cross-reference three different systems before giving a confident answer. This fragmentation slows operations, creates inconsistent customer service, and causes costly errors when an outdated document is mistakenly used over a newer one. SMEs cannot afford enterprise knowledge management systems, yet they bear the same operational cost of siloed, unstructured data.

CORE TECHNICAL COMPLEXITY

The system must ingest and parse multiple unstructured data formats — PDFs, Excel spreadsheets, and raw text or email files — and build a unified Retrieval-Augmented Generation (RAG) pipeline capable of semantic search across all of them simultaneously. The core challenge is ensuring that tabular data (pricing rows, inventory columns) is read and reasoned about with the same accuracy as dense prose (policy clauses, email threads). A further layer of complexity is conflict resolution: when an old email and a new policy PDF contain contradictory information, the system must autonomously detect the conflict, prioritize the most recently dated document, and explain its reasoning to the user rather than silently returning a wrong answer.

EXPECTED OUTCOME

A unified conversational agent where any employee can ask a natural language question — for example, 'What is our refund policy for bulk orders quoted to Acme Corp last month?' — and the AI retrieves a precise, cited answer linking back to the exact document, spreadsheet row, or email paragraph used. When conflicting information exists across sources, the system surfaces the conflict explicitly and states which source it trusted and why.

EXACT DELIVERABLES

- A data ingestion pipeline that handles at least one PDF, one Excel file, and a set of mocked emails, chunked and stored in a vector database.
- A RAG-based conversational interface where employees ask natural language questions and receive sourced answers.
- Accurate source attribution on every response — the AI must link back to the exact document and section used.

- A conflict detection module that identifies when two documents contradict each other, prioritizes by date, and explains the decision to the user.
- A mocked CRM integration where a retrieved answer auto-populates a Support Ticket form with the relevant context.

RECOMMENDED TECH APPROACH

Use a Large Language Model (LLM) as the reasoning core. Build a RAG pipeline using a vector database for semantic document retrieval. Use a document parsing library for multi-format ingestion. Design a lightweight chat UI for the employee-facing interface.

SME02 Autonomous RFP Response & Competitive Quotation Orchestrator

THE PROBLEM

Drafting a customized response to a client's Request for Proposal (RFP) takes SMEs days of manual work — pulling current inventory pricing, researching what competitors are charging, calculating margins, and formatting a professional document. During this delay, faster competitors win the contract. Furthermore, without a clear view of market rates, SMEs either underprice and erode their margins or overprice and lose the bid entirely. This bottleneck is particularly damaging for SMEs that receive multiple RFPs simultaneously and lack the sales staff to process them in parallel.

CORE TECHNICAL COMPLEXITY

This requires a true multi-agent pipeline where each agent has a distinct, non-overlapping responsibility. One agent parses the unstructured RFP and extracts requirements. A second queries the SME's internal pricing database. A third queries a mocked competitor pricing source and applies competitive analysis logic. A final drafting agent synthesizes all gathered inputs into a formatted PDF quotation. The critical complexity lies in the competitive logic: the system must recognize scenarios where a competitor drops below the SME's base cost and autonomously pivot to a value-differentiation strategy — bundling a free warranty or service — rather than blindly matching an unsustainable price.

EXPECTED OUTCOME

A fully automated pipeline where a user pastes or uploads an RFP and the system autonomously checks internal inventory, analyzes competitive pricing, applies the appropriate pricing strategy, and generates a ready-to-send PDF quotation — complete with the agent's pricing rationale — in minutes rather than days.

EXACT DELIVERABLES

- A multi-agent framework with clearly separated roles: RFP Parser Agent, Pricing & Competitor Analysis Agent, and Proposal Drafting Agent.
- Integration with a mocked competitor database or API that the analysis agent queries dynamically.
- Auto-generation of a clean, professionally formatted PDF quotation with accurate line items and margin calculations.
- An approval UI showing the agent's pricing logic — including the reasoning behind any competitive strategy adjustment.
- Multi-currency and tax handling: the system must dynamically apply currency conversion and regional tax brackets when the client is international.

RECOMMENDED TECH APPROACH

Use a multi-agent orchestration framework (agentic LLM pipeline). Use a relational or document database for internal pricing data. Use a PDF generation library for the final quotation output. Use a currency conversion API for international quotes.

SME03 Few-Shot Visual Quality Inspection System for Micro-Manufacturing

THE PROBLEM

Indian manufacturing SMEs — producing textiles, leather goods, ceramics, and auto-components — rely entirely on manual visual inspection to catch defects. Enterprise machine vision systems cost crores of rupees, making them inaccessible to small workshops. The deeper problem is that SMEs operate on small-batch production: the product line changes weekly. A traditional AI model fails here because SMEs cannot collect and label thousands of defect images every time they switch products. What they need is a system that can learn what 'perfect' looks like from just 10–20 photographs, then immediately flag anything that deviates — without ever having been shown a defective example.

CORE TECHNICAL COMPLEXITY

The system must implement unsupervised anomaly detection or few-shot deep learning — architectures such as PatchCore or PaDiM — trained exclusively on 'normal' (non-defective) images. It cannot rely on labeled defect data. Instead, it must learn the statistical distribution of a 'Golden' reference product by performing deep feature extraction on a small calibration set. During inference, it calculates distance metrics in the embedding space (such as Mahalanobis distance) to detect and precisely localize microscopic deviations — scratches, thread-pulls, dents — from the learned norm, in real-time on standard consumer hardware.

EXPECTED OUTCOME

A low-cost, edge-deployable visual inspection system. A workshop supervisor uploads 10–20 photos of a perfect product to calibrate the AI in under one minute. A standard smartphone or webcam mounted over the workbench then inspects items in real-time, instantly flagging defective pieces and generating a precise anomaly heatmap showing exactly where on the item the flaw appears.

EXACT DELIVERABLES

- A few-shot anomaly detection pipeline calibrated exclusively on normal images — no defect labels used at any stage.
- A calibration UI where a supervisor uploads fewer than 20 good images and the model adapts in under 60 seconds.
- An inference engine processing a live webcam feed or test image set, outputting a real-time visual anomaly heatmap that localizes the defect on the item.
- A lightweight dashboard logging daily defect rates and saving flagged frames for supervisor review.

RECOMMENDED TECH APPROACH

Use a pre-trained deep learning vision model for feature extraction (no training from scratch). Apply an unsupervised anomaly detection algorithm over the extracted embeddings. Use a computer vision library for webcam feed handling and heatmap overlay rendering. Optimize for inference speed on CPU-class hardware.

FinTech

Domain Code: FT

FT01 Webhook Reconciliation Engine with Autonomous Ledger Healing

THE PROBLEM

Payment gateways — Razorpay, Stripe, Cashfree — emit a webhook for every transaction state change: payment created, captured, refunded. In distributed systems under real-world conditions, these webhooks routinely arrive out of chronological order, get dropped silently during server downtime, or duplicate themselves under retry storms. Without a reconciliation layer, orders stall permanently in 'pending' state, ledgers drift from reality, double charges go undetected, and settlement mismatches accumulate until a compliance audit surfaces them at enormous cost. SMEs relying on digital payments cannot afford this silent ledger rot.

CORE TECHNICAL COMPLEXITY

The system must infer the true final state of a transaction even when intermediate lifecycle events are missing. It must maintain a per-transaction state machine, detect chronological gaps in event sequences, and trigger a fallback API poll to the payment gateway to reconstruct missing events — healing the ledger automatically rather than queuing indefinitely. A further layer of difficulty is resilience: when the gateway experiences an outage and floods the system with thousands of backlogged, duplicated webhooks simultaneously, the system must handle idempotency correctly to avoid double-counting, and must apply dynamic rate-limiting to prevent database locking under the surge.

EXPECTED OUTCOME

A reconciliation microservice that ingests a stream of webhook events, maintains a per-transaction event log, detects sequence gaps using state machine logic, auto-heals by polling a mocked payment gateway API, and exposes a live dashboard displaying drift rate, heal success rate, and any unresolvable anomalies flagged for manual review.

EXACT DELIVERABLES

- A webhook ingestion service with robust idempotency key handling and a per-transaction ordered event log.
- A state machine that maps the expected transaction lifecycle and detects any missing or out-of-order events.
- An auto-healing module that calls a mocked payment gateway fetch API to reconstruct missing events and resolve ledger gaps.

- A live dashboard showing drift rate (%), heal success rate (%), and a manual-review queue for unresolvable anomalies.

RECOMMENDED TECH APPROACH

Use an event streaming or message queue system for webhook ingestion. Implement a finite state machine for transaction lifecycle tracking. Use a relational database for the event log and a fast cache layer for idempotency. Build the dashboard with a lightweight frontend or data visualization library.

FT02 Real-Time MSME Credit Scoring via Alternative Business Signals

THE PROBLEM

Banks and NBFCs reject approximately 80% of MSME loan applications because new businesses — under two years old — have no formal credit history, no ITR filings, and no audited financials. Conventional underwriting is built entirely on lagging documents: GST registration certificates, Udyam papers, bank statements that are already months out of date. A 6-month-old manufacturer with ₹40 lakh in confirmed live orders receives the same rejection as an inactive shell company, because the model cannot distinguish between them. Meanwhile, real-time digital footprints — the velocity of GST e-invoice generation, UPI cash-flow cadence, e-way bill volume trends — paint an accurate picture of current business health that traditional underwriting completely ignores.

CORE TECHNICAL COMPLEXITY

The model must consume dynamic, near-real-time signals — specifically focusing on GST filing timeliness and velocity, UPI transaction frequency and flow patterns, and e-way bill volume trends — as primary features. The pipeline must handle sparse and missing data streams gracefully (a new business may have only 3 months of GST history). The model must produce a continuously updating credit score with full explainability: every score must be accompanied by the top contributing factors expressed in plain language, enabling a loan officer to understand and defend the decision. A fraud detection layer must also identify circular transaction topologies — where multiple MSMEs rotate the same UPI funds among themselves to artificially inflate their scores.

EXPECTED OUTCOME

A scoring API that accepts a GSTIN, ingests mocked live signals via simulated GST and UPI data pipelines, runs them through a feature engineering layer, and returns a credit score on a 300–900 scale, a risk band label, the top 5 plain-language reasons for the score, a recommended loan amount with tenure, and a score freshness timestamp.

EXACT DELIVERABLES

- A REST API endpoint accepting a GSTIN and returning the credit score, risk band, top-5 SHAP-driven reasons, recommended loan amount, and a freshness timestamp.
- A data pipeline mocking live ingestion of GST filing velocity, UPI transaction cadence, and e-way bill volume.
- A gradient boosting model with integrated explainability (SHAP values) producing human-readable scoring reasons.
- A fraud detection module that flags circular transaction topologies among linked MSMEs as high-risk.
- A dashboard visualizing feature contributions and score trends over time.

RECOMMENDED TECH APPROACH

Use a gradient boosting or ensemble ML model for scoring. Integrate a model explainability framework to generate human-readable SHAP reasons. Build a feature engineering pipeline for time-series financial signals. Use a REST API framework for the scoring endpoint and a lightweight frontend for the dashboard.

Healthcare & Accessibility

Domain Code: HC

HC01 Agentic Diagnostic Risk Assistant for ICU Complication Detection

THE PROBLEM

In fast-paced ICU environments, patients generate vast amounts of fragmented clinical data daily: continuously shifting lab results, changing vital signs, and unstructured notes written by different specialists across different shifts. Critical deterioration patterns — such as the early onset of sepsis — are often missed not because the data does not exist, but because no single physician has the time or computational capacity to synthesize all historical and real-time signals simultaneously. Shift changes introduce additional risk, as incoming clinicians inherit complex histories without full context. The result is delayed intervention for conditions where hours — sometimes minutes — determine survival outcomes.

CORE TECHNICAL COMPLEXITY

The system requires a multi-agent architecture to process temporal clinical data in parallel. Distinct agents must extract symptom histories from unstructured clinical notes, map shifting lab anomalies (WBC counts, lactate levels, creatinine trends) into a chronological timeline, and cross-reference these patterns against clinical guidelines using a Medical RAG pipeline. A 'Chief Synthesis Agent' must then integrate all agent outputs into a unified risk report — and critically, must detect when a new lab result is an outlier that contradicts three prior days of consistent data, flag it as a probable lab error, and refuse to alter the diagnosis until a confirmed redraw is received.

EXPECTED OUTCOME

An agentic clinical assistant that ingests a patient's complex ICU history — structured vitals, lab results, and unstructured clinical notes — builds a chronological timeline of disease progression, and generates a Diagnostic Risk Report flagging early warning signs of sepsis or organ failure with citations to the specific clinical guidelines that support each flag.

EXACT DELIVERABLES

- A multi-agent pipeline with clearly separated roles: Note Parser Agent, Temporal Lab Mapper Agent, Guideline RAG Agent, and Chief Synthesis Agent.
- A generated disease progression timeline showing how vitals and lab values shifted chronologically.
- A Medical RAG integration retrieving and citing specific clinical guidelines for each flagged risk.

- A Diagnostic Risk Report with explicit safety caveats stating that all outputs are decision-support only, not a clinical diagnosis.
- An outlier detection module in the Chief Agent that flags statistically anomalous lab results as probable errors rather than updating the diagnosis.

RECOMMENDED TECH APPROACH

Use an LLM-based multi-agent orchestration framework for agent coordination. Build a Medical RAG pipeline over a curated clinical guideline corpus using a vector database. Use the MIMIC-III open-access demo dataset for patient data simulation. Implement temporal reasoning logic for chronological lab trend mapping.

HC02 Contact-Free Cardiac Stress Monitoring via Facial Video Analysis (rPPG)

THE PROBLEM

Monitoring heart rate and cardiac stress requires wearable hardware — ECG patches, pulse oximeters, smartwatches — that are unavailable in disaster triage zones, remote telehealth scenarios, and low-income clinical settings. Yet the physiological signal of a heartbeat is already present in any standard smartphone camera: microscopic green-channel intensity fluctuations in facial skin caused by blood pulsing through capillaries. If this signal can be reliably extracted, every smartphone becomes a zero-friction biometric monitor, enabling cardiac triage without any additional hardware. The challenge is that this 'photoplethysmographic' signal is orders of magnitude smaller than the noise introduced by ambient lighting, subject movement, and video compression artifacts.

CORE TECHNICAL COMPLEXITY

Remote Photoplethysmography (rPPG) requires isolating a pulse signal from a video stream by analyzing per-frame green-channel intensity variations in a facial Region of Interest (ROI). The core difficulty is signal separation: teams must implement advanced noise rejection — such as Independent Component Analysis (ICA), the Plane-Orthogonal-to-Skin (POS) algorithm, or a deep rPPG neural network — to filter out ambient lighting flicker, subject head motion, and codec compression noise. The extracted Inter-Beat Interval (IBI) variability must then feed a stress classifier. The system must also include a signal quality indicator: when lighting or motion conditions degrade the signal below a confidence threshold, it must refuse to output a reading rather than silently returning an inaccurate one.

EXPECTED OUTCOME

A system that processes a 30-second selfie video from a standard smartphone camera and outputs: estimated Heart Rate in BPM, Inter-Beat Interval (IBI) variability as an autonomic stress proxy, a binary or graded stress classification, and a signal quality confidence score — refusing output when confidence is too low to be clinically meaningful.

EXACT DELIVERABLES

- A video processing pipeline that detects a facial ROI and extracts per-frame green-channel intensity signals.
- A noise rejection module implementing at least one advanced technique: ICA, POS algorithm, or a pre-trained deep rPPG model.
- A stress classifier trained on IBI variability features (RMSSD, SDNN, or LF/HF ratio).
- A signal quality confidence indicator that flags and suppresses unreliable readings caused by poor lighting or excessive motion.
- A live demo UI processing a 30-second test video and displaying BPM, IBI variability graph, and stress classification.

RECOMMENDED TECH APPROACH

Use a computer vision library for facial detection and ROI tracking. Use a signal processing library for bandpass filtering, FFT, and ICA. Optionally use a pre-trained deep rPPG model for the noise rejection step. Use an open rPPG benchmark dataset for validation (e.g., UBFC-rPPG or PURE).

HC03 Golden-Hour Emergency Triage & Constraint-Based Hospital Routing System

THE PROBLEM

During medical emergencies, ambulances are routinely dispatched to the nearest hospital — only to arrive and discover that the required ventilator is occupied, the relevant specialist is off-duty, or the ICU is at capacity. This reactive routing wastes the most critical minutes of a patient's 'golden hour.' The problem compounds during mass casualty events: a major accident can simultaneously flood a single trauma center while other capable facilities nearby sit underutilized. What is needed is a system that predicts what a patient will need before arrival and routes the ambulance to the nearest hospital that is actually capable of treating them right now.

CORE TECHNICAL COMPLEXITY

The system must combine two hard problems in real-time. First, a severity prediction model must ingest initial EMT-reported vitals and symptom data to predict the patient's likely critical care requirements — ICU bed, ventilator, neurosurgeon, cardiac catheterization lab. Second, a constraint-based optimization engine must simultaneously evaluate a live grid of regional hospitals against multiple competing constraints: predicted care requirements, equipment availability, current hospital load, specialist on-duty status, and estimated transit time — then output the optimal routing decision and explain it. During a mass casualty event, the engine must switch to batch optimization across dozens of simultaneous patients without concentrating all assignments on a single facility.

EXPECTED OUTCOME

A centralized emergency dispatch dashboard that accepts EMT-reported patient data, predicts critical care needs, and routes the ambulance to the optimal hospital in real-time — displaying the routing rationale, estimated arrival time, and dynamically recalculating if a hospital reaches capacity or if road closures change the route mid-journey.

EXACT DELIVERABLES

- A severity prediction model that takes initial vitals and symptom inputs and outputs predicted care requirements (ICU, ventilator, specialist type).
- A constraint-based optimization engine routing to the optimal hospital given equipment availability, load, specialist availability, and transit time simultaneously.
- An interactive map dashboard simulating ambulance routing with real-time hospital status updates.
- An explainability panel stating exactly why a specific hospital was chosen over alternatives.
- A batch-optimization mode that handles simultaneous multi-patient routing during a mass casualty event without overloading a single facility.

RECOMMENDED TECH APPROACH

Use a gradient boosting model for severity and care-needs prediction. Use a constraint-based optimization library for the hospital routing engine. Use a routing or maps API for real-time transit time estimation. Build an interactive map-based dashboard for the dispatch UI.

Education & Skill Development

Domain Code: ED

ED01 AI-Assisted Answer Clustering & Grading Acceleration Engine

THE PROBLEM

Faculty members spend hundreds of hours manually grading subjective exam answers. A professor marking 300 papers reads the same correct explanation 60 times, the same partial-credit answer 40 times, and the same conceptual error 30 times — never realizing the repetition because the papers are graded sequentially in random order. The cognitive fatigue this creates leads to grading inconsistency: the same answer receives different marks depending on whether the professor reads it at 9 AM or 11 PM. What is needed is not an AI that replaces the professor's judgment, but one that organizes the work so the professor grades each distinct answer type once and applies that decision to all matching papers simultaneously.

CORE TECHNICAL COMPLEXITY

The system must process 100 scanned handwritten answer sheets through an OCR pipeline to digitize the text. It must then use sentence embeddings to represent each answer as a semantic vector and apply clustering algorithms (K-Means or DBSCAN) to group answers by meaning — so that 20 students who used the same correct reasoning are grouped together regardless of different phrasing or handwriting. The system must also handle edge cases: identifying answers that contain the correct formula but an arithmetic error, and creating a distinct cluster for these 'half-credit' answers. A further complexity is multilingual input: students answering in a mix of English and a regional language (Hindi/Hinglish) must have their semantic meaning mapped across languages so that a correct Hindi answer clusters with its correct English equivalent.

EXPECTED OUTCOME

A Grader's Dashboard where a professor uploads 100 scanned exam papers and is immediately presented with semantically grouped answer clusters, rubric keyword highlights, and distinct edge-case groupings — enabling them to grade all similar answers in a single review action rather than reading each paper independently.

EXACT DELIVERABLES

- An OCR ingestion pipeline that digitizes handwritten answer sheets from a bulk PDF scan.
- A semantic clustering engine using sentence embeddings and a clustering algorithm to group answers by meaning.
- A rubric keyword highlighting module that marks matching terms in each student's answer to guide the teacher's eye.

- An edge-case cluster for 'formula correct, calculation wrong' answers, detected automatically via text parsing.
- Multilingual clustering support that maps semantically equivalent answers across English and at least one regional language into the same cluster.
- A cost and efficiency log showing LLM token usage and processing time per 100 sheets.

RECOMMENDED TECH APPROACH

Use an OCR library or API for handwritten text digitization. Use a sentence embedding model for semantic vectorization of answers. Apply an unsupervised clustering algorithm (K-Means or DBSCAN) for answer grouping. Use a multilingual embedding model to support cross-language semantic matching.

ED02 Real-Time Procedural Skill Coach Using Hand-Tracking & Spatial Analysis

THE PROBLEM

In vocational and professional training programs — nursing, dentistry, electrical engineering, culinary arts — students must master precise physical, hands-on procedural skills. A single instructor cannot simultaneously monitor 40 students' hand positions to ensure each is holding a scalpel at the correct angle, soldering a joint with the right pressure, or completing a multi-step safety procedure in the correct sequence. Errors go uncorrected until the instructor cycles back, by which point the wrong technique has already been practiced and partially embedded as muscle memory. Students also cannot self-assess: without objective feedback, they cannot distinguish 'my grip felt right' from 'my grip was right.'

CORE TECHNICAL COMPLEXITY

The system must perform real-time 3D hand landmark tracking from a standard smartphone or webcam video feed, extracting (x, y, z) coordinates for all finger joints at every frame. These coordinate sequences must be evaluated continuously against a JSON-defined procedure schema — checking grip angles between specific joints, spatial zone transitions (hand moving from Position A to Position B in the correct order), and minimum dwell time at designated safety checkpoints. Any deviation from the schema must trigger an immediate, specific alert stating exactly what is wrong and what correction is needed. An additional complexity is fatigue detection: by analyzing micro-tremor patterns (coordinate jitter) over extended practice sessions, the system must detect hand fatigue and enforce mandatory rest breaks.

EXPECTED OUTCOME

A real-time procedural coaching system that monitors a student's hands via a smartphone or webcam feed, overlays a live joint skeleton on the video, and provides instant, specific, objective feedback as

they practice — turning a checklist green for each correctly completed step and issuing precise corrective alerts for deviations.

EXACT DELIVERABLES

- A hand tracking pipeline extracting real-time (x, y, z) landmark coordinates from a video feed using a computer vision framework.
- A JSON-based procedure schema encoding a multi-step skill as spatial checkpoints with grip angle tolerances, zone transitions, and dwell time requirements.
- An error detection engine comparing live landmark data to the schema and classifying deviations by type with specific corrective messages.
- A real-time UI showing a live skeleton overlay and a step checklist that updates as each sub-step is completed correctly.
- A fatigue detection module analyzing coordinate jitter over time to detect hand tremor and enforce timed rest breaks.

RECOMMENDED TECH APPROACH

Use a real-time hand landmark tracking framework for (x, y, z) joint coordinate extraction. Use a computer vision library for video feed handling and skeleton overlay rendering. Implement the procedure logic as a state machine evaluated against a JSON schema. Design a web-based or desktop UI with live camera access.

Environment & Sustainability

Domain Code: EN

EN01 Satellite-Based Urban Tree-Equity Auditor & Planting Prioritization Engine

THE PROBLEM

Lower-income urban neighborhoods consistently have significantly less tree canopy coverage — a phenomenon called 'tree inequity.' The compounding effects are measurable: higher surface temperatures due to the urban heat island effect, elevated electricity costs for cooling, worse air quality from vehicle and industrial pollution that trees would otherwise absorb, and worse public health outcomes for residents who are already economically vulnerable. City planners and NGOs have limited tree-planting budgets and want to deploy them where a single tree will have the greatest combined environmental and social impact. Without spatial analysis tools, these decisions are made based on intuition rather than evidence, and the neighborhoods that most need intervention are the last to receive it.

CORE TECHNICAL COMPLEXITY

The system must perform pixel-level vegetation segmentation on multi-band public satellite imagery to compute NDVI (Normalized Difference Vegetation Index) and Land Surface Temperature (LST) at neighborhood granularity. These environmental layers must be fused with geocoded socioeconomic vulnerability data to produce a weighted Tree Planting Impact Score per neighborhood polygon. A further layer of optimization complexity arises when a budget constraint is introduced: the system must apply a Knapsack-style optimization algorithm to select only the highest-ROI intervention zones that fit within a maximum tree count limit. An additional constraint — water availability — must filter high-priority zones that cannot support sapling irrigation, demonstrating multi-constraint spatial optimization.

EXPECTED OUTCOME

An interactive geospatial dashboard that maps urban canopy deserts against thermal hotspots and social vulnerability layers, computes a ranked Tree Planting Impact Score for each neighborhood, and outputs a prioritized intervention list that respects budget and water availability constraints.

EXACT DELIVERABLES

- A satellite image processing pipeline that computes NDVI and LST from public multi-band imagery (e.g., Sentinel-2 via Google Earth Engine).
- A pixel-level land cover segmentation model distinguishing tree canopy, built surfaces, and bare ground.
- A data fusion layer combining NDVI, LST, and at least one socioeconomic dataset into a weighted Tree Planting Impact Score per neighborhood polygon.

- A budget optimization module using a Knapsack algorithm to select maximum-ROI zones within a hard tree count limit.
- A water constraint filter that removes high-priority zones lacking access to municipal water infrastructure for sapling irrigation.
- An interactive geospatial dashboard visualizing the canopy desert heatmap, prioritization scores, and final recommended intervention zones.

RECOMMENDED TECH APPROACH

Use a satellite data platform (e.g., Google Earth Engine) for multi-band imagery access and NDVI/LST computation. Use a deep learning segmentation model for land cover classification. Use geospatial libraries for polygon-level data fusion and mapping. Use a combinatorial optimization library for the budget-constrained planting selection.

EN02 Hyperspectral Methane Super-Emitter Detection, Quantification & Source Attribution

THE PROBLEM

Methane super-emitters — leaks exceeding 100 kg/hour from oil and gas infrastructure — account for 20–50% of the sector's total methane emissions despite representing fewer than 1% of all emission sources. Current satellite monitoring pipelines struggle on three fronts: excessive false positives caused by cloud and surface reflectance artifacts that mimic methane plume signatures, slow batch processing that delays regulatory response by days, and weak source attribution that can detect a plume but cannot reliably match it to a specific facility or pipeline segment. Without precise attribution, regulatory enforcement is impossible. Meanwhile, methane is 80 times more potent as a greenhouse gas than CO2 over a 20-year horizon, making super-emitter detection one of the highest-leverage interventions in near-term climate action.

CORE TECHNICAL COMPLEXITY

The system must execute a three-stage AI pipeline on hyperspectral satellite data (Sentinel-5P/TROPOMI or NASA EMIT). Stage one: a Vision Transformer or U-Net++ performs pixel-level plume segmentation with attention-based artifact removal to suppress cloud and surface reflectance false positives. Stage two: a physics-informed neural network (PINN) estimates emission flux rates from plume morphology combined with ECMWF wind vector data, incorporating atmospheric radiative transfer principles to improve quantification accuracy. Stage three: a Graph Neural Network matches each detected plume back to a specific facility polygon from infrastructure maps, attributing the source at facility level. When heavy cloud cover occludes part of the image, the system must use spatial interpolation or historical decay models to estimate the obscured plume extent.

EXPECTED OUTCOME

A real-time super-emitter monitoring system featuring a global heatmap of detected methane plumes, facility-level risk scores, confidence-bounded anomaly alerts, and an end-to-end pipeline containerized for reproducibility — with a public API endpoint accepting a geographic bounding box and returning detected plumes with flux estimates and source attribution.

EXACT DELIVERABLES

- A plume segmentation model (Vision Transformer or U-Net++) with artifact removal, targeting F1 > 0.85 on a validation set.
- A physics-informed flux quantification module estimating emission rates (kg/hr) from plume morphology and wind vectors.
- A source attribution module matching plumes to facility polygons from infrastructure maps, targeting > 80% attribution accuracy at facility level.
- A live dashboard showing a global super-emitter heatmap, facility risk scores, and confidence-bounded anomaly alerts.
- A containerized pipeline (Docker) with a REST API endpoint accepting a bounding box and returning plume detections with flux estimates and attribution.
- A reproducible Jupyter notebook documenting the end-to-end pipeline with uncertainty quantification.

RECOMMENDED TECH APPROACH

Use a satellite data platform for hyperspectral imagery access. Use a Vision Transformer or U-Net architecture for plume segmentation. Use a physics-informed neural network framework for flux estimation. Use a graph neural network library for source attribution. Containerize the full pipeline with Docker and expose it via a REST API.

Smart Cities

Domain Code: SC

SC01 Predictive Last-Mile Transit Synchronizer with Dynamic Fleet Positioning

THE PROBLEM

The 'last mile' gap — the distance between a metro station or bus stop and a commuter's actual destination — is the primary reason most urban Indians default to private vehicles despite the availability of public transit. Shared last-mile options (e-bikes, autos, cabs) are present at stations but unpredictably located and uncoordinated with arriving trains. When a train pulls in with 500 passengers, there are 3 autos waiting. When no train is expected for 20 minutes, 8 vehicles sit idle. The fundamental problem is that last-mile supply is reactive: vehicles respond to passengers they can already see, instead of positioning themselves for passengers who are still 4 minutes away on an incoming train.

CORE TECHNICAL COMPLEXITY

The system must solve a predictive multi-sided matching problem. It must ingest live GTFS transit data for incoming trains and buses and combine it with historical ridership patterns, time of day, and weather signals to forecast commuter exit volumes at each station exit point 5–15 minutes in advance. A constraint-based fleet optimization engine must then pre-position a simulated fleet of e-bikes, autos, and cabs to predicted demand hotspots before passengers arrive — not after. A dynamic matching algorithm must then assign arriving commuters to the nearest available vehicle in real-time. A critical operational complication must be handled: when an unexpected train delay causes a 45-minute gap followed by two simultaneous trains arriving with double-capacity crowds, the system must switch to a surge rebalancing mode without depleting vehicles from surrounding stations.

EXPECTED OUTCOME

An AI platform that synchronizes last-mile vehicle dispatch with public transit schedules — pre-positioning vehicles at predicted demand points before commuters exit — and provides both a commuter-facing matching interface and a city operations dashboard showing fleet status, demand forecasts, and matching efficiency metrics.

EXACT DELIVERABLES

- A demand prediction model forecasting passenger exit volumes per station node 5–15 minutes ahead, using live GTFS feed data and historical patterns.
- A fleet positioning optimization engine that pre-deploys vehicles to predicted demand hotspots before passenger arrival.

- A dynamic real-time matching algorithm assigning arriving commuters to the nearest available vehicle.
- A commuter-facing UI showing available vehicles, estimated wait time, and one-tap booking.
- A city operations dashboard with real-time fleet status, demand heatmaps, and matching efficiency metrics.
- A surge rebalancing mode that handles simultaneous high-volume arrivals without over-concentrating the fleet at one station.

RECOMMENDED TECH APPROACH

Use a GTFS feed parser for live transit data ingestion. Use a time-series ML model for demand forecasting. Use a constraint-based optimization library for fleet positioning. Build a real-time backend with WebSocket support for live updates to the commuter and operations UIs.

SC02 Drone Image-Based 3D Infrastructure Defect Detection & Severity Scoring

THE PROBLEM

Manual inspection of aging infrastructure — bridges, dams, flyovers, retaining walls — requires inspectors to access dangerous heights and confined spaces, making it slow, expensive, and hazardous. Using drones to capture photographs is safer and faster, but conventional 2D image analysis introduces a critical blind spot: depth. In a flat 2D photograph, a harmless surface water stain and a 15mm-deep structural fracture appear visually similar. An inspector viewing 2D images alone cannot reliably distinguish a cosmetic defect from a structurally critical one without expensive on-site verification. India has over 1.7 lakh bridges, many well past their design lifespan, and collapses like the 2019 Savitri Bridge incident demonstrate the life-threatening cost of missed defects.

CORE TECHNICAL COMPLEXITY

Rather than attempting full-scale 3D photogrammetric reconstruction (which requires specialized drones and massive compute), the system must derive pseudo-3D structural insight from standard 2D drone photographs. First, an instance segmentation model must precisely outline each detected defect — crack, rebar exposure, concrete spalling — at pixel level. Second, a monocular depth estimation model must infer a relative depth map from the same image. By fusing the 2D defect boundary with its AI-inferred depth profile, the system calculates a Structural Severity Score (1–5). A further complication arises from drone flight overlap: when multiple photos of the same crack are taken from different angles during a bridge pass, the system must stitch these overlapping images into a single orthomosaic before defect analysis to avoid counting the same defect multiple times.

EXPECTED OUTCOME

An automated inspection pipeline where an engineer uploads a sequence of standard 2D drone images and receives: an annotated output with each defect precisely outlined, an estimated depth profile for each detected defect, a Structural Severity Score per defect, and a consolidated inspection report recommending 'monitor,' 'repair,' or 'close for immediate assessment.'

EXACT DELIVERABLES

- An instance segmentation model detecting and outlining concrete cracks, spalling, and rebar exposure at pixel level.
- A monocular depth estimation module inferring relative depth maps from the same 2D input images.
- A severity scoring algorithm fusing segmented defect boundaries with depth map values to output a Structural Severity Score (1–5) per defect.
- An image stitching module using feature matching to merge overlapping drone images into a single orthomosaic before defect detection, preventing double-counting.
- A temporal crack growth analysis mode that aligns a historical image with a current image of the same location and calculates the percentage growth of a specific defect over time.
- An interactive inspection dashboard for image upload, annotated result viewing, and PDF report export.

RECOMMENDED TECH APPROACH

Use a pre-trained instance segmentation model (fine-tuned on infrastructure defect data) for crack and spalling detection. Use a monocular depth estimation model from a public model hub for depth map inference. Use a feature matching algorithm (ORB or SIFT) for image stitching and orthomosaic generation. Build a simple web-based dashboard for upload, visualization, and report export.