Elaboration - Risk Removal Plan - Background Research, Prototyping, Skill Building Proposal

Table of Contents:

1. **Purpose & Scope (Inception Report Recap)**

2. **Background / Related Work**
   2.1 Method (How We Searched)
   2.2 Project-wise Background Scan

3. **Research Questions & Proposed Methodology**

4. **Proposed Workproducts**
   4.1 Requirements & Analysis
   4.2 Design Support

5. **References**

1) Purpose & scope (Inception report recap)

For our project, as we discussed in the inception, we set a common plan to study where AI can genuinely help (or not) our engineering work across three active efforts—Animal Ecology, Adaptive Optics–OCT, and Digital Agriculture. We also defined our research objective to conduct a comprehensive analysis of AI use cases across the software development lifecycle—requirements analysis, design support, implementation, and project management—using both functional and non-functional measures and grounded in each project as a use case. Finally, we discussed the risks (tool reliability, privacy, over-automation, cost, and reproducibility) associated with our intended research. In short, the inception report gives us a shared yardstick and real scenarios to test and compare against.

2) Background / related work

2.1 Method (how we searched)

To identify relevant prior work, we adopted a hybrid search strategy that combined I-assisted literature discovery with manual screening, such as Google Scholar, Science Direct, and IEEE along with AI tools such as ChatGPT deep research, we queried keywords tailored to

each domain — for example, “AI-assisted software engineering in wildlife monitoring,” “detection of retinal microstructures using AO-OCT images,” and “edge-to-cloud orchestration in digital agriculture.”

We included papers and technical reports from the past decade (focusing more on the last 10 years for better relevance) that explicitly described software development practices, AI integration within engineering workflows, or documented performance/efficiency gains attributable to AI tools. Each candidate's work was reviewed for its domain relevance, clarity of AI involvement, and potential to inform our objectives of evaluating where and how AI contributes across the software development lifecycle.

2.2 Project-wise background scan:

2.2.1 Animal Ecology (Edge-cloud continuum):

We researched across the most suitable 10 papers from the last 10 years, and the following is a short summary of related work in the conservation-tech and edge–cloud continuum domain. Across this literature, a clear pattern emerges: small field devices (e.g., smart camera traps and acoustic sensors) run lightweight AI to filter out “empty” data and raise alerts only when something meaningful happens. Those alerts and selected media are buffered on the device and sent over whatever connectivity is available (cellular or satellite) to simple cloud dashboards that teams already use. In some cases, detections trigger targeted follow-ups like dispatching a drone to capture brief, higher-quality video, so rare events are recorded with minimal disturbance and manual effort.

Across these lines of work, AI is embedded inside system components like detectors on traps, acoustic classifiers, onboard drone vision, and multi-network backhauls but we see little to no emphasis on AI tools that assist the software development lifecycle itself. In other words, the literature focuses on AI for sensing and alerts, not on AI as a co-engineer that helps with requirements extraction, implementation support, maintenance, documentation, or project management

Our project aims to address these gaps by discussing the AI-assisted SDLC to the same event-driven manner from edge to near-edge and then cloud/HPC loop. Practically, that means turning research ideas into clear, testable engineering requirements; generating lightweight documentation and design artifacts. Using AI tools alongside research scientists to implement the components and then applying AI to maintain and evolve the application. The result is a smoother path from concept to production-ready systems in wildlife monitoring with less friction and greater efficiency.

2.2.2 Adaptive Optics–OCT (Biomedical Imaging and Disease Progression):

We examined ~10–15 influential papers from the last decade (with emphasis on the most recent 5 years) covering AO-OCT imaging and retinal disease analysis. A consistent trend emerges: pipelines have shifted from classical thresholding/morphology for the IS/OS (ISOS)

junction toward deep models (U-Net variants and, more recently, transformer backbones like SegFormer/Swin-UNet) that can delineate fine microstructures and support cone-level quantification. Most studies optimize for segmentation quality (e.g., Dice/F1) and clinical interpretability (retinal layer integrity, cone density maps).

What's largely missing is attention to how these systems are engineered and maintained at scale. The literature rarely addresses reproducible experiment setup, automated configuration management, traceable multi-experiment orchestration on HPC, or AI tools that assist the SDLC itself (requirements drafting, design artifacts, test scaffolding, docs). Reproducibility is often acknowledged but not operationalized.

Our AO-OCT project addresses these gaps by embedding AI-assisted SDLC into the biomedical workflow: we use AI to translate research ideas into concrete requirements; auto-generate training/evaluation configs for ISOS-Net (healthy vs. AMD folds); and orchestrate multi-run jobs on HPC with standardized logs, metrics, and report generation. Outcome: a reproducible, end-to-end framework for cone-level disease progression analysis with lower friction in maintenance and collaboration.

2.2.3 Digital Agriculture:

We focus on automating aerial data collection through two key modules: (1) a rectangular field path planner that generates drone-compatible waypoint lists (consumed by the OpenPass service), and (2) a transfer and orchestration module that moves captured media from edge devices to HPC systems and triggers downstream machine learning tasks like training or inference. Together, these components enable a reliable field-to-cloud pipeline for AI-ready datasets with minimal manual intervention.

Prior work in this space typically uses deterministic Coverage Path Planning (CPP) — especially lawnmower-style strips — for surveying rectangular areas, due to its simplicity and efficiency. Photogrammetry best practices translate flight height and overlap requirements into lane spacing and image intervals. For data transfer, tools like Tapis Workflows and Globus Flows are widely used to implement structured "capture → transfer → validate → compute" pipelines with retries, checksums, and conditional branching. While AI is central to the inference and training stages once the data is uploaded, it is not commonly used in the planning or transfer logic itself.

This creates space for innovation: Planners can emit "mission manifests" that support pre-HPC validation (e.g., checking image coverage via EXIF before scheduling jobs), and transfer modules can be upgraded to handle unreliable field networks with batching, retrying uploads, and smart triggers. Over time, logs from these systems could even inform learned scheduling or batching strategies, blending automation with domain-specific controls. A custom wrapper can be built for integration with OpenPASS that takes the mission requirements, such as height of flight, coordinates, percentage of overlap, etc.

3) Research questions & proposed methodology (Addressing all three projects)

- **RQ1: Where does AI help and where doesn't in our software development lifecycle?**
  We'll test how AI performs across different SDLC stages: from requirements and design to coding, testing, and deployment. For each stage, we'll pick small real-world tasks (e.g., writing user stories, drafting APIs) and perform them twice — once with AI assistance and once without. By measuring time taken, bugs found later, and overall usability, we'll assess whether AI speeds things up, improves quality, or introduces friction.
- **RQ2: How do different AI tools or setups compare in practice?**
  To evaluate productivity and quality, we'll test a few popular AI tools or configurations on the same set of tasks. Inputs will be kept identical (e.g., same prompt, same code base). We'll measure time to completion, test/lint outcomes, consistency of results (reproducibility), and cost. This helps identify which tool offers the best trade-off for our needs.
- **RQ3: What's the right balance between automation and human control?**
  An experiment can be with three modes for each task: *Suggest* (AI drafts, human edits), *Approve* (human confirms AI action), and *Auto* (AI runs automatically if simple checks pass). We'll begin conservatively and shift toward automation only if outcomes remain safe. Key metrics will include override frequency, bug escapes, incident handling, and overall speed.

A short, evidence-based take: AI can boost parts of the SDLC, especially coding—e.g., a controlled trial found developers using GitHub Copilot finished a task **~55.8% faster** than a control group Microsoft+1. But broad reviews show the effect varies by stage and task: recent systematic surveys catalog wins and limits for LLMs across requirements, design, coding, testing, ops, and docs (benefits are real but not universal) arXiv+1. In testing, empirical studies report LLM-generated unit tests can reach good coverage, yet quality depends heavily on prompts and setup, underscoring the need for careful evaluation arXiv. Reproducibility and consistency also matter—prompt sensitivity can shift outcomes, so pinning prompts/models/seeds is recommended ACL Anthology. For control vs. automation, human-in-the-loop frameworks remain the prudent default in production workflows, adding automation where guardrails and validations exist

4) Proposed workproducts

4.1 Requirements & analysis

We'll focus on two things that's important aspects that would be easier to compare across AI tools:

1. Functional scenarios -> user stories with acceptance criteria:
   We will evaluate how well (or not) AI tools translate our project brief and research

idea into a set of clear user stories, each capturing the persona, goal, and value and 2–4 testable Given–When–Then acceptance criteria per story.
2. Non-functional requirements (NFRs) with measurable targets:
   The tool should propose concrete quality attributes—performance, reliability, security/privacy with numbers and a way to verify them (e.g., job latency < 5 min; device storage < 200 MB/day).

Human-in-the-loop role:
We'll use AI to draft; humans will (a) ground details in our domain, (b) calibrate numbers to hardware/network reality, and (c) resolve conflicts—keeping the edits minimal but decisive.

Workproduct:
A lightweight Requirements Analysis document containing the stories+Acceptance Criteria table, an NFRs-with-metrics table.c

Prompts we provided to AI tools- User stories with Acceptance Criteria

You are a senior requirements engineer. Using ONLY the PROJECT CONTEXT above:

Task: Produce 8–12 user stories (persona + goal + value), each with 2–4 testable Given–When–Then Acceptance Criteria.

Rules:

- Make ACs objective and measurable (numbers/units/percentiles). No vague words like "fast/secure."
- Include at least one edge/exception flow across the set (e.g., connectivity loss, token expiry).
- If you must assume something not stated, list it under an Assumptions section, don't bury it in stories.
- Don't invent unrelated features.

Output format:

1. A Markdown table with columns: `ID | Persona | User Story | Acceptance Criteria (G-W-T, numbered)`.
2. A short Assumptions list (bulleted).
3. A Glossary (5–10 key terms, one-line each).

Prompt we provided to AI tools - Non-Functional Requirements

Using the same PROJECT CONTEXT, draft Non-Functional Requirements that are measurable and verifiable.

Include categories: Performance/Latency, Reliability/Availability, Security/Privacy, project specific constraints, Scalability, Observability/Logging, Cost (operational), Maintainability.

Rules:

- Every NFR must specify a numeric target (units, percentile if applicable) AND a verification method (how to measure, where instrumented).
- Keep targets realistic for the context; if uncertain, mark `TBD` with a proposed range and a validation experiment.

Output format: Markdown table with columns:
`NFR-ID | Category | Statement (with metric) | Verification Method | Where Measured | Owner/Role | Rationale`

4.1.1 Animal Ecology:

Project context:

Goal: Camera traps detect animal intrusions, score the captured images with on-device/object-detection models, and trigger a pipeline that (a) plans drone flights over polygon Areas of Interest (AOIs), (b) collects additional media, and (c) auto-transfers data to the Ohio Supercomputer Center (OSC) via Tapis to trigger training/inference jobs.

Aim: Reduce manual human intervention across detection, data collection, transfer, and job orchestration.

Constraints: Edge devices like Raspberry Pi camera traps and Parrot Anafi drones come with limited storage, network bandwidth, power, and compute.

Link: Requirement analysis_Animal Ecology

4.1.2 AO OCT

Project context

Goal: Build an automated, reproducible AO-OCT pipeline that segments the IS/OS junction and quantifies cone-level biomarkers to study AMD progression.

Scope: Data prep → training (healthy/AMD) → evaluation → visualization → progression summaries; AI assists SDLC (requirements, configs, SLURM, docs).

Aim: Reduce manual friction in experiment setup and documentation; ensure traceable, repeatable results across HPC runs.

Constraints: Large AO-OCT volumes; limited labeled AMD scans; GPU quotas/scheduler; strict reproducibility; privacy constraints.

Link: Requirement Analysis AO OCT

4.1.3 Digital Agriculture

Project context:

Goal:
Automate the aerial data collection process by (a) generating rectangular waypoint plans for Parrot Anafi drones based on user-defined plots, and (b) efficiently transferring collected images from edge devices to HPC systems (e.g., OSC) for downstream processing such as ML training or inference.

Scope:

- The Path Planner Module takes in rectangular field specifications and camera/flight parameters to output optimized waypoint sets and mission manifests.
- The Transfer Module monitors a drone mission folder in real-time, batches newly captured images, and triggers asynchronous uploads to HPC clusters using Tapis (or equivalent). Upon upload, it initiates relevant jobs (e.g., training, inference).

Aim:
Minimize manual involvement during flight planning and data transfer; ensure robustness in low-connectivity field settings; enforce correctness checks before triggering costly HPC jobs.

Constraints:

- Drone limitations: No internal AI processing; mission control is handled by OpenPass.
- Edge environment: Resource-constrained Linux device (e.g., laptop or NUC); intermittent field connectivity (Wi-Fi/LTE); Python-based stack.
- Networking: Transfers must support unreliable links, retries, and partial recovery.
- Storage: Limited space on edge for batching; jobs must avoid recomputation via deduplication and quality checks.

Link: **Digital_Agriculture_Requirements**

Observations on AI-Generated PRD Content :

Overview:

Across these projects, **ChatGPT**, **Perplexity**, **Gemini**, and **WriteMyPRD** reliably produced clear, well-structured PRD artifacts—user stories, G-W-T acceptance criteria, non-functional requirements (NFRs), assumptions, and glossaries. Most drafts were immediately usable as a first pass and required only light editing to harmonize terms and tighten numeric targets.

What worked well:

- **ChatGPT:** Best at sticking to a clear template and longer drafts. Gives clean user stories and, if asked to “use numbers,” it adds measurable ACs and NFRs across performance, reliability, and security.
- **Perplexity:** Great for quick fact-checks and definitions. Keeps things concise and can add citations or short reasons for chosen metrics.
- **Gemini:** Good for brainstorming and alternative phrasings. Starts broad, but with “no vague words” it gives more numeric, specific targets.
- **WriteMyPRD:** Delivers PRD-style sections (Problem, Goals, KPIs, Risks, Out-of-Scope) ready for stakeholders. Sometimes uses generic KPIs, so we plug in domain-specific numbers.

Where drafts needed refinement (common across tools):

- **Depth variability:** Numeric strictness varied run-to-run (e.g., “low latency” vs. “≤ 500 ms p95”). Enforcing a “numbers-only” rule in the prompt reduced this.
- **Terminology drift:** Terms like *event*, *batch*, *job*, *artifact* were used inconsistently across drafts/projects; a **shared glossary** fixed this quickly.
- **Traceability:** Links from **story -> AC -> NFR -> verification method** were often implicit.

Although the content generated solely by these tools isn’t shippable yet, it’s a strong starting point for development, and with iterations it has shown promising improvements that could make it nearly shippable.

4.2 Design support

Design support includes but is not restricted to turning requirements into a concrete system blueprint—architecture, component boundaries, interfaces, and data/control flows. It guides trade-off decisions (performance, reliability, security) and gives teams perspective through diagrams they can design, build, and operate against.

While design support is a vast topic, for this project we have limited it to two design artefacts: a high-level system architecture diagram and an end-to-end flow. These are also easy to compare across AI tools.

System architecture diagram
Evaluate how well AI tools turn a short brief into a clear, architecture view that shows major components, external systems, boundaries (client/edge/backend/cloud/HPC), and labeled interfaces (protocol/API and data direction). The diagram should be compact, consistent, and understandable at a glance.

End-to-end flow chart
Evaluate how well AI tools depict the key steps, decisions, and handoffs in the system—from event/input through processing to outputs—including at least one error/offline branch and points where logs/telemetry are emitted.

Human-in-the-loop role
Humans will (a) confirm partitioning and naming, (b) fix mislabeled or missing interfaces, and (c) ensure error paths reflect operational reality—keeping edits minimal but decisive.

Workproduct

- One **system architecture diagram**.
- One **end-to-end flow chart**.

4.2.1 Animal Ecology:

We have used four different AI tools such as ChatGPT 5 models, Gemini pro models, Miro (link) and Lucid (link) for the diagram creation, all the diagrams were created in the same project context that was used in the requirement analysis.

Link: DesignSupport_AnimalEcology.ipynb

The above link provides the work products in the form of system architecture and flow chart diagrams developed by the AI tools along with the manually created diagram.

4.2.1 AOOCT:

The link below contains flow diagrams

Untitled

4.2.2 Digital Agriculture:

The link below contains details on the subsystems, key prototypes, supporting architectural flow diagrams.

Link: System_Details_and_Diagrams

Observations on AI-Generated Diagrams:

While AI tools were helpful at turning text requirements into rough blueprints and clear explanations of components and interfaces. Diagrams generated through AI tools were not very helpful. AI tools tend to over-complicate and/or miss understanding the context. That being said, to a certain degree ChatGPT + Figma AI, Miro, Gemini and Lucid AI could generate basic block/flow diagrams from prompts, which sped up first drafts. However, even they struggled with accurate, detailed visuals, for example, getting interface contracts, data paths, and constraints (edge power/bandwidth, OSC/Tapis steps) exactly right and laying them out clearly. As a result, diagrams typically needed manual edits to fix node naming, connections, legend/keys, and non-functional constraints.

References:

- Edge Machine Learning for Wildlife Conservation: A part of the Ngulia project. (n.d.). *DiVA Portal.* https://www.diva-portal.org/smash/record.jsf?pid=diva2:1788498
- AI Camera Trap – Hack The Planet. (n.d.). *Hack The Planet.* https://www.hack-the-planet.io/project/ai-camera-trap
- Reliable and efficient integration of AI into camera traps for smart wildlife monitoring based on continual learning. (n.d.). *DOAJ.* https://doaj.org/article/17306a45fcb54bf18a8c2520b7afca0e
- FASTCAT-Edge – Cos4Cloud. (n.d.). *Cos4Cloud.* https://cos4cloud-eosc.eu/services/fastcat-edge-camera-trap/
- Faster Response, Smarter Decisions: How Sentinel and EarthRanger Are Empowering Conservationists to Save Wildlife. (n.d.). *EarthRanger.* https://www.earthranger.com/news/faster-response-smarter-decisions-how-sentinel-and-earthranger-are-empowering-conservationists-to-save-wildlife
- Otsuka, R. (2024). *Logbot-playback-2024* [Computer software]. GitHub. https://github.com/ryoma-otsuka/logbot-playback-2024
- How a new drone system may transform next-gen ecology research. (2025, April). *College of Engineering, The Ohio State University.* https://engineering.osu.edu/news/2025/04/how-new-drone-system-may-transform-next-gen-ecology-research
- Trailguard AI. (n.d.). *Global Conservation.* https://globalconservation.org/trailguard-ai
- Guardian Platform. (n.d.). *Rainforest Connection.* https://rfcx.org/guardian
- Saving Rainforests by Listening. (n.d.). *Huawei Tech4All.* https://www.huawei.com/en/tech4all/stories/rainforest
- arXiv.org. (2024). *Preprint 2412.01000.* https://arxiv.org/pdf/2412.01000
- Cabreira et al., Survey on Coverage Path Planning with UAVs (Drones, 2019): https://www.mdpi.com/2504-446X/3/1/4
- Elhadary et al., Influence of Flight Height and Overlap on UAV Imagery (IJA AA, 2022): https://mej.researchcommons.org/home/vol47/iss2/15/
- Freeman et al., Workflow management with Tapis Workflows (PEARC'22): https://dl.acm.org/doi/10.1145/3491418.3535142
- Chard et al., Globus Automation Services (FGCS, 2023): https://www.sciencedirect.com/science/article/pii/S0167739X23000183
- Vescovi et al., Linking scientific instruments and computation: Patterns (Patterns, 2022): https://www.sciencedirect.com/science/article/pii/S2666389922002318