Problem Statement and Goals
SFWRENG 4G06 - Capstone Design Process

Team 17, DomainX

Awurama Nyarko
Haniye Hamidizadeh
Fei Xie
Ghena Hatoum

September 20, 2025

Table 1: Revision History

Date	Developer(s)	Change
September 20, 2025	Awurama Nyarko, Fei Xie Ghena Hatoum Haniye Hamidizadeh	Created first draft of document

1 Problem Statement

1.1 Problem

Open-source neural network (NN) libraries, including training frameworks (e.g., PyTorch, TensorFlow), inference frameworks (e.g., ONNX Runtime), and companion tooling (e.g., data loaders, visualization utilities), are central to academic research and production AI systems. Despite their importance, there is currently no transparent, reproducible, and domain-tailored way to assess the state of software-development practice across these libraries.

Existing assessments are typically ad hoc: spreadsheet-based scoring and manual pairwise comparisons (e.g., AHP) that are difficult to trace, with evidence scattered across READMEs, documentation, tests, and CI pipelines. This creates challenges for:

1. Users (researchers and engineers), who cannot easily compare libraries on qualities such as installability, maintainability, understandability, usability, transparency, robustness, reliability, and reproducibility.
2. Maintainers, who lack cross-project feedback on practice gaps and improvement opportunities.
3. Researchers, who lack a consistent methodology and artifact trail suitable for replication or meta-analysis.

This project addresses these issues by adapting and applying a rigorous, repeatable state-of-practice methodology to the NN-library domain, and replacing fragile spreadsheets with a more traceable, auditable toolset that streamlines evidence collection, scoring, and ranking. The outcome will be a defensible comparison across software-engineering qualities and actionable recommendations for practitioners and maintainers.

1.2 Inputs and Outputs

Inputs

- Curated set of open-source NN libraries that meet inclusion criteria (public repositories with source, installation docs, runnable examples/tutorials).
- Public artifacts: source code, documentation, tutorials, tests, CI configs, issue trackers, release notes.
- Repository measures: commit/issue activity, language breakdown, lines-of-code counts, collected via automated scripts.
- Measurement template responses: structured question bank per quality, completed via automation and human inspection.
- Decision-model configuration: weights over qualities (installability, maintainability, etc.) using a method such as AHP for reproducible rankings.
- Optional practitioner/maintainer interviews, if time permits.

Outputs

- Traceable dataset: structured table recording evidence and normalized scores per software quality.
- Rankings and sensitivity analyses: transparent results with full evidence trail, plus stability testing under weight changes.
- Domain insights: summarized patterns, practice gaps, correlations across projects.
- Recommendations: prioritized, evidence-backed suggestions for maintainers and users.

- Lightweight toolset: simple replacement for spreadsheets to collect evidence, apply scoring, and export results (CSV/PDF).
- Research deliverables: (i) an academic-style report documenting methods, dataset, findings; (ii) an open data/software package (versioned with changelog) for reproducibility and reuse.

1.3 Stakeholders

- Supervisors and domain experts (primary): define scope, validate inclusion criteria, review methodology, and vet results.
- NN library maintainers and contributors (primary): benefit from diagnostics and recommendations; possible interview participants.
- Practitioners (researchers, engineers, data scientists) (primary): need trustworthy comparisons to select or advocate for libraries.
- Software engineering and RSE research community (secondary): gains reusable methodology, dataset, and toolset.
- Course instruction team (secondary): evaluates rigor, scope fit, and completeness of methodology and reporting.

1.4 Environment

1.4.1 Hardware & Execution Environment

- Storage: A SSD for fast data access
- CPU: Multi-core processor to handle user requests
- Web Server: Used to store, process, deliver content
- Operating System: Linux to follow industry standards for web pages

1.4.2 Software & Tooling

- Frontend: HTML, CSS, JavaScript, and React, to build user interface
- Backend: Python and Django, to handle back-end logic and data processing
- Database: MySQL to store and manage data
- APIs: GitHub API, to automatically fill in data
- Libraries: Data visualization and AHP libraries
- Version Control: Git to manage the project codebase

Assumptions

- Target libraries expose public repositories with source code.
- Installation documentation is available.
- Software qualities can be evaluated from surface artifacts within the project timeline.
- Stakeholders are available for periodic reviews.

Constraints

- Fixed academic timeline.
- Limited GPU access.
- Private or internal artifacts excluded.
- Interviews optional and subject to limited availability.
- Rapid upstream changes mitigated via timestamped data collection.

Out of Scope

- Benchmarking model accuracy.
- Benchmarking runtime or throughput performance.
- Domain-specific fitness evaluations beyond software-engineering qualities.

2 Goals

2.1 Research

At the end of our research we will have developed a research paper evaluate and understand the state of development practices within the NN libraries. Through this research paper we are aiming answer the questions mentioned in the Methodology for Assessing the State of the Practice for Domain X paper, which are as follows:

- Development and Technical Stack: What are the common tools and methodologies used to build and manage these software packages?
- Developer Experience: What are the main challenges that developer face and how could they be resolved to improve development process and software quality?
- Quality and Best Practices: What actions are developers taking to ensure key software quality (e.g. usability, maintainability, and reproducibility)?
- Quality Comparison: How does a library's technical quality, as designed by this research methodology, compare to it's reputation within the broader developer community?

2.2 Tool

To enhance the efficiency of this research, we will develop a tool with several key features to minimize manual effort and improve data management.

- Interactive Data Table: This feature will provide a visually appealing and intuitive table, allowing researchers to directly edit and manage the database with a user-friendly interface.
- Automated Data Collection: We will automate the collection of specific data points using APIs to reduce manual work and the chance of errors. While some metrics like installability and sustainability will still require manual expert ranking, the tool will automatically gather data such as number of commits and issue counts.
- Automate Analytical Hierarchy Process: The tool will implement an automated AHP on the tool to compare libraries, guiding users through a series of pairwise comparisons to weigh criteria and evaluate libraries against each other.
- Data Visualization and Download: The tool will enable researchers to view and filter results as graphs. Users will be able to download these graphs in various formats, including PNG files for presentations and LaTeX code for direct use in papers. It will also allow researchers to download the entire database for any scope as either an Excel or JSON file, providing greater flexibility and control over the data.
- Collaboration: This methodology requires close collaboration between software and domain experts. The tool will facilitate this by allowing domain experts to vet software lists and provide feedback on the results. To support simultaneous data entry, the user interface will be designed to allow up to two people to update the database concurrently.

3 Stretch Goals

The following are possible features that can be added to the tool however are not the focus.

- Traceability: Currently, the use of Excel sheets makes it difficult to track who made what changes and when. Our tool will include a feature that logs all updates, creating a clear history of changes and improving the accountability of the database.
- Research Paper Generation: This feature will allow researchers to write up their findings directly on the tool. It will automatically compile important graphs and tables and format the final document in a user's preferred style, eliminating the need for manual formatting or direct use of LaTeX.

- Increase Library Features: While the current methodology focuses on evaluating development practices, we will expand our data collection to include information on a library's specific features and capabilities. This will add a new level of functionality, ensuring the tool can recommend a library that not only follows best practices but also aligns with a user's specific needs.
- Interactive Dashboard: This dashboard will provide a centralized, real-time overview of the project's progress. It will visually highlight key milestones, identify the next steps, and clearly show any missing data or required feedback from domain experts.

4 Extras

Following a discussion with our supervisor, Dr. Smith, it confirmed that a research paper will serve as our extra and that there will be no need to have another. For more information on the paper's scope and purpose, please refer to Section 2.1 Research.

- An academic-style report documenting the methodology, dataset, and findings.

Appendix — Reflection

1. What went well while writing this deliverable?
   Awurama: I found that structuring the problem into the four subsections (Problem, Inputs/Outputs, Stakeholders, Environment) went smoothly. Having my draft already written in a google doc made it much easier to adapt to LaTeX.
   Ghena: I was able to shift my perspective on the problem after having a meeting with our supervisor, which lead to a more accurate understanding of the PS and goals.
   Fei: While writing the deliverable, we were able to as a team have a deeper understanding of what this project is about, and solidify what our own expectations were for this project, through the goals determined and problem statement.

2. What pain points did you experience during this deliverable, and how did you resolve them?
   Awurama: My main challenge was getting familiar with the LaTeX template and re-adjusting to the GitHub workflow after not using it for a while. I overcame this by researching LaTeX best practices and following the problem statement writing checklist to make sure I stayed on track.
   Ghena: Initially, my understanding of the problem statement was centered on the operational challenges faced by researchers, rather than the foundational importance of the research itself. A clarifying discussion with Dr. Smith provided the crucial insight needed to reframe my thinking, enabling me to grasp the project's broader significance. This adjustment allowed me to understand the problem statement and goals that was much more aligned with the supervisor's expectations and the project's core purpose. Need to get better at taking meeting notes.
   Fei: Due to the two parts of this project, some members of the team took longer to understand the research side, which affect the understanding of what is required for the tool (i.e. what does the data look like). While writing this deliverable, we were able to identify the aspects that weren't as clear, and schedule meetings with our supervisor to clarify it for everyone.

3. How did you and your team adjust the scope of your goals to ensure they are suitable for a Capstone project (not overly ambitious but also of appropriate complexity for a senior design project)?
   Awurama: As a team, we agreed to keep the project focused on assessing software-engineering practices of neural network libraries, and to exclude accuracy and performance benchmarking. This adjustment kept the scope realistic for the Capstone timeline and ensured it matched the course expectations.
   Ghena: Through our meeting with our supervisor, we were able to adjust the scope of our goals to ensure they are suitable for a Capstone project. We discussed what the expected results are and what would be possible

add-ons is we had time.

Fei: By interacting with our supervisor early on in the process about what the team though the expectations were, we were able to quickly narrow down the focus, levaging the expertise of our supervisor on their past expertise of overseeing numerous past capstone projects. Identifying what were the base functionality compared to "nice-to-have's" were also important. That way we can ensure we at least finish the project, instead of being stuck in development trying to implement something that is just a cool feature