Software Requirements Specification for SFWRENG 4G06 - Capstone Design Process: DomainX

Team 17, DomainX

Awurama Nyarko
Haniye Hamidizadeh
Fei Xie
Ghena Hatoum

October 8, 2025

Table 1: Revision History

Date	**Developer(s)**	**Change**
October 4, 2025	Awurama Nyarko	Added Open Issues
October 4, 2025	Awurama Nyarko	Added Off-the-Shelf Solutions
October 4, 2025	Awurama Nyarko	Added New Problems
October 4, 2025	Awurama Nyarko	Added Tasks
October 5, 2025	Ghena Hatoum	Added Product Boundary
October 5, 2025	Ghena Hatoum	Added Product Use Case Table
October 5, 2025	Ghena Hatoum	Added Individual Product Use Cases (PUC's)
October 5, 2025	Ghena Hatoum	Added Functional Requirements
October 5, 2025	Ghena Hatoum	Added Look and Feel Requirements
October 5, 2025	Ghena Hatoum	Added Usability and Performance Requirements
October 5, 2025	Ghena Hatoum	Added Sections: Product Boundary to Usability and Performance Requirements
October 5, 2025	Haniye Hamidizadeh	Added Sections: Operational and Environmental Requirements to Compliance Requirements
October 6, 2025	Fei Xie	Added Sections: Cost to Ideas for Solution
October 6, 2025	Fei Xie	First Draft of Sections: Cost to Ideas for Solution
October 8, 2025	Fei Xie	Added Sections: Purpose of the Project to Naming Conventions and Terms
October 8, 2025	Awurama Nyarko	Fixing Tables

Contents

1 Purpose of the Project

1.1 User Business

With the recent advancements in LLM, AI, and ML in general. There has been more interest in the ML area. Focusing on the Neural Network domain, it can feel like a wild west of software tools and packages available. Dealing with "bad" software can be frustating, it can be hard to install, lacking in documentation, or being outdated. This can take precious times away from the actual research and development of the software the user wants, with the time spent on debugging the libraries used.
Following the "Methdology to assess the state of best practice for Domain X" paper, we will apply the methdology to the Neural Network Libary domain. To assist in the creation of this paper, we will also create a tool, whose main purpose is to simplify the process of gathering, visualizing and storing the data. This tool will than be reusable for future domains.

1.2 Goals of the Project

The completion of this tool will make the data gathering process faster and more efficient for the current paper used to pilot the project. Future papers will than be able to use the tool to assist in their own data gathering for their domain.
We'll measure the success of the goals listed in the Problem Statement and Goals document by the research subteam's experience while using the tool. A summary of the goals is provided below, for more information, please visit the above hyperlink.

1. Interactive Data Table
2. Automated Data Collection
3. Automated Analytic Hierarchy Process (AHP)
4. Data Visualization and Download
5. Collaboration

2 Stakeholders

The following includes stakeholders of our project,

2.1 Client

CAS Supervisor: The CAS Supervisor is the main product owner of the tool. They are responsible for providing the main features requested, as well has having the final say in the presentation and features of the tool. When the tool is live, they will also have administrative access.

Research Subteam: The research subteam will be actively using the tool. They are responsible for providing feedback to the product subteam on any pain points experienced while using the tool. As well as accessing if the tool makes the current data gathering process easier, as compared to the previous ad-hoc method of using excel.

2.2 Customer

Research Subteam/Future Students: See Research Subteam.

Future Students: Future students who interested in accessing the state of best practice for their specific domain. They have the same responsiblities as the Research Subteam.

2.3 Other Stakeholders

Domain Expert: The consultant for the research subteam during their work on the paper. This person has deep domain knowledge but won't necessarily know the technical software aspects of the domain.

2.4 Hands-On Users of the Project

Research Subteam/Future Students: See Research Subteam.

Future Students: See Future Students.

Researchers/Industry Workers: These are people who are interested in

seeing the results of the research results for a domain. They are responsible for accessing the usability of the tool in the context of a read-only experience, and provide feedback to the product subteam.

2.5 Personas

2.5.1 Kevin - Master's Student at McMaster University

I'm Kevin, a master student at McMaster University currently working on the state of best practice for Domain X. I am extremely busy so I don't want to spend too much time so i want to use a tool to decrease the amount of manual data inputting I need to do. I am adequately familiar with software, such as using the terminal and version control. However, my interests are more in the science aspect of computer science, so I don't want to spent too long setting things up.

2.5.2 Ashley - Research Scientist at X company

I'm Ashley, a research scientist currently working in the ML/AI field. During my work, I need access to varies tools to assist me. I don't want to create all the tools myself, preferring to use off-the-shelve, pre-existing libraries so I can focus on what actually matters. I've read through several tech blogs but they all rank the libraries differently, or don't consider a big enough variety, and sometimes the libraries mentioned are outdated. I would like to have a consolidated source that can measure varying aspects of the library since as a researcher, I haven't had as much experience with the DevOps process of the software lifecycle. I like to use packages and libraries that are easy to learn, install, and use.

2.6 Priorities Assigned to Users

Key Users:

- Research Subteam
- Domain Expert

Secondary Users:

- CAS Supervisor
- Researchers/Industry Workers

2.7 User Participation

Research Subteam
Actively using the tool during the writing of the paper. They should provide honest feedbacks on the experience when using the tool. Including any shortfalls, UI disagreements, and bugs.

Domain Expert
Has knowledge of the NN domain, will provide feedback on the research subteam's list of libraries to access, as well as help verify the conclusions during the research process, to see if it is similiar from what they have personally experienced. Estimated participation time of: 1-2 hours.

CAS Supervisor
During the user acceptance testing phase of the product. The CASE Supervisor should provide their honest feedback on their experience while using the tool, as they are the main administrator of the tool once it's live.

2.8 Maintenance Users and Service Technicians

Project Maintainer
The person responsible for updating the tool with new features, or fixing any existing bugs once the initial creators of the tool leaves. They were not present during the development of the tool, and will most likely be the future students who uses the tool for their domain research.

Infrastructure Team
The infrastructure team at McMaster University will be responsible for hosting the project, and should ensure that the application is available for use and maintains data integrity of their database servers.

3 Mandated Constraints

3.1 Solution Constraints

Insert your content here.

3.2 Implementation Environment of the Current System

Insert your content here.

3.3 Partner or Collaborative Applications

Insert your content here.

3.4 Off-the-Shelf Software

Insert your content here.

3.5 Anticipated Workplace Environment

Insert your content here.

3.6 Schedule Constraints

Insert your content here.

3.7 Budget Constraints

Insert your content here.

3.8 Enterprise Constraints

Insert your content here.

4 Naming Conventions and Terminology

4.1 Glossary of All Terms, Including Acronyms, Used by Stakeholders involved in the Project

Insert your content here.

5 Relevant Facts And Assumptions

5.1 Relevant Facts

The existing workflow relies on Excel sheets and bash scripts, which do not support automated data gathering.

5.2 Business Rules

Domain experts generally work Monday to Friday, 9 AM – 5 PM, and are unlikely to respond outside these hours.

5.3 Assumptions

McMaster University will provide the infrastructure required to host the application and make the necessary databases available for use.

6 The Scope of the Work

6.1 The Current Situation

Following the methodology listed:

1. Domain is identified
2. Research team gathers a preliminary list of packages fitting the domain
3. Domain expert is consulted and helps determine if the packages fit the domain
4. Data gathering commences for the finalized packages used
    (a) Optional: Interviewing or surveying industry users of the packages
5. Research paper is written, with visuals generated from the data

Current ways to consult the domain experts or industry users are done non-systematically, left to the discretion of the research team. The most common method is through email, but this can be difficult to track when multiple messages are sent or several people are being contacted simultaneously.

The data gathering process is performed using Excel spreadsheets, with each entry manually typed. This introduces inefficiencies and potential errors, motivating the need for an automated solution to streamline data collection, improve traceability, and reduce manual workload.

6.2 The Context of the Work

The Neural Network Libraries (NNL) Assessment Tool is a web-based application designed to automate the collection, evaluation, and visualization of data for comparing open-source deep learning libraries. It operationalizes an existing research methodology developed to assess the state of practice in software engineering, applying it specifically to neural network libraries (NNLs).

There is currently no automated system performing this type of assessment; existing workflows rely on manual data collection and Excel-based analyses. The tool introduces automation, consistency, and traceability across data gathering, scoring, and visualization activities.

The tool operates within McMaster University's research environment and interacts with multiple adjacent systems for data input, authentication, hosting, and output management. These interactions define the boundaries and environment in which the tool functions.

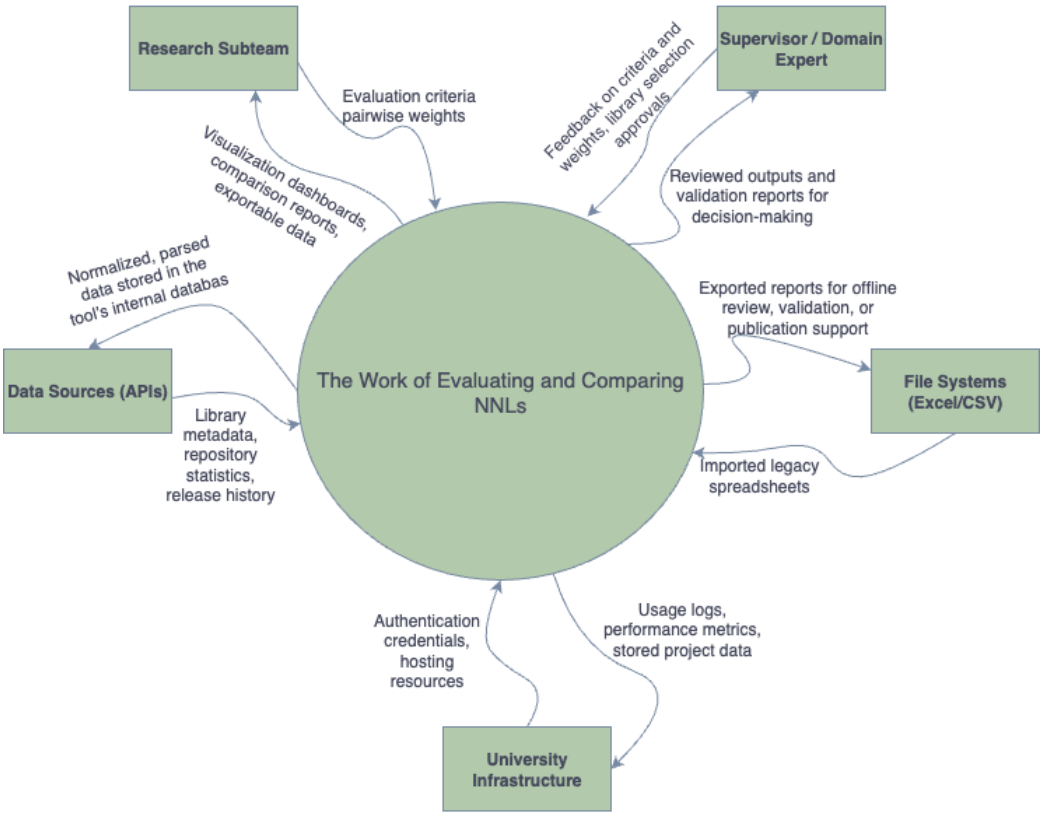

Figure 1: Work Context Diagram for the NNL Assessment Tool

Adjacent Systems:

- **Data Sources (APIs)** – GitHub, PyPI, and official library documentation, providing metadata, repository statistics, and release history.
- **Research Subteam** – Defines evaluation criteria and pairwise weights, initiates analyses, and reviews dashboards.
- **Supervisor / Domain Expert** – Reviews and validates library selections and output reports.
- **University Infrastructure** – Provides authentication services, hosting, and performance monitoring.
- **File Systems (Excel/CSV)** – Enables importing of legacy spreadsheets and exporting reports for offline review or publication.

Each interface between the tool and adjacent systems represents an exchange of data or actions, as shown in Figure 1. Inputs include user-defined criteria and API data; outputs include processed datasets, analytical dashboards, and reports.

All data interfaces will align with the definitions in the Data Dictionary (Section 7).

6.3 Work Partitioning

Table 2: Work Partitioning and Business Use Cases

Event Name	**Input & Output**	**Summary of BUC**
New Library/Package Added for Assessment	**Input:** Research subteam enters GitHub link or package name. **Output:** Tool retrieves metadata (stars, forks, releases, last commit) via API.	System automatically fetches repository metrics from GitHub and stores them in the database for analysis and visualization.
Library Becomes Archived / Deleted / Restricted	**Input:** API request fails or returns error. **Output:** Error log generated; library flagged for manual review.	System detects inaccessible data and notifies research team to investigate and update records.
Domain Expert Revises Library List	**Input:** Feedback from domain expert on selected packages. **Output:** Updated list of libraries for assessment.	System allows easy modification of included libraries; re-runs analysis with updated list.
Assessment Cycle Completed	**Input:** All data collected and validated. **Output:** Generated dashboards, ranking reports, and exportable visualizations.	System computes AHP-based scores and produces comparative visualizations and reports for research publication.

6.4 Specifying a Business Use Case (BUC)

Below is an example of a Business Use Case for a key event in the NNL Assessment Tool workflow.

Table 3: Business Use Case Example – New Package Added to Domain

Business Event	New library/package created for domain usage
Input(s)	GitHub repository link and metadata retrieved from public API
Output(s)	Stored library record in database; automated metrics collected (stars, forks, releases)
Trigger	Researcher adds new library or detects a new public repository
Primary Actor(s)	Researcher
Secondary Actor(s)	Domain Expert, System APIs
Preconditions	API access is available; library URL is valid
Postconditions	Library and metadata stored successfully; flagged for domain expert review
Main Flow	1. Researcher inputs new repository link. 2. Tool fetches metadata from GitHub and related APIs. 3. Data is validated and stored in the system database. 4. Confirmation and logs generated.
Alternative Flow(s)	- If API call fails, tool logs error and prompts user for manual data entry. - If duplicate record detected, tool alerts user and prevents duplication.
Business Rules	- All repositories must belong to open-source libraries under the defined domain. - Collected metrics must be timestamped for version tracking.
Exceptions	- API rate limits reached. - Network outage during data retrieval.

7 Business Data Model and Data Dictionary

7.1 Business Data Model

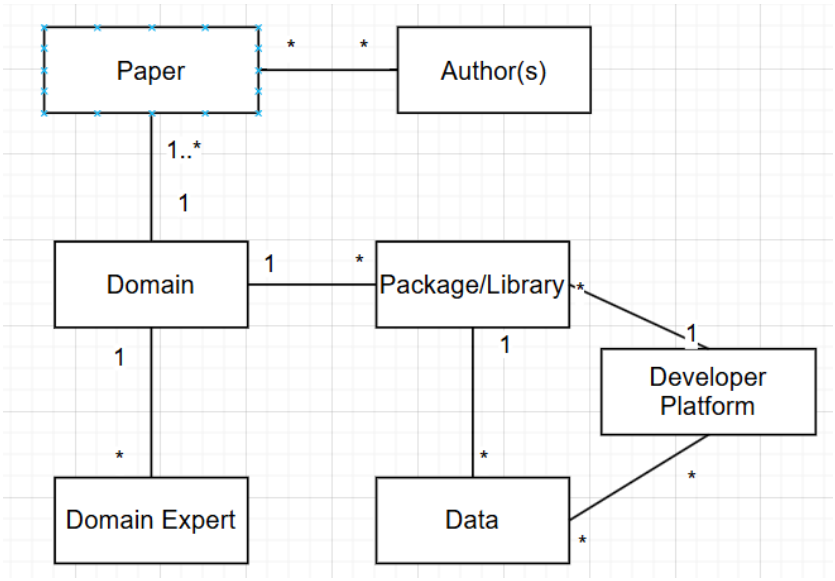

Figure 2: Business Data Model

7.2 Data Dictionary

Name	Content	Type
Paper	Paper Version + Paper Id + Paper Name	Class
Author/Contributor	Author Name, MacID	Class
Domain	Domain Name	Class
Packages/Library	Package Name + Github URL	Class
Data	Data description, Data Value	Class
Developer Platform	Package URL, Type, API Information	Class
Paper Version	*See Paper*	Attribute/Element
Paper Id	*See Paper*	Attribute/Element
Paper Name	*See Paper*	Attribute/Element
Package Name	*See Package*	Attribute/Element
Package URL	*See Package*	Reference
Data Description	*See Data*	Attribute/Element
Data Value	*See Data*	Attribute/Element
Package URL	*See Developer Platform*	Attribute/Element
Type	*See Developer Platform*	Attribute/Element
API Information	*See Developer Platform*	Attribute/Element

Table 4: Business Data Model Elements

8 The Scope of the Product

8.1 Product Boundary

This system aims to simplify and streamline the process of collecting, managing, and analyzing data to assess the state of the practice for various research domains. It replaces the existing, manual process (currently managed via Excel sheets) and formalizes the methodology outlined in the Methodology for Assessing the State of the Practice for Domain X paper. The final system will be a secure, accessible, data management and analysis tool.

The system will provide the following core functionality:

- **Security and User Management:**

  - Securely manage user accounts (login, password changes).
  - Accounts can only be created by an administrator invitation.
  - Enforce Role-Based Access Control (RBAC).
  - Ensure only authenticated users can modify data.
- **Data Integrity and Auditing:** Maintain an audit trail for all data entries, logging when a change is made.
- **Core Data Operations:** Facilitate the storing, updating, and retrieving of all core research entities: research domains, libraries, and collected metrics.
- **Data Collection:**
  - Support mass data upload from previous research work.
  - Automatically collect data to populate a specified data table (e.g., State-of-Practice Metrics Table).
- **Analysis and Visualization:**
  - Implement an AHP-based ranking of libraries within a domain based on their state of practice.
  - Visualize data as different 2D graphs, showing the difference between libraries in a domain.
- **User Experience (UX):**
  - Have an intuitive UI to improve overall user experience.
  - Ensure accessible read permissions for all users.
- **Data Export:** Allow users to download tables or graphs onto their local device.

The following capabilities are explicitly excluded from the initial scope:

- **Communication:** The system will not facilitate scheduling or hosting meetings between researchers and domain experts.
- **User Data:** Only username, email, and password will be collected for user profiles, no need for personal information.

- **Fine-Grained Permissions:** The system will not separate domain access by researcher. All authenticated researchers will be able to view and edit any domain.
- **Data/Library Recommendations:** The system will not give users suggestions on which library to use based on insufficient information.
- **Data Imputation/Handling:** The system will not automatically fill in missing data, nor will it incorporate a mechanism to assume the possibility of missing data during AHP calculation.
- **System Management:** The system will not include functionality for adding or removing administrator accounts.
- **AHP Validation:** The system will not include functionality to check the accuracy of the AHP calculations.

The following items are recognized as potential enhancements but are considered low-priority:

- **Data Auditing:** Logging who edited the data entries.
- **Conflict Resolution:** Creating a method for merging changes when multiple users edit the same data simultaneously.

8.2 Product Use Case Table

Use Case (Goal)	Primary Actor	Secondary Actor(s)	Key Functional Requirements (FRs)
Invite New Users	Admin	Researcher	Admin can input a new user email. New user will receive an email with a link that will allow them to setup an account, password, and username.
Domain Creation & Setup	Admin	Researcher	Create new Domain (must validate unique name). Optional: Add description to the domain.
Publish Domain	Researcher	General Users	Once the data collection is finished, allow anyone to see it (General Users).
Automation Library Data	Researcher	System	Auto-fill fields (e.g., creation date, commit date) upon entering a public URL. Check for and flag missing data that was expected to be collected.
Manually Input/Update Data	Researcher	System	Add missing data through the data table. Missing data must be highlighted in red. Only allow user to input right format (e.g., numbers, text). Maintain an audit trail of changes.
Ranking (AHP)	Researcher	System	System will run the integrated AHP tool using input quality scores and expert pairwise comparisons to automatically calculate the final package rankings. Run process automatically when the table is filled and updated.
Visualize & Export Results	General User / Researcher / Admin	System	Allow selection of domain, libraries, and metrics to 2D graphs. Allow download of collected data (JSON/Excel). Allow download of graphs (PNG/LaTeX).

8.3 Individual Product Use Cases (PUC's)

Important things to keep in mind when looking at the at the Use Case Diagram:

- Include: A include B means, if A is executed that means B will be executed as well.
  - Update Data include Login, if user wanted to Update Data they must Login.
  - Login include Verify Account, when user tries login the system will verify account.
- Extend: A extends B means, if A is executed that means that B could be executed as well but not in all cases.
  - Create New Domain extends Mass Data Upload:Create New Domain could also mean that the user wants to upload previous work, which would need a mass data upload, but not in all cases.
  - Login extends Failed Login: When trying to login the user might fail to do so, however it's not always the case.

9 Functional Requirements

9.1 Functional Requirements

FR1 The Admin will be able to extend an invitation to new researchers via the website, using the researcher's email address.

FR2 Researchers will be able to sign up using their invited email. The

process requires email validation and the creation of a unique username and password.

FR3 The system will allow Admin and Researchers to securely log in using their email and password.

FR4 Users will be able to reset their password after validating their email address.

FR5 Admin can create domains, using a unique domain name and optionally a short description.

FR6 Admin can publish a domain, once research is completed, for anyone to see and download data collected.

FR7 Researchers can add/modify data/libraries of existing domains, and download data/graphs regardless of completion status.

FR8 The system must validate all user input upon addition or update (e.g., ensuring numeric fields are numbers, dates are valid, and text limits are met).

FR9 The system must allow a Admin to bulk upload library and metric data using a predefined Excel template. The system must report detailed errors for any records that fail validation, such as feature names not being in the scope or wrong data type for a column.

FR10 When a Researcher enters a public GitHub URL for a library, the system must call the GitHub API to automatically retrieve and populate specific data points, such as the Repository Creation Date and the Last Commit Date.

FR11 The system must display a complete list of all metrics and their definitions for a selected domain.

FR12 The system will rank the libraries within a domain using the Analytical Hierarchy Process (AHP) and display the calculated results.

FR13 Users must be able to select a domain $\rightarrow$ libraries $\rightarrow$ metrics $\rightarrow$ graph type to visualize differences in libraries using comparative graphs.

FR14 All users will be able to download the collected data for a specific scope in JSON, Excel, or LaTeX code formats to their device.

FR15 Users must be able to download any generated graph, with the option to save it as a PNG file.

10 Look and Feel Requirements

10.1 Appearance Requirements

LF-AR1 The application will maintain a consistent visual hierarchy where primary action buttons (e.g., "Save," "Create New Domain") are visually distinct.

LF-AR2 The system must be fully responsive and function correctly on standard desktop monitors and laptops.

LF-AR3 Data visualization and editing displays must prioritize information density while remaining readable. Tables should have alternating row colors (zebra striping) to help tracking.

LF-AR4 The main user dashboard should provide a clear, high-level overview of active domains and tasks, with minimal visual clutter.

LF-AR5 Ensure all required input fields are clearly identified (e.g., with an asterisk). Input forms must give real-time feedback, such as a green checkmark for acceptable input and a red boundary for errors.

LF-AR6 All error messages (system errors, validation problems) must be clear, short, and contain practical guidance for resolving the problem.

10.2 Style Requirements

LF-SR1 The system must use a single, consistent style of icons (e.g., solid, outline, or filled) for all navigational elements, actions, and status indicators.

LF-SR2 Graphs must use minimalist design to prioritize data readability. Axes and labels must be clear and legible.

LF-SR3 Font sizing must follow a defined scale (e.g., large for titles, medium for body text, small for captions) and remain consistent on all screens.

11 Usability and Humanity Requirements

11.1 Ease of Use Requirements

UH-EU1 The primary navigational structure should be clearly defined and constantly available, with no prior training required for basic data viewing and download.

UH-EU2 Each library metric must have help icon that would tell the user what data type would be accepted.

UH-EU3 All errors will give clear and concise explanation.

11.2 Personalization and Internationalization Requirements

N/A

11.3 Learning Requirements

UH-LR1 The system must contain a brief lesson or guide that explains the precise steps for Bulk Data Upload and the AHP Ranking process and is available from the appropriate screens.

UH-LR2 All words referring to research entities (Domain, Library, Metric) must be used consistently across the interface and documentation.

11.4 Understandability and Politeness Requirements

UH-UP1 The system must contain a brief lesson or guide that explains the precise steps for Bulk Data Upload and the AHP Ranking process and is available from the appropriate screens.

UH-UP2 All words referring to research entities (Domain, Library, Metric) must be used consistently across the interface and documentation.

UH-UP3 All system-generated error messages will be clear, use non-technical language, and suggest a simple course of corrective action

11.5 Accessibility Requirements

UH-AR1 Critical status indicators and data visualizations must convey meaning through patterns or text labels in addition to color.

UH-AR2 Users must be able to change font size using regular browser options (up to 200%) without losing information or functionality.

12 Performance Requirements

12.1 Speed and Latency Requirements

PR-SL1 All key data viewing and dashboard pages must load completely and become interactive within 3 seconds.

PR-SL2 The system must finish the AHP ranking calculation for a standard domain in less than ten seconds.

12.2 Safety-Critical Requirements

PR-SC1 The system will encrypt all user passwords.

PR-SC2 If there have been multiple failed attempts to login to an existing account the user must be notified and lock account.

PR-SC3 The system will ensure that only researchers modify data.

12.3 Precision or Accuracy Requirements

PR-PA1 All data that the system automatically filled out, must be 100% accurate.

12.4 Robustness or Fault-Tolerance Requirements

PR-RFT1 The system must implement graceful error handling, ensuring that a failure in one function (e.g., a failed GitHub API call) does not crash the entire application or disrupt other users.

PR-RFT2 The system must use database transactions to ensure that data modifications are atomic, if an update operation fails in the middle, the data must be returned to its last validated state.

12.5 Capacity Requirements

PR-CR1 The system must be able to accommodate up to 20 concurrent active users (logged in and doing operations) with no visible performance decrease.

PR-CR2 The database must be constructed to accommodate at least 10 different study domains.

12.6 Scalability or Extensibility Requirements

PR-SE1 The system architecture must clearly divide the Data Management/Storage layer from the Analysis/Visualization layer so that either component may be improved or replaced independently.

PR-SE2 The authentication and authorization system must be adaptable enough to accommodate additional user roles (e.g., "Guest Analyst") and specialized rights without requiring code changes to core operations.

12.7 Longevity Requirements

PR-LR1 The system must be capable of running without the requirement for maintenance for two years.

13 Operational and Environmental Requirements

13.1 Expected Physical Environment

OE-EPE1 The tool is a web-based application that will be used mainly by our team, supervisors, and potentially other researchers or domain experts.

OE-EPE2 It is expected to run on a standard desktop or laptop computer with a reliable internet connection, in a normal indoor setting such as an office, lab, or home workspace.

OE-EPE3 No specialized hardware or rugged equipment is required. A keyboard, mouse, or touchpad, and a modern web browser (e.g., Chrome, Firefox, Edge) are sufficient.

13.2 Wider Environment Requirements

OE-WE1 The application depends on a stable internet connection for retrieving data from public repositories (e.g., GitHub) and for loading the hosted web interface if deployed to the cloud.

OE-WE2 It does not rely on any dedicated on-premises hardware.

OE-WE3 The tool should work on the latest two to three major versions of common browsers such as Chrome, Firefox, and Edge. No special environmental conditions (lighting, noise, temperature) are expected to affect usability.

13.3 Requirements for Interfacing with Adjacent Systems

The tool must be able to communicate with a few external systems and internal components to collect, store, and visualize data.

OE-IA1 Public Repository APIs (e.g., GitHub API): The tool must communicate with external repository services to automatically collect information about the selected neural-network libraries. For example, it needs to retrieve details such as how often the code is updated, the number

of open or closed issues and pull requests, and the main programming languages used. This data will help evaluate qualities like maintainability, transparency, and overall project activity. The information must be received in a standard machine-readable format (e.g., JSON over HTTPS) whenever a user triggers a scan or during scheduled updates.

OE-IA2 Database (MySQL): The backend must store scores, rankings, and evidence collected from repositories.

OE-IA3 Visualization Component: The frontend must render charts and comparisons (e.g., using Chart.js) from the data served by the backend.

OE-IA4 All connections must use standard web technologies (HTTP/HTTPS, JSON) and require only basic authentication methods such as API tokens for secure access to external repository APIs.

13.4 Productization Requirements

The tool will be delivered as an open-source web application hosted in the team's public GitHub repository.

OE-PR1 A clear README file must explain how to set up the backend (Python + Flask) and the frontend (React) using common package managers such as pip and npm.

OE-PR2 For developers running the tool locally, the repository must include a requirements.txt file for Python packages and a database schema file so they can create the required tables.

OE-PR3 When deployed on a cloud platform such as AWS or Google Cloud, users must be able to access the tool directly through a web URL without installing anything.

OE-PR4 The application must also provide options to export results in formats such as CSV for data tables and PNG/PDF for visualizations.

13.5 Release Requirements

OE-RR1 The project should follow the official capstone timeline: an internal test release before the Proof-of-Concept (PoC) demonstration in **November**, a Revision 0 Demonstration in **Weeks 18–19**, and the Final

Release (Revision 1) at **Week 26** along with the research paper and final dataset.

OE-RR2 The Final Release (Revision 1) must incorporate feedback collected during the Revision 0 Demonstration, including usability improvements, bug fixes, and supervisor/TA-requested changes.

OE-RR3 All releases must be published as GitHub Releases and include a short changelog describing the changes in each version.

OE-RR4 Releases must use a clear, descriptive version tag such as the `MAJOR.MINOR.PATCH` format (or an equally descriptive format):

- **MAJOR:** Increased only when a change breaks backward compatibility (e.g., a database schema change that makes older data unusable).
- **MINOR:** Increased when adding new features that remain fully compatible with previous versions.
- **PATCH:** Increased for bug fixes or small improvements that do not affect existing features.

Example version tags:

- `v0.1.0` → first working prototype for the Revision 0 Demonstration
- `v0.2.0` → adds a new feature such as exporting the table to a CSV file
- `v0.2.1` → fixes a small bug in the export feature (patch)
- `v1.0.0` → stable Final Release for Revision 1

14 Maintainability and Support Requirements

14.1 Maintenance Requirements

The tool is an open-source web application that will need occasional updates to fix bugs and to adapt if external APIs change.

MS-MR1 All code must follow the project's coding standards (PEP 8 for the Python backend; React + TypeScript style guide for the frontend).

MS-MR2 Automated tools (such as Black, Flake8, and Pylint) must remain part of the workflow so new contributors can easily read and update the code.

MS-MR3 Unit tests must be written for all new features and bug fixes. Developers should also run basic integration tests to make sure the full pipeline (API $\rightarrow$ database $\rightarrow$ visualization) still works after changes.

MS-MR4 Test coverage must be tracked and reported to ensure that critical parts of the backend are being tested.

MS-MR5 All Python and frontend packages must be listed in requirements.txt and package.json, with pinned versions so the same build can be reproduced.

MS-MR6 The dependency list must be reviewed and updated at least once each semester to keep it current and secure.

MS-MR7 Any changes to the dataset made through the interactive data table must be logged with a timestamp and the username so there is always a clear audit trail.

MS-MR8 Most routine fixes, such as small UI tweaks or bug fixes, should be finished within a couple of days. Larger updates, such as adding a new metric or a new visualization, are expected to take about one to two weeks.

14.2 Supportability Requirements

The tool is intended to be mostly self-supporting since it will be used primarily by our team, the supervisors, and potentially other researchers in the future.

MS-SR1 The README file must include step-by-step instructions so that a new developer can set up the backend (Flask + MySQL) and frontend (React) locally, or deploy it to a cloud platform (such as AWS or Google Cloud), in a consistent and repeatable way. A new developer should be able to follow these instructions and have the application running within about two hours.

MS-SR2 The repository must always include an up-to-date guide for installation, setup, the data-collection workflow, using the interactive data table, visualization, and exporting results.

MS-SR3 The backend must include logging to record API failures (such as rate-limit errors or unexpected data formats), database issues, and runtime exceptions so that maintainers can troubleshoot problems quickly.

MS-SR4 Users should use the GitHub Issues page to report bugs or ask questions. No printed manual will be needed; all documentation will remain online in the repository.

14.3 Adaptability Requirements

MS-AR1 The tool must remain flexible so it can grow with the project and adapt to future needs.

MS-AR2 The MySQL database must be set up so that if a new quality criterion needs to be tracked, it can be added by creating a new column, updating the data-collection script, and adjusting the UI without having to redesign the whole system.

MS-AR3 The data-collection module must allow new data sources (for example, another code-hosting site or a different metrics service) to be added with minimal extra code and without disrupting the existing GitHub integration.

MS-AR4 The tool must be able to run on standard operating systems(Windows, macOS, and Linux) by using widely supported technologies (such as Python, Node.js, and MySQL), and by avoiding any system-specific code.

MS-AR5 The system must be designed in a modular way so that parts like data collection, storage, and visualization remain separate. This makes it easier to add new features or update one part without affecting the rest of the tool.

15 Security Requirements

15.1 Access Requirements

SR-AC1 The tool must be open for anyone to view results, but only approved team members or domain experts may modify the data.

SR-AC2 The system must support two user roles:

- **Viewer:** Can view all interactive data tables that list the libraries, their scores, and rankings, along with all visualizations.
- **Contributor:** Has all Viewer permissions plus the ability to edit data in the interactive table.

SR-AC3 Editing or any other administrative actions must be available only to logged-in users with the **Contributor** role.

SR-AC4 User roles (Viewer / Contributor) must be assigned and updated correctly at signup or by an admin so that permissions always reflect the intended access level.

SR-AC5 The system must display clear error messages for login failures (e.g., invalid credentials, network errors) and provide a secure way for users to recover or reset their account credentials.

SR-AC6 API credentials such as keys or tokens for external services (e.g., repository APIs) must be stored securely outside the source code (e.g., in environment variables) and must never be exposed in the frontend or version control.

15.2 Integrity Requirements

SR-INT1 To ensure the accuracy and reliability of the collected data, all manually entered information into the interactive table must be checked before it is saved. For example, numbers must fall within valid ranges, and text must be cleaned so that it cannot be treated as code or scripts.

SR-INT2 The database must include safeguards so that if two users edit the same record at the same time, no changes are lost or overwritten.

SR-INT3 The system must detect simultaneous edits and either block one save or notify the users to resolve the conflict.

SR-INT4 When automated data updates conflict with a user's manual edits (if any are updated automatically), the system must warn the user or request confirmation before replacing existing data.

SR-INT5 The system must also store the raw evidence, calculated scores, and final rankings in separate fields so that the original data and results cannot be accidentally modified.

SR-INT6 Publishing visualizations must not occur at the same time as data edits or automated refreshes. The system must either block publishing or enforce downtime until updates are complete.

SR-INT7 If an export (e.g., CSV, PNG) fails or produces a corrupted file, the tool must alert the user and allow them to retry the download.

15.3 Privacy Requirements

The tool mainly works with open-source repository data, so there is very little personal data involved.

SR-P1 If basic user details are collected for login (such as name or email), they must be kept private and stored securely using widely accepted security standards.

SR-P2 User passwords must be stored only in hashed form using a secure one-way hashing algorithm so that the actual password is never saved.

No other personal or confidential information will be collected or stored.

15.4 Audit Requirements

SR-AU1 The system must keep a record of all important activity for accountability and troubleshooting. Every time a Contributor adds, edits, or deletes data in the interactive table, the system must save a log entry showing the user's ID, the time of the action, the type of action, and the fields that were changed.

SR-AU2 The system must also log key events such as login attempts (successful and failed), scheduled API-collection runs, and major errors so that they can be reviewed later.

SR-AU3 Access to these logs must be restricted to authorized project admins only.

15.5 Immunity Requirements

SR-IM1 The backend must prevent SQL-injection attacks by using the safe query methods provided by the database library instead of building raw SQL strings with user input.

SR-IM2 The frontend must display any text entered by users as plain text only and never allow it to run as code. For example, React's built-in escaping already takes care of this.

SR-IM3 The automated data-collection scripts must respect the rate limits of external repository APIs (e.g., GitHub). If a limit is reached, the scripts should pause and retry later rather than keep sending requests and risk overloading the service.

SR-IM4 All Python and React libraries must be kept up to date, so the project does not rely on outdated packages with known security issues. There should also be a tool in place to regularly check for known vulnerabilities in these libraries, such as GitHub Dependabot.

16 Cultural Requirements

16.1 Cultural Requirements

Since the tool will be shared mostly in an academic and research setting, we just need to make sure the interface stays clear, neutral, and easy to understand for anyone who uses it later.

CU-CR1 **Language:** All text in the user interface and documentation must be in plain English. We should avoid using region-specific jargon so that future collaborators from outside McMaster can understand it easily.

CU-CR2 **Dates and Numbers:** Dates must be displayed in a clear standard format such as ISO 8601 (YYYY-MM-DD) so they're unambiguous for anyone who uses the tool. Numbers must use a consistent style (for example, decimal point for fractions and thousands separated by commas).

CU-CR3 **Neutral Design:** Colours, labels, and icons must stay neutral and must not include any symbols or words that could carry unintended cultural or political meaning.

CU-CR4 **Open-Source Norms:** The repository must include a license notice and simple contribution guidelines to match common open-source practice.

17 Compliance Requirements

17.1 Legal Requirements

CR-LR1 The tool must include a copyright notice covering the code and documentation produced by the team.

CR-LR2 It must be distributed under an appropriate open-source license so that others can reuse it under clearly defined terms.

CR-LR3 Users of the tool or any data generated by it are required to provide proper citation or acknowledgement when they use it in their own work.

17.2 Standards Compliance Requirements

CR-STD1 N/A

18 Open Issues

Below is a list of factors and questions that remain unresolved and could significantly influence the design, functionality, or deployment of the Neural Network Libraries (NNL) Assessment Tool.

The purpose of identifying these open issues is to ensure they are tracked, managed, and resolved in a timely manner. Maintaining visibility of these uncertainties supports effective risk management and informed decision-making throughout the development lifecycle.

18.1 List of Open Issues

Table 5: Open Issues to Track and Resolve

Issue #	Summary	Cross-Reference	Stakeholders Involved	Action Required	Status
OI-01	Finalization of hosting environment (e.g., internal server)	Section 13 – Operational Requirements	CAS Supervisor, Infra Team	Confirm approved infrastructure with McMaster IT	Pending
OI-02	Confirmation of the final list of Neural Network Libraries	Section 6 – Scope of Work	Research Subteam, Domain Expert	Schedule review meeting with domain expert	Pending
OI-03	Selection of UI framework and visualization library	Section 9 – Functional Requirements	Development Team	Evaluate and finalize tech stack	In Progress
OI-04	Clarification of Excel integration level (import/export vs live sync)	Section 9 – Functional Requirements	Research Subteam, Developers	Define required use cases and confirm feasibility	Pending
OI-05	Decision on authentication method (SSO vs custom login)	Section 9 – Functional Requirements	CAS Supervisor, Development Team	Meet with IT to assess SSO feasibility	Pending
OI-06	Confirmation of data storage architecture (SQL vs cloud)	Section 13 – Operational Requirements	Developers, Infra Team	Evaluate hosting options and finalize design	Pending

19 Off-the-Shelf Solutions

This section identifies existing tools, software, and components that could be leveraged to reduce development time and cost for the Neural Network Libraries (NNL) Assessment Tool. It outlines reusable libraries and products that can be legally copied or adapted to accelerate development and ensure maintainability.

The goal is to reuse proven, reliable components and avoid reinventing existing functionality, thereby optimizing development effort and leveraging established best practices.

19.1 Ready-Made Products

Several existing platforms provide partial functionality aligned with the tool's goals, particularly in data visualization, analytics, and automation. However, none fully satisfy the requirement for automated data gathering, Analytic Hierarchy Process (AHP) analysis, and integrated visualization.

Table 6: Ready-Made Products Relevant to the NNL Assessment Tool

Product	**Description**	**Relevance to Project**	**Limitations**
Microsoft Excel	Spreadsheet with storage, formulas, and charts.	Used as baseline for current manual process.	Lacks automation, scalability, and centralized access.
Google Sheets	Cloud-based collaborative spreadsheet.	Enables multi-user editing and sharing.	Limited automation; manual data import.
Power BI / Tableau	Advanced analytics and visualization tools.	Suitable for dashboards and comparative graphs.	Licensing cost; limited AHP customization.
SurveyMonkey / Google Forms	Online data collection tools.	Useful for gathering expert feedback.	No direct database or analytics integration.

These products may serve as inspiration or integration points (e.g., Ex-

cel import/export) but cannot replace the custom automation and analysis required by the NNL Assessment Tool.

19.2 Reusable Components

The following open-source libraries and frameworks will be incorporated to support data collection, analysis, and visualization.

Table 7: Reusable Components

Component	**Purpose**	**Source**	**Justification**
Python Pandas	Data manipulation and analysis.	Open-source	Handles tabular data and transformations efficiently.
Requests (Python)	Retrieve data from GitHub / PyPI APIs.	Open-source	Automates collection of repository metrics.
Matplotlib / Plotly / Chart.js	Visualization libraries.	Open-source	Create interactive and exportable graphs for dashboards.
AHPy / ahpy	Analytic Hierarchy Process implementation.	Open-source	Automates pairwise comparison scoring.
Flask	Backend web framework.	Open-source	Manages API integration and data handling.
React	Frontend user interface framework.	Open-source	Enables responsive dashboards and visualization.
MySQL / SQLite	Database systems.	Open-source	Store collected data and evaluation results.

Reusing these components ensures consistency, leverages reliability, and minimizes custom development effort.

19.3 Products That Can Be Copied

Some open-source or academic research dashboards share functional similarities with the NNL Assessment Tool and may inform design or architecture.

Table 8: Products That Can Be Copied or Adapted

Product	**Description**	**Adaptation Potential**
Software Assessment Dashboards	Tools that analyze open-source software metrics.	Their structure can guide data collection, evaluation, and dashboard layout.
University Research Repositories	Academic dashboards for research analytics.	Useful for UI patterns, visualization approaches, and data categorization strategies.

Adaptation saves design time and provides validated frameworks for implementation while maintaining legal compliance.

Considerations:

- Licensing for third-party libraries must be reviewed before adoption.
- Integration testing is required to confirm compatibility.
- Long-term maintenance and documentation quality will influence selection.

Each reused or adapted product will be documented with:

- Name and source
- Functionality
- Integration plan
- Licensing details
- Selection status

20 New Problems

This section identifies potential conflicts, risks, or issues that may arise as a result of implementing the NNL Assessment Tool within McMaster University's research environment. The purpose is to anticipate and document any negative effects or dependencies introduced by the new system.

20.1 Effects on the Current Environment

The new tool will operate within McMaster University's research infrastructure and will rely on institutional hosting (e.g., internal servers). It may introduce additional server load and require IT resources for maintenance.

Motivation: To ensure the new tool integrates smoothly without disrupting existing research tools, storage systems, or academic workflows.

Examples:

- Increased data storage requirements may conflict with current quotas.
- Additional IT workload for managing hosting or user accounts.

Considerations:

- Coordination with the IT infrastructure team is required to confirm compatibility with university servers.
- Ensure the tool does not negatively affect access to existing research applications or networks.

Form: Documented assessment of integration impact with existing systems, supported by infrastructure review.

20.2 Effects on the Installed Systems

The new tool will interface with existing systems such as Excel, university authentication systems (SSO), and potentially university-hosted databases.

Motivation: Identify dependencies between the tool and existing platforms, ensuring stable coexistence and avoiding version conflicts.

Considerations:

- Compatibility with current Microsoft Excel versions.

- Security compliance when connecting to McMaster's SSO.
- Avoid introducing vulnerabilities or version mismatches.

Form: Integration map specifying systems affected, their current versions, and compatibility requirements.

20.3 Potential User Problems

Potential user challenges include onboarding, usability learning curves, and confusion regarding data visualization or interpretation.

Motivation: Ensure researchers and users can adopt the system efficiently without frustration or misinterpretation of outputs.

Considerations:

- Provide user documentation and tutorials.
- Offer training sessions or quick-start guides.
- Establish a support contact for reporting issues.

20.4 Limitations in the Anticipated Implementation Environment That May Inhibit the New Product

Possible constraints include limited hosting capacity, restricted access to certain cloud features, and dependence on McMaster's infrastructure approval.

Motivation: Identify environmental limitations that could delay deployment or reduce tool performance.

Examples:

- Hosting quotas may limit database scaling.
- University IT policies may restrict certain libraries or APIs.
- Limited access to high-performance computing resources.

Considerations: Execute a full review to confirm infrastructure readiness.

20.5 Follow-Up Problems

Potential long-term issues include sustaining the tool after project completion and keeping data updated as new Neural Network Libraries emerge.

Motivation: Ensure the system remains relevant and operational beyond the capstone timeline.

Considerations:

- Define ownership and maintenance responsibilities after project handover.
- Plan for version updates, new library integrations, and user feedback loops.
- Ensure continuity when original developers graduate.

21 Tasks

21.1 Project Planning

The Neural Network Libraries (NNL) Assessment Tool will be delivered using a hybrid development approach that combines Agile iterations with structured milestone-based deliverables aligned with the McMaster Software Capstone schedule.

The lifecycle is divided into major phases: Requirements, Design, Implementation, Testing, and Deployment, each building toward a functional and hosted tool that supports the research team in evaluating neural network libraries.

This approach ensures iterative feedback from supervisors and domain experts after each milestone, enabling continuous refinement. Development will be managed through GitHub for version control, VS Code for coding, and LaTeX for documentation.

The tool will be hosted on McMaster's internal infrastructure or an approved equivalent, with key non-functional activities such as user onboarding, data migration, and training planned in the later stages.

Table 9: Project Planning Phases

Phase	Description	Key Activities	Deliverables
Initiation	Define problem, scope, and objectives	Document review, team formation, feasibility assessment	Problem Statement, POC Plan, Development Plan
Requirements	Gather and formalize system requirements	Stakeholder interviews, drafting of SRS and Hazard Analysis	SRS Document, Hazard Analysis
Design	Establish system architecture and interfaces	UI mockups, data flow diagrams, schema design	Design Document
Implementation	Build core tool functionality	Develop modules, integrate APIs, implement Excel import/export	Proof of Concept Demo
Validation & Verification	Test and evaluate system performance	Unit and integration testing, review sessions, issue resolution	V&V Plan, V&V Report
Deployment	Deliver hosted solution	Deploy tool, prepare documentation, conduct final demo	Final Hosted Tool, Presentation, Final Report

21.2 Planning of the Development Phases

Each phase contributes to the development of a usable and reliable product. Feedback loops will be incorporated at each stage to ensure alignment with stakeholder expectations, compliance with requirements, and delivery of a secure, maintainable, and user-friendly system.

Table 10: Development Phase Plan

Phase Name	**Benefit to User**	**Operational Date**	**Operating Components**	**Functional Requirements**	**Non-Functional Requirements**
Requirements & Analysis	Clarifies system objectives and constraints	Week 1–6	GitHub, LaTeX	Requirements documentation	Accuracy, clarity
Design	Defines architecture and interfaces	Week 10–16	UML tools, wireframing software	UI and backend design	Maintainability
Implementation	Delivers core product functionality	Week 16–19	IDE, databases, APIs	Tool modules, automation scripts	Reliability, usability
Testing & Validation	Ensures product meets all criteria	Week 17–22	Test suites, CI/CD	Verification of features	Performance
Deployment	Provides accessible hosted tool	Week 22–26	Hosting platform, documentation	Hosted application	Security, accessibility

22 Costs

There is no development cost for this project, due to the nature of this being a capstone project.
The costs of hosting the required services, such as the database and the web application will depend on McMaster University's existing infrastructure.
However, estimating using a common cloud provider such as Amazon Web Services (AWS).
Using the Relational Database Service (RDS) for database and the Elastic Cloud Computing (EC2) service for hosting. Assuming around a maximum usage of 20 Hours/Month, the total cost of hosting is 11.63 CAD per month, as shown in Table 11.

Name	**Configuration**	**Estimated Usage**	**Total Cost**
RDS for MySQL	Two vCPU and 8GB of Memory	20 Hours/Month	9.76 CAD/Month
EC2	t4g.large Instance	20 Hours/Month	1.87 USD/Month

Table 11: Price estimates and total costs for two AWS services.

23 User Documentation and Training

23.1 User Documentation Requirements

23.1.1 User Manual

The user manual will highlight the key features of the product, and provide additional details for installing, setup and usage that wasn't previously covered in the project's README.

The maintenance of this document will be the responsibility of the development team. Changes to the product's features, such as adding new features or altering existing features must be reflected in the user manual upon release of the update.

23.1.2 Release Manual

The release manual will cover the release process required for future releases of the product, found in the Development Plan. It should include the whole CI/CD lifecycle, including team standards, release labelling, and more.

The maintenance of this document will be the responsibility of the development team. Changes to the CI/CD process, either from the development team or the broader McMaster University infrastructure team should be reflected before the next release.

23.2 Training Requirements

1. Users should be able to use the tool and utilize key features immediately after following the tutorial

2. Users should be able to find key features without consulting additional documentation 95% of the time.

The responsibilty of the training will first fall towards the development team of the tool. They must ensure the provided in-tool tutorial is up-to-date and sufficient to help a user understand all key features.
Additional training will be provided to the supervisor, hosted by the development team. Subsequent training will be the responsibility of the supervisor, if needed to train future users of the tool, in the format that the supervisor chooses.

24 Waiting Room

1. Comparing across Domains
The user should be able to compare two (or more) completed domain analysis against each other.

2. Versioning of Domains
Users can revist and update completed domains, adding new data or editing existing ones. Allowing users to view the evolution of the state of best practice for the domain.

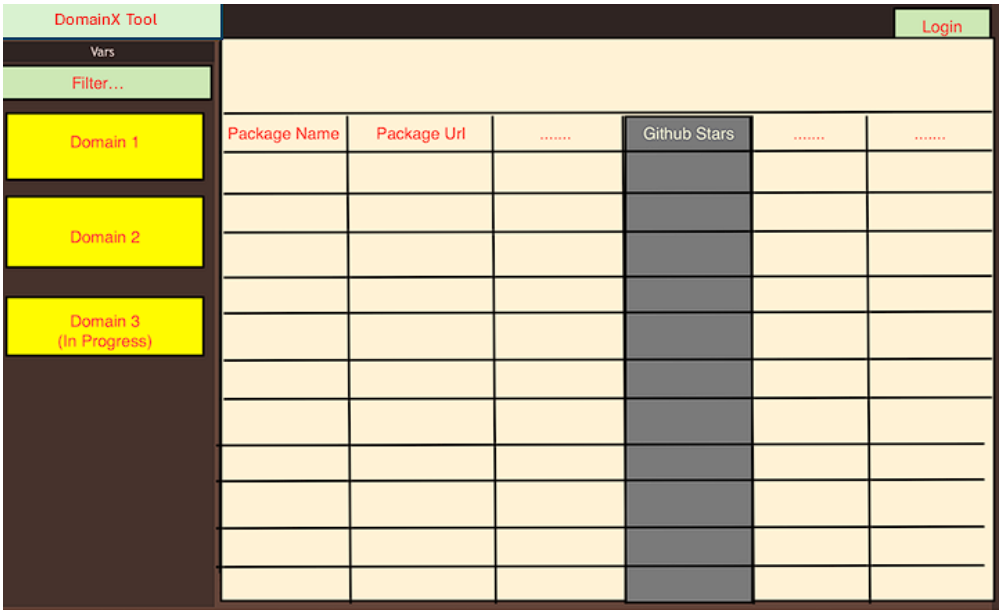

Figure 3: DomainX UI, inspired by Octave Online

25 Ideas for Solution

The following is the development team's idea for the user interface of the tool. Figure 3 shows the initial concept, drawing inspiration from Octave Online, the cloud IDE for Matlab.
The main section of the tool will be where the data is displayed and gathered for each domain. With the sections that can be automatically gathered differentiated using a different colour, such as the gray shown in Figure 3.
The left sidebar will contain all the domains, with indications on whether it's completed or not. As well as providing a filter to quickly search for a specific domain.

Appendix — Reflection

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?

2. What pain points did you experience during this deliverable, and how did you resolve them?

3. How many of your requirements were inspired by speaking to your client(s) or their proxies (e.g. your peers, stakeholders, potential users)?

4. Which of the courses you have taken, or are currently taking, will help your team to be successful with your capstone project.

5. What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project? Examples of possible knowledge to acquire include domain specific knowledge from the domain of your application, or software engineering knowledge, mechatronics knowledge or computer science knowledge. Skills may be related to technology, or writing, or presentation, or team management, etc. You should look to identify at least one item for each team member.

6. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?