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 6, 2025	Fei Xie	First Draft of Sections: Cost to Ideas for Solution

Contents

1 Purpose of the Project

1.1 User Business

Insert your content here.

1.2 Goals of the Project

Insert your content here.

2 Stakeholders

2.1 Client

Insert your content here.

2.2 Customer

Insert your content here.

2.3 Other Stakeholders

Insert your content here.

2.4 Hands-On Users of the Project

Insert your content here.

2.5 Personas

Insert your content here.

2.6 Priorities Assigned to Users

Insert your content here.

2.7 User Participation

Insert your content here.

2.8 Maintenance Users and Service Technicians

Insert your content here.

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

Insert your content here.

5.2 Business Rules

Insert your content here.

5.3 Assumptions

Insert your content here.

6 The Scope of the Work

6.1 The Current Situation

Insert your content here.

6.2 The Context of the Work

Insert your content here.

6.3 Work Partitioning

Insert your content here.

6.4 Specifying a Business Use Case (BUC)

Insert your content here.

7 Business Data Model and Data Dictionary

7.1 Business Data Model

Insert your content here.

7.2 Data Dictionary

Insert your content here.

8 The Scope of the Product

8.1 Product Boundary

Insert your content here.

8.2 Product Use Case Table

Insert your content here.

8.3 Individual Product Use Cases (PUC's)

Insert your content here.

9 Functional Requirements

9.1 Functional Requirements

Insert your content here.

10 Look and Feel Requirements

10.1 Appearance Requirements

Insert your content here.

10.2 Style Requirements

Insert your content here.

11 Usability and Humanity Requirements

11.1 Ease of Use Requirements

Insert your content here.

11.2 Personalization and Internationalization Requirements

Insert your content here.

11.3 Learning Requirements

Insert your content here.

11.4 Understandability and Politeness Requirements

Insert your content here.

11.5 Accessibility Requirements

Insert your content here.

12 Performance Requirements

12.1 Speed and Latency Requirements

Insert your content here.

12.2 Safety-Critical Requirements

Insert your content here.

12.3 Precision or Accuracy Requirements

Insert your content here.

12.4 Robustness or Fault-Tolerance Requirements

Insert your content here.

12.5 Capacity Requirements

Insert your content here.

12.6 Scalability or Extensibility Requirements

Insert your content here.

12.7 Longevity Requirements

Insert your content here.

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` $\rightarrow$ first working prototype for the Revision 0 Demonstration
- `v0.2.0` $\rightarrow$ adds a new feature such as exporting the table to a CSV file
- `v0.2.1` $\rightarrow$ fixes a small bug in the export feature (patch)
- `v1.0.0` $\rightarrow$ 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

The following unresolved items could materially affect the design, deployment, or operation of the NNL Assessment Tool. The issues in Table 2 will be tracked until closure to reduce delivery risk and surprises in later phases.

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.

These products may serve as inspiration or integration points (e.g., Excel import/export) but cannot replace the custom automation and analysis required by the NNL Assessment Tool.

Table 2: Open issues to track and resolve

Issue #	Summary	Cross-Reference	Stakeholders	Action Required	Status
OI-01	Finalize hosting environment for the tool (e.g., internal server or university-managed cloud).	Operational and Environmental Requirements	CAS Supervisor, Infrastructure Team	Confirm approved infrastructure with IT	Pending
OI-02	Confirm the final list of neural network libraries to include in the tool's database (based on domain-expert review).	Scope of the Work	Research Subteam, Domain Expert	Schedule and complete review with domain expert	Pending
OI-03	Select user interface (UI) framework and visualization library (e.g., React with Chart.js vs. D3).	Functional Requirements	Development Team	Evaluate options and record decision	In progress
OI-04	Decide Excel integration	Functional Requirements	Research Subteam, Develop-	Define use cases and confirm	Pending

Table 3: 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.

19.2 Reusable Components

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

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.

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.

Table 4: 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.

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.

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.

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 8.

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.

25 Ideas for Solution

The following is the development team's idea for the user interface of the tool. Figure 1 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 1.
The left sidebar will contain all the domains, with indications on whether

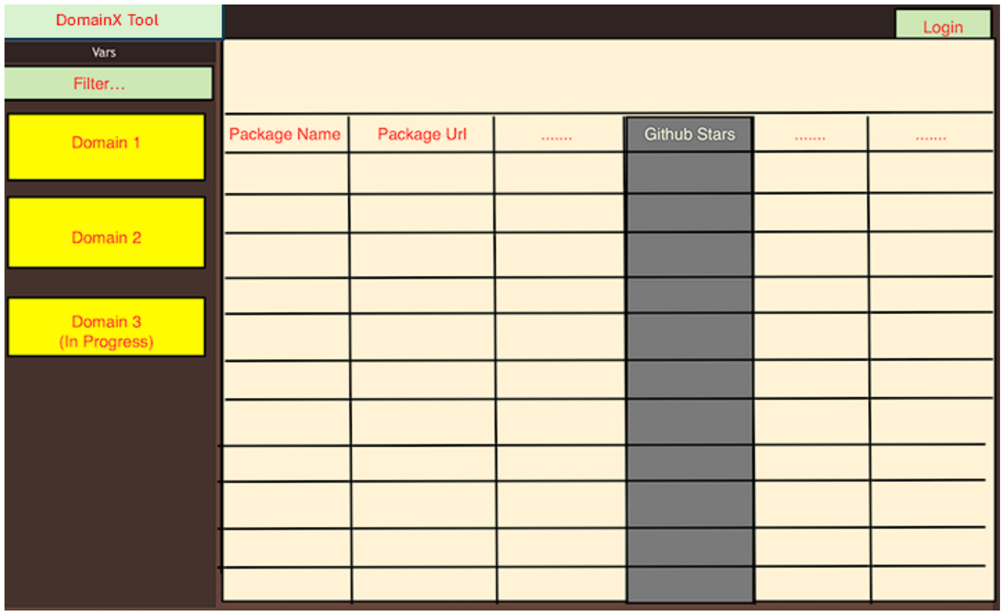

Figure 1: DomainX UI, inspired by Octave Online

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?

Table 5: Products That Can Be Copied or Adapted

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

Table 6: Project Planning Phases

Phase	**Description**	**Key Activities**
Initiation	Define problem, scope, and objectives	Document review, team formation, feasibility assessment
Requirements	Gather and formalize system requirements	Stakeholder interviews, drafting of SRS and Hazard Analysis
Design	Establish system architecture and interfaces	UI mockups, data flow diagrams, schema design
Implementation	Build core tool functionality	Develop modules, integrate APIs, implement Excel import/expor
Validation & Verification	Test and evaluate	Unit and integration testing, review sessions, issue resolution

Table 7: Development Phase Plan

Phase Name	Benefit to User	Operational Date	Operating Components	Functional Requirements	N F R q m
Requirements & Analysis	Clarifies system objectives and constraints	Week 1–6	GitHub, LaTeX	Requirements documentation	A cl it
Design	Defines architecture and interfaces	Week 10–16	UML tools, wireframing software	UI and backend design	M
Implementation	Delivers core product functionality	Week 16–19	IDE, databases, APIs	Tool modules, automation scripts	R us ab it
Testing & Validation	Ensures product meets all criteria	Week 17–22	Test suites, CI/CD	Verification of features	P
Deployment	Provides accessible	Week 22–26	Hosting platform, docu-	Hosted application	Se ac ce si bi

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 8: Price estimates and total costs for two AWS services.