Capstone Project Proposal Part II: Information Manager

CST 499 Computer Science Capstone

Julie Asato, Matthew Robertson, and Raymond Talmage

April 26, 2020

Executive Summary

Information and data in all its varying forms are an integral part of contemporary life. Whether you are a student, working, or simply managing your daily life; you undoubtedly expect to recall and utilize a variety of unstructured information down the line. To combat becoming overwhelmed with incoming material, people devise their own strategies of organization; whether it be through a digital medium or pen and paper. Currently, the typical software solution involves a prose-oriented approach. These can work satisfactorily for simple knowledge bases and on topics where you already have an adequate understanding of the domain. Yet if you are managing information for a complex, unfamiliar topic, they can fall short. On the other side of the spectrum, qualitative analysis software tools exist for information management professionals, but these tools have a significant barrier for entry for the average consumer. Not everyone has the skills, time, or money needed to use those products effectively; though a great many people would benefit substantially if they could.

To bridge the gap between the readily available and accessible consumer solutions and the hefty enterprise solutions, we have proposed a simple statement-oriented information manager. We plan to create an information manager that assists students and professionals with organizing their information in ways that are simple and effective without the difficulties associated with using professional-grade software.

Table of Contents

Part I

Introduction & Background

Project name and description

The project is currently unnamed. It is for anyone who has unstructured information they wish to manage. They may be students, working professionals or hobbyists focusing on a knowledge domain. Frequently these groups receive their information in loose unstructured formats like spoken word, in sections of prose, as outlines and in lists. Our project is designed to reform such information to be effortlessly managed and accessed; as the ease and swiftness of one's access to information are linked to overall productivity. As a student or professional, if you invest significant time in seeking a specific bit of information instead of working with it, you can miss deadlines—or at least have less time for more enjoyable endeavors.

Problem and/or issue in technology

Whether you are a student, working or simply managing your daily life; you undoubtedly expect to recall and utilize a variety of unstructured information down the line. To combat becoming overwhelmed with incoming material, people devise their own strategies of organization; whether it be through a digital medium or pen and paper. Currently, the typical software solution involves a prose-oriented approach. These can work satisfactorily for simple knowledge bases and on topics where you already have an adequate understanding of the domain. Yet if you are managing information for a complex, unfamiliar topic, they can fall short.

On the other side of the spectrum, specialized software tools exist for information management professionals, but these tools have a significant barrier for entry for the average consumer. Not everyone has the skills, time or money needed to use those products effectively; though countless people would benefit substantially if they could.

Solution to the problem and/or issue in technology

To bridge the gap between the readily available and accessible consumer solutions and the hefty enterprise solutions, we have proposed a simple statement-oriented information manager. In this project, the user would input statements and subsequently view/filter those statements. To preclude issues arising from prose-oriented approaches, the user enters the information as a statement, made in the form: Subject | Relationship | Object—which is simple to understand and offers a substantial power for searching and filtering. After the user has entered their data, they may search in the view pane of the application. When filtering the statements in the view, the user may enter one or multiple terms. In the multi-search case, intermediate statements are shown if they are part of a transitive relationship.

Example—search for: car, traction

car | has | tires

tire | provides | traction

Searching through notes made in this form can yield more concise and relevant results than manually searching and scanning through an entire document.

Evidence that the proposed project is needed

Storing, managing and searching through information has been vital well before computer systems were tasked to aid in the process. Now, many tools and resources exist for that purpose. Our information manager aims to address the shortcomings of what is currently available, while avoiding adding unnecessary complexity that professional level applications have.

Two of the most common prose-oriented approaches are personal wikis and store-everything notebooks e.g. Microsoft OneNote, Apple Notes or Evernote. Such software allows you to search for some text or a topic and read an excerpt that may or may not contain the sought material. This can quickly devolve into a time-consuming series of searches and skimming the text. Additionally, when adding to these information managers it is easy to introduce duplicate or even conflicting details. To prevent that, the stored content must be constantly reviewed and revised; a time-consuming process.

Contrary to these accessible but simple solutions, there are specialized tools within industry—programs like ATLAS.ti, QSR NVivo and Stanford Protégé. These can fit the information management use-case nicely, but they have some disadvantages. They all have steep learning curves, both in their conceptual models and in their application usage. Additionally, the dedicated qualitative analysis tools are expensive and their feature-sets are beyond what is needed for this project's basic use-case.

Project Goals and Objectives

Goals	Objectives
● Quickly take notes and find relative information in a personal application. ● Address the shortcomings of common, affordable software without introducing the complexities of professional-grade software.	● Create a input line for data entry ● Add a corresponding view pane to display entered statements ● Implement search functionality for the view pane to filter and show relevant information to the user ● Research and setup an embedded graph database for data storage and retrieval
● Empower users on most common platforms: Apple's macOS, Linux distributions with GUI and Microsoft Windows.	● Find cross-platform compatible GUI frameworks

Environmental Scan/Literature Review

Prose-oriented approaches

Evernote is a comprehensive store-everything notebook. It touts an impressive feature set including optical character recognition, cloud storage and syncing across multiple devices. Evernote has three pricing tiers: a free basic option, a premium offering at $7.99 per month and business level at $14.99 per user per month. Of interest to this project, Evernote provides several search and filter functionalities. These include the typical search phrase and exact match features. Additionally, Evernote allows for searching against a robust set of metadata fields; including notebook, tags, timeframe and even geo-location information from when the note was created (Evernote Corporation, 2018).

While these searching capabilities are impressive, they detract the user from the task at hand. In lieu of directly retrieving the sought information, users have to spend time remembering when the note was written and which tags they put on it. Also, if those notes each have many tags, then instead of not retrieving any results, an overwhelming number are returned.

Professional Qualitative Analysis Tools

ATLAS.ti is an enterprise level qualitative analysis tool. According to the official quick tour document: "[ATLAS.ti] lets you extract, categorize and interlink data segments from a large variety and volume of source documents. Based on your analysis, the software supports you in discovering patterns and testing hypotheses. With numerous output options and collaboration tools, your analysis is easily accessible to yourself and others" (Friese, 2019). Being a professional tool, ATLAS.ti is expensive, costing $1,840.00 for a single non-commercial license (cleverbridge, Inc., 2020).

As an application, much of ATLAS.ti's focus is source documents and maintaining their fidelity. This is critical when cross-checking analysis against sources to verify that a set of conclusions are sound. However, to accommodate that key facet of its workflow comes a difficult learning curve and major time investment in operating the application. In our project we assume the user is the de-facto source of information and they can invent facts and modify them as they see fit. This reduces the learning curve and usage overhead for users significantly.

Stanford Protégé is a free, open-source ontology development environment funded by sustaining grants from the National Institutes of Health. It began as a way to structure electronic knowledge bases and design better ones (Musen, 2015). It is presently used for managing terminology, visualizing relationships between terms (Rubin et al., 2007).

To appreciate how Protégé relates to our project we need to understand what an ontology is. Principally, ontologies are lists of entities with specific attributes and have related terms linked together thereby constituting the model. In Protégé, the terminology is a bit different. Ontologies are called frames. The entities within frames are called classes. The attributes are slots and facets—an additional concept—represents characteristics of the slots. A knowledge base in Protégé includes frames and their relationships (Rubin et al., 2007). To illustrate, suppose there is a dog class. That class has a brown fur slot as well as facets of thin fur, soft fur and clean fur. Then, a knowledge base would be multiple dogs with slots and facets for those slots.

Protégé is an excellent tool for exploring knowledge domains and examining their completeness and cogency. Our project, while it uses structures similar to ontologies, it does not need to evaluate the cogency of the dataset. Moreover, the conceptual model for our project is straightforward, so its learning curve is orders of magnitude lower than Protégé's three-day Short Course for beginners (Stanford University, 2016).

Stakeholders and Community

Note-takers

Our project would be incredibly useful to those in environments where they are expected to consume and digest large amounts of unstructured information—this includes students and professionals that routinely deal with unstructured information. Although these would likely comprise the majority of our clients, this tool could gain broader use beyond note-takers. Put simply: our end-user is anyone with access to a computer and info they wish to store and effectively search simple pieces of knowledge on.

These people will gain much needed efficiency in both creating and searching their notes. Additionally, they should be able to review a subject more effectively by employing the filter capabilities of the application. They should both find and learn from their notes more easily, quickly and thoroughly without having to continuously groom their older notes.

Risks for potential users include time invested learning the application; we expect most will be comfortable using it after thirty minutes. As with most note-taking applications, many users will want to migrate existing knowledge bases into this system. Since we will not develop an automated method importing notes from other systems, this process will have to be done manually; which could take lots of time. Furthermore, the system is currently designed to only handle Subject | Relationship | Object triples. If the user requires more complex statements they will be foiled. Last, as with any new software project, it could contain bugs that result in loss of valuable data; Beware!

Developers of the application

The developers will gain knowledge in product development and software engineering. In the Agile software development methodology, the developers will work with customers to build a comprehensive application. Application developers will additionally gain knowledge in project management tools like GitHub to assign work, prioritize features and perform code reviews. They will also gain knowledge in transforming innovative ideas into real-life products by building GUI, an application core and a graph database.

Approach and Methodology

- Define Minimum Viable Product (MVP).
    - Research comparable software to inform feature-set.
    - Create rough mockup of GUI.
    - Create rough design of data model.
- Research applicable technologies to implement GUI, database and application core.
    - Research JavaFX and alternatives for GUI creation.
    - Research Neo4j graph database and its alternatives.
- Use Agile development to create MVP.
    - Plan iterations in weekly meetings according to most valued features.
    - Use GitHub for code reviews.
    - Meet with potential users to elicit desirable enhancements.
    - Track tasks and issues in GitHub.
- Time permitting develop features beyond MVP, using the same methodology as above.

Part II

Ethical Considerations

For this project, there are no plans to conduct any human subject research. Its end product will be a standalone application running on an individual's personal computer. Accordingly, the application's users are its sole evaluators; they will have to assess the fitness of this application against potential issues they may face when operating and administering the application. We make no guarantees of the application's security or its ability to preserve their data—it could be

regarded as a developer's ethical responsibility to ensure application security and data consistency in the products they create.

During development of the project, there are a few potential avenues for ethical lapses. For the developers, a fair division of labor should be pursued so all participants can enjoy a satisfying learning experience. For the application testers, reasonable accommodations should be made to ensure they can follow and complete their testing tasks. Furthermore, it is possible testers will use this software with sensitive real-world information; they need to be advised on how to secure that information.

After the application has been deployed, some ethical concerns may trigger. However, those are either systemic—beyond this project's capacity to address—or could be ameliorated with future enhancements. Clearly, those without access to computers that can run the application would be unable to benefit from this project. For example, schools increasingly provide students with Google Chromebooks instead of full-blown laptops to minimize cost; those can't execute this application. In terms of accessibility, this application is not designed with visual impairment in mind and will not work well with a screen reader. Additionally, it may be completely unsuitable for work with some languages; only English will be used for the initial implementation. Many of those issues can be addressed with further development.

Legal Considerations

Copyrights, Patents and Permissions

This will be an independent, standalone project. It will exclusively consist of original work from the developers and libraries/assets with licenses that clearly allow the project to incorporate or link them. We plan to check every library used in the project for licenses,

copyrights and other dependencies both manually and with a program like ScanCode—a command line tool that helps identify and locate licenses (NexB/scancode-toolkit, 2015/2020). If the licensing terms for a given asset or library are incompatible with this project, an alternative will be used or created. Additionally, the development team will check assets against the United States Copyright Office Online System (at https://cocatalog.loc.gov). When naming the project, we will ensure it has not been trademarked by consulting the United States Patent and Trademark Office Website.

Project License and Other Legal Concerns

The application for this project will come with its own license, likely Apache 2.0. It will notably include disclaimers of warranty and limitations of liability. Other than that, the license will permit free usage and modification by other parties. Despite having license protections, it is essential we work to ensure user's sensitive data is protected, and its integrity is maintained.

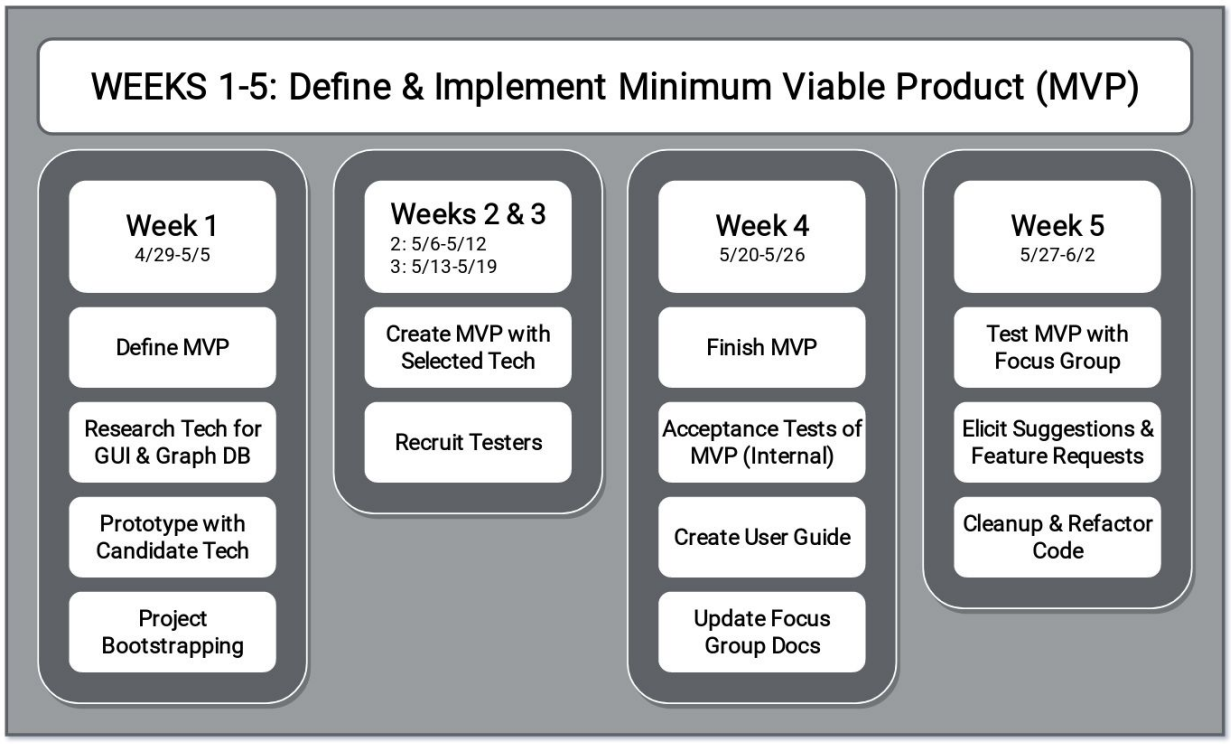
Since this is a desktop-only application, data is never communicated to any server, so that is a security aspect this project will not need to address. Since this is a desktop program, the user's operating-system login should be sufficient to secure their data. As a result, no encryption or authorization is needed in the application.

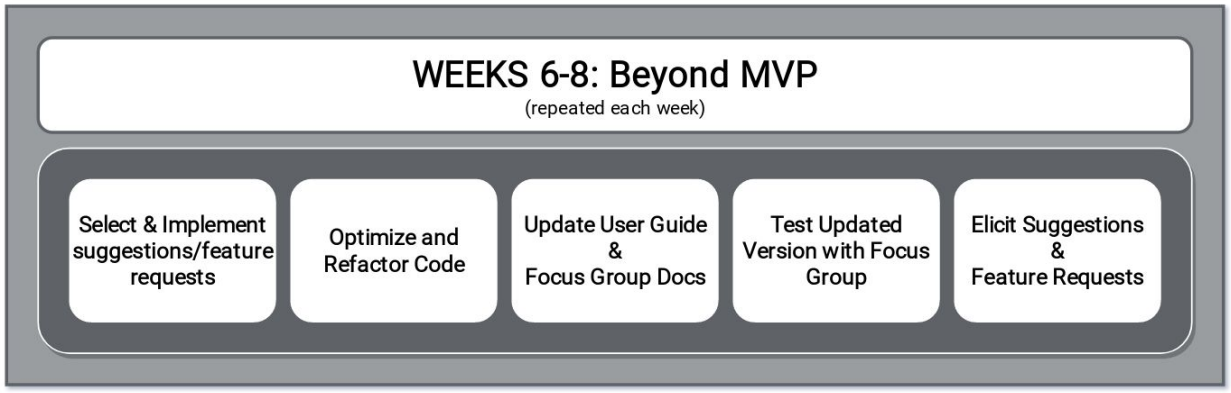
To safeguard data integrity, we will use various levels of software testing to verify that software defects are not the source of mangled user data. Likewise, we will provide extensive application logging, validate application data and ensure user data is backed up mid-transaction, if possible (Managing data integrity, n.d.).

Part III

Project Scope

Timeline





Budget/Resources Needed

This project has no budgetary demands. It will be implemented using only open source libraries, which are available at no cost. We will not purchase any software development tools since we have those tools needed to complete the project. The resulting application is designed to run standalone on a desktop computer, so we will not buy any domain names or web hosting services.

As this is a software-only solution, no hardware will need to be procured to aid in its realization. Every step of the project development can be completed with the materials and equipment already available to the team members. This includes: computers for development, writing/printing material, and internet communication equipment. Likewise, software testers shall possess the necessary hardware—a computer capable of executing the application.

Milestones

- Phase 1: Project Initiation (week 1)
    - Define Minimum Viable Product (MVP).
    - Evaluate and select technologies using prototypes.
    - Project Setup: Create Github repositories & Google Forms for Focus Group polling
- Phase 2: Implement MVP (weeks 2-4)
    - Iterate on the required feature set to complete the minimum viable product with a focus on testing with good code coverage
- Phase 3: Focus Group Testing (week 5)

    - Coordinate with Focus Group testers to validate if the MVP meets their typical use cases.
- Phase 4: Iterative Improvement (weeks 6-8)
    - Iterate on a new feature set informed by Focus Group testing and feedback.

Risks and Dependencies

The application must have both a GUI and a backing Graph Database to work. If an adequate framework for either is never found, this project will fail. Further, if either is difficult to understand or work with, then the project will be delayed or impossible to complete. One goal of this project is to provide a solution that works on various desktop operating systems. If no suitable GUI framework is available, support for some OSes will be jeopardized.

The graph database component is crucial for the success of the project. Since the application should be standalone, ideally it will include an embedded database; sparing users from installing and configuring that database component. Such embedding may be impossible. In that case, we will produce an automated installation of the graph database; potentially impacting the timeline.

Another fundamental goal for this application is it serves as a replacement for prose-oriented information managers while remaining easy to understand. Conceivably, the minimized scope of features for the initial product renders the application insufficient for some critical use cases. Namely, a lack of multi-object-relationship phrasings may make some concepts inexpressible within the application. In that case, those vital features will be incorporated—if time permits.

Final Deliverables

Upon project completion, a standalone desktop application will be delivered along with a supporting user guide. The guide will include a brief background and overview, a step-by-step usage tutorial and a frequently asked questions section. The application's source code will be available in a public git repository. Also, documents outlining focus-group feedback, survey summaries and planned enhancements will be released. A copy of the focus group survey will be included to clarify the evaluation criteria. Last, an informational video will be produced for the Computer Science Capstone Festival explaining the project's purpose and demonstrating its functionality.

This project has no official clients; we will have a focus group instead. The focus group will approve the application chiefly by saying they plan to use it instead of whatever they were using before. In order for that to be possible users need to be able to enter, view, filter information within the application and the filter needs to produce their anticipated results.

Usability Testing/Evaluation

The product's usability will be evaluated on two major levels: that the application works as designed and that the application meets its core use-case. If the project is successful, the testers will easily use the product: entering, viewing, filtering, and reviewing the results; and most importantly, they prefer this application over a prose-oriented one.

First, for technical correctness we will ensure the application operates as designed. That will be achieved with a combination of unit/integration testing and code coverage. Using an agile development approach, each developer will make certain that their code is sufficiently tested before integration into the mainline codebase.

Second and critically, we evaluate whether the application meets its core use-case; being a powerful, yet simple to use, replacement for prose-oriented information managers. This will be proven by actual usage and assessment by the developers and a focus group. This testing will be done by the development team continuously throughout phases two through four and joined by the focus group after fulfilment of MVP. Testers of various skill levels that represent the application's target audience will be recruited by the team to participate in the focus group.

Beyond meeting its core use cases, the application needs to be generally functional, bug-free and it needs to work as a typical user would expect. The testing plan by phase is outlined below. If there are problems found with usability, user feedback will be used to prompt features as well as direct improvements to the product's documentation.

Testing by Phase

- Phase 1: Project Initiation (week 1)
    - Not applicable.
- Phase 2: Implement MVP (weeks 2-4)
    - The graph database should be embedded within the application.
    - The GUI should be able to connect to the back-end and database.
    - The user should be able to enter data.
    - The user should be able to search their stored data terms and have the expected items displayed in a view panel.
- Phase 3: Focus Group Testing (week 5)
    - Assess whether MVP meets Focus Group's typical use-cases.
    - Focus Group testers should be able to complete the tasks from Phase 2.

- Phase 4: Iterative Improvement (weeks 6-8)
    - Iterate on a new feature set informed by Focus Group testing.
    - The tests depend on selected features.

Team Members

The team members for this project are Julie Asato, Matthew Robertson and Raymond Talmage. Tasks for this project will vary along with the timeline and be divided among team members according to the following outline. During the research and prototyping phase in the first week, all team members will define scope for the project's Minimal Viable Product (MVP). Also, Julie will research and prototype with the candidate GUI frameworks; Matthew will do the same for the Graph Databases. Raymond will bootstrap the project, including set up for the repositories, agile development tools and focus group surveys. In the MVP implementation phase—weeks 2-4—tasks will be divvied among team members using an agile methodology. Additionally each team member will be tasked with recruiting three or four testers for the focus group. In the fourth week, preparation work is needed for focus group testing. During that week Raymond will create the User Guide and update the focus group documents. Julie and Matthew will work on acceptance testing. The fifth week asserts the Focus Group Testing and Refactoring Phase. All team members will support the testing and feedback efforts and take on tasks for improving the code quality and efficiency. From the sixth week onward, team members will continue to iterate on the application making selected improvements guided by focus group assays and following an agile approach. In each iteration we will optimize and cleanup the codebase, update the various documentation, assist focus group testers and gather recommendations.

References

cleverbridge, Inc. (2020). ATLAS.ti Commercial Licenses. https://atlasti.cleverbridge.com/74/catalog/category.6302/language.en/currency.USD

Evernote Corporation. (2020). Compare plans and get started for free. Evernote. https://evernote.com/compare-plans

Evernote Corporation. (2018). How to use Evernote's advanced search syntax. Evernote Help & Learning. http://help.evernote.com/hc/en-us/articles/208313828

Friese, S. (2019). ATLAS.ti 8 Windows – Quick Tour. http://downloads.atlasti.com/docs/quicktour/QuickTour_a8_win_en.pdf

Neo4j, Inc. (2020). What Is a Graph Database and Property Graph | Neo4j. Neo4j Graph Database Platform. https://neo4j.com/developer/graph-database

Managing data integrity. (n.d.). Retrieved April 13, 2020, from https://www1.udel.edu/security/data/integrity.html

Musen, M. A. (2015). The Protégé Project: A Look Back and a Look Forward. AI Matters, 1(4), 4–12. https://doi.org/10.1145/2757001.2757003

NexB/scancode-toolkit. (2020). [C]. nexB. https://github.com/nexB/scancode-toolkit (Original work published 2015)

Rubin, D. L., Noy, N. F., & Musen, M. A. (2007). Protégé: A tool for managing and using terminology in radiology applications. Journal of Digital Imaging, 20 Suppl 1, 34–46. https://doi.org/10.1007/s10278-007-9065-0

Stanford University. (2016). Protégé Short Courses. https://protege.stanford.edu/short-courses.php