CSCI 370 Final Report

Longeviteam

Stone Amsbaugh
May Koch
Jake Staples

Revised 6/13/2025

CSCI 370 Summer 2025

Dr. Rob Thompson

Table 1: Revision history

Revision	Date	Comments
New	5/16/2025	Created document, wrote sections I-V
Rev – 2	5/23/2025	System Architecture (Section VI)
Rev – 3	5/29/2025	Software Quality & Ethical Considerations (Sections VII – VIII)
Rev – 4	6/7/2025	Draft of Final Report
Rev – 5	6/13/2025	Cleaning up and finalization

Table of Contents

I. Introduction

LEVL is an emerging startup that is targeting the biology of aging to create products that help individuals live longer while maintaining a higher standard of health. Their primary product is currently supplements that boost longevity and help customers with certain goals, such as sleep, cognition and metabolism. At the beginning of this project, they sought to create an application that delivers personalized 'protocols' to users. Protocols consist of a list of actions that users can follow to improve their longevity, such as supplements to take or habits to practice.

Additionally, they proposed the project of creating the 'matching algorithm' component of the application, which would compute these protocols given individual conditions, goals, determination and other data. This algorithm would require creating a knowledge graph of biomedical knowledge pertaining to longevity, delivering customized protocols to users, and an application with which stakeholders can interact with the algorithm.

II. Functional Requirements

- Create an accessible, easily updatable and expandable database of:
    - Modalities (interventions with the potential to improve outcomes)
    - Biomarkers, outcomes and other entities represented in biomedical literature
    - Client information (age, goals, pre-existing conditions, etc.)
    - Relationships between modalities, biomarkers, conditions, and other entities found in biomedical studies.
    - Scientific sources on the impacts and conditions of modalities
- Implement an algorithm to create a protocol (collection of modalities) based on user information and preferences.
- Implement an iterative loop to allow clients to give feedback on protocols and adjust accordingly.
- Create an export system to view protocols online, via email, or a downloaded PDF file.
- Create an interface with which these features can all be utilized.
- Expand the project, as time permits, to include:
    - Influencer Protocols
    - Display of Scientific Sources
    - Natural Language Modality Explanation
    - Data Visualization

III. Non-Functional Requirements

The final product must follow the following principles to ensure the viability of the product going forward:

- All information must be updatable by researchers without knowledge of code or codebase.
- Information must come from research rooted in scientific literature.
- Thorough documentation and clean architecture are crucial.

IV. Risks

The risks of this software are summarized below, along with their severities and mitigations.

Risk	Severity	Mitigation
Algorithm recommends modalities with harmful effects	5	Modalities should be well studied, and with well-known and generally minimized adverse effects. Perform sanity-check on protocols for known side effects and users' conditions. Warn users to place a limited amount of trust in the algorithm, to perform their own research and consult professionals before making potentially risky decisions.
Algorithm malfunctions in general	4	Ability to have the impacts of modalities to be explained in natural language to understand the output protocols.
Algorithm recommends modalities that are not effective enough	3	Disclaimer that the application is not a substitute for professional treatment and more research is still being conducted.
User interface crashes	2	Clean architecture and documentation to get the application back up as soon as possible.
Incorrect research is entered into the database/knowledge graph	4	All information that the knowledge graph and algorithm are founded on are modifiable without editing the code itself. Identifying this is aided by the natural language explanations and supporting literature features.
Personal information is susceptible to attacks	5	The algorithm will only work with de-identified user information.

Table 1: Risks and Mitigation

There were additionally many skills risks for this project. Only one team member had taken a databases course, and only one team member regularly used Python as their main programming language at the time. None of the team members had experience making web applications, creating databases, and were especially unfamiliar with frontier models and large language models. None of the team members were familiar with the biology surrounding aging. These would be mitigated via intensive research of necessary skills and domains, and by regularly communicating with the client to manage expectations.

V. Definition of Done

The minimum useful feature set for this project was the knowledge graph of entities/modalities and their relationships, the algorithm for identifying protocols, and a simple user interface with which to interact with it. In order for this knowledge graph to exist, we needed to establish methods for efficiently adding to it and editing it.

Before accepting the software, the client should certainly first try out our demo from a user's standpoint. They should experiment with a variety of user profiles and goals and observe the results. They should analyze the results by viewing the explanations and supporting literature for the modalities in the resulting protocols and confirm that this aligns with both their vision and their knowledge of the field. They should also test the software from a researcher's standpoint and add or update research in the database and observe its effects on the knowledge graph.

The product would be delivered by transferring the ownership of the GitHub repository for the project, as well as sending any other relevant files, and most importantly, documentation. Documentation would be provided with detailed instructions on how to run, deploy, and interact with every bit of work contained in the project. Should they choose to use their own PostgreSQL server, then we would use bulk loading to copy the contents of our database to theirs as a part of delivery.

VI. System Architecture

Our team created the following design for our system, based on our client's requirements and our current understanding of our capabilities with regards to the project.

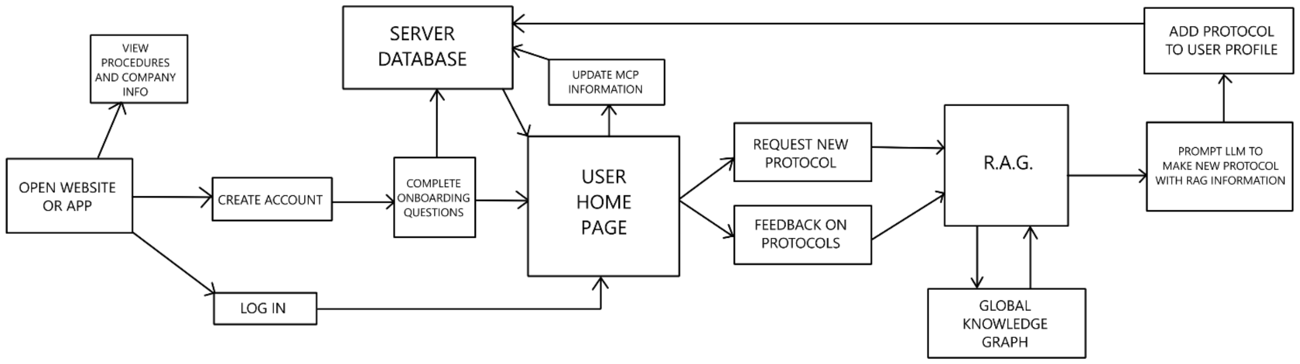


Figure 1: Diagram of User Input and Server Flow

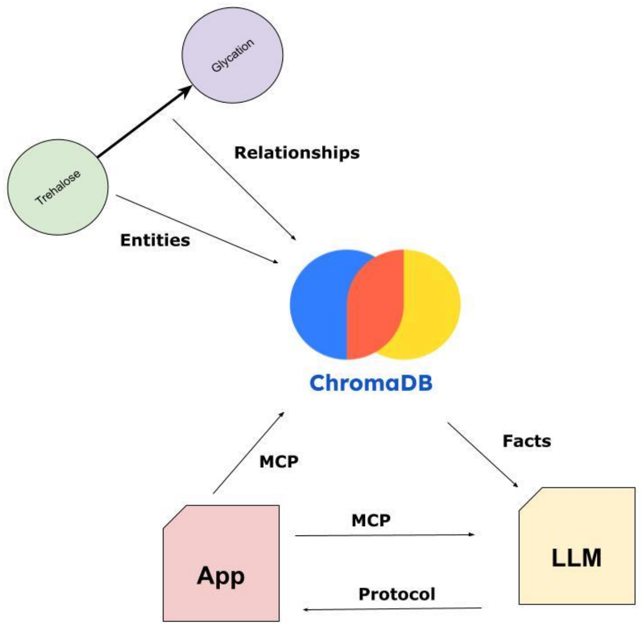


Figure 2: Diagram of RAG Process

The RAG (Retrieval Augmented Generation) has its own architecture to enhance the performance of the main functionality of the app; protocol generation. Routinely, all entities and relationships in the knowledge graph, as well as all of their attributes, are embedded as vectors (referred to as ‘facts’) and stored in a database using Chroma. When a user requests a protocol, the application sends Chroma the user’s MCP, and using these personalization configurations, Chroma can compute the most relevant facts from their embeddings. It returns the fifty (admin-specified value) most relevant facts to an LLM, which is also given the MCP. The LLM is tasked with computing a protocol using only the information provided to it, and it returns the protocol in JSON format back to the application to be read.

At the time of writing, several technical issues had been identified, and are listed here:

- Originally, we planned on hosting our relational database on AWS, however technical issues had arisen with the costs of this hosting, and an alternative relational database hosting platform needed to be connected.
- We observed issues with the entity resolution phase of the knowledge graph creation pipeline, as the LLM could assign slightly different names to different nodes that in reality refer to the same entity. We needed to look into other entity resolution processes that may be run.

VII. Software Test and Quality

The following are tests have been completed in order to ensure a high-quality product:

Graph Database Functionality	Ensure that all edits to the database made on the site are reflected in the Neo4J browser, and that the state maintained in Neo4j is always represented on the “Explore” page.
Literature Contribution	We read scientific literature to get an understanding of the claims it is making regarding biomedical information and entities as they relate to our project. Then, upload the literature, and ensure that the identified entities and relations match our understanding of the literature.
Protocol Generation	Generate many protocols for different user profiles, ensure the generated protocols adapt to match the users needs. Check that they seem safe with the client.
User Interface	We can test the user interface by simulating as many possible sequences of actions a user can take, and observing whether the application's behavior is expected and clean. This means accessing all pages from any kind of context and using them in any way available. Any bug, crash, or unexpected behavior/layout indicates a failure of the test.
Protocol Export	Export the protocol and ensure that the email consistently arrives and is in neat formatting.
LLM Integration	Ensure that throughout many uses of LLMs from any context, outputs are all in strict, valid JSON format.

VIII. Project Ethical Considerations

When working on this project, we were focused on the following potential ethical issues:
(The following codes are taken from the ACM Code of Ethics)

- USE OF LLMs: The usage of an LLM brings plenty of potential for error in our design, because even the most advanced models are not perfect. We needed to focus on ensuring that our use of the LLM. does not bring any unnecessary risk of error. Also, relying on artificial intelligence, especially on a larger scale, is energy intensive, so we needed to be mindful of that as well.
    - 3.13. Be careful to use only accurate data derived by ethical and lawful means, and use it only in ways properly authorized.
    - 1.07. Consider issues of physical disabilities, allocation of resources, economic disadvantage and other factors that can diminish access to the benefits of software.
- CUSTOMER PRIVACY: Our design relies on keeping track of any relevant user information, and such information can be personal medical information that should be kept as secure as possible so as to not leak any sensitive user information.
    - 2.05. Keep private any confidential information gained in their professional work, where such confidentiality is consistent with the public interest and consistent with the law.
    - 3.14. Maintain the integrity of data, being sensitive to outdated or flawed occurrences.

- ERRONEOUS INFORMATION: The information that we are handling in this project has the potential to be false or misleading, so we needed to be mindful of the potential of storing erroneous information that could be harmful to our users if not properly maintained.
    - 3.13. Be careful to use only accurate data derived by ethical and lawful means, and use it only in ways properly authorized.

IX. Project Completion Status

At the end of this project, we have successfully completed the minimal definition of done outlined above. These minimally viable features have been completed with high quality and reliability. In addition, we were able to complete other features, such as the email export, a more interactive webapp, high levels of functionality for administrators and researchers to work within the application, and more features that interact with a large language model to make the app more immersive.

The core functionality of the app allows the user to login, fill out onboarding questions, and generate protocols for themselves. Along with path they can also export their protocol to an email, talk with AI about the items in their protocol, and expand their modalities to see more details.

Basic administrative functionality includes adding literature to be processed and inserted into the knowledge graph and editing what has been inserted. We completed comprehensive functionality for editing the graph from an administrative perspective, made the program compatible with text input as well as txt, pdf and JSON upload. We added the option to download JSON files to store and avoid future recomputing, and a confirmation screen so you can cancel operations where the LLM did not find accurate information. We added a maintenance page in order for them to perform general graph health operations and to operate an additionally requested feature of having modalities that are always included.

We set up pages for them to manage users and all account information, the JSON data stored for users personalization and protocols, and configure the sites secrets such as API keys and passwords.

We provided the client with helpful and thorough guides on how to set up and use the application for various purposes, as well as a guide to the code base and to future work to give any future workers a good starting point. We used many of their technical suggestions throughout the process, such as making use of RAG and MCPs, which are thorough technologies boosting the quality of our program.

The primary feature that was originally requested that we did not have time to complete was the feedback loop, where users can provide feedback with their protocols and get adaptive recommendations, and this is a primary point of future work.

We set up hosting for the site and it is now online and accessible to anyone to use, which makes it especially complete and convenient for our client.

X. Future Work

There were a number of functions that we were not able to implement in time for this assignment. We were initially tasked with creating a feedback loop for our protocol generator, where the user could give positive or negative feedback regarding their modalities, and the protocol generator would take that information and make changes to their protocol as necessary. Unfortunately, we realized that it was unlikely for us to implement this feature in time, however the client assured us that such a feature would be fine implemented another time. Another feature was exporting a user's modalities as a PDF file and allowing them to download their protocol from the webpage. This didn't seem like a tough function to implement; however, our team is unfamiliar with such a system, and we realized that we most likely could not add it before the end of the project timeline. Finally, there was the case of edge aggregation for our knowledge graph that would be too time consuming to manage before the end of the project unless we were to focus on only that, which we could not.

Additionally, there were a few functionalities that came up in meetings that were mostly ideas for the project moving forward outside of our hands. One such functionality was creating a daily or weekly schedule-like format for the user's protocol, detailing the time of day that each modality should be performed. This way the user could have an easier view of what modalities they should do on a specific day and at a specific time.

XI. Lessons Learned

We learned a lot of lessons throughout the development of this project. Some key examples are:

- Python being an extremely popular programming language works in our favor when designing new concepts. Throughout the project there have been various times where we wanted to implement a new idea into our code, and nine times out of ten the solution was to download a Python module that already did the work for us.
- Working with Large Language Models in your design can be both impressive and tedious at the same time. It took us a long time to properly implement artificial intelligence into our knowledge graph builder and our RAG retrieval system; however, once we got the LLM properly set up and operational, it works very well in a way that raw programming wouldn't be able to.
- The Django module for Python is an incredibly useful resource when designing a web interface from scratch. Even with nobody on our team being familiar with the module, it took a shockingly short amount of time to get the web server up and running on one of our systems.
- There is a significant cost associated with hosting servers and databases. One consideration when choosing a framework in the future is that having a python-based application requires all operations to be done server-side, which can be more expensive than performing operations client-side using JavaScript. Database costs also needed to be planned for and is a reason to store data efficiently.

XII. Acknowledgements

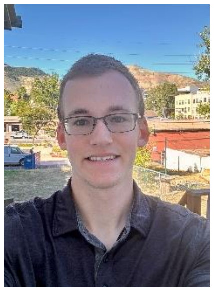
Our team would like to extend a great deal of gratitude to Kylen McClintock, Dan Callen and everyone else at LEVL that worked with us throughout the duration of this project. Our team appreciated the opportunity to gain insight not only into the professional world of software development work, but also into their research and outlooks on design.

Additionally, we would like to thank our advisor Rob Thompson for his continued guidance navigating the project and his technical advice, his assistance is reflected in the improved design and management skills of our team members.

XIII. Team Profile

Jake Staples
Developer

Jake is from Fountain, Colorado and is a Computer Science major with a focus in Computer Engineering. He is very passionate about the inner workings of a computer, digital and physical. He has other hobbies as well such as drawing and learning the piano.



Stone Amsbaugh
Developer and Client Liaison

Stone is from Dillon, Colorado and is a Computer Science major with a focus in Data Science. He loves to hike, is part of the Mines Speedcubing Club and actively competes and organizes cubing competitions for the local community.

May Koch
Developer and Advisor Liaison

May is from Houston, Texas. She is a Computer Science student with a focus in Computer Engineering. She is a teacher's assistant for one of Mines' C++ courses, plays a variety of instruments such as guitar, piano, and drums, and enjoys watching movies and playing video games in her spare time.

Appendix A – Key Terms

Include descriptions of technical terms, abbreviations and acronyms

Term	Definition
Protocol	*Collection of recommended modalities for a user to implement in their lifestyle.*
Modality	*Recommended action or intervention, such as a nutritional supplement, exercise, or some other habit or routine, intended to increase longevity in an individual.*
Knowledge Graph	*Network of entities that can be anything from lipids to genes to diseases, and how they relate to each other.*
RAG	*Retrieval Augmented Generation. A method of querying a LLM with both a prompt and providing it with a method of obtaining information from an external source. In our case, this source is the Knowledge Graph.*
MCP	*Model Context Protocol. A connection for user personalization data to be conveyed to an LLM, the context under which that model will operate. In our case, this is a lightweight JSON of this data that travels with all requests.*
Longevity	*The attainment of a longer and healthier life.*