4 Design

4.1 Design Context

4.1.1 Broader Context

Our project is primary focused on two communities: security researchers in academia and information security roles in industry. Although these communities will be directly affected by our project, the entire world will be indirectly affected as well. The increased efficiency of gathering cybersecurity knowledge will allow our target communities to better do their job and, increasing safety, security, privacy, and integrity in all software applications. As societies around the world grow more dependent on technology, our goal is to make the software they interact with safer.

Area	**Considerations**
Public health, safety, and welfare	All users of a technology companies' products are indirectly affected by our project due to the utility we provide information security staff. Helping these roles increases the public safety and welfare in their interaction with technology. It could also harm job opportunities of these positions, as an effective product would require less staff to research and fix problems. In the same way, researchers' use of our product will also improve the general populations' interaction with technology utilizing their discoveries.
Global, cultural, and social	Our project accurately reflects the values of our target cultural groups including security researchers in academia and information security staff in industry. Using extracted information to build a knowledge graph that can be queried is in line with practices to streamline cybersecurity information gathering.
Environmental	The environmental effects of this project are indirect. The software will be executed and hosted on servers that use a lot of electricity of unknown origin (renewable vs. nonrenewable). There is a potential impact of training ML models with GPU clusters in terms of energy usage, although our project would have to scale magnitudes larger for this to become a reasonable concern.
Economic	This project being successful will have an impact of increased productivity by information security roles in industry by getting them access to recent, condensed, and relevant information quicker. One pending consideration is if the project proves extremely useful what obligation we have to make it available to good actors in terms of cost.

4.1.2 Prior Work/Solutions

There have been research papers on the idea of using Knowledge Graphs to store information in the Cyber Security domain. One example is "TINKER: A framework for Open source Cyberthreat Intelligence". This research paper delves into creating a knowledge graph that is used primarily to "infer threat information from the [cybersecurity] text corpus". This differs from our project in that it attempts to strictly capture malware information from CTIs (Cyber Threat Intelligence). Our project is aimed more for researchers and industry professionals to be able to query a knowledge graph of many entity-types (companies, vulnerably, malware) that is consistently updating with new information automatically.

Some of the pros and cons of our target solution would be:

1. Pro: Web Interface with query input and data visualization
2. Pro: Updating periodically (e.g., every hour) with new cybersecurity information
3. Con: Information sourced from reputable cybersecurity blogs instead of CTIs.

Our project is not following previous work of any Senior Design project.

4.1.3 Technical Complexity

The design includes several components stored in Docker containers that collaborate to scrape website articles and construct the knowledge graph. These components include:

1. Web scraper and parser: uses frameworks to make a request for pages in the source list and get the raw HTML. Then uses programming organization principles to find and extract the useful information from the document. This requires structural knowledge of web pages and analysis of how to clean the pages into useable data for the following step.
2. Taxonomy definition and training Name-Entity Recognition Model: using the nltk and spacy libraries with Python along with our taxonomy definitions in order to perform natural language processing (NPL) and queries on the data. This requires scientific and mathematical knowledge of language processing and article tagging in order to create relationships for the graph.
3. Relationship extraction to construct the graph and store in a database: Using the relationship data created above, use computer science principles to build a knowledge graph that has meaningful relationships between the nodes and is in a readable format.
4. Running natural language queries on the completed knowledge graph: requires computer science and mathematical principles regarding AI and language comprehension that can query the graph with a search term and return meaningful results.
5. Displaying knowledge graph and queries on frontend for consumption: Includes web programming principles and standards in order to host a page in the browser that can render the graph and provide a search feature for the natural language queries. Will need to efficiently display the results and scale based on the graph or query size.

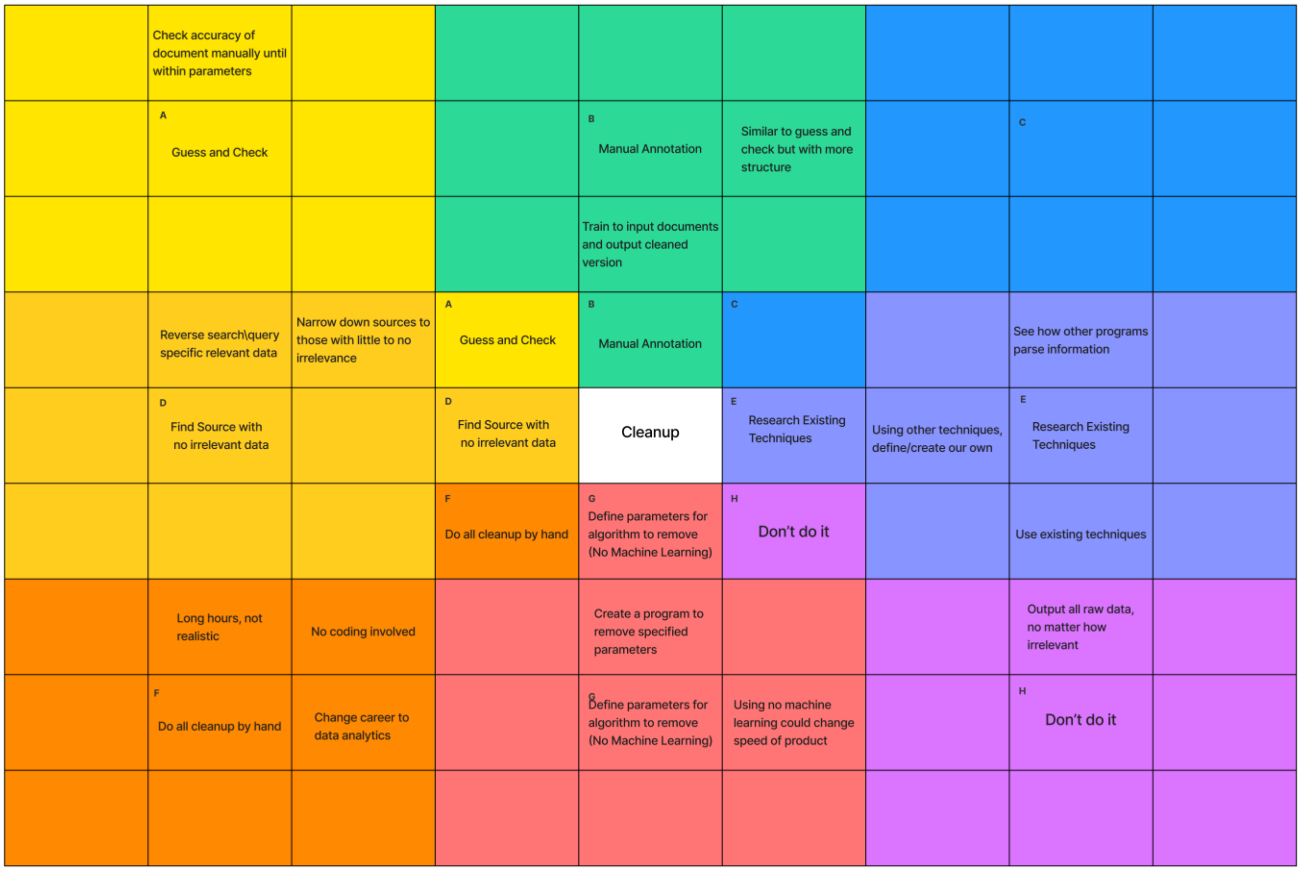
4.2 Design Exploration

4.2.1 Design Decisions

1. Potential Sources: The biggest decision in terms of potential sources we ran into were whe ther or not to include blogs as a source in our project as many cyber security professionals don't have or never needed to obtain a college degree so the information, they put out isn't in a well-structured or well-defined manner. Excluding these sources could be a problem later in the project because these cyber security blogs are rich with information.

2. Cleanup: The cleanup of the text from sources is for training our Name-Entity-Recognition Model which is a crucial part of our project. This decision is important because we need a well-formatted text to train the NER. Avoiding the best decision for our cleanup process would be detrimental to training the NER.
3. Taxonomy Definitions: Taxonomy is important in determining the scope of the project and which topics we should cover in the knowledge graph. We plan on utilizing the broader topics such as Vulnerabilities, Malware, and breaches but the decision comes down to including some of the smaller topics into our taxonomy like threat actors and phishing.

4.2.2 Ideation



	Check accuracy of document manually until within parameters							
	A Guess and Check			B Manual Annotation	Similar to guess and check but with more structure		C	
				Train to input documents and output cleaned version				
	Reverse search\query specific relevant data	Narrow down sources to those with little to no irrelevance	A Guess and Check	B Manual Annotation	C		See how other programs parse information	
	D Find Source with no irrelevant data		D Find Source with no irrelevant data	Cleanup	E Research Existing Techniques	Using other techniques, define/create our own	E Research Existing Techniques	
			F Do all cleanup by hand	G Define parameters for algorithm to remove (No Machine Learning)	H Don't do it		Use existing techniques	
	Long hours, not realistic	No coding involved		Create a program to remove specified parameters			Output all raw data, no matter how irrelevant	
	F Do all cleanup by hand	Change career to data analytics		G Define parameters for algorithm to remove (No Machine Learning)	Using no machine learning could change speed of product		H Don't do it	

For cleaning up documents to remove irrelevant information after they've been scraped and parsed, we considered several different options using the lotus blossom method for ideation.

- Guess and Check – Repeatedly attempt to clean up the document until the output meets certain criteria or is within certain parameters.
- Manual Annotation – Clean up a small set of documents by hand and then use that sample data to train a machine learning model to clean up documents.
- Find sources with no irrelevant data – Get information from sources that are known to only contain relevant information, or are in a structured format (e.g., JSON, XML, etc.) which can be queried for relevant information in a standard way.
- Do all cleanup by hand – Manually clean up all new articles every time one is added for as long as the project is running. Not a realistic solution.

- Define parameters for an algorithm – Figure out how to clean up each source and write a program to do it automatically. Program may need to be rewritten when sources change website design.
- Don’t do it – Output all data, even if it is irrelevant. The easiest “solution” to this problem, but may make later steps in the process more difficult since we have to work around all of the noise.

4.2.3 Decision-Making and Trade-Off

Cleanup Methods		Accuracy		Development Time		Development Complexity		Run Time	
Weights	Total	3		2		2		1	
Guess and Check	21	3	*9*	2	*4*	2	*4*	4	*4*
Find perfectly clean sources	26	4	*12*	4	*8*	1	*2*	4	*4*
By-Hand Cleanup	22	3	*9*	1	*2*	4	*8*	3	*3*
Manual parameters	23	3	*9*	3	*6*	2	*4*	4	*4*
Don't Clean up	22	1	*3*	4	*8*	4	*8*	3	*3*
Research Existing Techniques	26	4	*12*	2	*4*	3	*6*	4	*4*
Manual Annotation	25	3	*9*	2	*4*	4	*8*	4	*4*

For deciding which data cleanup method to use, we made a table like above. The different methods added to a weighted decision matrix. We chose four main factors: accuracy, development time, development complexity and run time. Accuracy represents how accurate we predict the method to be. A lower number indicates that the method might have a high chance of improperly cleaning the data, leaving mistakes and invalid data. Accuracy was chosen as the most important factor, as good data is important for building machine-learning algorithms. Next, was development time, which represents how long it will take to implement the cleanup method. A lower number indicates that it will take a longer time to implement. Next is development complexity, representing how difficult the method is to develop or use. Lastly, is run time, which represents how long the cleanup method will actually run when cleaning up data.