1.1 Requirements & Constraints

List all requirements for your project. Separate your requirements by type, which may include functional requirements (specification), resource requirements, physical requirements, aesthetic requirements, user experiential requirements, economic/market requirements, environmental requirements, UI requirements, and any others relevant to your project. When a requirement is also a quantitative constraint, either separate it into a list of constraints, or annotate at the end of requirement as "**(constraint)**." Ensure your requirements are realistic, specific, reflective or in support of user needs, and comprehensive.

Functional Requirements

- The product should use no less than 3 sources for cybersecurity information.
- The product should crawl sources for new information at least once a day.
- The product should extract information using HTML parsing, OCR, and/or API.
- The product should perform Information Extraction by using a trained model.
- The product should produce a knowledge graph containing cybersecurity entities and relations.
- The product should persist this knowledge graph using the graph database Neo4j.

Non-Functional Requirements

- Extensible by future developers by using interfaces and writing modular code.
- Code should have documentation on function definitions and up-to-date README.
- The crawl speed should be rated as to not disturb the resources of the sources.
- Only open-source libraries should be used in the product.

Resource Requirements

- Low-medium power GPU cluster for NER model training

Aesthetic Requirements

- Responses to queries should support being displayed in graph or tree form.

User Experiential Requirements

- The user should be able to build queries using date, company, and vulnerability filters.
- Query results should be returned in less than 30 seconds.

1.2 Engineering Standards

What Engineering standards are likely to apply to your project? Some standards might be built into your requirements (Use 802.11 ac wifi standard) and many others might fall out of design. For each standard listed, also provide a brief justification.

- PEP8
  - PEP 8 is the official style guide for Python code. Our team's code will follow this style guide to ensure maximum readability and avoid developer issues due to different coding styles in the same project.
- NIST

    - NIST has many definitions on various cybersecurity topics that we will utilize in our project. Our project will provide quick and easy information that has ties to NIST Cybersecurity Framework (CSF) and NIST Risk Management Framework (RMF), thus these standards will be incorporated into the end product.
- CVE
    - Common Vulnerabilities and Exposures is a standard for computer security flaws our project will need to follow to obtain, process, and serve our cybersecurity information. Using CVE will keep vulnerabilities tied to their initial reports, which include a description of the flaw, making it easier for users to follow a chain of information from our knowledge graph.
- CVSS
    - The Common Vulnerability Scoring System will be used in our project due to our interaction with vulnerabilities. This framework has three metrics that are used to rate the severity of vulnerabilities our knowledge graph will contain. We will interpret and display data in a CVSS format to remain consistent with other sources of information and improve comprehension by our users.