## Requirements

**1 – UI**

1.1 – The system must display an input field for the user to enter data into.

1.2 – There shall be buttons allowing the user to activate the lookup system, and activate filters for their query

* Unfortunately, we ran out of time to implement search filters.

1.3 – Upon query generation, a list of security analysis tools will be provided to the user for them to use.

1.3.1 – Virus Total, and shall be included.

1.3.2 – GreyNoise, Shodan, and Censys may be included.

1.4 – The query result shall include the last date and time an IOC was looked up.

1.5 – The application shall have a dark mode that can be toggled on or off.

1.6 – The layout of the application must be predictable and simple, allowing users to input data and analyze results with the fewest clicks necessary.

1.7 – Users must have the option to export IOCs into a CSV file via a button.

1.7.1 – There must be a check box for each IOC being exported to state whether it is defanged or live.

1.7.2 – The exported IOCs shall be kept in the same order they are in when input into the application.

* Unfortunately, we ran out of time to implement an export to CSV button.

1.8 – The application must be readable and the information in it distinguishable for color-blind individuals.

1.9 - There shall be a tab feature so that users can keep the results of multiple queries on a single page.

1.9.1 – Each tab shall represent a separate query, with the results of each query listed in a table within the tab

1.9.2 – There must be a button in the tabs section allowing the user to create a new tab.

**2 – Functionality**

2.1 – User input (IOCs) must be passed through the application into the back-end API; the user shall not interact with the API directly.

2.2 – User input shall be returned to the application via the API as a JSON file that must be parsed and displayed appropriately.

2.3 The system shall log user activity, including the IOCs that were looked up, the user who performed the lookup, the date and time of the lookup.

**3 – Technology Stack**

3.1 – The front and back ends of the application shall be programmed using the Go programming language.

* We switched to using React.js for our front-end to speed up development and utilize the many components available with it. This allowed us to provide a more robust and reliable front-end.

3.2 – The Go scripts shall interact seamlessly with the provided API.

3.3 – The application must have proper error handling, with error codes provided via alerts and a designated error code.

**4 – Compatibility and Accessibility**

4.1 – The application must work on ***COX designated web browser*** within the set performance goals.

4.2 – The application must be usable within COX’s internal networks.