**General Requirements:**

**UI Component:**

Accepts the following input:

1. Server IP and Port to listen in on.
2. Any number of rules for triggering actions. Rules should be able to handle:
   1. Detecting type of traffic
      1. SQL Server (TCP/IP -> TDS)
      2. HTTP POST / PUT / DELETE Requests (TCP/IP)
   2. Detecting contents of traffic based off types of conditions listed below (content functions)
      1. Contains
      2. Starts with
      3. Ends with
      4. Doesn’t Contain
      5. Doesn’t start with
      6. Doesn’t ends with
      7. Wildcard/Fuzzy searches
   3. The above content detection must be expandable to multiple rules. Functionality to add/edit/remove such rules must exist.
   4. Results from each one of these rules should be storable within a variable that’s used later on in section 3.
      1. Example: If resulting match has 5 columns coming back in SQL, the variable called **returnedVariable**  should be reusable in section 3 with syntax similar returnedVariable[0], returnedVariable[1], returnedVariable[2], returnedVariable[3], returnedVariable[4]. **(This needs to be discussed with the team)**
   5. Detecting contents of traffic based on regular expression\*
3. Any number of actions of the following types
   1. Call to API (postman style with ability to pass data from packet capture)
      1. Must support GET / POST / PUT / DELETE http requests
      2. Must make available all fields retrieved in section 2d as configurable variables in API call parameters.
   2. Call to another action
      1. SQL
         1. SQL Server URL
         2. SQL Server port
         3. SQL Server db name?
         4. SQL Server Query to run with variables defined in 2d as parameters (in set or where clauses).
            1. All fields retrieved in section 2d must be available as accessible variables for the runnable SQL Server query.
   3. Call outside, self-contained code written in JS or C#

Front End Application must:

1. Store all ports, rules, and actions in a SQL database.
2. Support success and failure conditions based on response codes.

**Listening Component:**

The listening component of the software must:

1. Listen in on ports specified in the UI.
2. Analyze traffic over configured port
   1. At minimum the software must handle SQL traffic and plain text http requests
      1. SQL Server (TCP/IP -> TDS)
      2. HTTP POST / PUT / DELETE Requests (TCP/IP)
3. Trigger configured actions based on the configured rules
4. Support configured failure conditions
   1. If success, run action A. If failure, run action B.
5. Support input types listed below:
   1. Network traffic over specified port
6. Support output types listed below:
   1. Any configured action.
7. Log all triggered actions.
   1. Can log in to either a DB table or a txt log file.
   2. Location should be configurable in the UI.

**SQL Requirements:**

If the traffic coming over the port is SQL traffic, we should be able to handle the following cases:

1. INSERT Query
   1. Defined by traffic coming through the port containing “INSERT INTO” as the key words.
   2. Rules should be able to handle what table is being inserted into and what data is being inserted.
   3. Rule Examples:
      1. If traffic is data being inserted into the customer table, fire action A.
      2. If traffic is data being inserted into the customer table and has an id of 20, fire action B.
      3. If traffic is data being inserted into the location table and has an address of 123 Road St, fire off action C.
2. UPDATE Query
   1. Defined by traffic coming through the port containing “UPDATE” as the key word.
   2. Rules should be able to handle what table is being updated and the fields being updated.
   3. Rule Examples:
      1. If traffic is data being updated in the customer table, fire action A.
      2. If traffic is data being updated in the customer table and the customer id is 20, fire action B.
      3. If traffic is data being updated in the location table and has the address of 123 road St, fire off action C.
3. DELETE Query
   1. Defined by traffic coming through the port containing “DELETE FROM” as the key words.
   2. Rules should be able to handle what table is being deleted from and what record is being deleted.
   3. Rule Example:
      1. If traffic is data being deleted in the customer table, fire action A.
      2. If traffic is data being deleted in the customer table and the customer id is 20, fire action B.
      3. If traffic is data being deleted in the location table and has the address of 123 road St, fire off action C.

**HTTP Request Requirements:**

If traffic coming over the port is an HTTP request, we should be able to handle the following cases:

1. GET
2. POST
3. PUT
4. DELETE
5. OPTIONS

For each of the above HTTP requests, we must be able to break down the data into the headers and the body of the request. From there we should be able to use the “content” functions outlined above in order to search the contents of either the header or the body of the request.

**Action Requirements:**

Actions must be able to carry out the following actions:

1. An external call to an API.
   1. Must be able to pass all fields contained in insert or update query.
   2. Must be able to submit to any end point.
   3. Must be able to customize fields getting posted to. (POSTMAN style/JSON body)
2. An internal call to another action
   1. Reference another action and call it.
3. An external call to a piece of code.
   1. Either JS or C#

All action should be carried out only after a confirmation message is received.