**Documentation - Asset Tracking System**

**Non-Functional Requirements:**

1. The software system should be designed and developed to track assets withing the given company.
2. Information must be stored in a database which could be used by other company systems in the future.
3. All actions must be created to be used to manipulate all asset data. Actions included are add, view, edit or delete.
4. Software must be designed to automatically acknowledge key data regarding hardware from the system on which it is running. Key data should include (system name, model, manufacturer, type, IP Address, etc.)
5. Software can also include a component which can hold a capture of any physical stickers, which hold useful information.
6. Software must also include an extra space for any data that the company may wish to provide regarding a specific asset.
7. Data should be intuitively corresponded.
8. Response time and data update must take no more than X milliseconds.
9. UI Design must be intuitive and easy to understand.
10. Manipulating data stored in the database, must be very easy and fast to learn. \*User training time should be less than 30 minutes.
11. Operating should be on a Microsoft Environment, which is able to run on a Windows 10, 11 machine.

**Functional Requirements:**

**Introduction:** This section will cover all functionalities, proposed from the client. Requirements will be fully detailed, which must be converted into a real implementation in the application. Validation will be held, to create test cases from specification, to assure implementation is according to specification.

**1. Asset Tracking – Used Components**

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Status** | **Last Modified** |
| **1.1** For the creation of the database section the application team will use Microsoft SQL Server Management Studio to handle the Data. | **NOT IMPLEMENTED** | **24/10/2022** |
| **1.2** The application itself will be developed via Visual Studio as an Application Project. | **NOT IMPLEMENTED** | **24/10/2022** |
| **1.3** All functionalities must be technically described and for more complex use cases, UML diagrams must be used for visualization. | **NOT IMPLEMENTED** | **24/10/2022** |

**2. Home Section – Layout and Functionality**

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Status** | **Last Modified** |
| * 1. User must be able to see layout which consists of text and buttons. Text information must be relative to the function of each button visible on the canvas. | **IMPLEMENTED** | **24/10/2022** |
| **2.2** Each button must trigger an event from “***3. Event Triggering***”. | **IMPLEMENTED** | **24/10/2022** |
| **2.3** Empty non-usable buttons should not be visible. | **IMPLEMENTED** | **24/10/2022** |
| **2.4** All panels must contain a button directing to the “Home” page. | **IMPLEMENTED** | **30/10/2022** |
| **2.5** At start of application, algorithm must show details of machine on which it is running. If information could not be retrieved, it should output “Unknown:”. | **IMPLEMENTED** | **30/10/2022** |
| **2.5.1**  Information should be:  Name, Model, Manufacturer, IP Address | **IMPLEMENTED** | **30/10/2022** |

**3. Event Triggering – Functionality**

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Status** | **Last Modified** |
| **<CLASS DIAGRAM OF SYSTEM HERE> TO DO** | **IMPLEMENTED** | **24/10/2022** |
| 3.1 **Add Asset Functionality**  *-This functionality shall offer the user to add a new asset and save it to the database.* | **IMPLEMENTED** | **24/10/2022** |
| 3.1.0 Inputs | **IMPLEMENTED** | **24/10/2022** |
| 3.1.1  Prefix “I” shall be used for naming variables which are indicating inputs. Prefix “o” shall be used for naming variables which are indicating outputs. | **IMPLEMENTED** | **24/10/2022** |
| 3.1.2  String iname – Information from input field “name” | **IMPLEMENTED** | **24/10/2022** |
| 3.1.3  String imodel – Information from input field “model’ | **IMPLEMENTED** | **24/10/2022** |
| 3.1.4  String imanufacturer – Information from input field “manufacturer” | **IMPLEMENTED** | **24/10/2022** |
| 3.1.5  String itype – Information from input field “type” | **IMPLEMENTED** | **24/10/2022** |
| 3.1.6  String iIPAddress – Information from input field “IP Address” | **IMPLEMENTED** | **24/10/2022** |
| 3.1.7  String iPurchaseDate – Information from input field “Purchase Date” | **IMPLEMENTED** | **24/10/2022** |
| 3.1.8  String iExtraInfo – Information from input field “Extra Information” | **IMPLEMENTED** | **24/10/2022** |
| 3.1.9  String oResponse – Text indicating  OPTION 1: “You have successfully added an asset!”  *OR*  OPTION 2: “Please Add All the Required Information”. | **IMPLEMENTED** | **24/10/2022** |
| 3.1.10 Button “Add” – Triggering the writing to the database. | **IMPLEMENTED** | **24/10/2022** |
| 3.1.11 Expected Behavior | **IMPLEMENTED** | **24/10/2022** |
| 3.1.12 User must be able to add an asset by clicking a button “Add a new asset”. This shall trigger an event, leading to a panel, which visualizes input boxes for the needed information and button “Add”. | **IMPLEMENTED** | **24/10/2022** |
| 3.1.13 After “Add Asset” panel is visualized, input boxes shall be visible for *system name, model, manufacturer, type, IP Address, purchase date* (optional), *extra information* (optional). | **IMPLEMENTED** | **24/10/2022** |
| 3.1.14 Button “Add” must trigger an event, which shall check if all “required” information is written. If it is successful, information shall be sent to the database and *oResponse* text shall indicate OPTION 1 (from 3.1.9). If information is not written, triggering shall be declined and *oResponse* text shall indicate OPTION 2. | **IMPLEMENTED** | **24/10/2022** |
| 3.1.15 **<USE CASE DIAGRAM SHOWING HOW IT WILL WORK HERE> TO DO** | **IMPLEMENTED** | **24/10/2022** |
| 3.2 **View Asset Functionality**  *-This functionality shall offer the user to view an asset/asset from the database.* | **FUTURE** | **FUTURE** |
| 3.2.0 Inputs | **FUTURE** | **FUTURE** |
| 3.2.1 Prefix “I” shall be used for naming variables which are indicating inputs. Prefix “o” shall be used for naming variables which are indicating outputs. | **FUTURE** | **FUTURE** |
| 3.2.2 Int iId – The variable in which the user will enter an Id, to find a specific item | **FUTURE** | **FUTURE** |
| 3.2.3 Boolean iSpecific – True if the user is looking for a specific item, and False if user Is searching for all items.  If iSpecific == True  Then user must input a valid asset id  Else If iSpecific == False  User MAY not input a valid asset id, as all items must be output. | **FUTURE** | **FUTURE** |
| 3.2.4 oResult[] – Array which will output the results. Either only [0] will output the specific asset if (ISpecific == true (see more 3.2.3), or it will fill the whole array with all assets stored. Text must contain all details of the asset which are stored. | **FUTURE** | **FUTURE** |
| 3.2.5 Button “Find” – Will trigger a search in the database, either looking for a specific ID (depending on iSpecifc (see more 3.2.3) or looking to return all assets, stored in the database. | **FUTURE** | **FUTURE** |
| 3.2.6 Expected Behavior | **FUTURE** | **FUTURE** |
| 3.2.7 User must be able to view a specific asset or all which are stored in the database. A toggle/radio must be used to check what the user has chosen. The result from the radio/toggle will be used as input for iSpecific. | **FUTURE** | **FUTURE** |
| 3.2.8 If the user has chosen ISpecifc == True,  Then an input field must be visible, indicating that the user must input a specific asset id. After the user has entered a value, the user must click the button “Find” to trigger the event in 3.2.10 | **FUTURE** | **FUTURE** |
| 3.2.9 If the user has chosen ISpecifc == False, then user must only click the button “Find” to trigger the event, described int 3.2.10 | **FUTURE** | **FUTURE** |
| 3.2.10 Button “Find” must be triggered to output the results from the search by requesting information from database. The event must only be triggered if (3.2.8 or 3.2.9) are fulfilled. If they are not oResult[0] must output “Something went wrong!”. | **FUTURE** | **FUTURE** |
| 3.2.11 **<USE CASE DIAGRAM SHOWING HOW IT WILL WORK HERE> TO DO** | **FUTURE** | **FUTURE** |
| **3.3 Edit Asset Functionality**  *-This functionality shall offer the user to edit an asset and update it to the database.* | **FUTURE** | **FUTURE** |
| 3.3.0 Inputs | **FUTURE** | **FUTURE** |
| 3.1.1  Prefix “I” shall be used for naming variables which are indicating inputs. Prefix “o” shall be used for naming variables which are indicating outputs. | **FUTURE** | **FUTURE** |
| 3.3.2  Int iId – Information from input field “ID” | **FUTURE** | **FUTURE** |
| 3.3.3  String iname – Information from input field “name” | **FUTURE** | **FUTURE** |
| 3.3.4  String imodel – Information from input field “model’ | **FUTURE** | **FUTURE** |
| 3.3.5  String imanufacturer – Information from input field “manufacturer” | **FUTURE** | **FUTURE** |
| 3.3.6  String itype – Information from input field “type” | **FUTURE** | **FUTURE** |
| 3.3.7  String iIPAddress – Information from input field “IP Address” | **FUTURE** | **FUTURE** |
| 3.3.8  String iPurchaseDate – Information from input field “Purchase Date” | **FUTURE** | **FUTURE** |
| 3.3.9  String iExtraInfo – Information from input field “Extra Information” | **FUTURE** | **FUTURE** |
| 3.3.10  String oResponse – Text indicating result of event | **FUTURE** | **FUTURE** |
| 3.3.11 Button “Update” – Triggering the writing to the database. | **FUTURE** | **FUTURE** |
| 3.3.12 Expected Behavior | **FUTURE** | **FUTURE** |
| 3.3.13 The user must fill all required input field, otherwise triggering event from Button “Update”, must not be possible. | **FUTURE** | **FUTURE** |
| 3.3.14 Button “Update” must check for (3.3.13). If information is fulfilled, event shall trigger updating the asset with specific IId (3.3.2). | **FUTURE** | **FUTURE** |
| 3.3.15 If after requesting to update from database, asset with “iId” could not be found, oResponse must output “Cannot Find Asset with Such ID” | **FUTURE** | **FUTURE** |
| 3.3.16 If after requesting to update from database, asset with “iId” is found and the update is successful, oResonse must output “Asset Successfully Updated” | **FUTURE** | **FUTURE** |
| 3.3.17 **<USE CASE DIAGRAM SHOWING HOW IT WILL WORK HERE> TO DO** | **FUTURE** | **FUTURE** |
| **3.4 Delete Asset Functionality**  *-This functionality shall offer the user to delete an asset from the database.* | **FUTURE** | **FUTURE** |
| 3.4.0 Inputs | **FUTURE** | **FUTURE** |
| 3.4.1 Prefix “I” shall be used for naming variables which are indicating inputs. Prefix “o” shall be used for naming variables which are indicating outputs. | **FUTURE** | **FUTURE** |
| 3.4.2 Int iId – The variable in which the user will enter an Id, to find a specific item | **FUTURE** | **FUTURE** |
| 3.4.3 Button “Delete” – Will trigger a delete request in the database, looking for a specific asset with “iId” | **FUTURE** | **FUTURE** |
| 3.4.5  String oResponse – Text indicating result of event | **FUTURE** | **FUTURE** |
| 3.4.4 Expected Behavior | **FUTURE** | **FUTURE** |
| 3.4.5 Button “Delete” shall trigger a request to delete item from the database ONLY if iId is found in the database. If deletion is successful oResponse must output “Successfully Deleted the Asset”. If iid is not found or null, request shall not be triggered and oResponse must output “Something went wrong.” | **FUTURE** | **FUTURE** |
| 3.4.6 **<USE CASE DIAGRAM SHOWING HOW IT WILL WORK HERE> TO DO** | **FUTURE** | **FUTURE** |