|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **System Name:** | | | | |
| **Author:** Anke Brits | **Date: 21 July 2024** | | | **Version:** 1.0.0 |
|  | | | | |
| **Use Case Name:** | Request inventory item | | **Use Case Type** | |
| **Use Case ID:** | 4.4 | | Abstract: ◻ | |
| **Priority:** | High | | Extension: ☒ | |
| **Source:** | Client study (Hestico) | |  | |
| **Participating Actors:** | Technician | | | |
| **Description:** | This use case describes the process of adding parts to the work order.  The technician requests to add parts to the work order. The system requests the technician to select the relevant inventory items and enter a quantity that is requires. The technician submits the request, and the system adds them to the work order.  The use case concludes when the inventory work order items have successfully been added. | | | |
| **Pre-condition:** | The technician must be logged in  The work order must already exist | | | |
| **Typical Course**  **of Events:** | **Actor Action** | **System Response** | | |
|  | Step 1: The system loads the "Add Parts" screen that contains the following elements:  A heading with the text “Add Parts” at the top of the screen.  An input textbox with a placeholder “Search...” allowing the user to search for inventory items. The system filters the inventory list as the user types.  A table that displays all the inventory that has the following columns   * Name * Description * Select   For each row in the Inventory table, there is a Select button in the Select column.  A pagination bar with a previous button with the text “Previous” on the left and a next button with the text “Next” on the right below the representatives table in the centre. Labels with the respective number display between previous and next button to indicate the page. The pages start and increment each new page.  Buttons   * Confirm * Cancel   The system will send a request form the Angular frontend to the InventoryWorkOrder service where the service will make a http get request to the .NET Core backend which makes use of a Lambda LINQ Query which creates a SQL Select query to retrieve the inventory work order from the InventoryWorkOrder Entity and the corresponding information from tables that are referenced by the foreign keys. The tables referenced by the foreign keys are described below.  The system links the InventoryWorkOrder table to the Work\_Order table using the foreign key Work\_Order\_Id. The Work\_Order table has the following attributes:   * Work\_Order\_Id (PK) * Service\_Request\_Id (FK) * Machine\_Type\_Id (FK) * Work\_Order\_Status\_Id (FK) * Employee\_Id (FK) * Reason * Date\_Started * Date\_Completed * Invoice\_Id (FK)   The system links the InventoryWorkOrder table to the Inventory table using the foreign key Inventory\_Id. The Inventory table has the following attributes:   * Inventory\_Id (PK) * Inventory\_Status\_Id (FK) * Name * Inventory\_Description * Price * Quantity   The system links the Inventory table to the Inventory\_Status table using the foreign key Inventory\_Status\_Id. The Inventory\_Status table has the following attributes:   * Inventory\_Status\_Id (PK) * Name * Description   The system links the InventoryWorkOrder table to the InventoryWorkOrderStatus table using the foreign key Inventory\_Work\_Order\_Status\_Id. The InventoryWorkOrderStatus table has the following attributes:   * Inventory\_Work\_Order\_Status\_Id (PK) * Name * Description   The system logs the following when inventory is viewed:   * user performing the operation * Transaction Type: * The description   In the following Audit\_Trail entity has the following attributes:   * Audit\_Trail\_Id (PK) * Date\_Time * User\_Name * Transaction\_Type * Description   The Audit\_trail\_Id is automatically incremented.  [ALT] | | |
| Step 2: The technician enters the search criteria for the inventory items that they are searching for. | Step 3: The system searches for the Inventory item in the database from the Inventory table using ASP.Net 7 Web API controller by using a LINQ query and displays the records that match the inputted information in the Inventory table on the “Add Parts” screen.  [ALT] | | |
| Step 4: The technician clicks the “Select” button for the inventory items that they want to add to the Work Order Update. | Step 5: The system changes the text on the Select buttons to “Selected” and changes the colour of the button to green. | | |
| Step 6: The technician clicks the “Confirm” button.  [ALT] | Step 7: The system stores the selected inventory items in an array and stores the array in local storage. | | |
|  | Step 8: The system clears the content on the “Add Parts” screen and displays the following elements:  A heading with the text "Selected Parts" at the top of the screen.  A table displaying the selected inventory items with the following columns:   * Name * Description * Required Quantity (input field to specify the quantity needed)   A "Save" button to finalize and save the selection of parts.  A “Cancel” button.  The system retrieves the selected inventory items from local storage and displays them in the newly generated table. | | |
| Step 9: The technician enters the required quantity for each inventory item in the table. |  | | |
| Step 10: The technician clicks the “Save” button.  [ALT] | Step 11: The system validates the entered information when the technician clicks the "Save" button on the "Add Parts" screen. The system uses Angular form checking to ensure that all required information has been inputted by checking that the input fields for the required quantity are not null and meet the specified criteria  Required Quantity:   * Required * Minimum value of 1 * Is a number   The system uses Angular to bind the input fields to the Selected Inventory array and allows the technician to edit the required quantity for each selected item.  The system uses Entity Framework Core to save the entered information in the InventoryWorkOrder table with the following attributes:   * InventoryWorkOrder\_Id (PK) * Work\_Order\_Id (FK) * Inventory\_Id (FK) * Quantity * Inventory\_Work\_Order\_Status\_Id (FK)   The system will auto increment the InventoryWorkOrder\_Id.  The system displays the selected inventory items by linking the InventoryWorkOrder table to the Inventory table using the foreign key Inventory\_Id. The Inventory table has the following attributes:   * Inventory\_Id (PK) * Inventory\_Status\_Id (FK) * Name * Inventory\_Description * Price * Quantity   The system links the InventoryWorkOrder table to the InventoryWorkOrderStatus table using the foreign key Inventory\_Work\_Order\_Status\_Id. The InventoryWorkOrderStatus table has the following attributes:   * Inventory\_Work\_Order\_Status\_Id (PK) * Name * Description   The system links the InventoryWorkOrder table to the Work\_Order table using the foreign key Work\_Order\_Id. The Work\_Order table has the following attributes:   * Work\_Order\_Id (PK) * Service\_Request\_Id (FK) * Machine\_Type\_Id (FK) * Work\_Order\_Status\_Id (FK) * Employee\_Id (FK) * Reason * Date\_Started * Date\_Completed * Invoice\_Id (FK)   The system saves the inventory work orders and updates the inventory quantities by using Entity Framework Core to manage the related data.  If the inventory required is more than the inventory quantity that is in stock, two entries for the InventoryWorkOrder table in the database. The one entity has a quantity equal to the remaining stock and has a InventoryWorkOrderStatus of “Used”. The remaining stock will be added to the second entry and will have a status of “Order Requested”.  [ALT] | | |
|  | Step 12: The system redirects the user to the “Update Work Order” screen.  The system loads the “Update work order” screen which contains the following elements:  A heading with the text “Work Order Update Details” at the top of the screen.  A card displaying the work order update details with the following information:   * Date Started * Work Order ID * Customer (Employee Name and Surname) * Machine Type * Reason * Technician (Employee Name and Surname) * Start Travel Time button * End Travel Time button * Start Work Order Time button * End Work Order Time button   A table displaying the details of a newly created inventory work order items with the following columns:   * Name * Description * Quantity * Status   The system displays the work order update details by using Entity Framework Core to retrieve only the related data in the Work\_Order\_Update table with the following attributes:   * Work\_Order\_Update\_Id (PK) * Work\_Order\_Id (FK) * Start\_Travel\_Time * End\_Travel\_Time * Start\_Work\_Order\_Time * End\_Work\_Order\_Time * Work\_Completed * Date\_Completed   The system links the Work\_Order\_Update table to the Work\_Order table using the foreign key Work\_Order\_Id. The Work\_Order table has the following attributes:   * Work\_Order\_Id (PK) * Service\_Request\_Id (FK) * Machine\_Type\_Id (FK) * Work\_Order\_Status\_Id (FK) * Employee\_Id (FK) * Reason * Date\_Started * Date\_Completed * Invoice\_Id (FK)   The system links the Work\_Order\_Update table to the InventoryWorkOrder table using the foreign keys Work\_Order\_Id and Inventory\_Id. The InventoryWorkOrder table has the following attributes:   * InventoryWorkOrder\_Id (PK) * Work\_Order\_Id (FK) * Inventory\_Id (FK) * Quantity * Inventory\_Work\_Order\_Status\_Id (FK)   The system links the InventoryWorkOrder table to the Inventory table using the foreign key Inventory\_Id. The Inventory table has the following attributes:   * Inventory\_Id (PK) * Inventory\_Status\_Id (FK) * Name * Inventory\_Description * Price * Quantity   The system links the InventoryWorkOrder table to the InventoryWorkOrderStatus table using the foreign key Inventory\_Work\_Order\_Status\_Id. The InventoryWorkOrderStatus table has the following attributes:   * Inventory\_Work\_Order\_Status\_Id (PK) * Name   Description | | |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |
| **Alternate Courses:** | [ALT] Step 1: There are no records in the inventory table from the database. The system will display an error message with the text “No item were found” | | | |
| [ALT] Step 3: There is no records in the inventory table from the database that matches the input search criteria. The system will display a notification to state “Data not found”. Technician clears search criteria, go to step 2. | | | |
| [ALT] Step 6a: The technician did not select any inventory items. The system displays a notification to state “Please select an item”. | | | |
| [ALT] Step 6b: The technician selects the cancel button. The technician is redirected to the “Update Work Order” screen. The use case is terminated. | | | |
| [ALT] Step 10: The technician selects the cancel button. The technician is redirected to the “Update Work Order” screen. The use case is terminated. | | | |
| [ALT] Step 11:The system fails to validate because the entered values does not match the specification for each required field. The system will notify the technician that provided information is invalid with popup notification that input information is invalid. Go to step 9 so technician can re input the information. | | | |
| **Post-condition:** | The technician is redirected to the “Update Work Order” screen. | | | |