|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **System Name:** | | | | |
| **Author:** Anke Brits | **Date:** 21 July 2024 | | | **Version:** 1.0.0 |
|  | | | | |
| **Use Case Name:** | Update work order | | **Use Case Type** | |
| **Use Case ID:** | 4.2 | | Abstract: ◻ | |
| **Priority:** | High | | Extension: ☒ | |
| **Source:** | Client study (Hestico) | |  | |
| **Participating Actors:** | Technician | | | |
| **Description:** | This use case describes the process of the technician creating a work order update.  The technician logs all the relevant details required for the work order update. The technician then has the option to add parts to the work order update. The technician then clicks the submit button and the work order update is created. They are then redirected to the “View Work Orders” screen.    This use case concludes when a work order update as been created. | | | |
| **Pre-condition:** | The technician must be logged in | | | |
| **Typical Course**  **of Events:** | **Actor Action** | **System Response** | | |
|  | Step 1: 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 (deactivated) * Start Work Order Time button (deactivated) * End Work Order Time button (deactivated) * Cancel button   A table displaying the details of a newly created inventory work order items with the following columns:   * Name * Description * Quantity * Status   The system will send a request form the Angular frontend to the Work Order 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 quote from the Work Order 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 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   [ALT] | | |
| Step 3: The technician clicks the “Start Travel Time” button | Step 4: The system stores the logs start travel time in local storage and deactivates the button so it cannot be selected again and activates the “End Travel Time” button. | | |
| Step 5: The technician clicks the “End Travel Time” button | Step 6: The system stores the logs end travel time in local storage and deactivates the button so it cannot be selected again and activates the “Start Work Order” button. | | |
| Step 7: The technician clicks the “Start Work Order” button | Step 8: The system stores the logs start work order time in local storage and deactivates the button so it cannot be selected again and activates the “End Work Order” button. | | |
| Step 9: The technician clicks the “End Work Order” button | Step 10: The system stores the logs end work order time in local storage and deactivates the button so it cannot be selected again. | | |
| Step 11: The technician enters a description of the work completed in the “Work Completed” textbox. |  | | |
| Step 12: The technician clicks the “Add Parts” button.  [ALT] | Step 13: The system redirects the user to the “Add Parts” screen.  The system invokes Use Case 4.4 Request inventory item. | | |
| Step 14: The technician selects the “Submit” button.  [ALT] | Step 15: The system validates the entered information by using Angular form checking to ensure that all required information has been inputted by checking that the dropdowns and input textboxes are not null. The system also validates the information in the ASP.Net 7 Web API controller by using a LINQ query to ensure that the work order update does not already exist.  The system validates that the provided information meets these requirements:  Start Travel Time:   * Required * Must be a valid datetime   End Travel Time:   * Required * Must be a valid datetime   Start Work Order Time:   * Required * Must be a valid datetime   End Work Order Time:   * Required * Must be a valid datetime   Work Completed:   * Required * Maximum length of 5000 characters * Is a string   The system will send a request form the Angular frontend to the Work Order 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 Insert query to retrieve the work order update from the Work Order Update Entity and the corresponding information from tables that are referenced by the foreign keys. The tables referenced by the foreign keys.  The system logs the following when work order update is created:   * user performing the operation * Transaction Type: * The description which contains the work order id, start travel time, end travel time, start work order time, end work order time.   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 16: The system redirects the technician to the “View Work Orders” screen.  The system loads the “View Work Orders” screen that contains the following elements  A heading with the text “Work Orders” top of the screen to the left.  An input textbox for the admin to input the information they want to search for below the “Work Orders” heading to the left.   * A placeholder within the input textbox with the label “Search..”   A card for each work order that contain the following information:   * Work Order ID * Service Request ID * Machine Type * Status * Employee * Reason * Date Started * Date Completed * View Updates button * View Parts button   The attributes Work Order ID, Service Request ID, Machine Type, Status, Employee, Reason, Date Started, and Date Completed will be displayed in the attribute’s respective places in the card.  The system will send a request form the Angular frontend to the Work Order 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 work order from the Work Order 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 displays the work order details by using Entity Framework Core to retrieve only the related data in the Work\_Order table with 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 table to the Service\_Request table using the foreign key Service\_Request\_Id. The Service\_Request table has the following attributes:   * Service\_Request\_Id (PK) * Service\_Request\_Status\_Id (FK) * Service\_Type\_Id (FK) * Technician\_Id (FK) * Machine\_Type\_Id (FK) * Representative\_Id (FK) * Preferred\_Date * Alternative\_Date\_1 * Alternative\_Date\_2 * Alternative\_Date\_3 * Problem\_Description * Finalised\_Date * Work\_Order\_Id (FK)   The system links the Work\_Order table to the Machine\_Type table using the foreign key Machine\_Type\_Id. The Machine\_Type table has the following attributes:   * Machine\_Type\_Id (PK) * Name * Description * Serial\_Number   The system links the Work\_Order table to the Work\_Order\_Status table using the foreign key Work\_Order\_Status\_Id. The Work\_Order\_Status table has the following attributes:   * Work\_Order\_Status\_Id (PK) * Name * Description   The system links the Work\_Order table to the Employee table using the foreign key Employee\_Id. The Employee table has the following attributes:   * Employee\_Id (PK) * Employee\_Type\_Id (FK) * UserId * Name * Surname * Address * Phone\_No * Gender * Race * Preferred\_Name   The system links the Work\_Order table to the Invoice table using the foreign key Invoice\_Id. The Invoice table has the following attributes:   * Invoice\_Id (PK) * Customer\_Reference\_Number * Discount\_Id (FK) * Work\_Order\_Id (FK) * Quote\_Id (FK) * Invoice\_Status\_Id (FK) * Date\_Issued * Date\_Paid * Total\_Amount * Description   If there is no Invoice associated with the Work Order that is displayed, then the Invoice\_Id is not displayed.  If the Date\_Completed value is set to Null, then the Date\_Completed will not be displayed on the card.  The system logs the following when work order update is created:   * user performing the operation * Transaction Type: * The description which contains the work order id, start travel time, end travel time, start work order time, end work order time.   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. | | |
|  |  | | |
|  |  | | |
|  |  | | |
| **Alternate Courses:** | [ALT] Step 1: There is an error in retrieving the information from the Work Order table in the database. The system displays an error message. A label with the text "There is an error with the system, please try Again Later”. | | | |
| [ALT] Step 12: The technician has no parts that need to be added to the work order. Proceed to step 15. | | | |
| [ALT] Step 15: 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 12 so technician can re input the information. | | | |
|  | | | |
|  | | | |
|  | | | |
| **Post-condition:** | The work order update has successfully been created. | | | |