| **System Name: HestiService system** | | | | | |
| --- | --- | --- | --- | --- | --- |
| **Author:** Anke Brits | **Date:** 21 July 2024 | | | **Version:** 1.0.0 | |
|  | | | | | |
| **Use Case Name:** | View Work Order | | **Use Case Type** | | |
| **Use Case ID:** | 2.17 | | Business Requirements: ◻ | | |
| **Priority:** | High | | System Analysis: ◻ | | |
| **Source:** | Client study (Hestico) | | System Design: ☒ | | |
| **Primary Business Actor (PBA):** | Customer | | | | |
| **Primary System Actor (PSA):** | None | | | | |
| **Other Participating Actors:** | None | | | | |
| **Other Interested Stakeholders:** | Admin | | | | |
| **Description:** | The use case describes the process of viewing a work order.  The customer will be able to view the work orders. The customer will click on the work order link in the navigation bar and the system will direct the customer to the View Work Order screen. The customer can also view work orders according to specific employee names. The customer will be able to view the updates associated with a particular work order. The customer will click on the "View Updates" button from the work orders list, and the system will direct the technician to the "View Work Order Updates" screen. The customer will be able to view the inventory work orders associated with a particular work order. The technician will click on the "View Inventory" button from the work orders list, and the system will direct the customer to the "View Inventory Work Orders" screen.  The use case ends when the work order has been viewed. | | | | |
| **Pre-condition:** | * The customer needs to be logged in. * The work order must exist to be viewed. | | | | |
| **Trigger:** | * The customer wants to view all their work orders and clicks the “Work Orders” button on the navigation bar. | | | | |
| **Typical Course**  **of Events:** | **Actor Action** | **System Response** | | | |
| **Manual Action** | | | **Automated Action** |
| Step 1: The customer wants to view all their work orders and clicks the “Work Orders” button on the navigation bar. |  | | | Step 2: 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 orders 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   The system logs the following when work order is viewed:   * user performing the operation * Transaction Type: * The description which contains the customer id.   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 3: The customer enters the employee’s name, employee’s surname, Service Request Id, Machine Type Name, Work Order Status, or Reason of the Work Order they want to search for. |  | | | Step 4: The system searches for the Work Orders in the database from the Work Orders table using ASP.Net 7 Web API controller by using a LINQ query and displays the records that match the inputted information in the Work Order table on the View Work Order screen.  The system logs the following when work order is viewed:   * user performing the operation * Transaction Type: * The description which contains the search filter (value entered to use for search).   In the following Audit\_Trail entity has the following attributes:   * Audit\_Trail\_Id (PK) * Date\_Time * User\_Name * Transaction\_Type * Description.   The Autdit\_trail\_Id is automatically incremented.  [ALT] |
| Step 5: The customer successfully views the work orders. |  | | |  |
| Step 6: The customer decides to view all the work order updates associated with their selected work order and clicks the “View Updates” button.  [ALT] |  | | | Step 7: The system loads the “View Work Order Updates” screen that contains the following elements:  A heading with the text “Work Order Updates” at the top of the screen.  A card for each work order update containing the following information:   * Update ID * Work Order ID * Start Travel Time * End Travel Time * Start Work Order Time * End Work Order Time * Date Completed * Work Completed   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 Update 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 updates 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 logs the following when work order update is viewed:   * user performing the operation * Transaction Type: * The description which contains the work order id   In the following Audit\_Trail entity has the following attributes:   * Audit\_Trail\_Id (PK) * Date\_Time * User\_Name * Transaction\_Type * Description.   The Autdit\_trail\_Id is automatically incremented.  [ALT] |
| Step 8: The customer successfully views the work orders updates. |  | | |  |
| Step 9: The customer decides to view all the parts associated with their selected work order and clicks the “View Parts” button.  [ALT] |  | | | Step 10: The system loads the “View Parts” screen that contains the following elements:  A heading with the text “Inventory Work Orders for Work Order ID” at the top of the screen.  A table displaying the inventory work orders with the following columns:   * Inventory ID * Inventory Name * Description * Price * 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 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 displays the inventory work orders by using Entity Framework Core to retrieve only the related data 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 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 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 work order is viewed:   * user performing the operation * Transaction Type: * The description which contains the work order id   In the following Audit\_Trail entity has the following attributes:   * Audit\_Trail\_Id (PK) * Date\_Time * User\_Name * Transaction\_Type * Description.   The Autdit\_trail\_Id is automatically incremented.  [ALT] |
| Step 11: The customer successfully views the work orders associated parts. |  | | |  |
| **Alternate Courses:** | [ALT] Step 2: There are no records in the Work Order table from the database. The system will display an error message with the text “No items were found” | | | | |
| [ALT] Step 2:  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 4: There is no records in the Work Order table from the database that matches the input search criteria. The system will display a notification to state “Data not found”. Admin clears search criteria, go to step 2. | | | | |
| [ALT] Step 6: The customer does not want to view anything further. The use case terminates. | | | | |
| [ALT] Step 7: There are no records in the Work Order Update table from the database. The system will display an error message with the text “No items were found”. | | | | |
| [ALT] Step 7: There is an error in retrieving the information from the Work Order Update 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 9: The customer does not want to view anything further. The use case terminates. | | | | |
| [ALT] Step 10: There are no records in the Inventory Work Order table from the database. The system will display an error message with the text “No items were found”. | | | | |
| [ALT] Step 10:  There is an error in retrieving the information from the Inventory 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”. | | | | |
| **Conclusion:** | The customer can view the Work Orders. | | | | |
| **Post-condition:** | The system displays the records for Work Orders. | | | | |
| **Business Rues:** | * None | | | | |
| **Implementation Constraints and Specifications:** | * None | | | | |
| **Assumptions:** | * None | | | | |
| **Open Issues:** | * None | | | | |