

|  |
| --- |
| School of Information and Communications Technologies |
| |  | | --- | | Work Order Manager Application |   System Document  **Instructor**  Achala Vinakota |

**Project Team**

Trevor Erixon

Jaehan Kim

Zennon Weleschuk

Maria Laura Diaz Pena

Rylan Cook

Saeid Ghavamabadi

**Aug 02, 2021**

Contents

[Preface 5](#_Toc79869941)

[Purpose of Document 5](#_Toc79869942)

[Documentation Standards 5](#_Toc79869943)

[System Overview 6](#_Toc79869944)

[System Purpose 6](#_Toc79869945)

[Clients 6](#_Toc79869946)

[End-Users 6](#_Toc79869947)

[Main Functions of the System 6](#_Toc79869948)

[User Requirements 6](#_Toc79869949)

[Business Overview and Objectives 6](#_Toc79869950)

[Project Overview 6](#_Toc79869951)

[Statement of the Problem 6](#_Toc79869952)

[System Requirements 7](#_Toc79869953)

[Use Case Diagram 7](#_Toc79869954)

[Use Case Descriptions and Scenarios 7](#_Toc79869955)

[Non-Functional Requirements 16](#_Toc79869956)

[System Interface Requirements 16](#_Toc79869957)

[Maintainability Requirements 16](#_Toc79869958)

[Usability Requirements 17](#_Toc79869959)

[Problem Domain Class Diagram 17](#_Toc79869960)

[System Design 19](#_Toc79869961)

[Layered Architecture 19](#_Toc79869962)

[Hardware Architecture 19](#_Toc79869963)

[Hardware Platform 20](#_Toc79869964)

[Software Platform 21](#_Toc79869965)

[Interaction Model 21](#_Toc79869966)

[Persistence Model 23](#_Toc79869967)

[Class Diagram 25](#_Toc79869968)

[Interaction Sequence Diagrams 30](#_Toc79869969)

[State-chart Diagrams 32](#_Toc79869970)

[Activity Diagrams 33](#_Toc79869971)

[Data Dictionary of Problem Domain and Application Domain 33](#_Toc79869972)

[Testing 37](#_Toc79869973)

[Test Plans 37](#_Toc79869974)

[System Deployment 64](#_Toc79869975)

[Data Conversion 64](#_Toc79869976)

[Changeover Strategy 64](#_Toc79869977)

[System Administration and Maintenance 65](#_Toc79869978)

[Security 65](#_Toc79869979)

[Operation 65](#_Toc79869980)

[Backup and Restore 69](#_Toc79869981)

[Data Archival 69](#_Toc79869982)

[System Maintenance 69](#_Toc79869983)

[Deficiency List 69](#_Toc79869984)

[Requirements Not Met 69](#_Toc79869985)

[Non-Functioning Components 69](#_Toc79869986)

[Appendix 69](#_Toc79869987)

[Program Listings 69](#_Toc79869988)

[Glossary 70](#_Toc79869989)

[Index 70](#_Toc79869990)

[References 71](#_Toc79869991)

# Preface

The underlying foundation for this project is the determination to apply the skills we have learned during our schooling to real world problems in the IT industry. This project was undertaken as part of the graduation requirements of the Software Development Major from the Information Technology Diploma at Southern Alberta Institute of Technology.

The process of gathering information about this project and defining its requirements has been extensive and has awoken questions about how to proceed but has also given us clarity. Our Object-Oriented System Analysis and Design instructor, Ali Moussa, has always been available for us and willing to answer our questions.

We would like to thank our instructors for all the support and counsel they have provided to us during this process. We also would like to thank our client for the opportunity he has given us to work on an application that will be beneficial to him and his company. Without his cooperation we would have not been able to direct this analysis.

## Purpose of Document

The purpose of this document is to provide a brief overview of the project requirements that we previously identified from the meetings with the client and the information given to us by him. After conducting an analysis on this information, we can, not only identify the requirements, but also define the project scope and its objectives, as well as the specifications on how to achieve success. The document aims to facilitate understanding of the reader on what the client needs the application to do and how we can translate that into solid deliverables. We can reach these deliverables by breaking down the requirement analysis into User Requirements and System Requirement.

## Documentation Standards

For this document, we ensure consistency throughout all the structure and presentation by applying documentation standards such as keeping to one style. For the headings, the font is Calibri (Headings) and the colour is the default Blue, Accent 1, and the tone of this blue and size of the heading depends on the type of heading. Headings of type 1 maintain a style of 50% darkness of blue and a font size of 18px. Headings of type 2 maintain a style of 25% darkness of blue and a font size of 16px. Headings of type 3 maintain a style of 25% darkness of blue and a font size of 14px. For the body text, the font is Calibri (Body), font size is 11px and the colour is black. There are tables included in this document which explain the actors and roles identified from the information given by the client, as well as tables describing the use cases later used in the diagrams. We used Software Ideas Modeler (SIM Tool) to create the diagrams to portray the problem domain. The diagrams will be used later on the project design and execution. The diagrams are the following:

* **Use Case Diagrams**: This diagram shows the application requirements satisfied and the interaction between actors.
* **Class Diagram**: Problem domain classes are defined, and each class has attributes, associations and operations derived from the analysis of the requirements.
* **Interaction Sequence Diagrams**: Displays the interaction within all the use cases and three other diagrams showcase specific use cases.
* **State Machine Diagram**: The time dependent object behaviour within our system.
* **Activity Diagram**: Identifies the complete processing of our system.

# System Overview

## System Purpose

The purpose of this system is to allow for field workers and office staff to have a single data share to track, request, update and monitor work orders, and tracking any interactions a field worker has with the work order (Operations). The system will give users information regarding the status of work orders and make updates to any operations that are currently in progress.

## Clients

The client for the system is an organization that needs to track field work done by their workers and be able to manage and make new request for maintenance and other such situations. This must also work for field workers knowing what they must do within the organization.

## End-Users

There are 2 major end users of this system. The first would office workers, who may manage the status of current work orders, set up accounts for new users, or create new work orders. The second will be field workers, who will do the assigned work, report on work orders, and make closing notes after the workorder is complete.

## Main Functions of the System

The system will allow for admin users to create a “Notification” which holds any details about why it is created, the location of the problem and what technical object it affects. From this a Work center lead will be able to create a work order from it and assign field workers a job. The field worker will be able to see all that has been done on the work order, update their own status and leave notes for themselves or other field workers. It will also provide coordinates for the location of the damage.

# User Requirements

## Business Overview and Objectives

We had originally engaged with PriceWaterhouseCoopers (PwC), the second-largest professional service network in the world [1], to provide a solution to the Fleet and Asset Management division that would assist with remote management of work orders. Due to some unforeseen circumstances, our client was less accessible and unable to support us as originally anticipated so we are no longer working with him or PwC; however, the focus of our application remains the same. We have essentially proceeded with creating a solution to the original problem, but have had to make some design changes to reflect an alternate deployment of the finished product.

## Project Overview

### Statement of the Problem

Currently, PwC does not have a solution that would allow their clients to remotely access details of the jobs they have been assigned to previously, therefore they cannot update the information or track the progress of completion of said job in real time. This means that some details can be lost or forgotten, and with no records of these details, mistakes can be prone to happen. While we are no longer providing our solution to PwC, this is still the problem that we aim to solve with our application.

# System Requirements

## Use Case Diagram

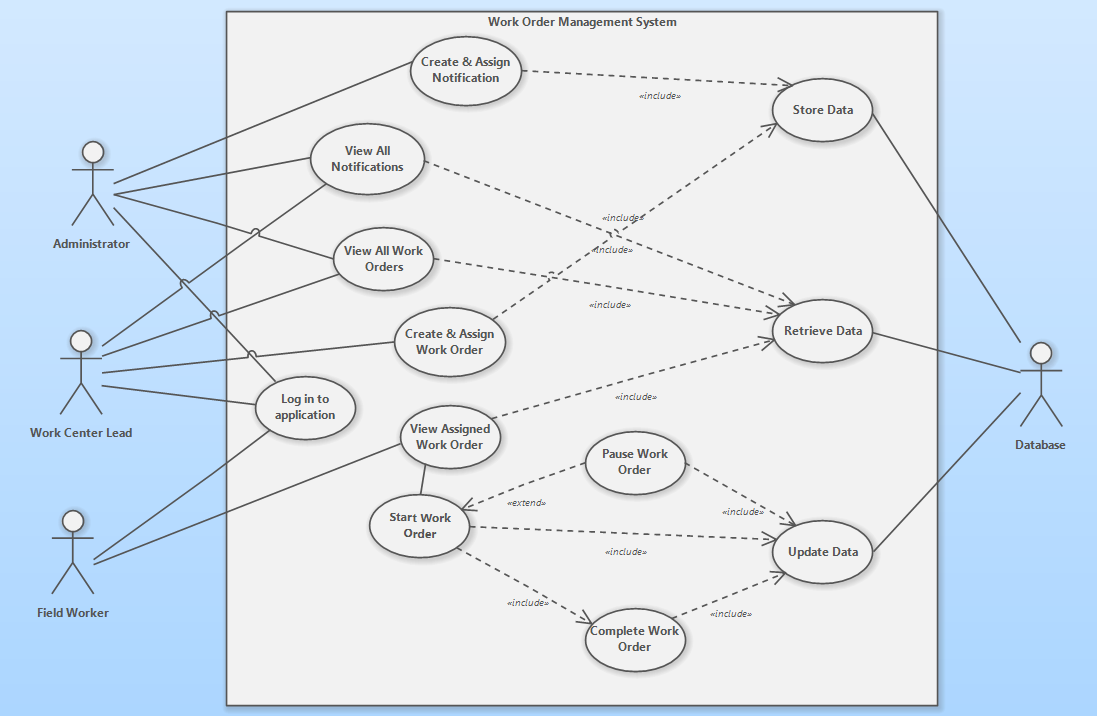


Figure 1 - Use Case Diagram

## Use Case Descriptions and Scenarios

|  |  |
| --- | --- |
| ***Actors*** | ***Roles*** |
| Administrator | Retrieve all notifications, work orders, operations and users. Create notifications and assigned the new notifications to a work center. Delete notifications. Create and delete users. |
| Field Worker | Review work order details, identify and travel to function location of work order, perform any necessary work, update status as required, and close work order once complete |
| Work Centre Lead | Retrieve all notifications assigned to work center. Retrieve all work orders for work center. Create work order from notification ID. Create and assign work order operations. |
| Database | Storing notification and work order information, including recurrent updates |

Figure 2 - Use Case Description and Scenario Table

**Use Case #1: Login to application**

Precondition: -

Postcondition: User is redirected to the appropriate page

Limitation: Network connection required

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1.1 User enters credentials to log into application | 1.2 Depending on the type of user, the system will redirect to appropriate page. |
| 1.1.1 If User is Admin | 1.2.1 System will redirect the user to the admin page and save the user information in the current session.  1.2.2 System will retrieve all notifications, work orders, operations and users and display to user. |
| 1.1.2 If User is Work Center Lead | 1.2.3 System will redirect the user to the work center page and save the user information in the current session.  1.2.4 System will retrieve all notifications and work orders assigned to the work center and display to user. |
| 1.1.3 If User is a Field Worker | -or-  1.2.5 System will redirect the user to the worker page and save the user information in the current session.  1.2.6 System will retrieve and display all the work orders assigned to the user. |

|  |  |
| --- | --- |
| **Use Case Name:** | Log in to application |
| **Primary Actor:** | All Users |
| **Stakeholders and Interests** | All Users – need to be authenticated to gain access to the system to be able to perform their required jobs. |
| **Brief Description:** | This use case describes the user logging in to the application and being redirected to the page appropriate to them. |
| **Relationships:**  **Include:** | Association: All Users  - |
| **Actor Action** | 1. User enters credentials to log into application. |
| **System Response** | 1. Depending on the type of user, the system will redirect user to appropriate page. |

Figure 3 – Use case: log in to application

**Use Case #2: Create & assign a notification**

Precondition: User is authenticated and is Admin

Postcondition: A new notification entity is assigned to a specific work center

Limitation: Network connection required

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 2.1 User selects the required options from dropdown menus to create a new notification. The required selections are Cause, Damage, Plant, Technical Object and Work Center. | 2.2 System takes user selections as parameters to create a new notification object, a new ID is assigned to the object.  2.3 System stores this notification in the database and displays it at the end of the list of all notifications in the admin page. |

|  |  |
| --- | --- |
| **Use Case Name:** | Create and Assign Notification |
| **Primary Actor:** | Admin Users |
| **Stakeholders and Interests** | Admins and Work Center Leads – need to assign the notification to a work center so that the Lead is able to create a new work order. |
| **Brief Description:** | This use case describes the admin user selecting all the requirements for a new notification to be passed as parameters for notification creation. |
| **Relationships:**  **Include:** | Association: Admin users.  Include: Storing the new notification object in the database. |
| **Actor Action** | 1. User selects the required options from dropdown menus to create a new notification. |
| **System Response** | 1. System takes user selections as parameters to create a new notification. |

Figure 4 – Use case: Create and assign notification.

**Use Case #3: View All Notifications**

Precondition: User is authenticated and is either Admin or Work Center Lead

Postcondition: All notifications are displayed.

Limitation: Network connection required

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 3.1 User retrieves all Notifications after being authenticated: | 3.2 Depending on user type, the System will display: |
| 3.1.1 If User is Admin | 3.2.1 System retrieves all notifications from all work centers to the user. |
| 3.1.2 If User is a Work Center Lead | 3.2.2 System retrieves only notifications assigned to the user’s work center. |

|  |  |
| --- | --- |
| **Use Case Name:** | View All Notifications |
| **Primary Actor:** | Admins and Work Center leads |
| **Stakeholders and Interests** | Admins and Work Center leads – need to retrieve all the notifications to perform required operations in the system. |
| **Brief Description:** | This use case describes the user accessing all the Notifications. |
| **Relationships:**  **Include:** | Association: Admins and Work Center leads  Include: Retrieving data from the database. |
| **Actor Action** | 1. User is authenticated into their appropriate page. |
| **System Response** | 1. Depending on the type of user, the system will retrieve all notifications or all notifications for the current user’s work center. |

Figure 5 – Use case: View all notifications.

**Use Case #4: View All Work Orders**

Precondition: User is authenticated and is either Admin or Work Center Lead

Postcondition: All work orders are displayed.

Limitation: Network connection required

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 4.1 User retrieves all Work Orders after being authenticated: | 4.2 Depending on user type, the System will display: |
| 4.1.1 If User is Admin | 4.2.1 System retrieves all work orders from all work centers to the user. |
| 4.1.2 If User is a Work Center Lead | 4.2.2 System retrieves only work orders assigned to the user’s work center. |

|  |  |
| --- | --- |
| **Use Case Name:** | View All Work Orders |
| **Primary Actor:** | Admins and Work Center leads |
| **Stakeholders and Interests** | Admins and Work Center leads – need to retrieve all the work orders to perform required operations in the system. |
| **Brief Description:** | This use case describes the user logging in to the application and being redirected to the page appropriate to them. |
| **Relationships:**  **Include:** | Association: Admins and Work Center leads  Include: Retrieving data from the database. |
| **Actor Action** | 1. User is authenticated into their appropriate page. |
| **System Response** | 1. Depending on the type of user, the system will retrieve all work orders or all work orders for the current user’s work center. |

figure 6 – Use case: View all Work orders.

**Use Case #5: Create & assign a Work Order**

Precondition: User is authenticated and is a Work Center Lead

Postcondition: A new notification entity is assigned to a specific work center

Limitation: Network connection required

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 5.1 Work Center Lead select a notification | 5.2 If selected notification does not have a corresponding work order, System retrieves notification data and displays it the Work Order creation form. |
| 5.2 Work Center Lead fills up the rest of the Work Order creation form. | 5.3 System takes and passes the input from the user as parameters to create a new Work Order entity, then displays the new entity in the work center page. |
| 5.4 Work Center Lead navigates to the page of the newly created work order. | 5.5 System displays all the Work Order information including the Notification information. |
| 5.6 Work Center Lead creates and assigns a new operation for the newly created work order to a field worker that belongs to the work center. | 5.7 System takes and passes the user input as parameters to create a new operation for the specific work order. |

|  |  |
| --- | --- |
| Use Case Name: | Create and Assign a Work Order |
| Primary Actor: | Work Centre Lead |
| Stakeholders and Interests | Work Centre Lead/Field worker – need to have a work order assigned to be able to complete the work order and close the notification. |
| Brief Description: | This use case describes the Work Center Lead creating a new work order from a selected notification and then creates and assign operations to workers from work center. |
| Relationships:  Include: | Association: Work Centre  Include: Store work order data |
| Actor Action | 1. User creates a work order from selected notification. 2. User creates operations to new work order. |
| System Response | 1. System retrieves the user inputs as parameters to create the work order entity and the corresponding operations. |

Figure 7 – Use Case: Create and assign a work order.

**Use Case #6: Retrieve Work Order**

Precondition: User is authenticated into the application

Postcondition: User sees a full list of assigned work orders

Limitation: Network connection required or device must have memory cache capabilities.

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 6.1 User presses button to request all assigned work orders | 6.2 System retrieves and displays assigned work orders from the database. |

|  |  |
| --- | --- |
| **Use Case Name:** | Retrieve Assigned Work Order |
| **Primary Actor:** | Work Centre Lead/Field Worker |
| **Stakeholders and Interests** | Work Centre Lead/Field worker – needs to get the work order into the application in order to begin the process of completing it |
| **Brief Description:** | This use case describes how the work order data is accessed by a field worker. |
| **Relationships:**  **Include:** | Association: Work Centre Lead and Field Worker  Include: Storing data from database |
| **Actor Action** | 1. The user receives the work orders assigned to them by the work center lead. |
| **System Response** | 1. System retrieves the work order information. |

Figure 8 - Use case table: Retrieve work order

**Use Case #7:**  **Start work order**

Precondition: User has navigated to the function location and is ready to begin work

Work order status must be “Assigned” or “Paused”

Postcondition: Work order and corresponding notification is changed to “Started” status

Work Order and Notification entities are updated in the database.

Limitation: Device must have memory cache capabilities.

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 7.1 User clicks on the Start button in the assigned work order page. | 7.2 System updates work order status to “Started”.  7.3 System sends status change to the database and update the Work Order and Notification entities.  -or-  7.4 If there is no network connection, work order and notification status are cached for later database update. |

|  |  |
| --- | --- |
| **Use Case Name:** | Start Work Order |
| **Primary Actor:** | Field worker |
| **Stakeholders and Interests** | Field worker – upon reaching destination, work order operations can begin.  Database – will update the work order data based on the field worker updates. |
| **Brief Description:** | Field worker begins to perform operations of the work order |
| **Relationships:**  **Association:**  **Include:**  **Extend:** | Association: Field Worker  Include: Close work order  Extend: Pause work order |
| **Actor Action** | 1. User clicks Start in the assigned Work Order page. |
| **System Response** | 1. Update work order status in database 2. Display a confirmation message on the screen |

Figure 9- Use case table: Start work order

**Use Case #8:**  **Pause work order**

Precondition: Work order status is active.

Condition exists that prevents Field Worker(s) completing the work order.

Postcondition: Work order status updated to “Paused”.

Limitation: Device must have memory cache capabilities.

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 8.1 When not able to complete the Work Order, User clicks on “Paused” to pause the Work Order. | 8.2 System updates work order status to “Paused”.  8.3 If there is a network connection, work order status is updated in database  -or-  8.4 If there is no network connection, work order status is cached for later database update |

|  |  |
| --- | --- |
| **Use Case Name:** | Pause Work Order |
| **Primary Actor:** | Field Worker |
| **Stakeholders and Interests** | Field Worker – will need to ensure the progress is updated properly in the database.  Database – will update the work order data based on the field worker updates |
| **Brief Description:** | Field Worker is not able to complete/continue on current Work Order therefore they have to pause the job. |
| **Relationships:**  **Association:**  **Include:**  **Extend:** | Association: Field Worker  Include: Update Data from Database.  Extend: Start Work Order |
| **Actor Action** | 1. User pauses the work order. |
| **System Response** | 1. Update the tracked work order data into the database. 2. Display the confirmation message on the screen |

Figure 10 - Use case table: Pause work order

**Use Case #9:**  **Close Work Order**

Precondition: Work Order Status is “Active”

All Operations in Work Order are finished.

Postcondition: Work Order and Notification status are updated to “Closed” in the database.

Limitation: Device must be mobile with memory cache capabilities

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 9.1 User completes the last Operation from current Work Order.  9.3 User clicks the Close button in the Work Order page. | 9.2 If there is internet connection, System updates the status of the last Operation to “Finished”.  9.4 System checks that all the operations are complete and then changes the status of the work order and notification object to “Closed” and updates in database.  -or-  9.5 If there is no network connection, work order status is cached for later database update |

|  |  |
| --- | --- |
| **Use Case Name:** | Close Work Order |
| **Primary Actor:** | Field Worker |
| **Stakeholders and Interests** | Field Worker – all the updated data by the field worker will be stored in work order object to update in the database.  Database – will update the work order data based on the field worker updates |
| **Brief Description:** | Field Worker finishes the last Operation in the Work Order and then closes the Work Order. |
| **Relationships:**  **Association:**  **Include:** | Association: Field Worker  Include: Store Data in Database  Update Data in Database |
| **Actor Action** | 1. Field Worker finished the last Operation not completed in the Work Order. 2. Field Worker closes the Work Order. |
| **System Response** | 1. Update the database will all details of the updated work order and corresponding Notification. 2. Display the confirmation message on the screen. |

Figure 11 - Use case table: Close work order

**Use Case #10:**  **Store Data**

Precondition: Authenticated user performs a transaction such as Inserting or Updating Data.

Postcondition: New or updated data is store in the database and accessible for users.

Limitation: Network connection required.

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 10.1 If User is Admin, User Creates a new Notification object or a new User. | 10.2 System takes the user input as parameters for the object creation and stores this new data in the database. |
| 10.3 If User is Work Center Lead, User Creates a new Work Order object from a Notification or commits changes to an existing Work Order/Operation. | 10.4 System takes the user input as parameters for the object creation or update and stores this new or modified data in the database. |
| 10.5 If User is a Field Worker, User commits changes to the Work Order object | 10.6 After updating the data, the System stores the modified data in the database. |

|  |  |
| --- | --- |
| **Use Case Name:** | Store Data |
| **Primary Actor:** | All Users |
| **Stakeholders and Interests** | All Users – Perform transactions such as Inserting or Updating data to complete their jobs.  Database – Receives and stores new/modified data. |
| **Brief Description:** | After a user has committed a change to the data or created new data, this data is stored in the database. |
| **Relationships:**  **Association:** | Association: Database |
| **Actor Action** | 1. User makes changes to the data in the System, such as creating data or modifying existing data |
| **System Response** | 1. System stores the new or modified data in the database. |

Figure 12- Use case table: Store Data

**Use Case #11:**  **Retrieve Data**

Precondition: User is authenticated.

Postcondition: All appropriate data is displayed to user.

Limitation: Network connection required.

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 11.1 After being authenticated, User enters their corresponding page. | 11.2 System retrieves the corresponding data from the database. |

|  |  |
| --- | --- |
| **Use Case Name:** | Retrieve Data |
| **Primary Actor:** | All Users |
| **Stakeholders and Interests** | All Users – need access to data to perform their required jobs.  Database – Retrieves the data. |
| **Brief Description:** | After being authenticated, the user is shown the data they are authorized to see. |
| **Relationships:**  **Association:**  **Extends:** | Association: Database  Extends: Store Data in Database. |
| **Actor Action** | 1. User logs in to their authorized page. |
| **System Response** | 1. System retrieves the data from the database. |

Figure 13- Use case table: Retrieve Data

**Use Case #12:**  **Update Data**

Precondition: User is authenticated and has made changes in the data.

Postcondition: Modified data replaces old data.

Limitation: Network connection required.

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 12.1 User commits modifications done to data, such as changing the status in a Work Order/Notification object or delete an existing entity. | 12.2 If modifying existing data, System replaces the old data with the new one in the database. 12.3 If deleting existing data, System removes the data from the database |

|  |  |
| --- | --- |
| **Use Case Name:** | Update Data |
| **Primary Actor:** | All Users |
| **Stakeholders and Interests** | All Users – need that all the data is accurate and up to date.  Database – Retrieves the data. |
| **Brief Description:** | After a user makes changes to existing data, such as modifying or deleting data, these changes are then made to the Database. |
| **Relationships:**  **Association:**  **Include:** | Association: Database  Include: Store Data in Database. |
| **Actor Action** | 1. User makes changes to existing data. |
| **System Response** | 1. System updates the database. |

Figure 14- Use case table: Update Data

Non-Functional Requirements

* Reliability: Application flow must be simple and easy to navigate, with clear goals for the user in mind. The application will be tested for usability. Reliability is measured by the data handling and tracking the progress which will update the database.
* Performance constraints: Our application must perform in real time, allowing the worker to update the job details or its progress immediately within the database. In the event of lost connection, the updated data will be cached, and the cached data will be sent to the database once the connection is restored.

System Interface Requirements

* Our application will be deployed to AWS and work order data will be generated within the application, which will be accessible on desktop and mobile devices.

Maintainability Requirements

* Maintainability: There will be extensive documentation of bug fixing, security vulnerabilities, replacement capabilities and possible enhancements to the software, to facilitate a seamless implementation into the main platform. The software will be compatible with the existing system.
* Backup/restore will be managed via cache capture for individual user activities. Long term backup should be established on a per client basis; however, weekly backups of the database are recommended.

Usability Requirements

* Given the wide variety of clients that could potentially make use of this application, the profile of the users that will be accessing it may vary from inexperienced to tech-savvy. They may represent a wide age demographic, ranging from young and new to their field to approaching retirement and set in their ways. Older users who may have less experience with technology might be less willing to learn new technology.
* Users may also be on billable time so the user interface should be simple and efficient to use to avoid unnecessary time spent trying to navigate the application.
* The driving motivator behind development of this application is based on the experiences and observations of our client that many field workers have faced difficulty with data entry into mostly manual systems, especially after finishing a job. Our interface should minimize, if not eliminate, this limitation.

Problem Domain Class Diagram

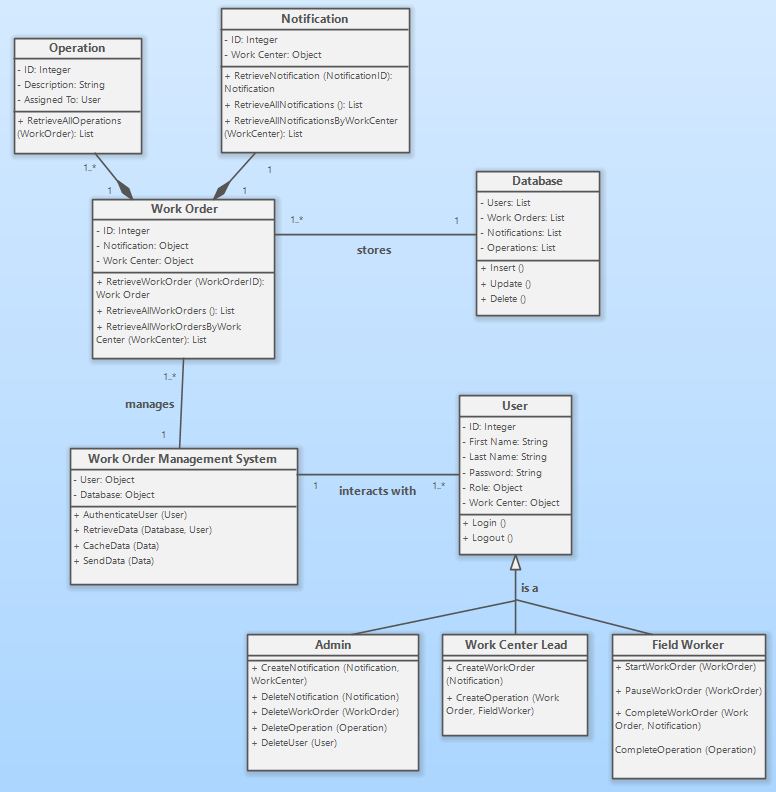


Figure 15 - Class diagram

System Design

Layered Architecture

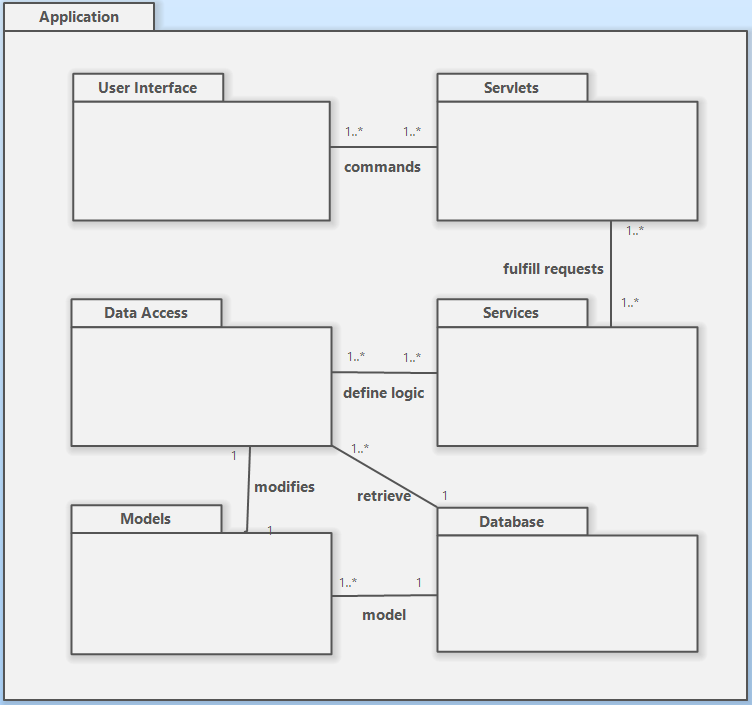


Figure 16 – Package Diagram

Hardware ArchitectureThe architecture of our application considers the requirements of the user, the developers and the future owner of the software product.   
User requirements focus on usability, which includes time for updating information on the pages based on changes made and the ability to switch between pages.  
Developer’s requirement center on the performance, scalability and speed of development. This means easily introducing new features, rearrange the structure of the code and minimize the server response time, while providing consistent and available data.   
Future owner of this software product concentrates on its functioning, such as maintenance, network infrastructure and security.  
To develop our Java-based web application, we put it in a dedicated server, MySQL server, which contains the tables and schemas we are to use to build and store data. Like most Java web applications, ours is not running directly on the server but inside a web container on the server. The web container we are using is Tomcat, which helps support the execution of Java Servlets and JavaServer Pages. We are also deploying a Java EE container, which supports additional functionality.

This is graphically explained in the following diagram:

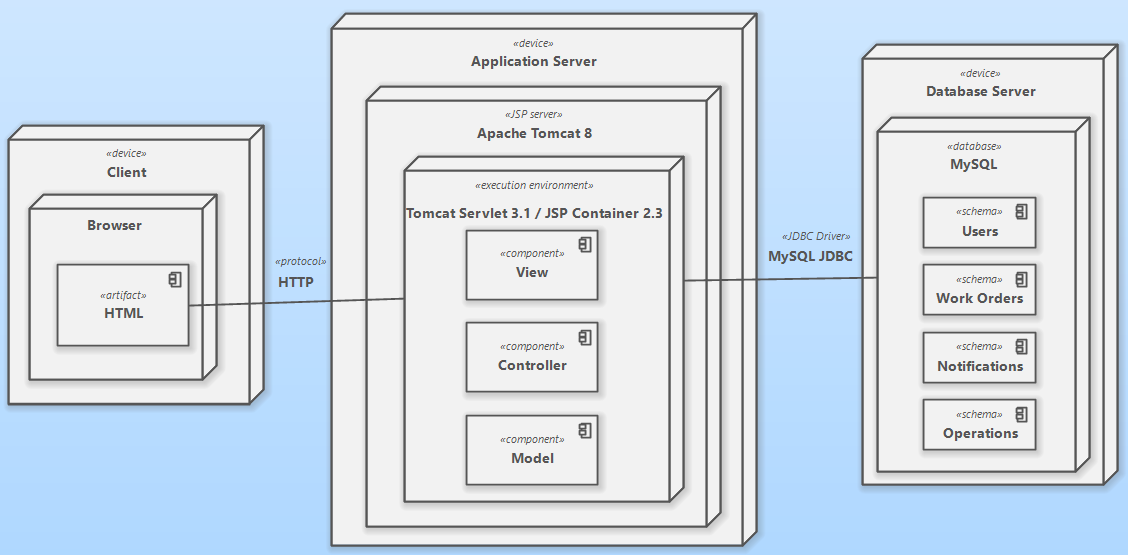


Figure 17 – Deployment Diagram

In the deployment diagram (Figure 17), the process can be defined as:

1. The user sends the command to the server through the internet, using the browser.
2. The web server is responsible for forwarding the command to the desired server.
3. The server finds the results of the requested commands by the user, wither by processing data or by querying the database.
4. The web application delivers the processed information to the server.
5. The server provides the user with the requested data.

Hardware Platform

Laptop:

* Processor: 32 bit
* Memory: 256MB RAM, 500MB Hard Disk (minimum)

512MB RAM, 1GB Hard Disk (recommended)

Personal Device:

* Processor: XScale
* Memory: 64MB RAM, 128MB fast flash ROM (minimum)

128MB RAM, 128MB fast flash ROM (recommended)

Software Platform

Laptop:

* *MS Windows XP*
* *MS Windows Vista*
* *MS Internet Explorer v6.0/7.0*
* *SAP MaxDB 7.6*
* *Sun Java RunTime Environment 6.0 (1.6.0\_07 and higher)*

Personal Device:

* Microsoft Windows Mobile 2003 SE, Microsoft Windows Mobile 5.0, and Microsoft Windows Mobile 6.0.
* Microsoft Pocket Internet Explorer
* NSIcom crEme 4.1

The server we are using is MySQL and the web server is Apache Tomcat 8.0.27.0, type TomEE.

Interaction Model

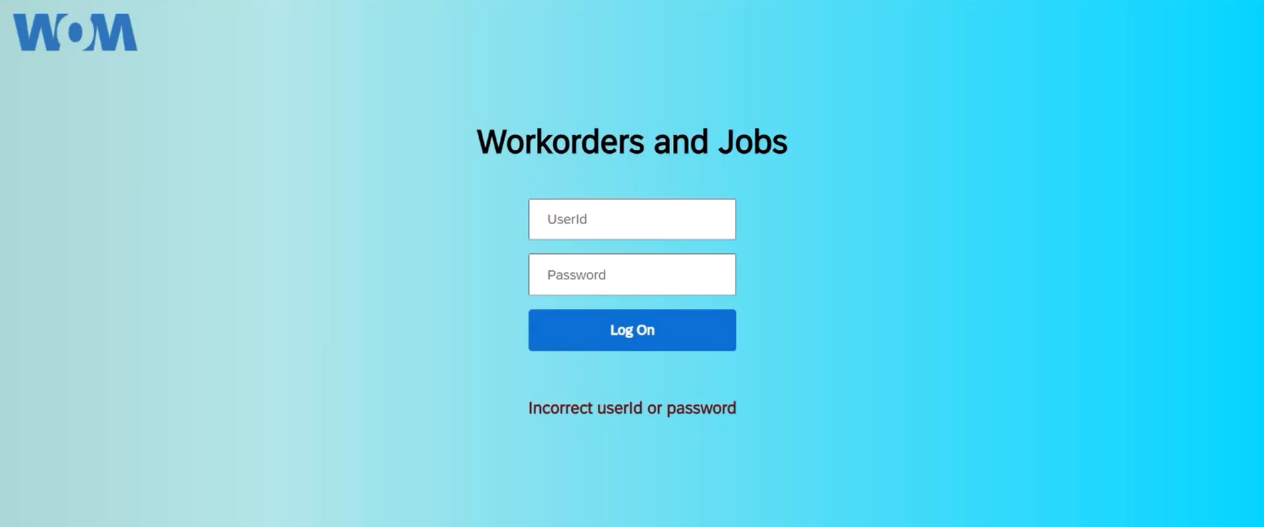


Figure 18 – Log in page

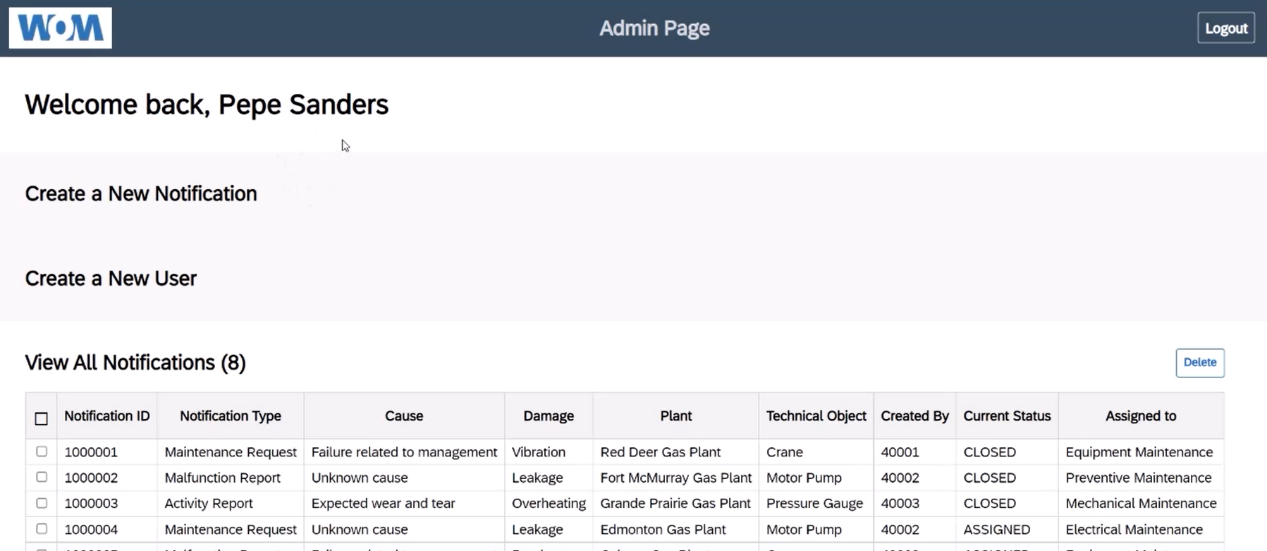


Figure 19 – Admin Page

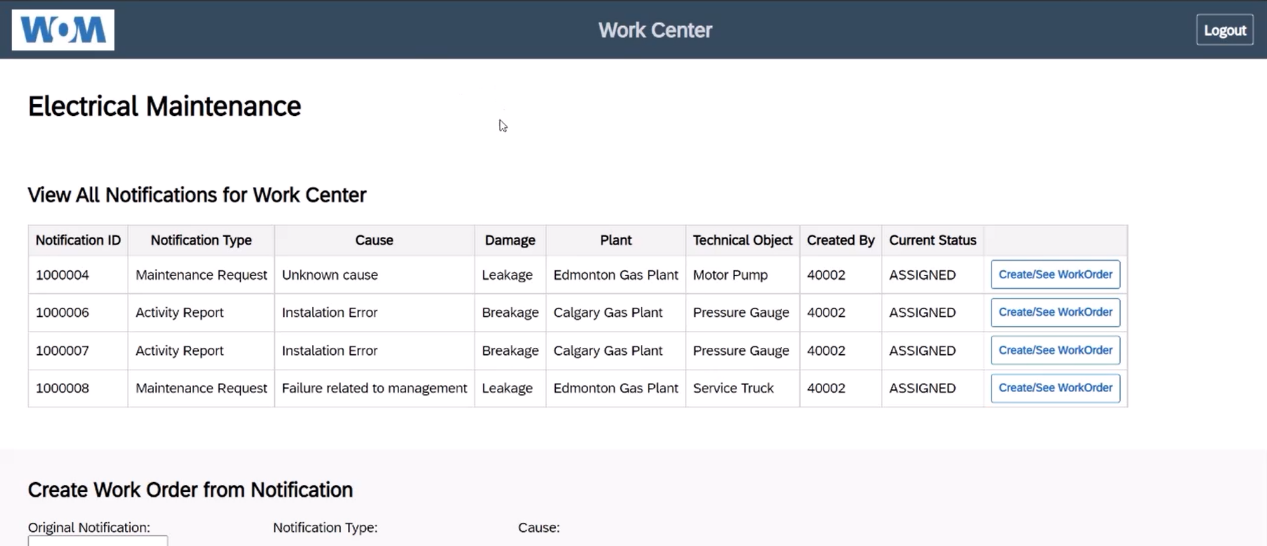


Figure 20 – Work Center Page

Persistence Model

For our Capstone Project, we will be utilizing the EclipseLink JPA object-relational persistence model along with a MySQL Database server and an Apache TomCat web server in order to provide the framework of our data management services.

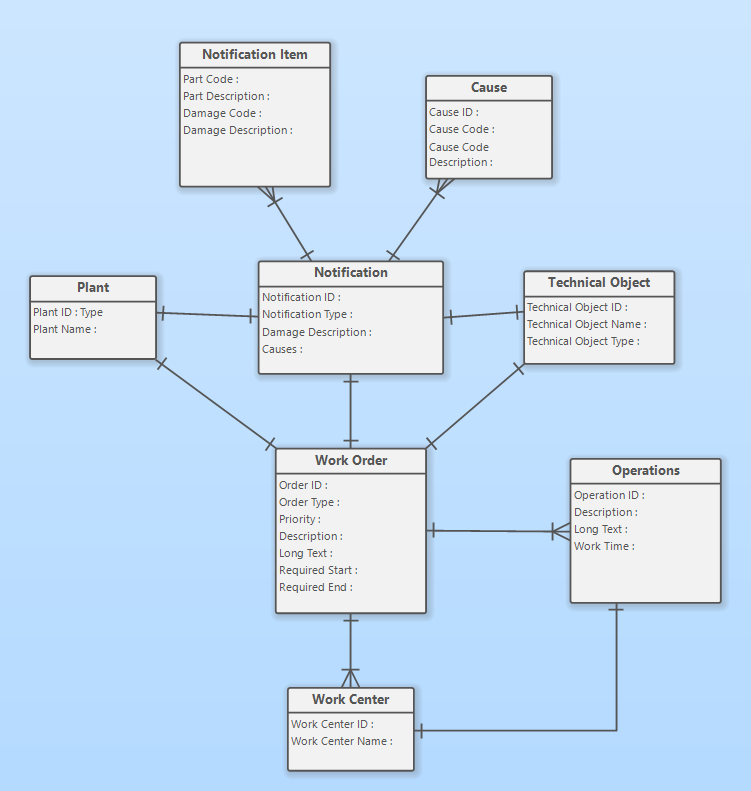


Figure 21 - Conceptual Schema ERD (as per Systems Analysis and Design course)

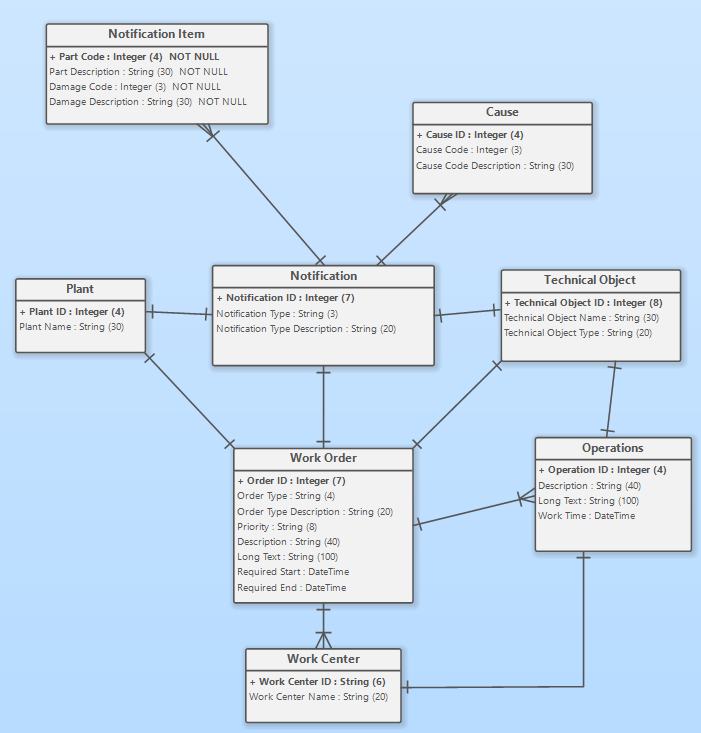


Figure 22 - Internal Schema ERD (3NF physical model as per Database courses)

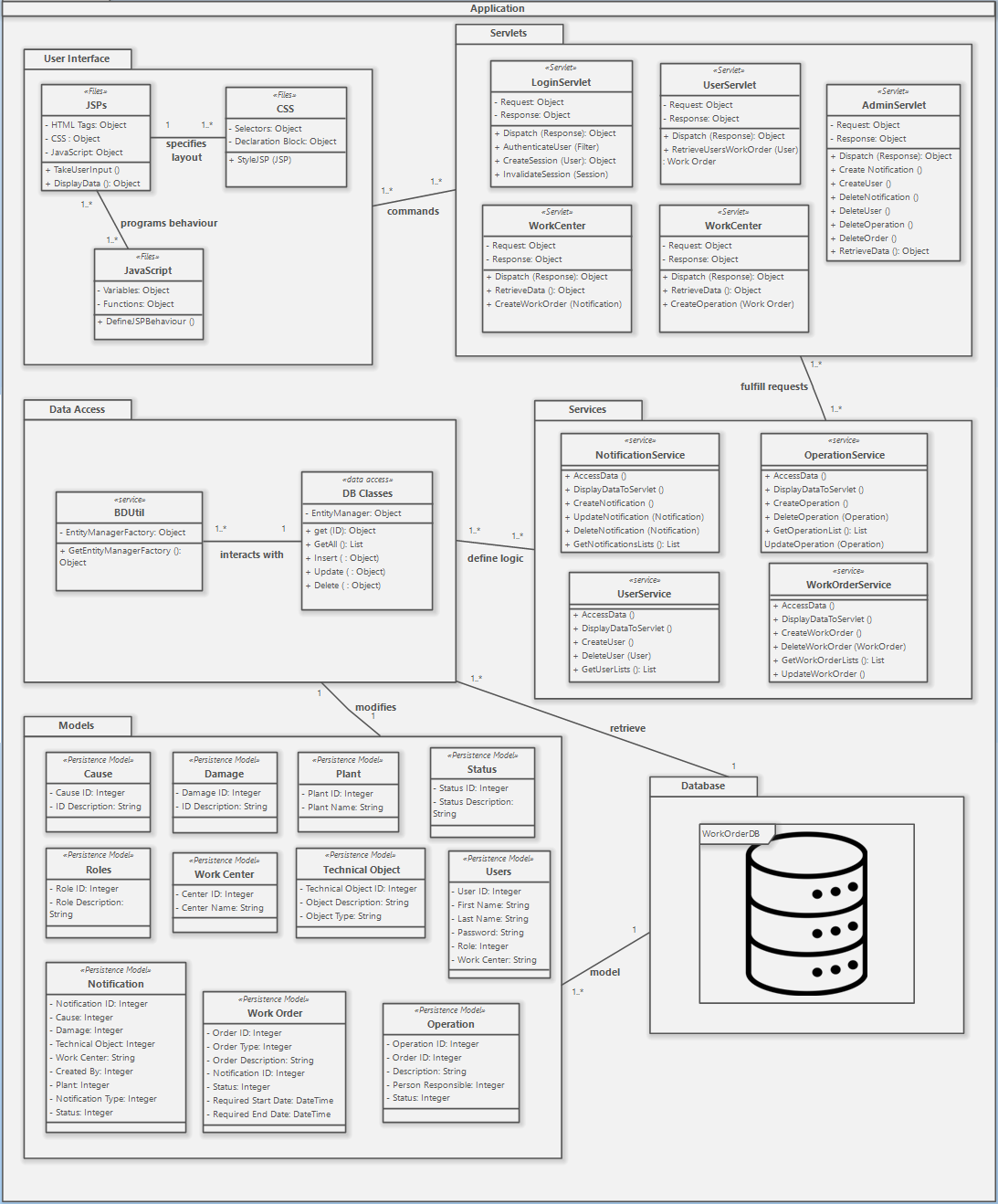
Class Diagram

Figure 23 – Complete Application – Detailed Class Diagram

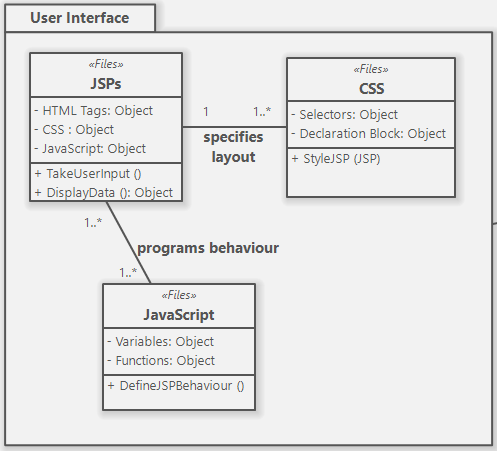


Figure 24 – User Interface layer/Package within the application

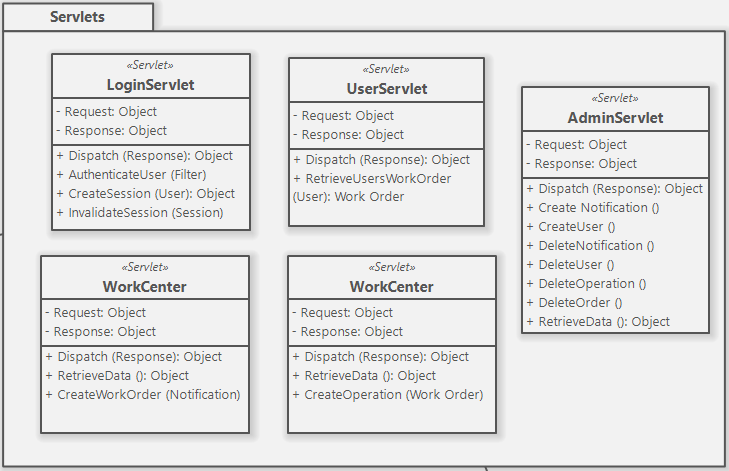


Figure 25 – Servlets layer/package within the application

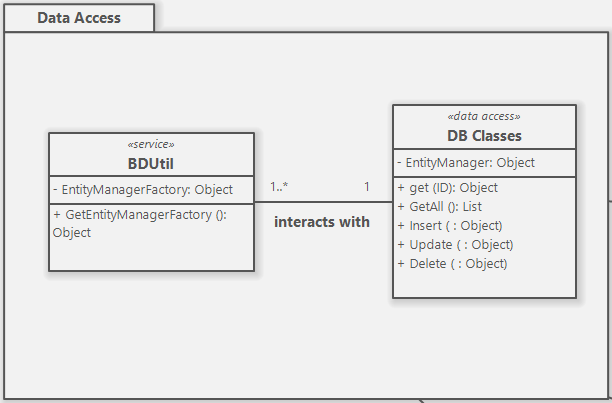


Figure 26 – Data Access layer/package within the application

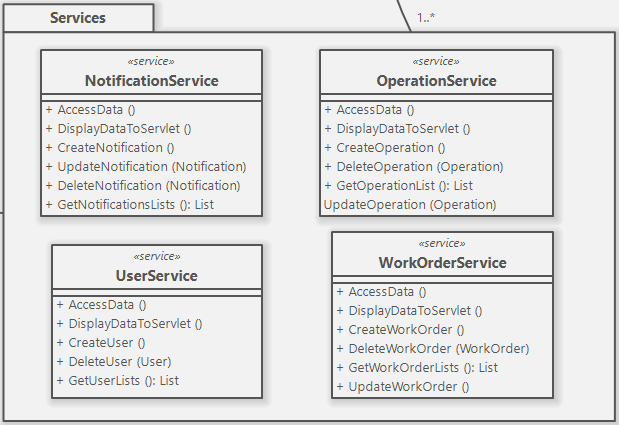


Figure 27 – Services Package within Application Layer

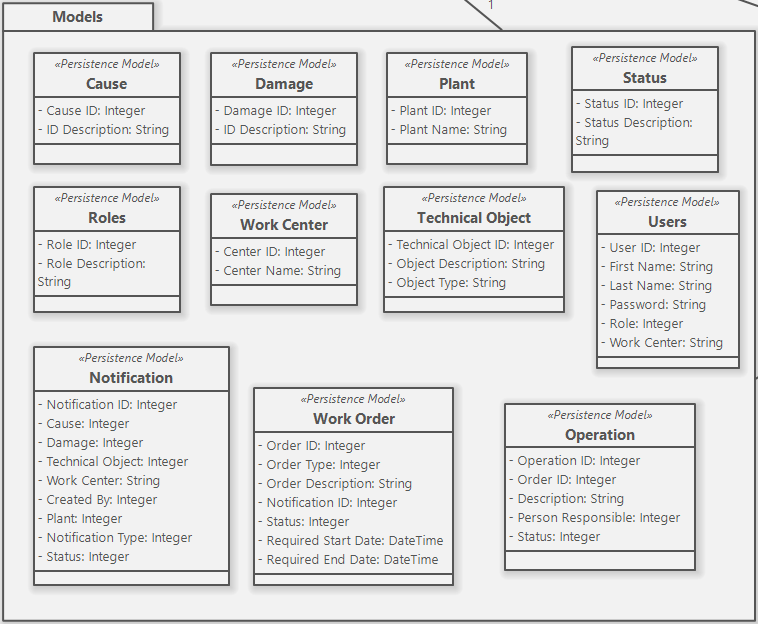


Figure 28 – Persistence Models layer/package within the application

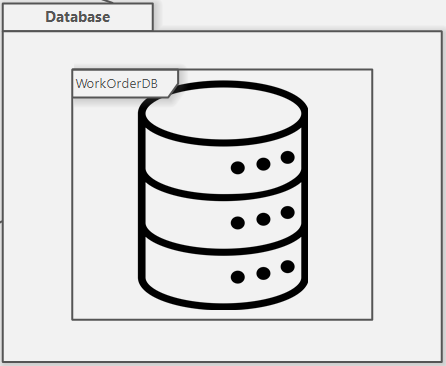


Figure 29 – Database layer/package within the application

Interaction Sequence Diagrams

## 

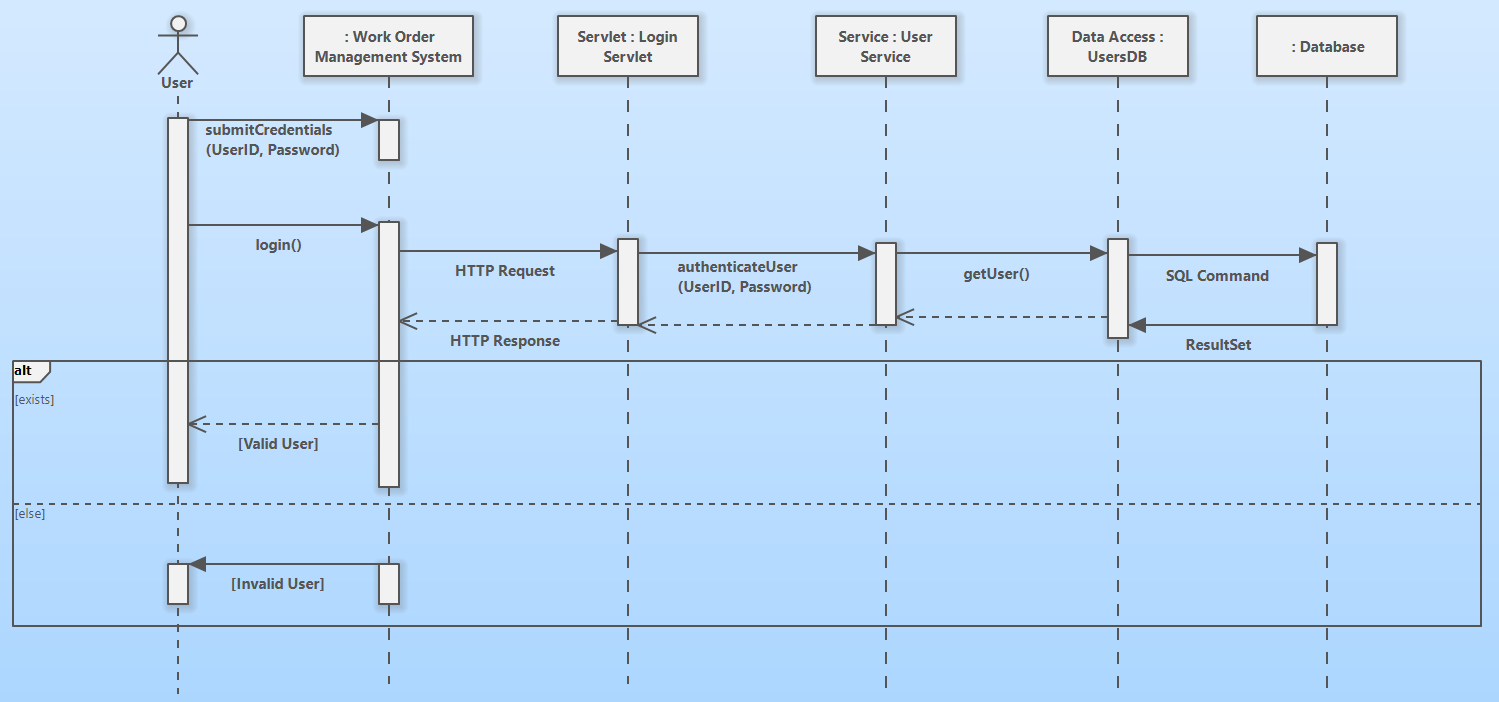


Figure 30 – Login Use Case Sequence Diagram

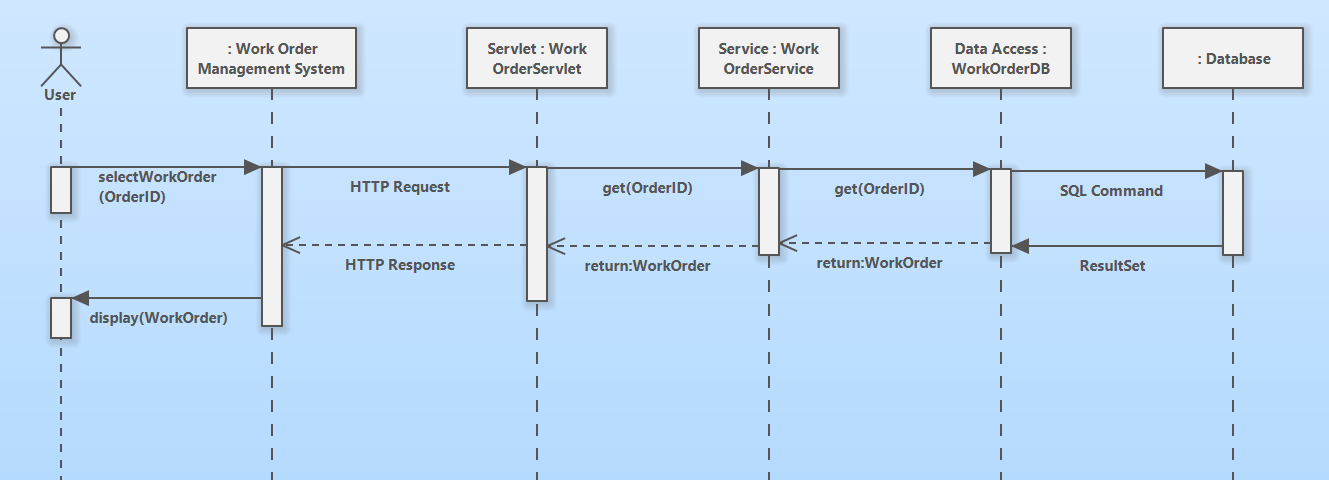


Figure 31 – Retrieve and View Work Order Use Case Sequence Diagram

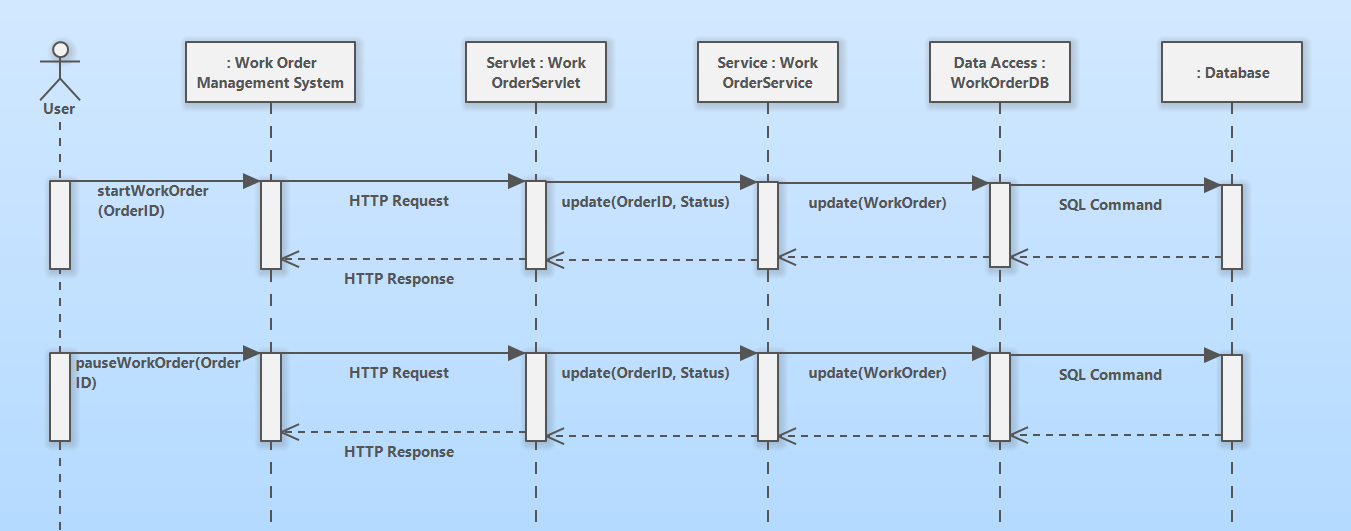


Figure 32– Start and Pause Work Order Use Case Sequence Diagram

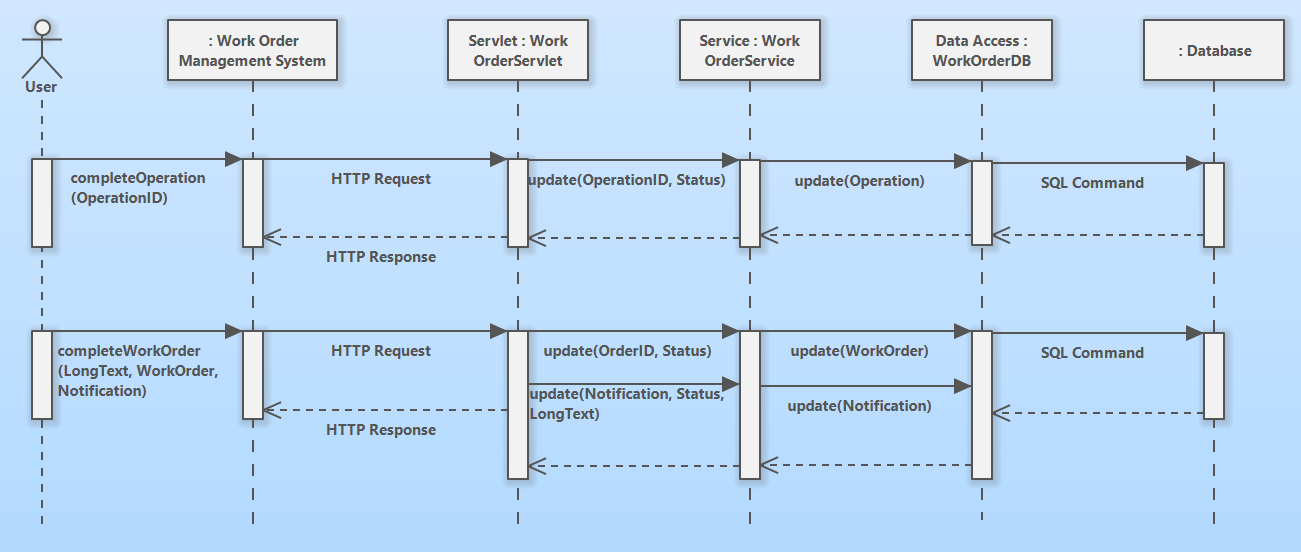


Figure 33 – Complete Work Order Use Case Sequence Diagram

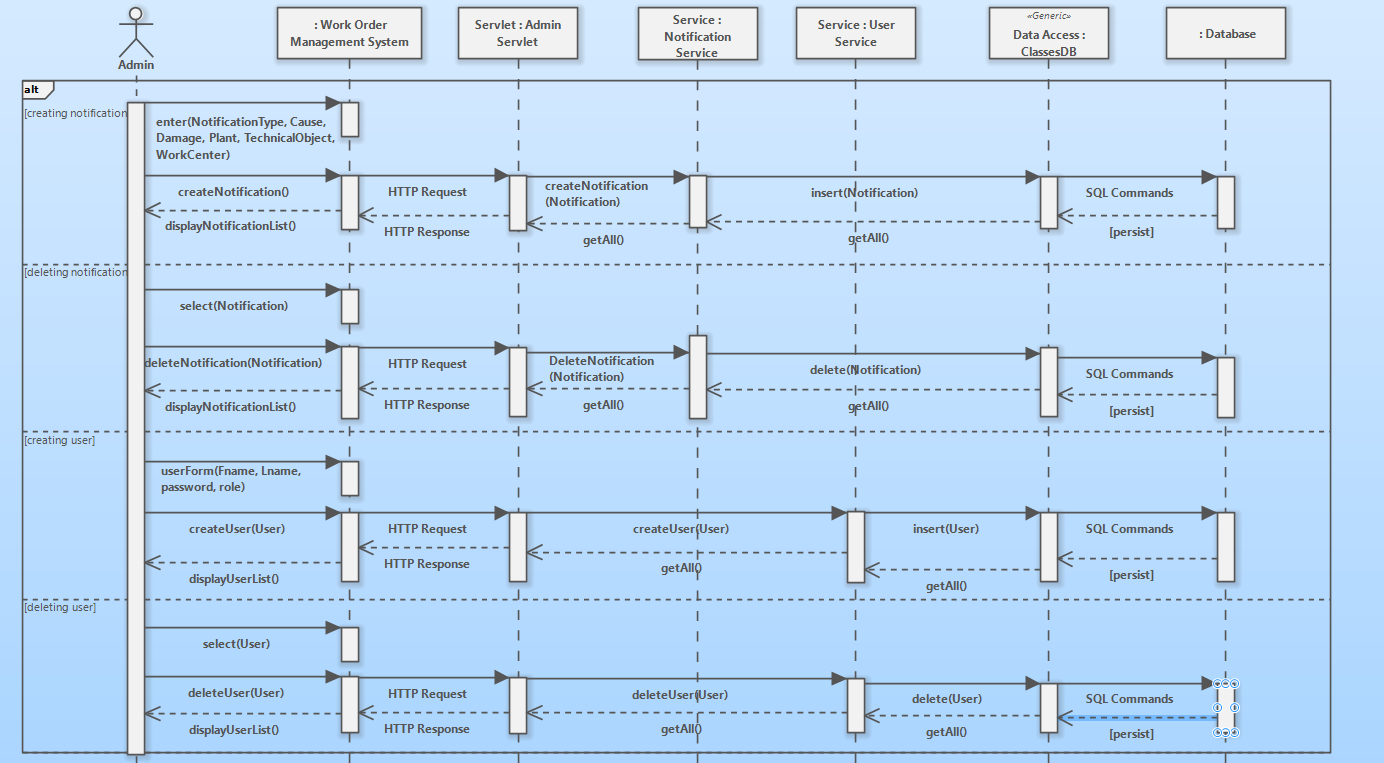


Figure 34 – Start to Finish Application Flow Sequence Diagram – ADMIN USER

State-chart Diagrams

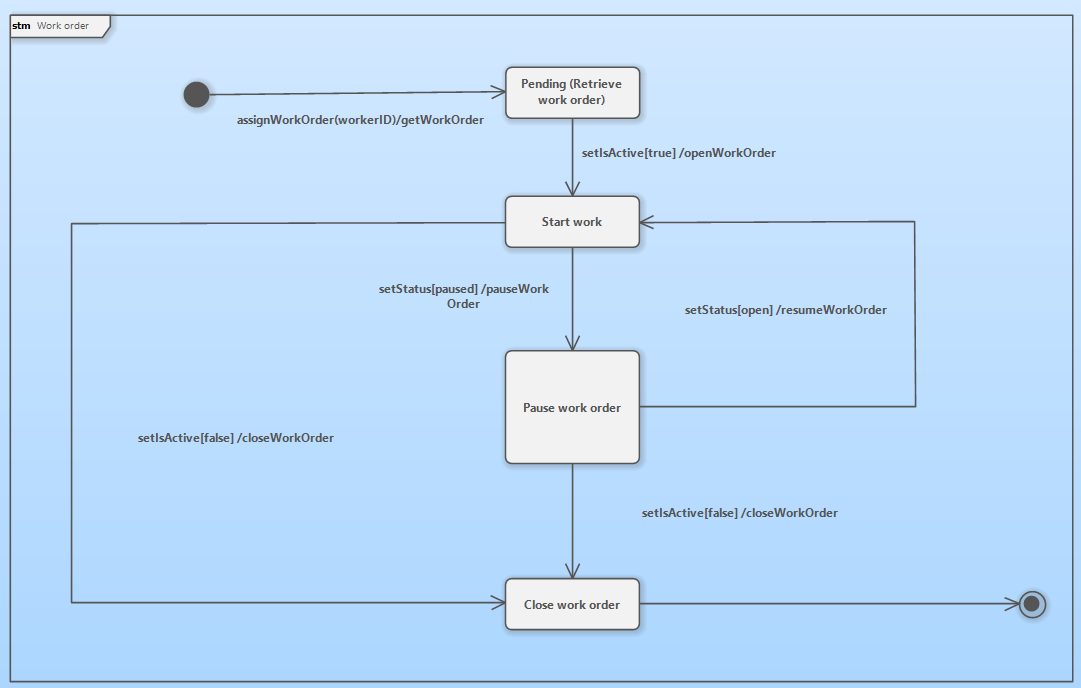


Figure 35 - State Chart Diagram

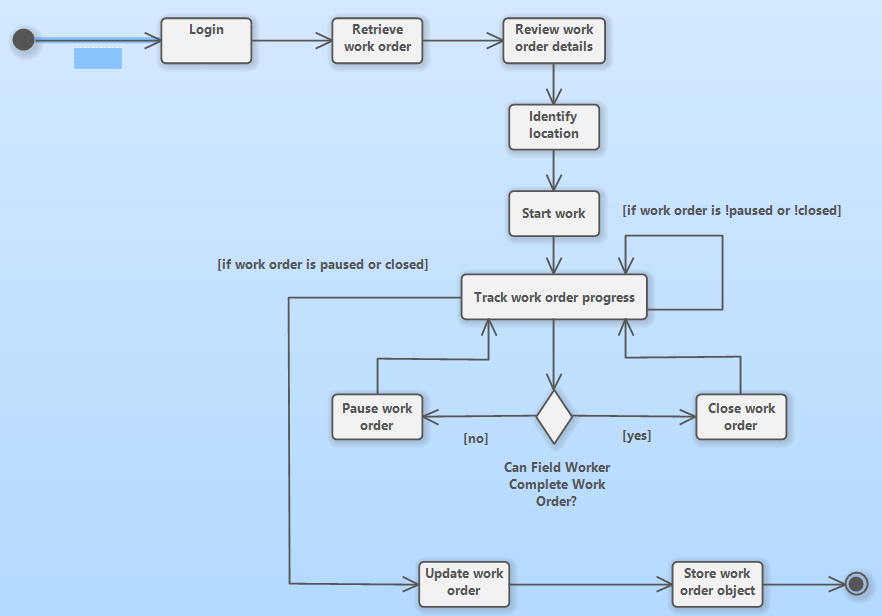
Activity Diagrams 

Figure 36 - Activity diagram

Data Dictionary of Problem Domain and Application Domain

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Worker** | | | | | |
| ***Field Name*** | ***Component*** | ***Data / Return Type*** | ***Validation*** | ***Description*** | ***Example*** |
| *workerName* | *attribute* | *String* |  | *Worker's name attribute* | *John Smith* |
| *workerID* | *attribute* | *Integer* | *WorkerID > 0* | *Worker's ID (Unique)* | *1234* |
| *worCentreID* | *attribute* | *Integer* | *WorkCentreID > 0* | *Work Centre ID (Unique)* | *5678* |
| *workerPhone* | *attribute* | *String* |  | *Worker's phone number* | *403-123-4586* |
| *workerEmail* | *attribute* | *String* | *WorkerEmail.contains(“@”) == true* | *Worker's email address (unique)* | *Abc@abc.ca* |
| *workerHireDate* | *attribute* | *Date/Time* | *DD/MM/YYYY* | *Worker's hired data* | *08/03/2020* |
| *getWorkerName* | *operation* | *String* |  | *Getter method to get worker’s name* | *f1.getWorkerName();* |
| *setWorkerName* | *operation* | *void* |  | *Setter for worker’s name* | *f1.setWorkerName(“John”);* |
| *getWorkerID* | *operation* | *Integer* |  | *Getter for workerID* | *f1.getWorkerID();* |
| *setWorkerID* | *operation* | *void* |  | *Setter for workerID* | *f1.setWorkerID(1234);* |
| *getWorkerPhone* | *operation* | *String* |  | *Getter for worker’s phone number* | *f1.getWorkerPhone()* |
| *setWorkerPhone* | *operation* | *void* |  | *Setter for worker’s phone number* | *f1.setWorkerPhone(4032341234)* |
| *getWorkerEmail* | *operation* | *String* |  | *Getter for worker’s email* | *f1.getWorkerEmail()* |
| *setWorkerEmail* | *operation* | *void* |  | *Setter for worker’s email* | *f1.setWorkerEmail("b@b.com”)* |
| *getWorkerHireDate* | *operation* | *Date/Time* |  | *Getter for worker’s hired date* | *f1.getWorkerHireDate()* |
| *setWorkerHireDate* | *operation* | *void* |  | *Setter for worker’s hired date* | *f1.setWorkerHireDate(09/02/2021)* |
| *getWorkCentreID* | *operation* | *Integer* |  | *Getter for work centre ID* | *f1.getWorkCentreID()* |
| *setWorkCentreID* | *operation* | *void* |  | *Setter for work centre ID* | *f1.setWorkCentreID(1972)* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Utility** | | | | | |
| ***Field Name*** | ***Component*** | ***Data / Return Type*** | ***Validation*** | ***Description*** | ***Example*** |
| *workerObject* | *attribute* | *Object* |  | *Worker's detail is stored in the object type* | *{workerName: ‘John Smith’, workerID: 1234, …}* |
| *workOrderObject* | *attribute* | *Object* |  | *Worker order object is stored in the object type* | *{Notification:new Notification(), operationList: new OperationList()….}* |
| *getWorkerObject* | *operation* | *Object* |  | *Getter for worker object* | *utility.getWorkerObject();* |
| *setWorkerObject* | *operation* | *void* |  | *Setter for worker object* | *utility.setWorkerObject(w2);* |
| *getWorkOrderObject* | *operation* | *Object* |  | *Getter for workOrder object* | *utility.getWorkOrderObject();* |
| *SetWorkOrderObject* | *operation* | *void* |  | *Setter for workOrder object* | *utility.setWorkOrderObject(wo1);* |
| *CacheData()* | *operation* | *Object* |  | *Load stored the worker order detail* | *utility.cacheData();* |
| *SendData(workOrderID)* | *operation* | *Object* |  | *Send updated worker order data* | *utility.sendData(1234);* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **WorkOrder** | | | | | |
| ***Field Name*** | ***component*** | ***Data / Return Type*** | ***Validation*** | ***Description*** | ***Example*** |
| *notification* | *attribute* | *Object* |  | *Notification object data* | *{Location: new Location(), Description: …}* |
| *operationList* | *attribute* | *Object* |  | *Operation object data* | *{workCentreID: 5678, operationDetail: … }* |
| *Start\_time* | *attribute* | *Date/Time* | *Hour:Min, DD/MM/YYYY* | *Worker's start working time* | *19:30, 01/02/2021* |
| *End\_time* | *attribute* | *Date/Time* | *Hour:Min, DD/MM/YYYY* | *Worker's end working time* | *23:30, 01/02/2021* |
| *workOrderID* | *attribute* | *Integer* | *WorkOrderID > 0* | *Work order ID (unique)* | *456891* |
| *assetToBeRepaired* | *attribute* | *String* |  | *Description for the asset should be repaired* | *"gas pipe in section A”* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Notification Object** | | | | | |
| ***Field Name*** | ***component*** | ***Data Type*** | ***Validation*** | ***Description*** | ***Example*** |
| *Location* | *attribute* | *Object* |  | *Details of work site location is stored in the object* | *{region: “Okotoks”, siteNumber: “O-2”, section: “B”, latitude: ….}* |
| *Description* | *attribute* | *String* |  | *Description of notification* | *"work order is closed”* |
| *Cause* | *attribute* | *String* |  | *Description of the cause of the event* | *“Field worker (Id:1234) closed the work order”* |
| *Status* | *attribute* | *Boolean* |  | *Whether the notification is handled or not* | *true* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operation** | | | | | |
| ***Field Name*** | ***component*** | ***Data Type*** | ***Validation*** | ***Description*** | ***Example*** |
| *workCentreID* | *attribute* | *Integer* | *WorkCentreID > 0* | *Work Centre ID (Unique)* | *5678* |
| *operationDetails* | *attribute* | *String* |  | *Operation details* | *"repair gas pipes in the section B of the site O-2”* |
| *crewSize* | *attribute* | *Integer* | *CrewSize > 0* | *The number of crew members for this operation* | *10* |
| *operationID* | *attribute* | *integer* | *OperationID > 0* | *Operation unique ID number* | *0012* |
| *operationComplete* | *attribute* | *Boolean* |  | *Boolean to check whether the operation is completed or not* | *true* |

Testing

Test Plans

|  |  |
| --- | --- |
| **Client**: N/A |  |
| **System**: Login |  |
| **Objective**:  Take the user’s credentials input and validate it to redirect the user to the page they are authorized. | |
| **Tester Name:** Laura Diaz | |
| **Developer Name:** Trevor Erixon, Jaehan Kim, Maria Laura Diaz Pena, Zennon Weleschuk, Rylan Cook, Saeid Ghavamabadi | |

|  |  |  |
| --- | --- | --- |
| **Component**: Accept Valid User | | **Test Date: 07/15/2021** |
| **Test** | **Description** | **Actual Result** |
| 1. | ***Login.jsp*** is the first page shown to the user.  **Procedure**:  Open the program and enter your credentials on the *User ID* and *password* fields on the ***Login*** page.  **Result**:  The credentials are displayed in the input fields, the *password* is censored. | The User credentials are displayed in the input fields, the password is censored. |
| 2. | User can now click on ***“Login”*** button.  **Procedure**:  Click on the ***“Login”*** button.  **Result**:  ***Login*** page redirects the user to the appropriate page. | * If user is Admin, the admin.jsp is shown. * If user is advanced user, the workcenter.jsp is shown. * If user is Basic User, user.jsp is shown |
| **Comments**:  Successful test. | | |
| **Client Signature of Approval**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |
| **IT Use Only:**  Redesign Complete:  Yes  No  Retest Complete:  Yes  No  Details if Retest not complete: | | |
| **Date: 07/15/2021** | | |
| **Developer Name and Signature:** | | |

|  |  |  |
| --- | --- | --- |
| **Component**: Reject Invalid User | | **Test Date:** |
| **Test** | **Description** | **Actual Result** |
| 1. | ***Reject invalid user***  **Procedure**:  Open the program and enter invalid credentials on the *User ID* and *password* fields on the ***Logi*n** page.  **Result**:  Credentials are displayed in the login page. | Credentials are displayed in the login page |
| 2. | User can click on ***“Login”*** button.  **Procedure**:  Click on the ***“Login”*** button.  **Result**:  The message “Incorrect userID or password” is displayed to the user. | The message “Incorrect userID or password” is displayed to the user. |
| 3. | User inserts null values in either input field in the ***login*** page.  **Procedure**:  Open the program and type a *user ID* in the *UserID* field in the ***login*** page and leave the *password* field blank.  **Result**:  *User ID* is displayed in the input field. | *User ID* is displayed in the input field. |
| 4. | User can click on ***“Login”*** button.  **Procedure**:  Click on the ***“Login”*** button.  **Result**:  Message “Please fill out both fields” is displayed to the user. | Message “Please fill out both fields” is displayed to the user. |
| **Comments**:  Successful test. | | |
| **Client Signature of Approval**: | | |
| **IT Use Only:**  Redesign Complete:  Yes  No  Retest Complete:  Yes  No  Details if Retest not complete: | | |
| **Date: 07/15/2021** | | |
| **Developer Name and Signature:** | | |

|  |  |  |
| --- | --- | --- |
| **Component**: Secure Logout | | **Test Date:** |
| **Test** | **Description** | **Actual Result** |
| 1. | ***Secure Logout***  **Procedure**:  On the page shown to the user (Admin.jsp for admin users, WorkCenter.jsp to advanced users and user.jsp to regular users), look for the “Logout” button located at the right side of the navigation bar, at the top of the page.  **Result**:  User is redirected from current page. | User is redirected from current page. |
| 2. | User is successfully logged out.  **Procedure**:  Click on the “Logout” button in the navigation bar.  **Result**:  User is taken back to the ***login*** page and the message “You have successfully logged out” is displayed under the ***login*** form. |  |
| **Comments**:  Successful test. | | |
| **Client Signature of Approval**: | | |
| **IT Use Only:**  Redesign Complete:  Yes  No  Retest Complete:  Yes  No  Details if Retest not complete: | | |
| **Date: 07/15/2021** | | |
| **Developer Name and Signature:** | | |

|  |  |  |
| --- | --- | --- |
| **Component**: Admin user | | **Test Date: July 15, 2021** |
| **Test** | **Description** | **Actual Result** |
| 1. | ***Add a new user account***  Tested by: Jaehan Kim  **Procedure**:  Login with an admin account (40002, password)  Click ‘Create a New User’ tab to unfold the ‘Create a New User’ function.  Filling out all the blank columns and click ‘Create’ button.  Leave a blank or two columns and click ‘Create’ button to check the error detection  **Result**:  User will be created successfully.  If the user missed any column, the error message will be popped up with an instruction. | 1. Successfully added when fill in a form 2. Error messages indicate the cause of errors correctly. 3. Password is not encrypted by \*. |
| 2. | ***Delete user account(s)***  Tested by: Saeid Ghavamabadi  **Procedure**:  Click on existing user account. Click delete button.  Click on more than one existing user account. Click delete button.  **Result**:  User will be deleted successfully  Multiple users will be deleted successfully | 1. User is successfully deleted WHEN they aren’t apart of any outstanding work orders. 2. Multiple users are successfully deleted WHEN they aren’t apart of any outstanding work orders. 3. If multiple users are selected but any one of them has an outstanding work order, none of the users will be deleted. |
| 3. | ***Display work order***  Tested by: Saeid Ghavamabadi  **Procedure**:  Scroll down to the work order table.  Examine work order table if there’s any missing data.  Click ‘Order Details’ to connect to the order detail page  **Result**:   1. All work orders are displayed on the table 2. The page will be transferred to the work order detail page if user click ‘Order Details’ link button. | 1. All work orders are displayed on the table 2. The page is transferred to the work order detail page when I click ‘Order Details’ link button. |
| 4. | ***Delete work order***  Tested by: Jaehan Kim  **Procedure**:  Scroll down to the work order table.  Click one of checkboxes of work orders and click ‘delete’ button.  Click multiple checkboxes of work orders and click ‘delete’ button  **Result**:   1. One specific work order will be deleted if the user check one work order. 2. Multiple work orders will be deleted if the user check multiple work order to delete. | Work order is successfully deleted as I intended. (Single or Multiple) |
| 5. | ***Lock user account(s)***  Tested by: Saeid Ghavamabadi  **Procedure**:  Click on an existing user account.  Click on the *lock* button.  Click on multiple existing user accounts.  Click on the *lock* button.  **Result**:   1. User account will be locked. 2. Multiple user accounts will be locked. | Lock user function not yet added. |
| 6. | ***Create notification***  Tested by: Jaehan Kim  **Procedure**:  Login with an admin account (User: 40002 Password: password)  Click “Create a New Notification” tab to unfold the create a notification function.  Pick drop down options for notifications.  Click ‘Create’ button.  **Result**:  New notification will be created. | Notification is successfully created as intended. |
| 8. | ***Delete notification***  Tested by: Jaehan Kim  **Procedure**:  Scroll down to the notification table.  Click one of checkboxes of notifications and click ‘delete’ button.  Click multiple checkboxes of notifications and click ‘delete’ button  **Result**:   1. One specific notification will be deleted if the user selects one notification. 2. Multiple notification will be deleted if the user selects multiple notifications to delete. | Notification is successfully deleted as I intended. (Single or Multiple) |
| **Comments**:  Most functions are working correctly except lock account. | | |
| **Client Signature of Approval**: | | |
| **IT Use Only:**  Redesign Complete:  Yes  No  Retest Complete:  Yes  No  Details if Retest not complete: | | |
| **Date: 07/18/2021** | | |
| **Developer Name and Signature:** | | |

|  |  |
| --- | --- |
| **Client**: N/A |  |
| **System**: Regular User experience |  |
| **Objective**:  Confirmation of full, correct functionality of all regular user experiences within the application. | |
| **Tester Name:** Trevor Erixon | |
| **Developer Name:** Trevor Erixon, Jaehan Kim, Maria Laura Diaz Pena, Zennon Weleschuk, Rylan Cook, Saeid Ghavamabadi | |

|  |  |  |
| --- | --- | --- |
| **Component**: Regular user | | **Test Date: July 19, 2021** |
| **Test** | **Description** | **Actual Result** |
| 1. | ***Land on user’s page***  **Procedure**:  Successful login to application, landing on user’s page.  **Result**:  Land on user’s page with user’s name, role, and work center correctly displayed along with a list of work orders that the user has assigned tasks within. Work orders should be accessible via a link that will take the user to the work order page. | User 40016: Landed on user’s page with all information correctly displayed. The link to the work order details is active.  User 40013: same result |
| 2. | ***Navigate to page(s) for each work order that the user is assigned to***  **Procedure**:  Click on any of the ‘order details’ links to be taken to the page for each individual work order.  **Result**:  Land on the page for the work order corresponding to the link that was clicked. All details of the work order should be populated and there should be at least one operation assigned to the user. The ability to add additional operations to the work should not be available to the regular user. | User 40016: Landed on the correct work order page. All details were correctly populated and there was one operation assigned to the user. No ability to add additional operations.  User 40013: same result |
| 3. | ***Update status/comments of assigned operation***  **Procedure**:  Click on the status dropdown menu to select an appropriate status for the operation. Comments can/should be added in the input box when the status of the operation is updated. Once these two fields are ready for update, click the update button.  **Result**:  The details of the operation should now default to the updated values so if the user changes pages and returns to this work order, these details will have persisted. User should not be able to modify other user’s operations. | User 40016: User was able to update the operation status to each different available option. Each time the status was changed, I navigated away and came back to the page – status and comment text correctly persisted.  User can no longer update or change the status of another user’s operation(s).  Attempting to inject HTML tags is rejected. Data previously in this field in the database is overwritten with an empty string.  User 40013: same result |
| 4. | ***Enter closing notes/update work order***  **Procedure**:  Once all operations are closed, the closing notes field will display at the bottom of the page along with a button to update/close the work order. Update the closing notes field with any relevant details and click the update work order button.  **Result**:  The closing notes field should update and display the closing notes. The work order status should change to closed. | User 40016: Closing notes were captured in the database and the closing notes field updates to reflect that a note was added. Work order status correctly updated to closed.  Attempting to inject HTML tags is rejected. Data previously in this field in the database is overwritten with an empty string.  User 40013: same result |
| 5. | ***Navigate using ‘Home’ link***  **Procedure**:  Click on Home in the top right corner of the page.  **Result**:  Land on user’s page with all information correctly displayed. | User 40016: Landed on user’s page with all information correctly displayed.  User 40013: same result |
| 6. | ***Navigate using WOM icon link***  **Procedure**:  Click on WOM icon in the top left corner of the page.  **Result**:  Land on user’s page with all information correctly displayed. | User 40016: Landed on user’s page with all information correctly displayed.  User 40013: same result |
| 7. | ***Navigate using Logout link***  **Procedure**:  Click on 'Logout’ link in the top right corner of the page.  **Result**:  Land on logout page with logout message. | User 40016: Landed on logout page with a logout message correctly displayed.  User 40013: same result |
| 8. | ***Navigate to geolocation page for a work order***  **Procedure**:  Click on link corresponding to the latitude/longitude of the plant location for a work order.  **Result**:  Land on a page that displays a map with a pinned location? | Not yet ready to test |
| 9. |  |  |
| 10. |  |  |
| **Comments**:   * We should consider displaying the role description instead of the id number on the user’s page. * All tests for the regular user were generally successful; however, there are still a few opportunities for improvement to make the application more secure and more user-friendly for a regular user. | | |
| **Client Signature of Approval**: | | |
| **IT Use Only:**  Redesign Complete:  Yes  No  Retest Complete:  Yes  No  Details if Retest not complete: | | |
| **Date: July 19, 2021** | | |
| **Developer Name and Signature:** | | |

|  |  |  |
| --- | --- | --- |
| **Component**: Advanced user | | **Test Date: July 15** |
| **Test** | **Description** | **Actual Result** |
| 1. | ***Create work order from notification***  Tested by: Zennon Weleschuk  **Procedure**:  Create notification from the workcenter page as an advanced user.  tests:   1. input all valid fields 2. input valid notification id, all other fields blank 3. all blank 4. Invalid notification id = 99 5. Invalid string for date field   **Result**:   1. input all valid fields => work order creation successful 2. input valid notification id, all other fields blank => reject successful 3. all blank => reject successful 4. notification id = 99 (ie invalid) => reject successful 5. invalid string for date field => reject successful   ALL TEST PASSED SUCCESSFULLY | 1. Notification created 2. Rejection handled 3. Rejection handled 4. Rejection handled 5. Rejection handled |
| 2. | **Add Operation to specific work order**  Tested by: Zennon Weleschuk  **Procedure**:  Add operation to a work order as an advanced user  Tests:   1. All valid inputs 2. All blank inputs 3. String in operation id, rest of inputs valid   **Result**:   1. All valid inputs => not added [failed] 2. All blank inputs => unable to test 3. String in operation id, rest of inputs valid => unable to test | 1.  2.  3. |
| 3. | Access page to show geolocation for the job  Tested by: Zennon Weleschuk  **Procedure**:  Access geolocation page and ensure all information is displaying properly.  Tests:   1. Navigate via link 2. Navigate via url   **Result**:   1. Navigate via link => page does not exist yet [failed] 2. Navigate via url => page does not exist yet [failed] |  |
| **Comments**: | | |
| **Client Signature of Approval**: | | |
| **IT Use Only:**  Redesign Complete:  Yes  No  Retest Complete:  Yes  No  Details if Retest not complete: | | |
| **Date:** | | |
| **Developer Name and Signature:** | | |

|  |  |  |
| --- | --- | --- |
| **Component**: Notification Creation | | **Test Date: July 16, 2021** |
| **Test** | **Description** | **Actual Result** |
| 1. | ***Immediate display of newly created notification***  Tested by: Jaehan Kim  **Procedure**:  Open a new application.  Login with an admin account (40001 / 40002 / 40003, password)    Click ‘Create a New Notification’ tab to unfold the ‘Create a New Notification’ function.    Create a new notification by setting each columns (Notification Type, Cause, Damage, Plant, Technical Object, Assign to).  Scroll down to see if the new notification is successfully created and added on the notification table with auto-generated notification number.  Create another new notification.  **Result**:  New notification will be created and added immediately on the notification table as the user intended. | Newly created notification is updated right away. |
| 2. | ***Automatic generation of a notification number.***  Tested by: Jaehan Kim  **Procedure**:  Open a new application.  Login with an admin account (40001 / 40002 / 40003, password)    Click ‘Create a New Notification’ tab to unfold the ‘Create a New Notification’ function.    Create a new notification by setting each columns (Notification Type, Cause, Damage, Plant, Technical Object, Assign to).  Scroll down to see if the new notification is successfully created and the notification Id which is increased by 1 from the previous notification id.  **Result**:  New notification will be created and added on the notification table as the user intended.  Notification ID will be incremented by 1 automatically if a new notification is created. | Newly created notification is updated on the notification table and the notification ID is incremented by 1. |
| 3. | ***Work Center assigned to the notification***  Tested by: Jaehan Kim    **Procedure:**    Open a new application.    Login with an admin account (40001 / 40002 / 40003, password)    Click ‘Create a New Notification’ tab to unfold the ‘Create a New Notification’ function.    Create a new notification by setting each columns (Notification Type, Cause, Damage, Plant, Technical Object, Assign to).    Scroll down to see if the new notification is successfully created and Create another new notification and check the Assigned to column.    **Result:**    New notification will be created and added on the notification table as the user intended.    Assigned to will be the same as user intended to create the table. | Newly created notification is updated on the notification table and the ‘assigned to’ section’s value is same as I submitted to create a new notification. |
| 4. | ***Notifications displayed on the work center page***  Tested by: Jaehan Kim  **Procedure**:  If the user is already logged in, click ‘logout’ button on the top right of the page otherwise open a new application  Login by considering ‘assigned to’ value selected for creating a new notification as a previous step. Please refer the user id as below:  Electrical Maintenance: 40004  Mechanical Maintenance: 40005  Equipment Maintenance: 40006  Preventive Maintenance: 40007  **Result**:  All notifications related to the specific work center will be displayed on the page | A newly created notification is updated on the notification table and the ‘assigned to’ section’s value is same as I submitted to create a new notification. |
| 5. | **Automatic “ASSIGNED” status after notification creation**  Tested by: Jaehan Kim  **Procedure:**    Open a new application.    Login with an admin account (40001 / 40002 / 40003, password)    Click ‘Create a New Notification’ tab to unfold the ‘Create a New Notification’ function.    Create a new notification by setting each columns (Notification Type, Cause, Damage, Plant, Technical Object, Assign to).    Scroll down to see if the new notification is successfully created and see the current status is ‘assigned’ on the current Status    **Result:**    New notification will be created and added on the notification table as the user intended.    Current status of newly created notification will be ‘assigned’ | A newly created notification is updated on the notification table and the current status of the new notification is assigned |
| 6. | ***Delete notification***  Tested by: Jaehan Kim  **Procedure:**  Scroll down to the notification table.  Click one of checkboxes of notifications and click ‘delete’ button.  Click multiple checkboxes of notifications and click ‘delete’ button  **Result:**  One specific notification will be deleted if the user selects one notification.  Multiple notification will be deleted if the user selects multiple notifications to delete. | Notification is successfully deleted as I intended. (Single or Multiple) |
| **Comments**:  All tests are successful. | | |
| **Client Signature of Approval**: | | |
| **IT Use Only:**  Redesign Complete:  Yes  No  Retest Complete:  Yes  No  Details if Retest not complete: | | |
| **Date:07/16/2021** | | |
| **Developer Name and Signature:** | | |

|  |  |  |
| --- | --- | --- |
| **Component**: Work Order Creation | | **Test Date: 2021-07-19** |
| **Test** | **Description** | **Actual Result** |
| 1. | Work order is created after clicking button beside work no work order notification.  **Procedure**:  Create work order using one of work center lead account (40004, 40005, 40006, 40007)  After signing into the work center lead for that type of notification, scroll down to verify a work order is not created by the notification id which the user intends to use for the work order creation.  Click the “Create/See WorkOrder” button  **Result**:  The “Original Notification” box on the “Create Work Order from Notification” is filled with the correct notification and the rest of the unchangeable fields are filled with that notification’s attributes. | Able to enter details for new workOrder. |
| 2. | WorkOrder dates only accept valid inputs.  **Procedure**:  Create work order using a work center lead account (40004, 40005, 40006, 40007)  After signing into the work center lead for that type of notification, scroll down to verify a work order is not created by the notification id which the user intends to use for the work order creation.  Click the “Create/See WorkOrder” button.  Enter a date in the not show format. And try to create it  **Result**:  Creation does not happen if dates do not follow format. Does not perform any checks on verification of date. | If WorkOrder has valid inputs, it will be created. If not, it will be rejected |
| 3. | WorkOrder Id generation.  **Procedure**:  Create work order using a work center lead account (40004, 40005, 40006, 40007)  After signing into the work center lead for that type of notification, scroll down to verify a work order is not created by the notification id which the user intends to use for the work order creation.  Click the “Create/See WorkOrder” button  Enter valid inputs and press “Create”  **Result**:  Work order should be created with correct WorkOrder ID | Work order is created with correct WorkOrder ID |
| 4. | Work order is assigned to correct notification and given correct status.  **Procedure**:  Create work order using a work center lead account (40004, 40005, 40006, 40007)  After signing into the work center lead for that type of notification, scroll down to verify a work order is not created by the notification id which the user intends to use for the work order creation.  Click the “Create/See WorkOrder” button  Enter valid inputs and press “Create”  **Result**:  WorkOrder Notification ID should match notification it was created from.  WorkOrder Status should be “Assigned” | WorkOrder Notification ID matches notification it was created from.  WorkOrder Status is correct as “Assigned” |
| 5. | Clicking on “Create/See WorkOrder” with a notification that has a WorkOrder takes you to the WorkOrder.  **Procedure**:  Click on a “Create/See WorkOrder” for a notification that already has a created WorkOrder.  **Result**:  Should land on the page for the selected WorkOrder. | Gets sent to that WorkOrder page. |
| 6. | Work Order shows the notification it is for.  **Procedure**:  Open any WorkOrder  **Result**:  Notification details should display with correct information. | Notification details are correctly displayed. |
| 7. | **Add Operations to the Work Order**  Tested by: Laura Diaz  **Procedure:**  From the Work Center page, as a Work Center Lead, click on the newly created Work Order’s details link. This will take the user to that particular Work Order’s page.  In the Work Order page, insert the Operation ID and Operation Description in the input fields and select the Worker responsible for the Operation’s fullfilment and the status for that Operation.  Click on “Add Operation” button.  **Result:**  The new Operation should be displayed to the user. | The new Operation is displayed to the user in the Work Order Page. |
| 8. | **Update Status information on existing Work Order once finished**.  Tested by: Jaehan Kim  **Procedure:**  Login as an admin user (40001, 40002, 40003) or work center lead (40004, 40005, 40006, 40007).  Move to work order detail page  Change all status of operation in the operations to close and click update for each operation.  When all the operation status change to ‘CLOSED’ the closing note will be activated  Leave any comment in the close notes and click ‘update work order’  Login back as an admin user (40001, 40002, 40003) and scroll down to see work order table  **Result:**  The status of notification will change to ‘CLOSED’  The status of workorder will changed to ‘CLOSED’ | The work order status changed successfully |
| 9. | **Delete an existing work order**  Tested by: Laura Diaz  **Procedure:**  Login as an admin user (40001, 40002, 40003) and go to the Admin page.  Scroll down to the View All Work Orders table, select a Work Order and click on the checkbox of said Work Order.  Scroll up to the top of the table and click on the “Delete” button.  **Result:**  Work Order is deleted from the table and from the database. | Work Order is deleted from the table and from the database. |
| **Comments**:  Successful test. | | |
| **Client Signature of Approval**: | | |
| **IT Use Only:**  Redesign Complete:  Yes  No  Retest Complete:  Yes  No  Details if Retest not complete: | | |
| **Date: 2021-07-19** | | |
| **Developer Name and Signature: Rylan Cook** | | |

|  |  |  |
| --- | --- | --- |
| **Component**: Database Maintenance | | **Test Date: July 15** |
| **Test** | **Description** | **Actual Result** |
| 1. | **Get User object for specified user**  Tested by: Zennon Weleschuk  **Procedure**:  A specified user object will be returned when called from a service layer  **Result**:  User object return successfully  TEST PASSED SUCCESSFULLY | User object (id: 40017) returned successfully |
| 2. | **Get list of all users in DB**  Tested by: Zennon Weleschuk  **Procedure**:  A list of all users in the DB will be returned when called from the userDB from a service layer  **Result**:  List of user objects returned  TEST PASSED SUCCESSFULLY | List of users returned when on admin page |
| 3. | **Get list of all users from specified work center**  Tested by:  **Procedure**:  A list of all users that belong to a specifed work center will be returned when called from the userDB from a service layer  **Result**:  List of user objects returned |  |
| 4. | **Insert new user into DB**  Tested by: Zennon Weleschuk  **Procedure**:  Inserts a new user into the DB  **Result**:  New user inserted successfully  TEST PASSED SUCCESSFULLY | User inserted from admin page |
| 8. | **Get WorkCenter object for specified work center**  Tested by: Zennon Weleschuk  **Procedure**:  Get work center object for specified work center  **Result**:  Work center object returned  TEST PASSED SUCCESSFULLY | Work center returned for user 40017 |
| 9. | **Get WorkOrder object for specified work order**  Tested by: Zennon Weleschuk  **Procedure**:  Get WorkOrder object for specified work order  **Result**:  Work order object returned  TEST PASSED SUCCESSFULLY | Work order returned for user 40017 |
|  | **Get WorkOrder object by notification**  Tested by: Zennon Weleschuk  **Procedure**:  Get WorkOrder object by notification  **Result**:  Work order object returned  TEST PASSED SUCCESSFULLY | Work order returned for user 40017 |
|  | **Get list of all workorders in DB**  Tested by: Zennon Weleschuk  **Procedure**:  Get list of all work orders in DB  **Result**:  All work orders returned  TEST PASSED SUCCESSFULLY | All work orders returned for user 40001 on admin page |
|  | **Insert new WorkOrder into DB**  Tested by: Zennon Weleschuk  **Procedure**:  Insert new work order into DB  **Result**:  New work order inserted  TEST PASSED SUCCESSFULLY | New work order inserted with user 40007 (advanded user) |
|  | **Delete workorder from DB**  Tested by: Zennon Weleschuk  **Procedure**:  Delete work order from DB  **Result**:  Delete work order from DB  TEST PASSED SUCCESSFULLY | Work order deleted from admin page |
|  | **Get list of all notification object in DB**  Tested by: Zennon Weleschuk  **Procedure**:  Get list of all notification objects in DB  **Result**:  List of all notification objects in DB  TEST PASSED SUCCESSFULLY | All notification objects displayed on admin page |
|  | **Get list of all notification object by work center**  Tested by: Zennon Weleschuk  **Procedure**:  Get list of all notification object by work center  **Result**:  list of all notification object by work center  TEST PASSED SUCCESSFULLY | All notification objects for user 400017 |
|  | **Insert new notification object into dB**  Tested by: Zennon Weleschuk  **Procedure**:  Insert new notification object into DB  **Result**:  New notification object inserted into DB  TEST PASSED SUCCESSFULLY | New notification object inserted from admin page |
|  | **Delete notification object from DB**  Tested by: Zennon Weleschuk  **Procedure**:  Delete notification object from DB  **Result**:  Notification object deleted from DB  TEST PASSED SUCCESSFULLY | Notification object deleted from admin page |
| **Comments**: | | |
| **Client Signature of Approval**: | | |
| **IT Use Only:**  Redesign Complete:  Yes  No  Retest Complete:  Yes  No  Details if Retest not complete: | | |
| **Date:** | | |
| **Developer Name and Signature:** | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| **Component**: Navigation | | **Test Date: July 20th** |
| **Test** | **Description** | **Actual Result** |
| 1. | ***Navigation Links***  **Procedure**:  Click on every available link on every JSP and make sure they take the user to the correct destination.  **Result**:  Test passed. | All navigation links work. |
| 2. | ***Buttons***  **Procedure**:  Click on every available button on every JSP and make sure they do the proper action required.  **Result**:  Test passed. | All buttons work. |
| **Comments**:  Successful test. | | |
| **Client Signature of Approval**: | | |
| **IT Use Only:**  Redesign Complete:  Yes  No  Retest Complete:  Yes  No  Details if Retest not complete: | | |
| **Date: 07/15/2021** | | |
| **Developer Name and Signature:** | | |

|  |  |  |
| --- | --- | --- |
| **Component**: Quality of Life | | **Test Date: July 20th** |
| **Test** | **Description** | **Actual Result** |
|  | **Feedback**  **Procedure:**  Execute a function / action on every available page and record whether a feedback message is displayed when action is taken.  **Result:**  Certain functions have no feedback and need to have feedback added. | **Admin Page**  **Create a notification: No feedback**  **Create a user: No feedback**  **Delete notification: No feedback**  **Delete work order: No feedback**  **Delete user: No feedback**  **Forced password reset: Feedback**  **Logout: Feedback**  **Login Page**  **Invalid Login: Feedback**  **Valid Login: No Feedback**  **User Page:**  **Logout: Feedback**  **Work Order Page:**  **Update: Feedback**  **Add Operation: No feedback**  **Work Center Page:**  **Create W.O from Notification: No feedback**  **Delete W.O: No feedback** |
| **Comments**:  Successful test. | | |
| **Client Signature of Approval**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |
| **IT Use Only:**  Redesign Complete:  Yes  No  Retest Complete:  Yes  No  Details if Retest not complete: | | |

System Deployment

Data Conversion

Any data currently within our system is fictitious and not related to any organization. Thus, we will be giving a barebones MySQL that will load the minimum requirements for the DB to function and one admin account that will be used to create the rest of the required accounts, notifications and workorders.

Changeover Strategy

For the changeover of the system, we will work with the client to help them put this on AWS/other hosting sites (excluding Heroku), and work to update the default SQL file to be more suited for their business. We will add new plants, causes, damages, technical objects, Workcenter's, and types for notifications and work orders. If there is no one that can use sql within the organization, we will provide them with a script that will allow for uploading new plants, causes, damages, technical objects, Workcenter's, and types for notifications and work orders through a GUI. We will also update the context.xml so that it will connect to the database.

System Administration and Maintenance

Security

All accounts can be locked, including admins. A password reset or locking of accounts will force the user out of the system and require a login and, if a reset is requested, a change to the current password. All account passwords must contain 1 uppercase letter, 1 lowercase letter and a number. Database passwords are encrypted when stored in the database.

Operation

Administrative User

Administrative users have several specific functions available to their role:

1. Full summary visibility to all data within the application
2. Notification creation
3. New user creation
4. Password reset
5. Lock user account
6. Delete functionality
7. Full summary visibility to all data within the application

* Tables include all Notifications, Work Orders, assigned Operations, and Users
* Tables can be sorted/filtered for ease of access

1. Notification creation
   * Select the ‘Create a New Notification’ dropdown to expose required fields for completion.
   * Required fields:
     1. Notification type
        + Classification to provide some basic information about the request for repair/attention
     2. Cause
        + The cause of the damage
     3. Damage
        + The type of damage
     4. Plant
        + The plant location of the asset
     5. Technical object
        + The asset or equipment in need of attention
     6. Assign to
        + The work centre that will be notified
   * Each field has been designed as a drop down to aid in the speed of creation
   * To complete the notification creation, click on the ‘Create’ button
2. New user creation
   * Select the ‘Create a New User’ dropdown to expose required fields for completion
   * Required fields:
     1. First name
        + The new user’s first name
     2. Last name
        + The new user’s last name
     3. Initial password
        + The new user’s initial application password
     4. Role
        + The new user’s role
        + Current available selections: Director (Admin), Department Manager (Admin), Work Center Lead (Work Center super-user), Field Worker (Regular)
     5. Assign to:
        + The work centre that the user will be a member of
   * The name and password fields are all text-based; the remaining fields are dropdown menus
   * To complete the user creation, click on the ‘Create’ button
3. Password reset
   * Locate the user from the ‘View All Users’ table by scanning from top to bottom or by utilizing the sort/filter functionality at the top of the table
   * Select the checkbox that corresponds to the user requiring password reset and then select the ‘Reset Password’ button on the same line
   * The Users table will update to reflect that a forced password change has been initiated
   * Upon the user’s next attempt to login, they will be required to change their password to something different from their previous password
4. Lock user account
   * Locate the user from the ‘View All Users’ table by scanning from top to bottom or by utilizing the sort/filter functionality at the top of the table
   * Click on the ‘Lock User Account’ button at the end of the row corresponding to the desired user
   * Upon the user’s next attempt to login, they will be notified that the account is locked and to contact an admin
5. Delete functionality
   * Select one or more checkboxes from a single table to identify the records to be deleted
   * Click on the ‘Delete’ button in the upper right corner above the table
   * Each ‘Delete’ button corresponds to a specific table so it is not possible to delete a user and a notification with the same button click. To delete data from multiple tables, you must repeat these steps for each table

Work Center Super-User

Work Center super-users have several specific functions available to their role:

1. Full summary visibility to all data within their work center
2. Work order creation from a notification
3. Work order operation assignment
4. Delete functionality
5. Full summary visibility to all data within their work center
   * Tables include all Notifications, Work Orders
   * Ability to view a specific notification or work order
6. Work order creation from a notification
   * Select a notification from the notification table. If it does not already have a corresponding work order, it will automatically populate the known fields of the ‘Create Work Order from Notification’ dropdown menu. Alternatively, you can manually enter the notification id number
   * Selecting a notification that already has a corresponding work order will take you directly into the work order
   * Required fields:
     1. Order Type
        + Additional classification to fine tune type of work order being requested
     2. Order Description
        + Additional details to provide regarding the general goal of the work order
     3. Required Start Date
        + In MM/DD/YYYY format
     4. Required End Date
        + In MM/DD/YYYY format
   * To complete the work order creation, click on the ‘Create’ button
7. Work order operation assignment
   * From the ‘View All Notifications’ table
     1. Click on the ‘Create/See Work Order’ button for any notification that already has a work order created
   * From the ‘View All Work Orders’ table
     1. Click on the ‘Detail’ button for the work order you wish to view
   * Once on the page for the desired work order, navigate to the ‘Operations’ table at the bottom of the page
   * Required fields:
     1. Operation ID
        + The next operation id number
     2. Assigned To
        + The field worker that is assigned to the operation
     3. Description
        + Detailed description of the specific task required to be completed
     4. Status
        + Current status of the operation
     5. Comments
        + Knowledge capture opportunity for the field worker handling the operation
   * To complete the operation assignment, click on the ‘Add Operation’ button
8. Delete functionality
   * Select one or more checkboxes from a single table to identify the records to be deleted
   * Click on the ‘Delete’ button in the upper right corner above the table
   * Each ‘Delete’ button corresponds to a table so it is not possible to delete a work order and a notification with the same button click. To delete data from multiple tables, you must repeat these steps for each table

Regular User

Regular users have several specific functions available to their role:

1. Update operations
2. Update closing notes/close work order
3. Update operations
   * Required fields:
     1. Status
        + The current status of the operation
        + Changing operation status will change the notification/work order status to maintain consistency (i.e. if an operation is ‘paused’, the work order will also be paused)
     2. Comments
        + Knowledge capture opportunity for any status changes of the operation
   * Users can only make changes to operations that are assigned to them
4. Update closing notes/close work order
   * Once all operations have been addressed, the closing notes section of the work order will unlock and a large text box will appear
   * Required field:
     1. Closing Notes
        + Knowledge capture opportunity for summarizing any additional relevant details with respect to closing the work order
   * To complete the work order and set the status to closed, click on the ‘Update Work Order’ button

Miscellaneous

Any user that is able to access a work order has the ability to see a map showing the location of the work order.

1. Click the ‘Map’ button found in the Navigation column of the Notification table

Backup and Restore

Unless changes are made to the files themselves, the only thing that must be backuped within the system is the database as all data for operations is store in it. The organization will decide on a backup and restore plan.

Data Archival

We currently have no systems in place to archive data other than backing it up using MySQL. We plan to implement a data archival system, where users and notification can be hidden from view, but still stored within the database, and have implemented search functionality to help with data build up.

System Maintenance

System maintenance will not have to be done on the hosting system as no data is stored within it. The maintenance will be done on the database to ensure that all entries are valid and not causing corruption within the system.

Deficiency List

Requirements Not Met

This application solves the requirements given to us by the Organization. It allows to for tracking of repair jobs. Unfortunately, due to the fallout with the previous client and the movement from SAP to a private DB, our application is not fully usable through the application itself. Some of the problems that are currently lacking for a self-standing system include:

* Inability to add new locations, damages, causes, custom status’s, object (cranes, pump, etc), new workcenter’s and types for workorders and notifications.
* Roles are currently hard coded for access, meaning even adding a new role within the database, will not help to give user specific access to objects.
* Users and other created objects have no way to be edited through the web app.

Non-Functioning Components

There are no non-functioning components within the database. Everything that is implemented is working and tested.

Appendix

Program Listings

Developer environment is in a zip file called: “Final Version Capstone Dev environment”. This dev environment suitable for NetBeans 8.2, due to library dependencies. Use in a more updated version of NetBeans may result in library errors.

Final Versions is in a war file called “ROOT”. This is name in a way that when used with the Elastic Beanstalk CLI, is will automatically update the environment.

Barebones SQL file for a MySQL database is in the file “workorderdb.sql”.

Connection to database is in “context.xml” and must be filled with the JDBC connection link, username, and password. The account that is being used must allow for SELECT, UPDATE, INSERT and removal of data.

Java documentation is located within the javadoc folder.

Glossary

|  |  |
| --- | --- |
| SAP | An Enterprise Resource Planning (ERP) software |
| GUI | **Graphical User Interface: computer program that enables a person to communicate with a computer using symbols, visual metaphors, and pointing devices.** |
| OS | **Operating System: system software that manages computer hardware, software resources, and provides common services for computer programs.** |
| Platform Agnostic | **Product that runs equally well across more than one platform.** |
| UI | **User interface: the design field for human-computer interaction.** |
| Work Centre | **Specialized group of field workers in the organization.** |
| Field worker | **A worker who travels to different locations and work outside the office. Job duties vary based on organization.** |
| Database | **Organized collection of data.** |
| Actors | **Roles within the system.** |
| Cached data | **Files, scripts, images and other multimedia stored in a device after opening an application.** |
| ABAP | **SAP ABAP is a high-level programming language.** |
| HANA | **SAP HANA is a high-performance in-memory database that speeds data-driven, real-time decisions and actions.** |
| Platform-as-a-Service | **Category of cloud computing services that allows users to manage a computing platform and one or more applications.** |

Index

Activity Diagram 4, 20

ABAP 38

Class Diagram 4, 15

HTML5 38

JavaScript 38

Milestones 37

PwC 4

SAP 5

SAPUI5 38

Sequence Diagram 16-18, 35-36

State machine Diagram 4, 19

Use case Diagram 4, 6

References

[1] ”PricewaterhouseCoopers” Wikipedia, Mar. 10, 2021. [Online]. Available: <https://en.wikipedia.org/wiki/PricewaterhouseCoopers> [Accessed: Mar. 26, 2021].

[2] ”PwC Canada,”PwC, n.d. [Online]. Available: <https://www.pwc.com/ca/en/about-us.html> [Accessed: Mar. 26, 2021].

[3] ”Configuring SAPNetWeaver Mobile,” SAP, n.d. [Online]. Available: <https://help.sap.com/viewer/41b87e9d7a504eb7bb0cf31939ec83a8/7.5.6/en-US/482759a7d00258d7e10000000a421937.html> [Accessed: Apr. 22, 2021].

[4] “ABAP CDS Entities”, SAP, n.d [Online]. Available: <https://help.sap.com/doc/2d51e3a4c6ed4a0f82bb09c42053d09a/1610%20001/en-US/frameset.htm?4ed2245c6e391014adc9fffe4e204223.html> [Accessed: Apr. 22, 2021]