NDT-Link

Master's Thesis

Master in Web App and Website Development

Author: Chakir Mrabet

Consultant Professor: Carlos Caballero González

Professor: César Pablo Córcoles Briongos

June 06, 2021

Credits/Copyright

Dedication

To my beloved wife *Amal*, for convincing me to take this journey, and for being a source of strength when I most needed it. To my wonderful kids *Adam* and *Sara*, for all the moments I took from them and gave to this thesis. To my mentor and friend *Borja*, for his support and for believing in me when others didn't.

Abstract

NDT-Link is a web based cloud service that provides a complete suit of productivity tools to companies that own instruments and systems manufactured by Innerspec Technologies[1], a US based company leader in NDT[2] solutions. Innerspec products are used in a wide range of applications, including defect detection in manufacturing processes and measurement of material properties such as thickness and residual stress. Innerspec customers usually own several instruments and systems which they need to configure and manage individually. Also, Innerspec customers don't have a tool that centralizes the data generated by all their instruments and systems. NDT-Link will provide a centralized location from where Innerspec customers can manage all their instruments and systems. NDT-Link will also serve as a hub where Innerspec customers can upload, review and share data generated by the instruments and systems they own.

Key Words: NDT, cloud, service, defect, thickness, stress, centralization, configuration, management, data.

1 Innerspec Technologies, Inc. (http://www.innerspec.com).

2 Non-Destructive Tests

Index

Figures

Tables

1. Introduction

NDT-Link is a web based cloud service that provides a complete suit of productivity tools to companies that own products manufactured by Innerspec Technologies. NDT-Link will serve as a hub where Innerspec customers can manage several aspects of the products they own, store and share data generated from them, and receive direct support provided by Innerspec personnel. The products owned by customers will also synchronize automatically their user information, settings, and licensing credentials with the information managed from NDT-Link.

Innerspec Technologies is a manufacturer of NDT solutions that include integrated systems for manufacturing plants, and portable solutions for in-service applications. Innerspec pioneered commercial applications using Ultrasound generated with EMAT[3] in the mid-90s, becoming the world leader in this technology with hundreds of systems installed worldwide. More recently, Innerspec has included other techniques to serve customers with all their advanced non-destructive testing needs.

Innerspec is headquartered in Forest, Virginia (USA), and has offices in Mexico, Europe and China, along with representatives around the globe that provide commercial and technical support with factory trained personnel.

The author of this thesis is the head of the Software team at Innerspec Technologies.

3 Electro-Magnetic Acoustic Transducer

1.1. Pain Points

Innerspec products are mainly used to find defects caused during the production process of different types of parts and components, or to measure specific material properties such as thickness, stress, and electrical conductivity. Innerspec products range from large automated systems installed in factory production lines, to small portable instruments that NDT inspectors use in the field to inspect the integrity of pipes, tubes and tanks used to transport or store oil, gas and other corrosive elements. Portable instruments are also used by R&D professionals who research new inspection and measurement methods.

Figure 1: NDT professional inspecting corrosion in a pipe using an Innerspec instrument

Innerspec products can perform different types of inspections and measurement which generate large amounts of data presented to the user as graphical charts. In the case of applications for defect detection, the software installed in the integrated systems will analyze the generated data automatically in order to tell the user where the defect is located on the inspected part, which later can be either discarded or repaired. For portable instruments, the user will analyze the generated data to determine where the defect is located on the part in order to determine if it needs repair.

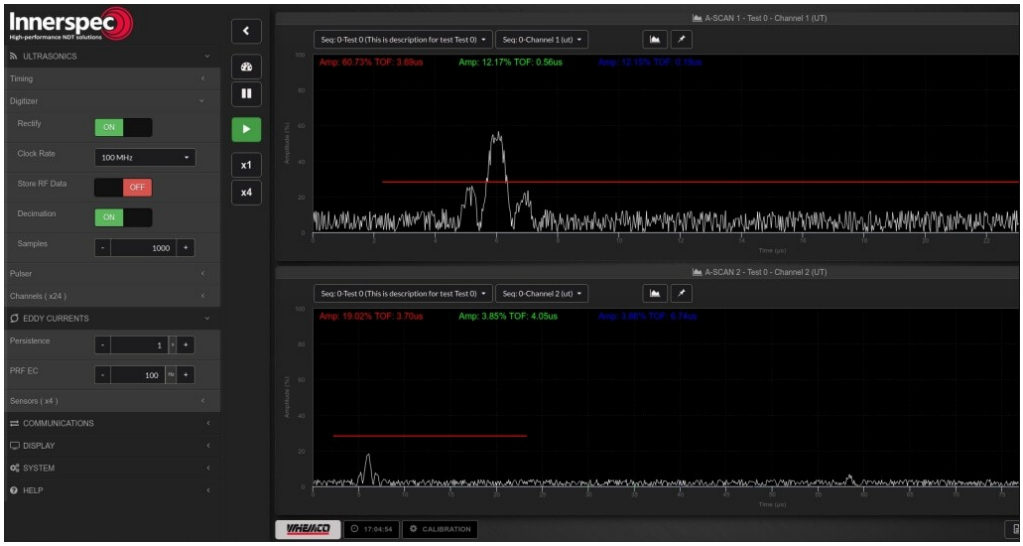


Figure 2: Strip chart showing the location of a defect when inspecting a part

Innerspec customers usually have several systems installed in one single factory or several instruments deployed in the field at the same site where an In-Service job is performed. All these systems and instruments work isolated from each other which introduces several limitations:

1. **No centralized user account management***:*
   - The software that powers Innerspec products requires user authentication and role based authorization in order to grant access to its functionality. Customers that have several products used by multiple operators don't have a centralized location from where they can manage and push user accounts. Customers need to do this from each one of the products they own.
2. **No centralized setup management***:*
   - Innerspec products can be configured to work with different types or materials and parts. The configuration consists of a group of settings entered by the operator in the user interface of the product. In many situations, customers inspect the same material or part with multiple products, requiring them to re-create the same settings in each one of them.
3. **No centralized status information, usage statistics and inspection data***:*
   - Innerspec customers don't have a tool that aggregates status information, usage statistics and inspection results from all the products they own. Customers need to access their products individually in order to get this information.
4. **No centralized location for product documentation***:*
   - Innerspec customers don't have an automatic and reliable way to access user guides, manuals, spares lists and other product documentation. Up to now, customers need to rely on either contacting Innerspec directly to ask for the desired documentation, or hope to get contacted by Innerspec personnel when there is new documentation available.
5. **No centralized location for support***:*
   - When users experience problems with Innerspec products they are forced to either call Innerspec or use their desktop computers or laptops to send an email to help@innerspec.com describing the problem they have. Customers don't have an option

to report problems directly from the Innerspec products they own, which adds an extra level of disconnection with the user.

6. **No software based option to review inspection data***:*
    - Innerspec doesn't have a reliable tool that allows customers to review inspection data generated by the products they own. If customers need to share inspection data with other team members, they have to extract it first from the instrument or system, and manually send it to their team members, who then need to import the data into another product of the same type.
    - This limitation also affects Innerspec personnel when they need to assist the customer by reviewing their inspection data. Innerspec is currently overcoming this problem by establishing remote connections to the customers' instruments and products, which is in many situations, a slow and unreliable process.
7. **No option to limit access to specific software features***:*
    - The software that powers Innerspec products provides several levels of functionality. Some of these levels are not critical for the operation of the product, but provide extra tools that ease the job of the operator or enhance the resultant inspection data. Innerspec wants to start monetizing the access to specific software levels with a pay-as-you-go system, managed from a centralized location where customers can purchase daily, monthly or annual recurrent subscriptions.

1.2. Proposed Solution

NDT-Link will provide the centralization that is currently missing in the ecosystem of products provided by Innerspec, bringing users, machines and Innerspec staff together in one single application hosted in the cloud.

Among other features, NDT-Link will allow Innerspec customers to manage user accounts and push them to their instruments and systems, access aggregated status and usage information uploaded from them, access to updated product information, track issues reported from the Innerspec products, or report new ones, view and share inspection data, and buy subscriptions to premium software modules.

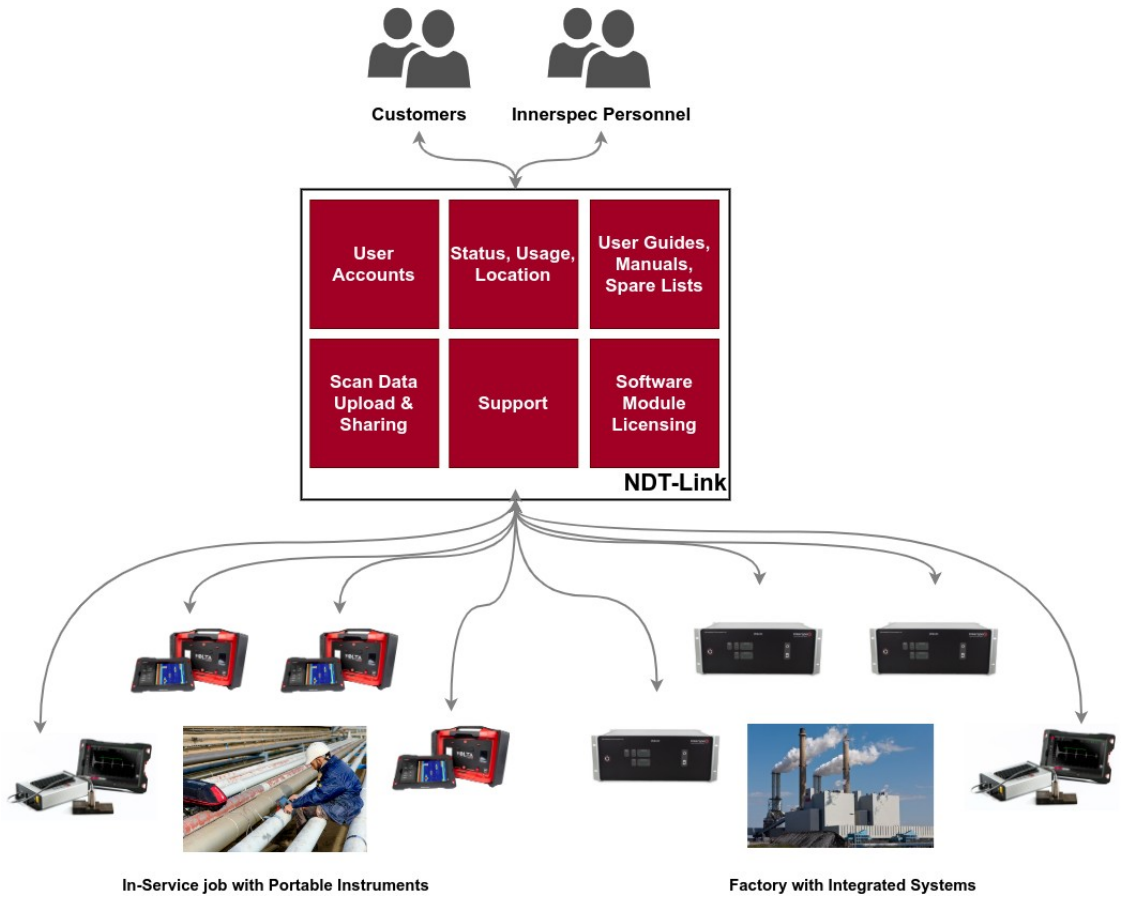


Figure 3: NDT-Link centralization for Innerspec products

2. Description

2.1. Application Structure

NDT-Link will be composed of the following main components:

1. *Hosted in the cloud by 3rd party services:*
    1. Web client (front-end).
    2. REST API (back-end).
    3. Mail Service.
    4. Payment Gateway.
    5. Database Service.
2. *Hosted locally in each Innerspec product:*
    1. Product Connector.

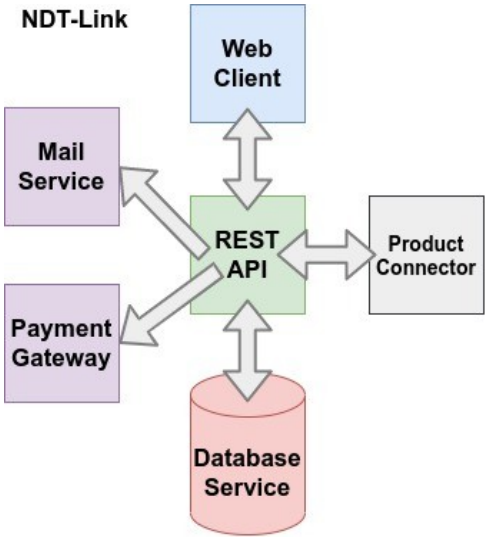


Figure 4: NDT-Link main structure

The web client will be implemented as a responsive SPA[4] that will be served from https://www.ndt-link.com. The web client will communicate with a back-end that provides a REST[5] API[6] exposing endpoints for the front-end and for all the Innerspec products deployed in the field. All the content provided by the back-end will be persisted in a database service. The back-end will be listening for incoming requests from https://api.ndt-link.com .

A separate service will handle the transmission of transactional emails that are queued by the back-end as a response to certain actions performed by users. For processing purchases of subscriptions for premium software modules, a payment third party payment gateway will be used.

4 Single Page Application

5 Representational State Transfer

6 Application Programming Interface

Innerspec products will communicate with the REST API through the Product Connector, which will authenticate in the REST API using a unique Asset Tag and Asset Secret.

Product Connector

This will be an application installed as a service in each Innerspec product. It will be continuously running in the background and performing the following actions when there is Internet connectivity:

1. **For all products***:*
   - Send a GET request to the REST API to check if new users have been assigned to the system or existing ones have been removed. If there are changes, apply locally (create new accounts for the users that has been added, or delete the accounts for the users that has been removed).
   - Send a POST request to the REST API with scan data for inspections selected by the operator.
   - Send a GET request to the REST API to check the software licenses available for the system. Lock or unlock software features based on the response.
2. **Portable Instruments***:*
   - Send a POST request to the REST API with the system status (new geolocation and usage since the last request).
3. **Integrated Systems***:*
   - Send a POST request to the REST API with the system status (system is running, system is idle, system is in error state).
   - Send a POST request to the REST API with the current production statistics (loaded setup, number of inspections since the last request, number of pass and fail inspections).

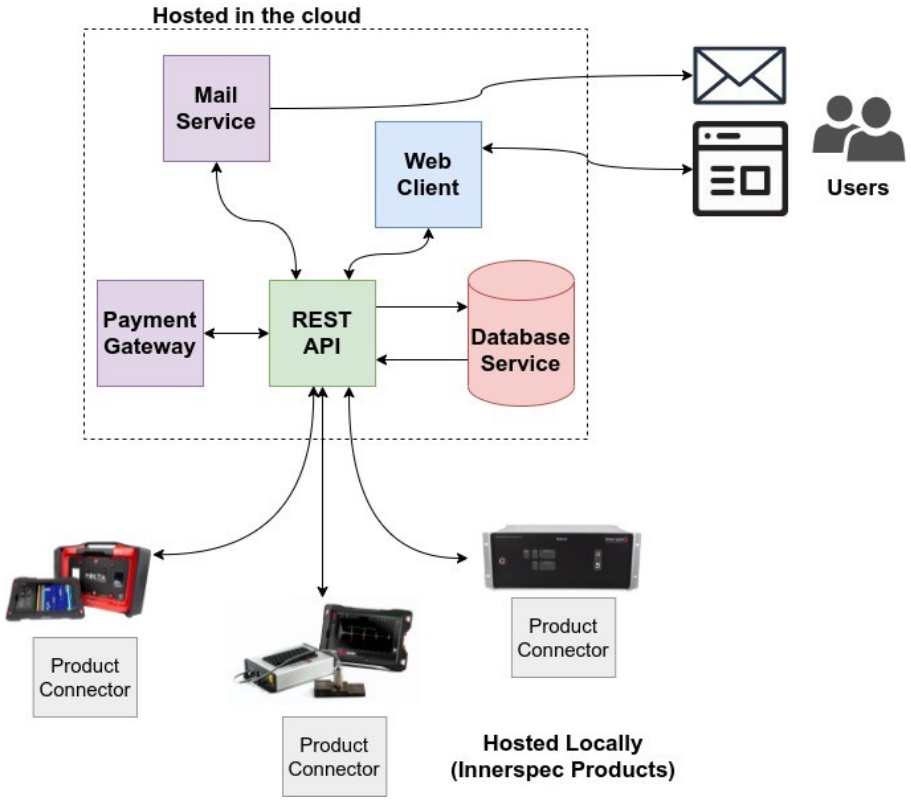


Figure 5: Communication flow between NDT-Link components

2.2. Modules

The content and functionality of NDT-Link will be organized in five different modules.

User Management

1. Register new users, edit, delete and block existing ones.
2. Assign and remove roles from existing users.
3. Create new groups and companies, link users to companies, and companies to groups.
4. Manage contact information for existing groups and companies.

User Asset Management

1. List all the products owned by groups and companies.
2. See detailed information for each product:
    1. For portable products, history of usage and localization.
    2. For integrated products, current status and production statistics.
3. Assign and remove users from products.
4. Access to product documentation (manuals, drawings and spare lists).
5. List all the available software updates for a product.
6. Purchase software licenses for a product.
7. Purchase one-to-one Customer Support time (provided by phone), and track its usage (number of hours used, number of hours left).
8. Report and track issues found with the products.

Innerspec Product Management

Innerspec users will be able to:

1. Populate NDT-Link with all the available products, their technical specifications, and the software packages they can run.
2. Publish product documentation (manuals, drawings, and spare lists).
3. Register products as new assets of the location that purchased them.

Help Desk (Ticketing System)

1. Report a problem (open a ticket) or bug that a user is experiencing when using Innerspec products.
2. Track the resolution of reported problems.
3. Subscribe for email notifications when the status of the reported problems changes.
4. List all reported problems by a group, by a company, by a user, or for a product, with options to filter by status and importance.

Innerspec users will also:

1. Keep the user updated on the resolution of a reported problem by updating their open tickets.

Data Sharing

This module will allow customers to upload to NDT-Link the resultant data of selected inspections performed with Innerspec products, so it can be then shared with other NDT-Link users from their same groups and companies:

1. From the products, users will select a specific inspection and upload it to NDT-Link (done by the Product Connector described later in this section).
2. From NDT-Link, under the product profile, users will be able to manage the list of uploaded inspections. Users will be able to select one of the inspections to see its details, or to generate a shareable link for other users.
3. When an inspection is selected, users will see its meta data (timestamp, setup used, inspection disposition), as well as plot data based on the types of scans performed.

2.3. Authentication/Authorization

All content in NDT-Link will be private and will require user authentication for access. For authentication, NDT-Link will provide the following public paths:

1. User Log In
2. User Password Recovery

Users will be able to change their passwords and other contact information from their profile pages once they have authenticated in NDT-Link. The authentication strategy will be based on email and password confirmation. The emails used to create new accounts will be unique since they will be used as username or unique identifier for each account.

Profiles

NDT-Link will support three different user profiles that will limit the actions the users can perform:

1. *Innerspec:*
    - It will be assigned to accounts created for Innerspec users or personnel.
    - It will will allow access to the content of all groups and companies as well as other content and pages for used for back-office operations.
2. *User:*
    - It will be assigned to accounts for Innerspec customers, partners and representatives.
    - It will only allow access to content that belongs to the user's company and group.
3. *Product:*
    - This profile will used by Innerspec products to communicate with NDT-Link in order to download user account and asset information, and to upload status and usage information, as well as inspection data.

Roles

NDT-Link will provide two main roles that will be used to authorize the actions the users can perform:

1. *Manager:*

- Innerspec profiles: Creation, edition, and deletion of any content.
- User profiles: Creation, edition and deletion of any content that belongs to the user's company.

2. *Employee:*
   - Innerspec profiles: Read any content, creation and deletion of content in the Customer Support module, registration of new systems when purchased by customers.
   - User profiles: Read any content that belongs to the user's company, report and update issues in the Customer Support module.

The User profile will have two extra roles:

1. *Group:*
   - It will allow the user to read content from all the companies that are part of the group.
2. *Group Manager:*
   - It will allow the user to read content from all the companies that are part of the group.
   - It will allow the users to assign the role Group to other users.

The Innerspec profile will also have one extra role:

1. *Superuser:*
   - It will give full control to settings that configure NDT-Link.
   - It will allow changing the password for the built-in Superuser account.
   - It will allow assigning the profile Superuser to other Innerspec profile based users.

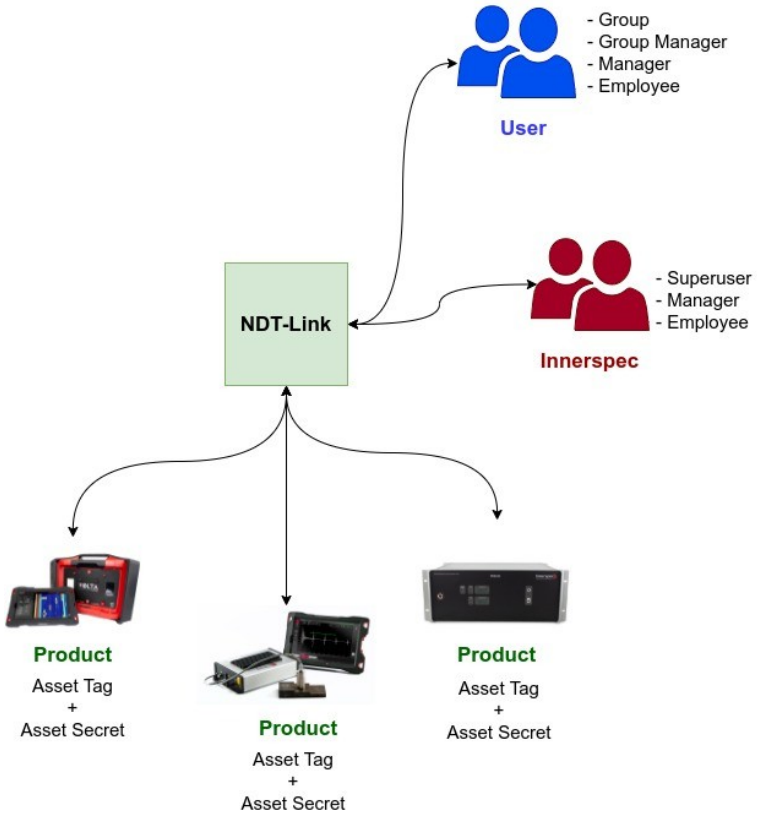


Figure 6: Authentication/Authorization profiles and roles

2.4. Account Creation

New user accounts will be created as follows:

1. The REST API will provide a built-in account with the Innerspec profile and the Superuser role. This account will be used to create other Innerspec accounts with roles Manager and Employee as needed in order to populate NDT-Link with accounts for Innerspec personnel.
2. Innerspec accounts with role Manager will populate NDT-Link creating Groups and Companies for all Innerspec customers.
3. Innerspec accounts with role Manager will create a User profile account with role Manager for each company. This process will send an automatic email to the users with their username and password.
4. Innerspec accounts with role Manager will assign the Group Manager role to one of the User accounts with role Manager.
5. From this point on, any User profile account with role Manager can create more User profile accounts for their companies, assigning the roles Employee or Manager as needed. Also, the User profile account elevated with the role Group Manager can assign the Group role to other users in its group as needed.

NDT-Link accounts for products will be created by Innerspec personnel as follows:

1. Innerspec profile based users will register a new system under the company's profile by entering its unique serial number, Asset Tag, purchase order number, and purchase date.
2. NDT-Link will generate a unique Asset Secret for the system.
3. Innerspec based profile users will then enter the generated Asset Secret in the Product Connector service running in the system. The Product Connector will use this information to authenticate in NDT-Link REST API in order to download and upload data.

2.5. Purchasing Subscriptions

Subscriptions to paid Software Modules will be handled by an external payment gateway:

1. User profiles with the role Manager access the detail page of one of their systems.
2. Under the section where the active subscriptions are listed, the user will press a button to add a new one.
3. A dialog box will show up listing all the software modules available for the software running on the system, with the exception of those that are already active.
4. The user will select the desired software module and pick one of the available payment plans (daily, weekly, monthly, annual).
5. The user accepts the dialog and is forwarded to the payment gateway's website, on which he enters his payment method to process the purchase.
6. Once the purchase is processed, the user will be forwarded back to the product's page on NDT-Link.
7. NDT-Link REST API will check with the payment gateway that the purchase has been admitted, and then it will enable the software module for the target product.

8. NDT-Link REST API will inform the Product Connector installed in the target product of the new purchase, so the new software module can become available to the users who can access it.
9. NDT-Link will run a recurrent daily task that will check expired licenses in order to update target Product Connectors so they can revoke local access to paid software modules. If a Product Connector can't reach NDT-Link REST API at least once in 24 hours, the Product Connector will make the local software to lock the paid software modules, and ask users to connect the system to the Internet for synchronization.

3. Objectives

3.1. Primary

1. Implement user authentication and authorization based on different profiles and roles.
2. Implement password recovery.
3. Implement user account management for local products.
4. Implement asset management.
5. Implement global management of Groups, Companies, Users, Products, Software Platforms and Software Modules for Innerspec profiles.
6. Implement subscriptions to Software Modules and the communication process with a payment gateway.
7. Implement ticketing system for User and Innerspec profile based users.
8. Implement basic data sharing between systems and NDT-Link (status and usage information only).
9. Implement REST API with endpoints for front-end and for the Product Connector.
10. Deploy front-end, back-end and database service on the cloud.

3.2. Secondary

1. Implement transactional emails triggered by different actions performed on the front-end.
2. Implement geolocation on the Product Connector and representation of the systems' geographic location on maps in the front-end.
3. Expand data sharing between products and NDT-Link with inspection data and setups.
4. Implement private messaging between users.

3.3. Other

The author of this thesis will implement NDT-Link with the exception of the Product Connector, which will be implemented by other employees at Innerspec Technologies. The author of this thesis will supervise and manage the implementation of the Product Connector to make sure it satisfies NDT-Link requirements.

4. Methodology

NDT-Link will be implemented following a Waterfall methodology, modified to fit the life cycle of a product or service creation. The traditional sequential implementation in separate and consecutive stages of the Waterfall methodology will be respected, but each stage will focus on producing different versions of NDT-Link that gradually grow with new functionality.

The exception of this rule will be the stages dedicated to definition of requirements and design specifications. Also, the maintenance stage will not be part of the scope of this project since Innerspec Technologies will be responsible for implementing it once the product is finished and released.

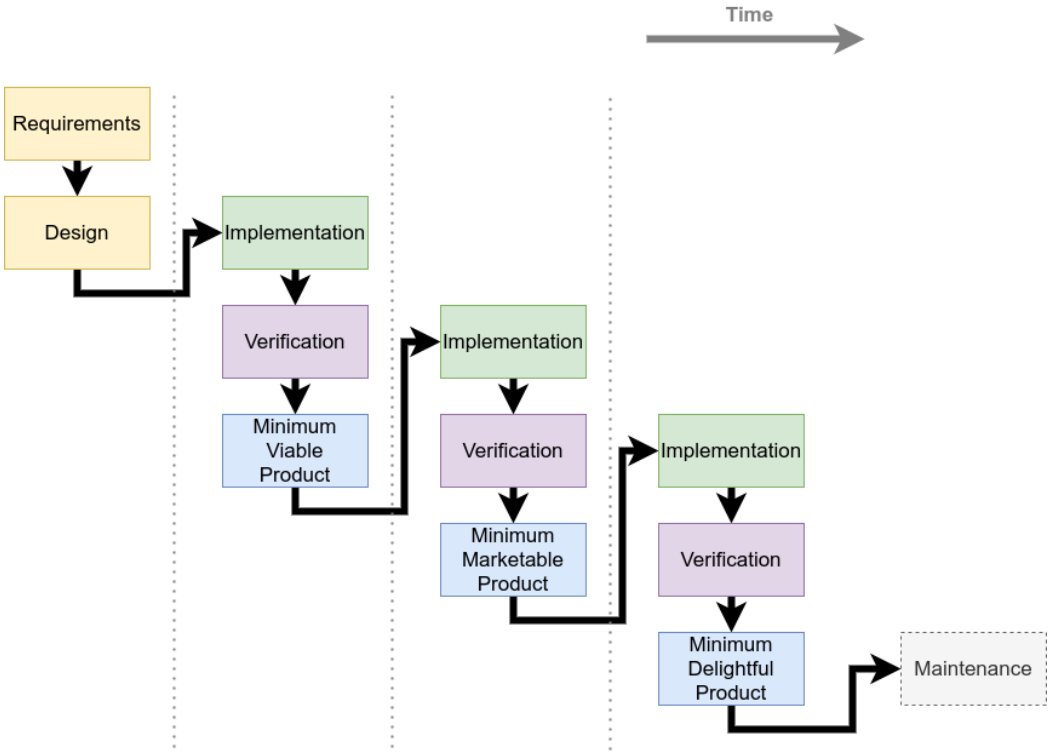


Figure 7: Implementation Methodology for NDT-Link

5. Application Architecture

5.1. Preliminary Information Architecture

NDT-Link will provide two main menus for authenticated users:

1. **User Menu**, located at the top of the screen and containing the following options:
    1. Log Out.
    2. Messages:
        1. It will show a list of messages sent by other users from their profile pages.
    3. Notifications:
        1. It will show a list of events, such as a new reply posted to a support ticket.
    4. My Profile:
        1. Will forward users to a screen where they can update personal information (name, phone number, title, and department).
        2. All NDT-Link users can visit the profile page of other users by clicking the usernames that will appear in different screens.
        3. From this page, users will be able to send messages to each other.
2. **Navigation Menu**:
    1. It will be located on the left side of the screen and will provide links that change based on the profile of the authenticated user.

The Navigation Menu for User profile accounts will have the following links:

1. **Dashboard** (default screen):
    1. It will show a summary of relevant information:
        1. Total number of systems, instruments and users in the company.
        2. Total number of open and in-progress support issues reported by the user.
        3. Total number of open and in-progress support issues reported by the company.
    2. It will show a map containing the locations of all the Portable systems owned by the company. Each marker in the map will be a link that forwards the user to the detail page of the system.
    3. It will show a table with the following information for all the Integrated systems owned by the company:
        1. Serial Number, type of the system, and software version installed. This will be a link that forwards the user to the detail page of the system.
        2. Current status (Idle, Running, in Error state).
        3. Total number of parts inspected in the last production shift (8 hours).
2. **My Group** (only accessible by users with the role Group and Group Manager):
    1. It will show general information about the group (name, description, notes).
    2. It will show a map with the locations of all the companies that belong to the group, as well as a table with the following information from each one of them:
        1. Name.

2. Country.
3. Number of users.
4. Number of Portable Systems.
5. Number or Integrated Systems.

3. It will show a table listing all the Integrated Systems owned by all the companies in the group, with the following columns:
    1. Serial Number and type of the system.
    2. Based on the selected range of time from filters exposed by the table (current shift, last 24 hours, this week, this month, this year):
        1. Total parts inspected.
        2. Total parts that have passed the inspection.
        3. Total parts that have failed the inspection.
        4. Scrap percentage (fail/total * 100).

3. **My Company**:
    1. This screen will show general information about the company (name, description and type among others).
    2. It will provide a CRUD section to manage the locations o addresses the company can have (shipping, accounting, etc.).
    3. It will provide a CRUD section to manage the users that belong to the company:
        1. User profiles with the Role Manager will be the only ones allowed to create new users or delete existing ones.
    4. It will show a table listing all the Portable Instruments and Integrated Systems with a link to their detail pages.
    5. It will show a table listing all the open support issues reported for all products that belong to the company.
4. **Equipment**:
    1. This page will show tables listing all the Integrated Systems and Portable Instruments that have been registered to the company by Innerspec personnel. These tables will contain the following columns:
        1. Product Serial Number, Part Number and Type.
        2. Number of user accounts active in the product.
        3. Software version installed.
        4. Number of support issues reported for the product (open, in-progress, closed).
        5. Number of inspections uploaded to NDT-Link.
        6. For Integrated Systems:
            1. Production totals based on selected time range.
            2. Current status (idle, running, error state).
        7. For Portable Instruments:
            1. Number of hours that the instrument has been in use.
            2. Last calibration date.
    2. If the user clicks a row in these tables, they will be forwarded to the detail page of the product, which will be as follows:

1. It will show general information about the product (serial number, product number, name, description, purchase date, purchase order, manufacturing date, and notes).
2. It will show a CRUD section for managing the users that are allowed to sign in each product:
    1. Users with role Managers will be able to add new users or remove existing ones.
3. It will show a table listing all the subscriptions purchased for each product:
    1. Status (active, inactive), date of the purchase, purchase order number and recursion type (daily, monthly, annual).
        1. Users with role Manager will be able to cancel subscriptions and buy new ones.
4. It will show a table with all the documentation available for the product (manuals, user guides).
5. It will show a table listing all the software updates available for the product, with a link to download their installer.
6. It will show a table listing all the support issues reported from the product:
    1. Users will also be able to report new issues from this page:
        1. They will be forwarded to a screen where they can enter a subject summarizing the problem, a description, and urgency.
    2. Clicking on one of the rows in the table will forward the user to the detail page of the support ticket show the following:
        1. Subject, description, urgency and date the ticket was reported.
        2. Status (open, in progress, fixed, etc..).
        3. Name of the Innerspec agent assigned to work on the ticket.
        4. Messaging section, that will contain a series of comments from the user and the Innerspec agent during the resolution of the ticket.
7. Integrated Systems:
    1. It will show general information about the system (Serial Number, Product Number, Name, Description, Purchase Date, Purchase Order, Manufacturing Date, and notes).
8. Portable Instruments:
    1. It will show the localization history of the product as a series or marks on a map.
    2. Clicking on each marker a box will provide a total of usage hours at the location, start date and end date.

The Navigation Menu for Innerspec profile accounts will have the following links:

1. **Dashboard** (default screen):
    1. It will show a summary of relevant information:
        1. Total number of systems, instruments and users in all companies.
        2. Total number of open and in-progress support issues reported by all companies.
    2. It will show a map containing the locations of all the Portable products owned by all companies. Each marker in the map is a link that forwards the user to the detail page of the product.

2. **Groups:**
    1. It will provide CRUD operations to manage all groups. Only Innerspec profiles with the role Manager will be able to create new ones and delete existing ones.
    2. Clicking a group will forward to its detail page that will provide the same functionality described for User profiles.
3. **Companies**:
    1. It will provide CRUD operations to manage all companies. Only Innerspec profiles with the role Manager will be able to create new ones and delete existing ones.
    2. Clicking a company will forward to its detail page that will provide the same functionality described for User profiles.
4. **Users:**
    1. It will provide CRUD operations to manage all users. Only Staff profiles with the role Manager will be able to delete users or to create new ones and assign them to a company.
    2. Clicking a company will forward to its detail page that will provide the same functionality described for User profiles.
5. **Products:**
    1. It will provide CRUD operations to manage types of products. Only Staff profiles with the role Manager will be able to create new ones and delete existing ones.
    2. When creating new product types, the following information will be recorded:
        1. Part number, name, type (Integrated or Portable), description.
        2. Software Platform that runs on the product.
6. **Software Platforms:**
    1. It will provide CRUD operations to manage types of software platforms. Only Staff profiles with the role Manager will be able to create new ones and delete existing ones.
    2. When creating new software platforms, the following information will be recorded:
        1. Part number, name, description.
    3. Clicking a software platform on the CRUD table will forward the user to a detail page that shows the following:
        1. Part number, name and description of the software platform.
        2. A CRUD section where Staff profiles with the role Manager can add or remove the modules that compose the software platform and their licensing type:
            1. The information to record when creating a new module will be the following:
                1. Name, Description, License Type.
                2. License Type can be Free or Paid. When Paid is selected, the user will need to enter the cost for the different types of subscriptions (daily, weekly, monthly, annual). If no amount is entered the subscription will not be available for that period of time.
7. **Help Desk:**
    1. This page will provide a full ticketing system listing all the issues reported from all products by all the customers. The table will expose filters that Innerspec profiles can use to narrow down the list by type of product or company.

2. Each item in the list will show the ticket's subject, product type, company, username of the reporter, date, status and urgency.
3. Clicking on an item in the list will forward to the tickets detail screen with the same functionality described previously for User profiles, with the following additions:
    1. Option to assign the Innerspec account that will handle the ticket (agent).
    2. Option to change urgency.
    3. Option to categorize the ticket as bug, feature request or question.
4. Every time the ticket is updated by the reporter or agent, an automated email will be sent to the other party with details about the change.

The following sections show sitemaps for Innerspec and User based profiles based on their correspondent Information Architecture.

Preliminary Information Architecture for Innerspec based profiles

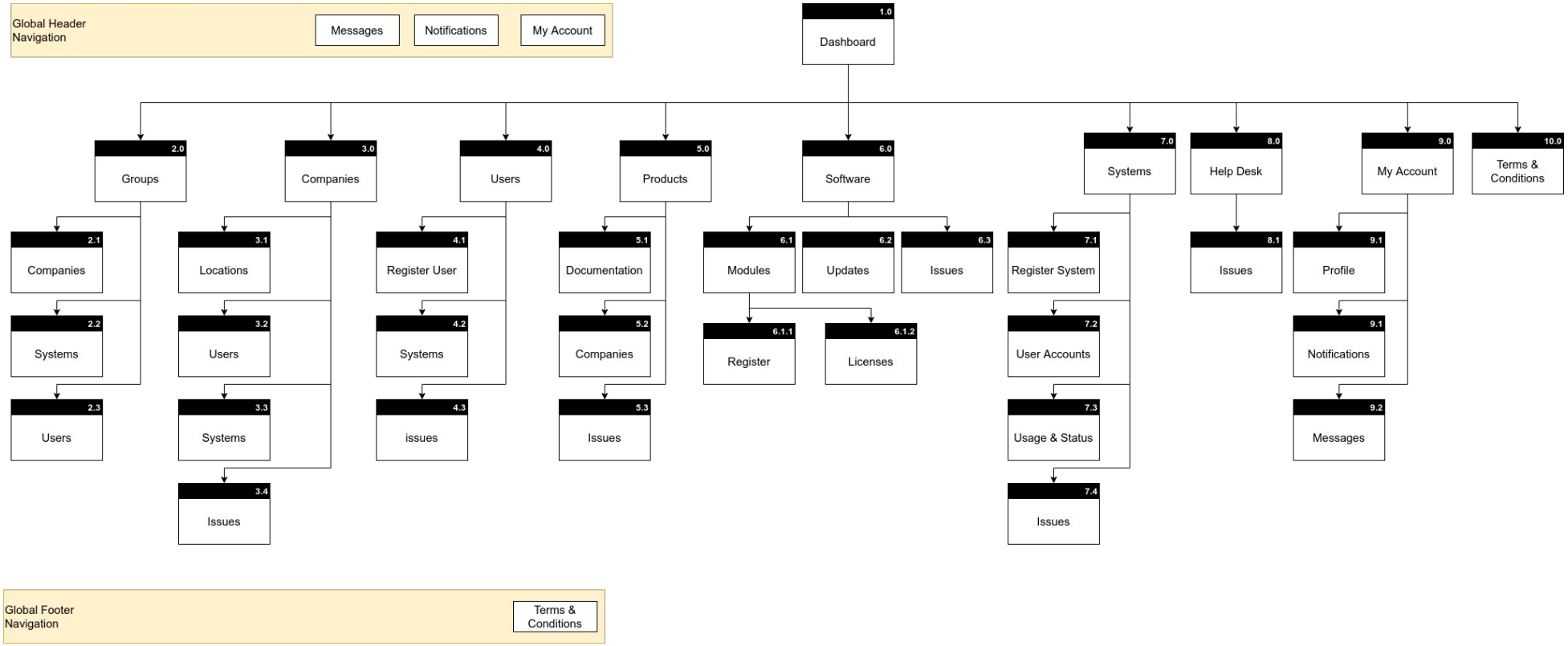


Figure 8: Preliminary Information Architecture for Innerspec based profiles

Preliminary Information Architecture for User based profiles

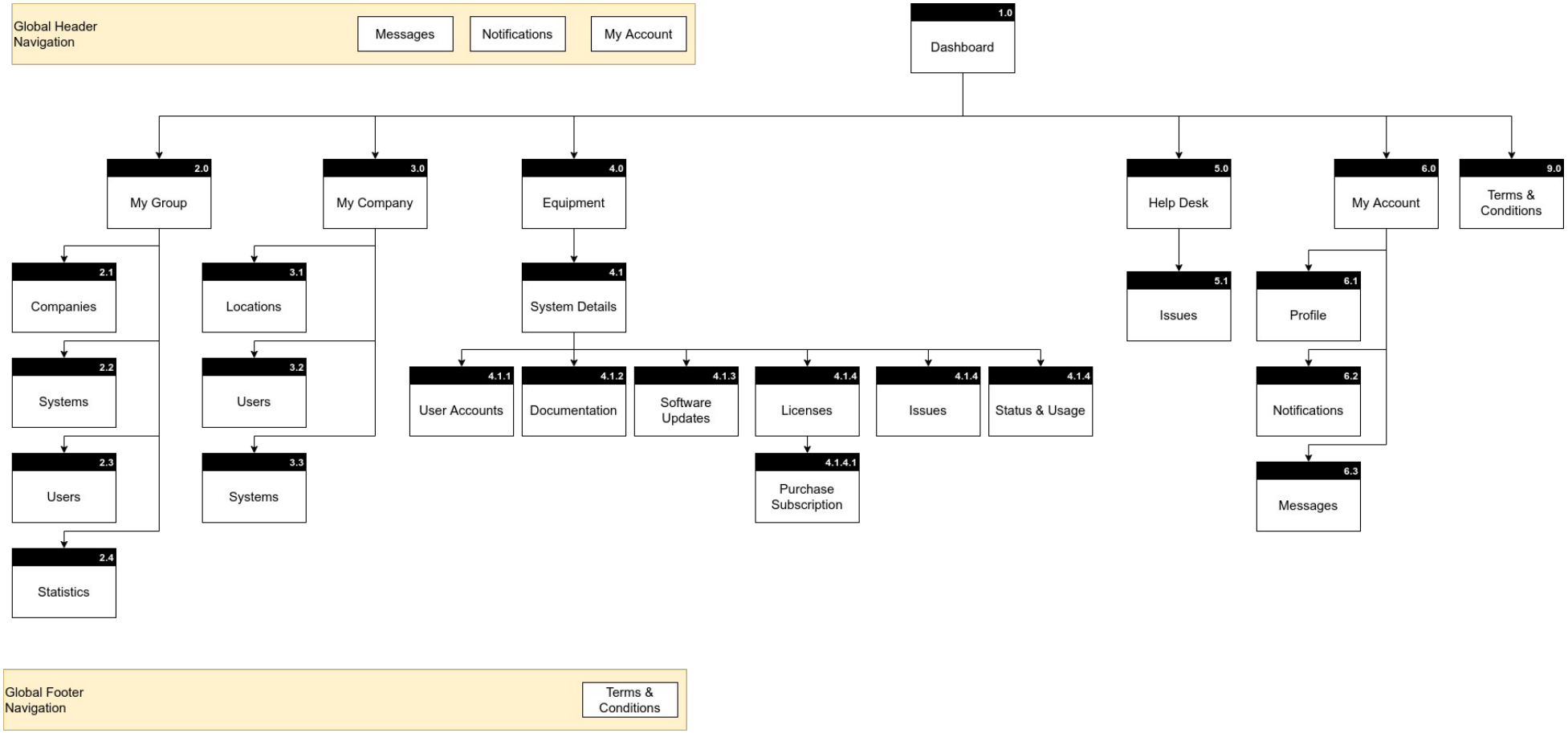


Figure 9: Preliminary Information Architecture for User based profiles

5.2. Preliminary Information Architecture Validation

Tree Testing was conducted to validate the preliminary Information Architecture. The testers were selected from within different departments of Innerspec, and they were given several tasks to perform using the *Optimal Workshop*[7] platform.

Since NTD-Link has two different user profiles, one for Innerspec personnel and another for Innerspec customers (User), two different tests were conducted with two different groups of testers. For the Innerspec profile, the test was taken by members of the Sales, Customer Support, Software and Assembly teams. For the User profile, members of the Sales team took the test.

The main reason to select all the testers from within Innerspec was due to the constraint in time to finish the project. Innerspec customers are very diverse and belong to a wide spectrum of industries and needs. Studying and defining specific groups of target users to produce customized tests would be a very complex and time consuming task that would require coordination with several external companies and groups. In order to meet the deadline set to publish the first version of NDT-Link, validation and other usability tests were performed internally using feedback from those members who are closer to Innerspec customers. Once NDT-Link is published and used for a reasonable amount of time, Innerspec will conduct more usability tests if needed.

Test Setup

The sample for the test is as follows:

1. **Innerspec profile**:
    1. 10 people:
        1. 2x from Sales team.
        2. 1x from Assembly team.
        3. 1x from Customer Support team.
        4. 6x from Software team.
    2. Objectives:
        1. Find customer related information (groups, companies, issues).
        2. Find systems related information.
        3. Manage Innerspec assets (products, software, software modules).
        4. Manager customer's assets (register new systems, register users).
2. **User profiles**:
    1. 2x from Sales team.
    2. Objectives:
        1. Find information related to the user's organization.
        2. Find information about systems owned by the user's organization.
        3. Report issues with systems owned by the user's organization.

7 https://www.optimalworkshop.com/

Two different test trees were created for the User and Innerspec profiles (Staff).

Figure 10: Test Trees for User and Innerspec profiles

Two sets of tasks were were created for the User and Innerspec profiles (staff).

	Task	Correct Answer
1	You are the admin of the NDT-Link account for your company. Register two new users from your team so they can access NDT-Link too.	My Company > Users
2	Add one of your teammates to VOLTA SN 12345 so they can sign in the instrument to use it in an upcoming job.	Equipment > System Details > User Accounts
3	Download the user guide for your PRIMO SC system.	Equipment > System Details > Documentation
4	Buy a subscription for the software module Circular Scan for your VOLTA unit with SN 12345 (this module is an extra, it's not included in the basic software that comes with the unit).	Equipment > System Details > Licenses > Purchase Subscription
5	One of your colleagues uploaded a scan from a job site and wants you to review. Find the scan to review.	Equipment > System Details > Inspection Data
6	You are having issues with your VOLTA unit SN 12345. The software doesn't seem to be working correctly for certain features. Report the problem so the Support Team at Innerspec can review and get back to you.	Equipment > System Details > Issues Help Desk
7	Update the shipping address for your company.	My Company > Locations
8	Mike Smith no longer works for your company. Remove him from all the systems in which he had access.	Equipment > System Details > User Accounts
9	Get a list of the systems owned by the group your company belongs is part of.	My Group > Systems
10	Send a message to one of your teammates.	My Account > Messages
11	Review the last 3-month history for VOLTA unit 12345.	Equipment > System Details > Status and Usage
12	Check that all the TEMATE SI-WB systems in your plant are working with no issues	Equipment > System Details > Status and Usage
13	You are a manager and have access to company group wide data. Review the amount of parts inspected from all the plants in the group you belong to.	My Group > Statistics
14	Check the status of all the issues you have reported.	Help Desk
15	You received an automated email from NDT-Link informing that there is a new software update available for your PRIMO SC unit. You don't want to connect this unit to the Internet for automatic updates, instead, you want to update the unit manually. Download the update package from from NDT-Link.	Equipment > System Details > Software Updates

Figure 11: Tree Test - Tasks for User profiles

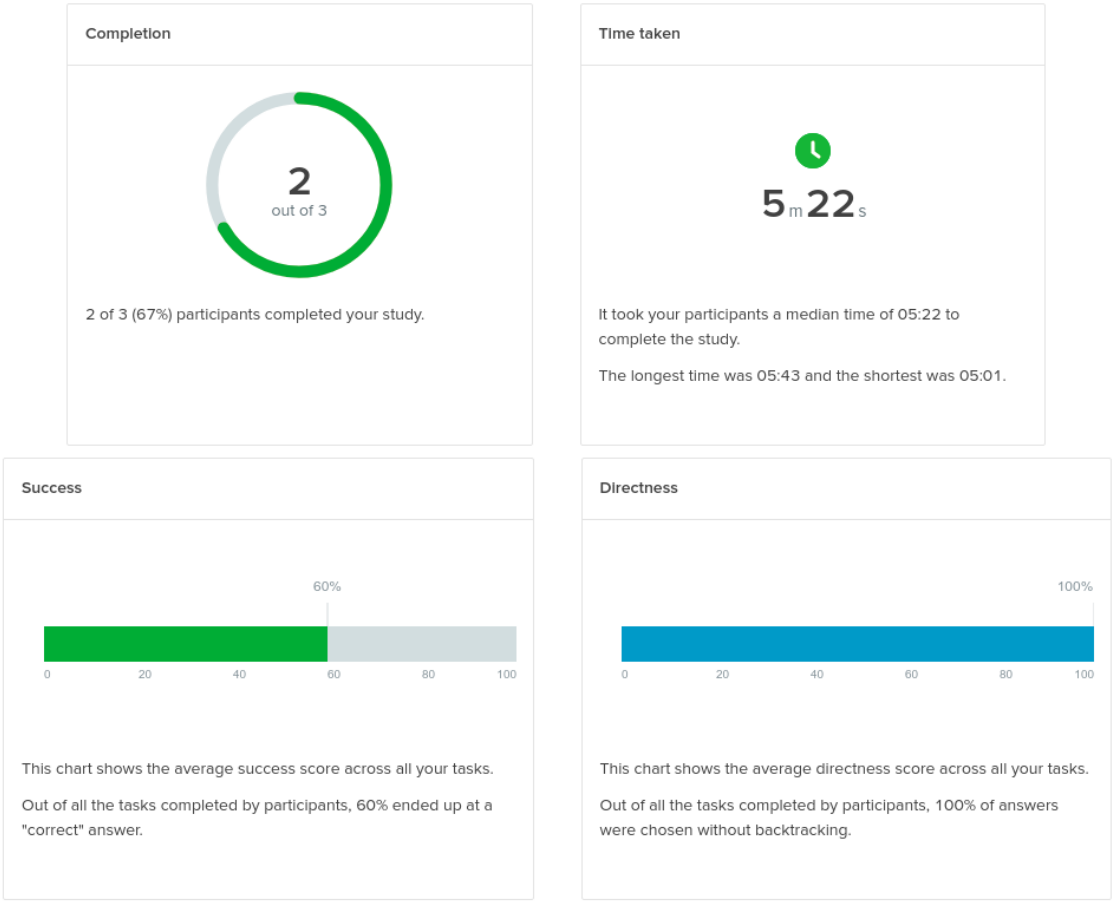
	Task	Correct Answer
1	Find all the users that ArcelorMittal group has registered in NDT-Link.	Groups > Users
2	You need to ship spares to company Shiloh Pendergrass. Find its shipping address.	Companies > Locations
3	List all the open issues reported by all customers for the software VOLTA	Software > Issues
4	There is a new version of the TEMATE SI-WB user guide. Upload it to NDT-Link.	Products > Documentation
5	A new VOLTA unit has been assembled for SpaceX. Register it on NDT-Link before it can be shipped.	Systems > Register New
6	The Software Team has uploaded a new update for the Rollmate software. Find it so you can download it.	Software > Updates
7	The Software Group has created a new module on the VOLTA software focused on circular inspections on plates. Register this new module on NDT-Link so users can purchase licenses to use it.	Software > Modules > Register
8	The Sales team has decided to start charging for the MRUT software module on the VOLTA software. Update the license type for this module on NDT-Link.	Software > Modules > Licenses
9	Solblank is having troubles with some of their TEMATE SI-WB systems. You agreed to monitor their status from NDT-Link and call them in case you see something strange. Find where to monitor these systems.	Systems > Usage and Status
10	Update your contact information.	My Account > Profile
11	List all open issues reported by users.	Help Desk > Issues
12	You received an automated email from NDT-Link informing that Mike Smith with US Steel has reported a problem with one of their systems. Find the issue to review and respond to the customer.	Companies > Issues Users > Issues Help Desk > Issues
13	Register a user account.	Users > Register New/Edit/Block/Delete
14	Find all the users who have access to VOLTA unit with S/N 12345.	Systems > User Accounts
15	An in-service customer is having issues interpreting the results of a scan performed with VOLTA on a pipe. The scan has been uploaded to NDT-Link automatically by the instrument. Review the scan so you can come back to the customer with your feedback.	Systems > Inspection Data
16	Check all the active software licenses for VOLTA unit with SN 12345.	Software > Modules > Licenses

Figure 12: Tree Test - Tasks for Innerspec profiles

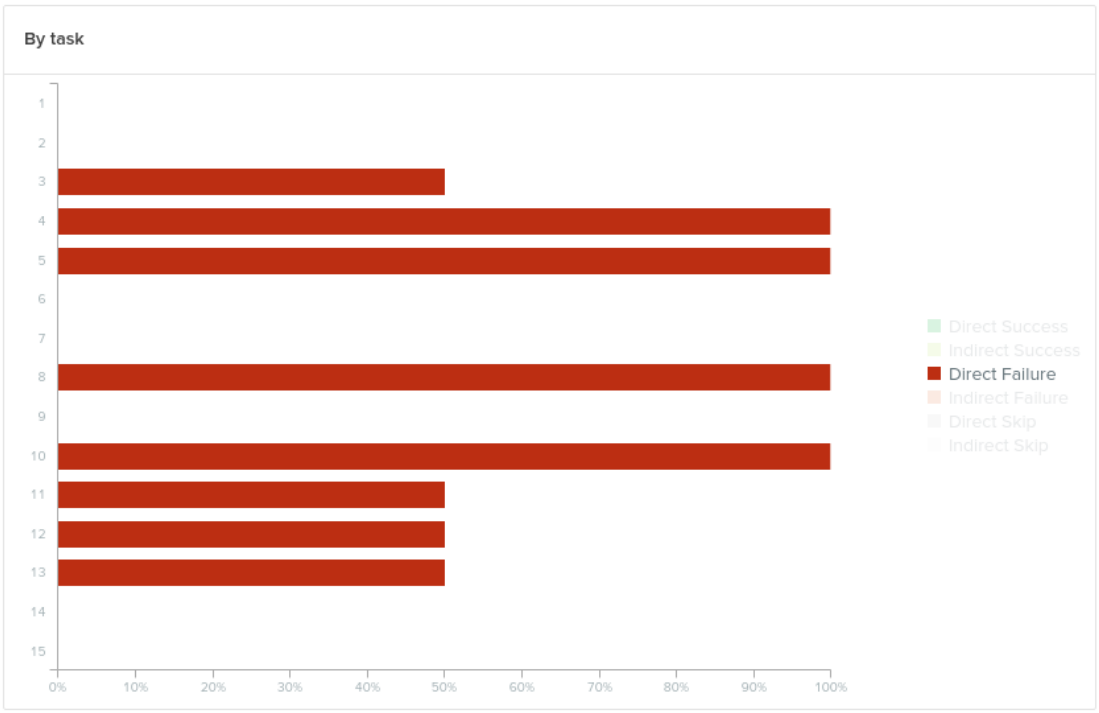
Test Results

This is the first time Innerspec conducts usability tests, so the experience was new for all the team. Some of the testers didn’t complete the test, so the final result sample ended up being smaller than expected.

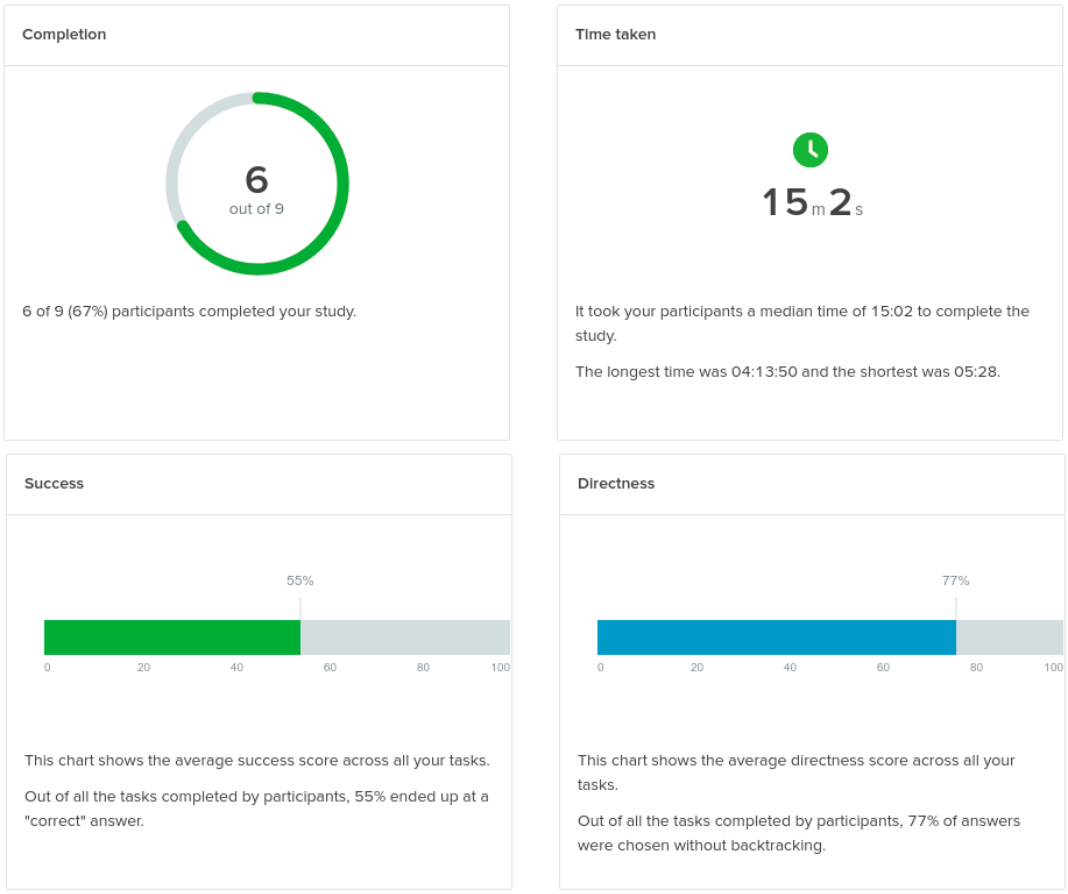
For the User test, the average success rate was 60% and time needed to complete the tasks was almost 5 minutes and a half:



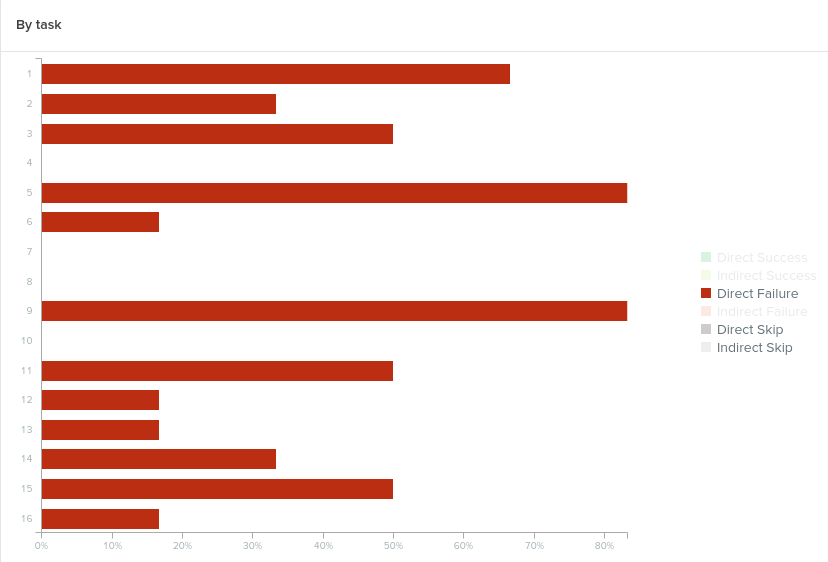
The direct failures were all related to tasks that required to understand the difference between groups, companies, product and software.



As for the Innerspec test, the average success rate was slightly lower, 55%, however the time needed to complete the tasks was almost three times the one for the User test:



The direct fails were almost identical as with the test for the User profile.



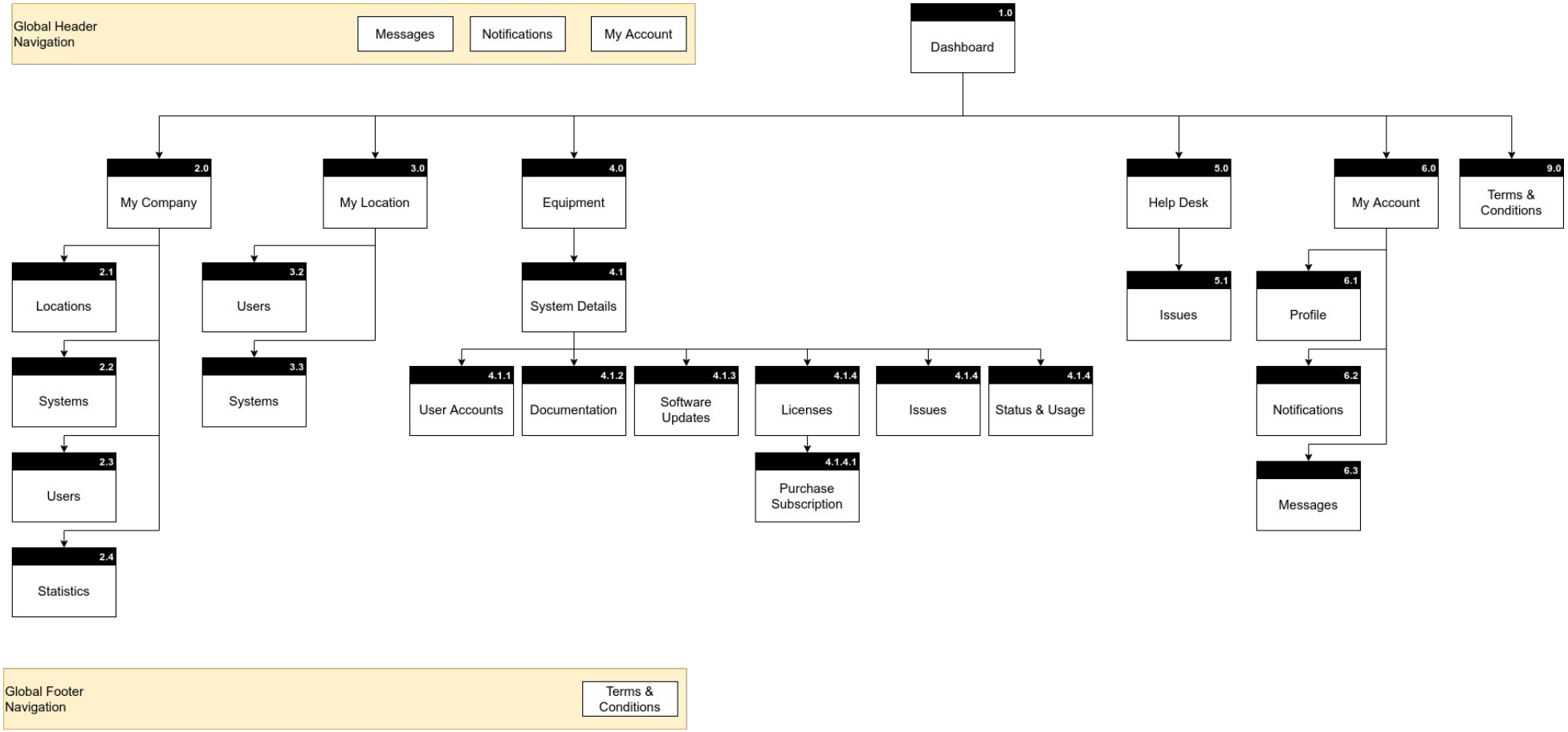
From these results and analyzing each path taken by the testers for each task, it was concluded that the testers in most of the cases were confusing the term Group (a set of companies) with Company, and the term Software (generic) with Product, and sometimes even with System (one particular instance of a product). Also, in some occasions, the testers didn't understand the difference between User generated content and Innerspec managed content.

The results of these tests are provided as ANNEX 5 to this document.

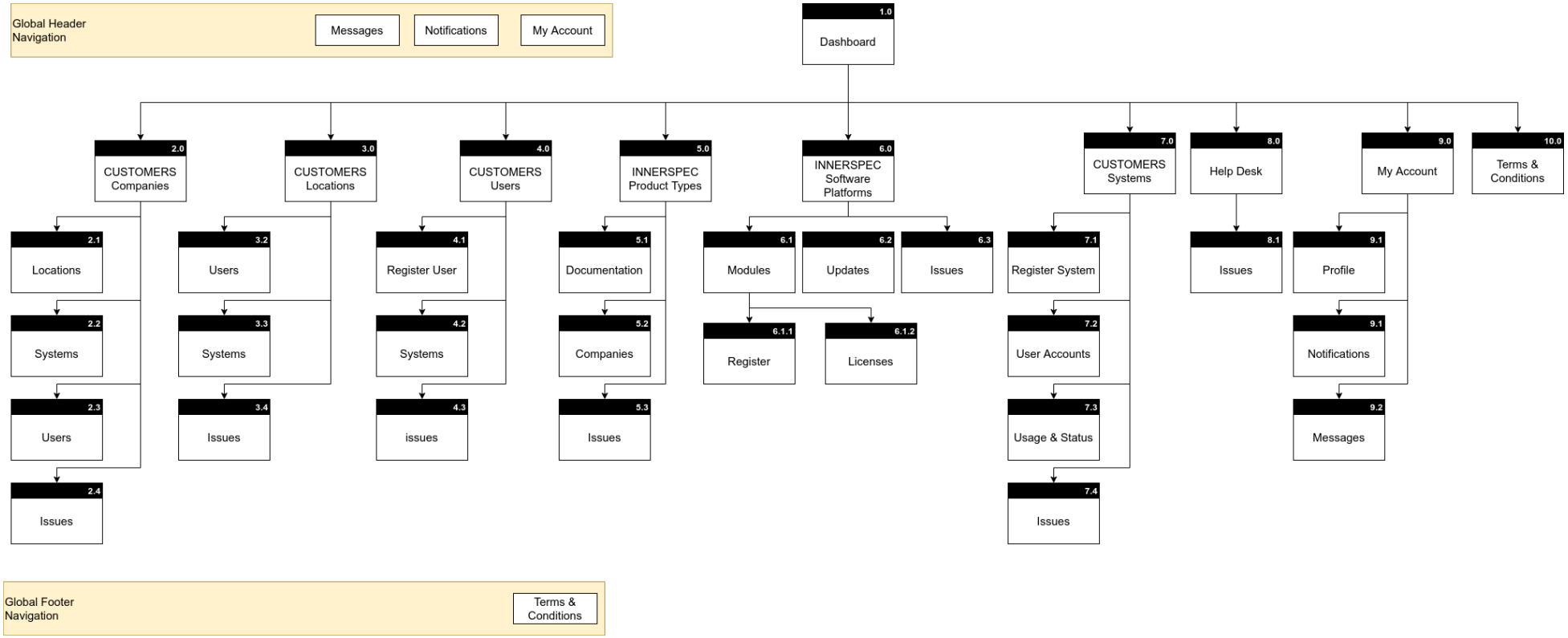
5.3. Improved Information Architecture

Based on the results from the Tree Tests, the Information Architecture was modified as indicated in the following sections.

Improved Information Architecture for User profiles



Improved information Architecture for Innerspec profiles

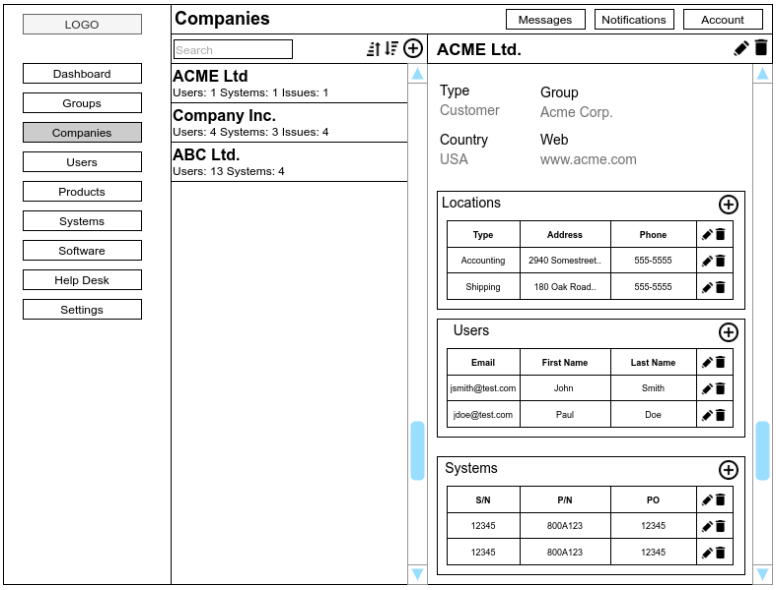


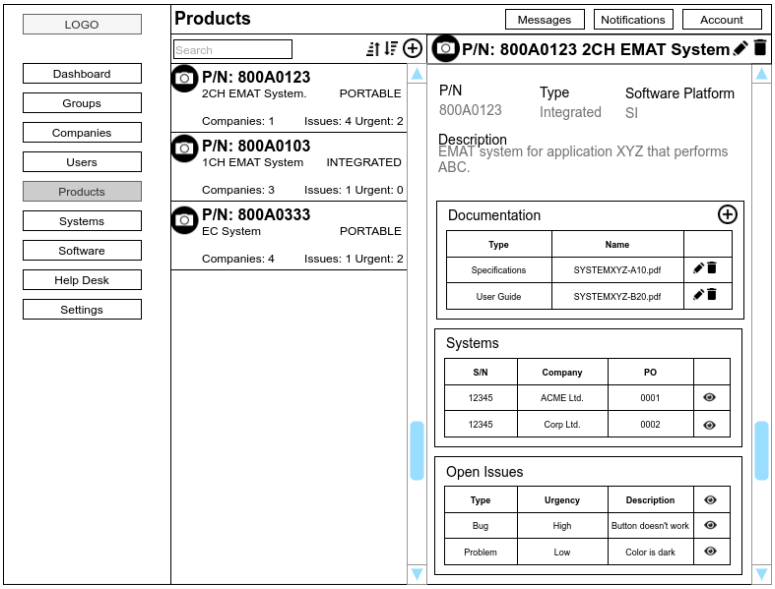
6. Prototypes

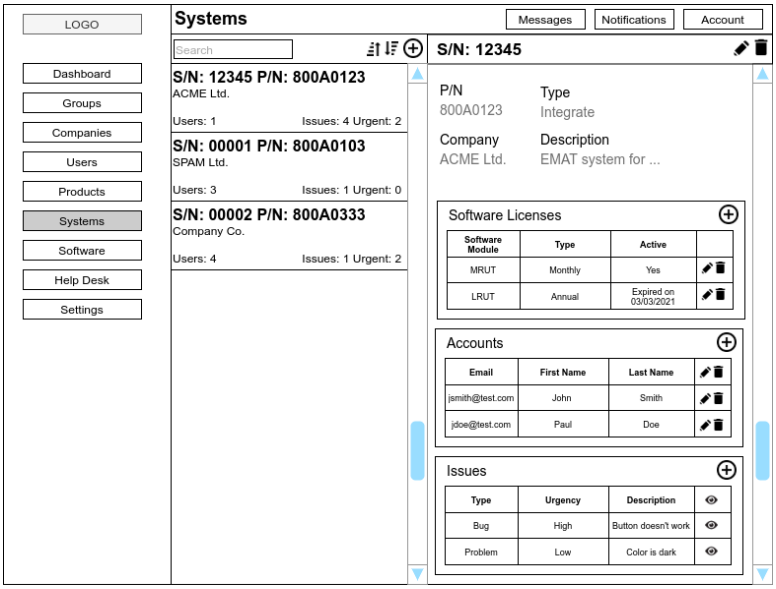
NDT-Link is an application meant to be used mainly on desktop computers due to the several complex operations that can be done on it and the amount of data that can it can display. This is why a desktop-first approach was selected for the design and implementation of prototypes.

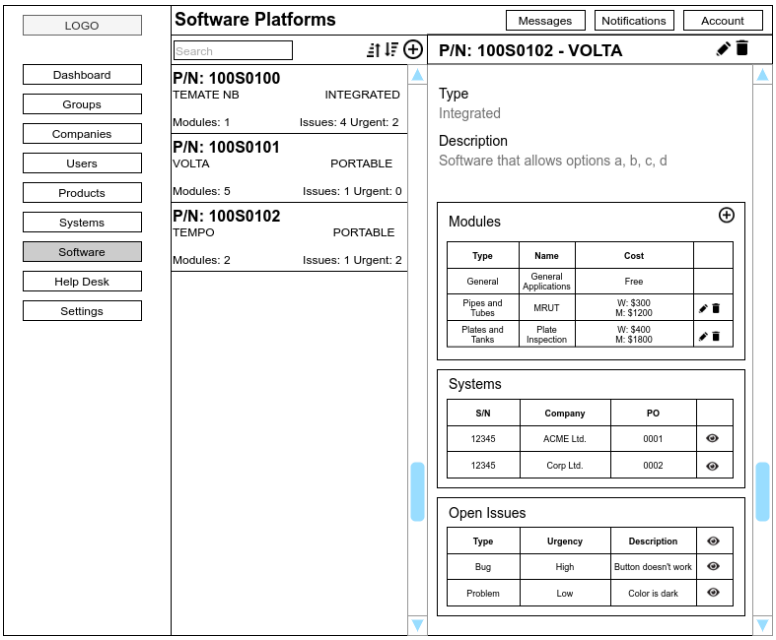
6.1. Lo-Fi

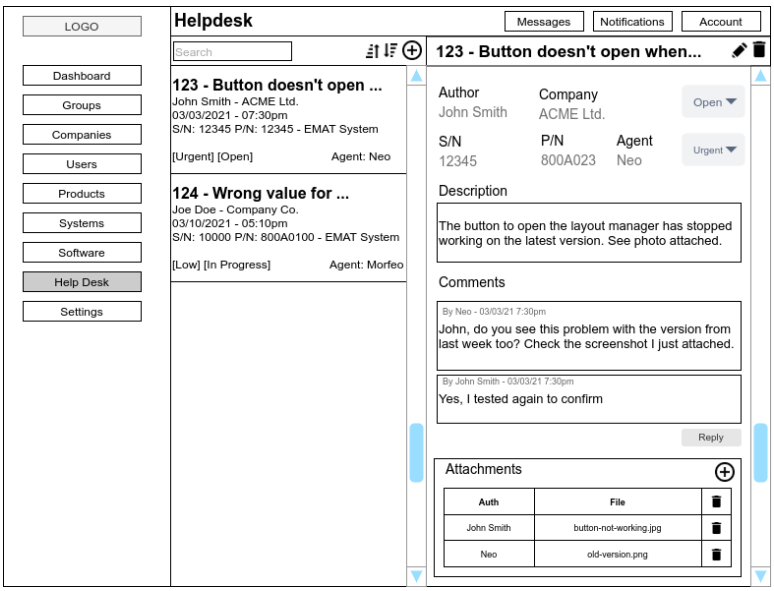
The following prototypes were designed for the main view used to list, create, edit and delete content in all sections. The structure is similar to a mail browser, where items are listed in one side of the screen, and clicking on them will show their details on the other one. The detail view provides several actions to be performed on the selected item, as well as embedded lists of its related content.






LOGO
Dashboard
Groups
Companies
Users
Products
Systems
Software
Help Desk
Settings
Systems
Messages
Notifications
Account
Search
S/N: 12345 P/N: 800A0123
ACME Ltd.
Users: 1
Issues: 4 Urgent: 2
S/N: 00001 P/N: 800A0103
SPAM Ltd.
Users: 3
Issues: 1 Urgent: 0
S/N: 00002 P/N: 800A0333
Company Co.
Users: 4
Issues: 1 Urgent: 2
S/N: 12345
P/N
800A0123
Type
Integrate
Company
ACME Ltd.
Description
EMAT system for ...
Software Licenses
Software Module
Type
Active
MRUT
Monthly
Yes
LRUT
Annual
Expired on 03/03/2021
Accounts
Email
First Name
Last Name
jsmith@test.com
John
Smith
jdoe@test.com
Paul
Doe
Issues
Type
Urgency
Description
Bug
High
Button doesn't work
Problem
Low
Color is dark


LOGO
Dashboard
Groups
Companies
Users
Products
Systems
Software
Help Desk
Settings
Software Platforms
Messages
Notifications
Account
Search
P/N: 100S0100
TEMATE NB
INTEGRATED
Modules: 1
Issues: 4 Urgent: 2
P/N: 100S0101
VOLTA
PORTABLE
Modules: 5
Issues: 1 Urgent: 0
P/N: 100S0102
TEMPO
PORTABLE
Modules: 2
Issues: 1 Urgent: 2
P/N: 100S0102 - VOLTA
Type
Integrated
Description
Software that allows options a, b, c, d
Modules
Type
Name
Cost
General
General Applications
Free
Pipes and Tubes
MRUT
W: $300
M: $1200
Plates and Tanks
Plate Inspection
W: $400
M: $1800
Systems
S/N
Company
PO
12345
ACME Ltd.
0001
12345
Corp Ltd.
0002
Open Issues
Type
Urgency
Description
Bug
High
Button doesn't work
Problem
Low
Color is dark

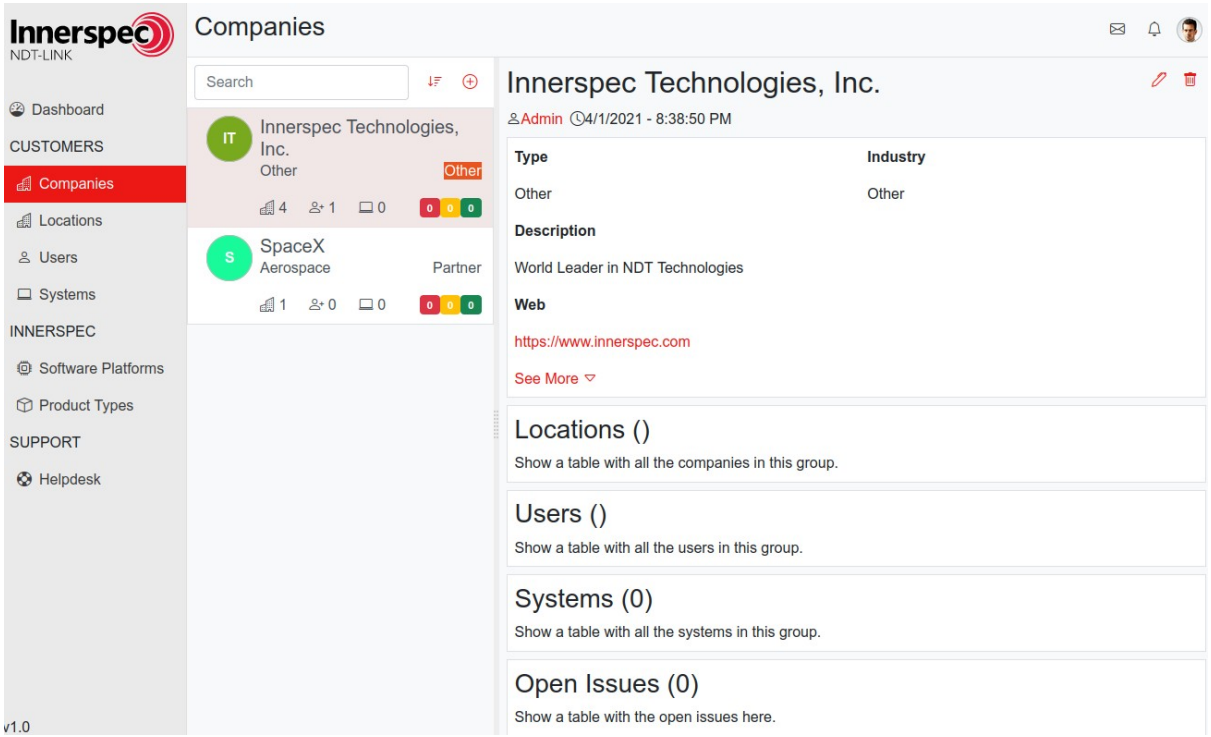

LOGO
Dashboard
Groups
Companies
Users
Products
Systems
Software
Help Desk
Settings
Helpdesk
Messages
Notifications
Account
Search
123 - Button doesn't open ...
John Smith - ACME Ltd.
03/03/2021 - 07:30pm
S/N: 12345 P/N: 12345 - EMAT System
[Urgent] [Open]
Agent: Neo
124 - Wrong value for ...
Joe Doe - Company Co.
03/10/2021 - 05:10pm
S/N: 10000 P/N: 800A0100 - EMAT System
[Low] [In Progress]
Agent: Morfeo
123 - Button doesn't open when...
Author
John Smith
Company
ACME Ltd.
Open
S/N
12345
P/N
800A023
Agent
Neo
Urgent
Description
The button to open the layout manager has stopped working on the latest version. See photo attached.
Comments
By Neo - 03/03/21 7:30pm
John, do you see this problem with the version from last week too? Check the screenshot I just attached.
By John Smith - 03/03/21 7:30pm
Yes, I tested again to confirm
Reply
Attachments
Auth
File
John Smith
button-not-working.jpg
Neo
old-version.png

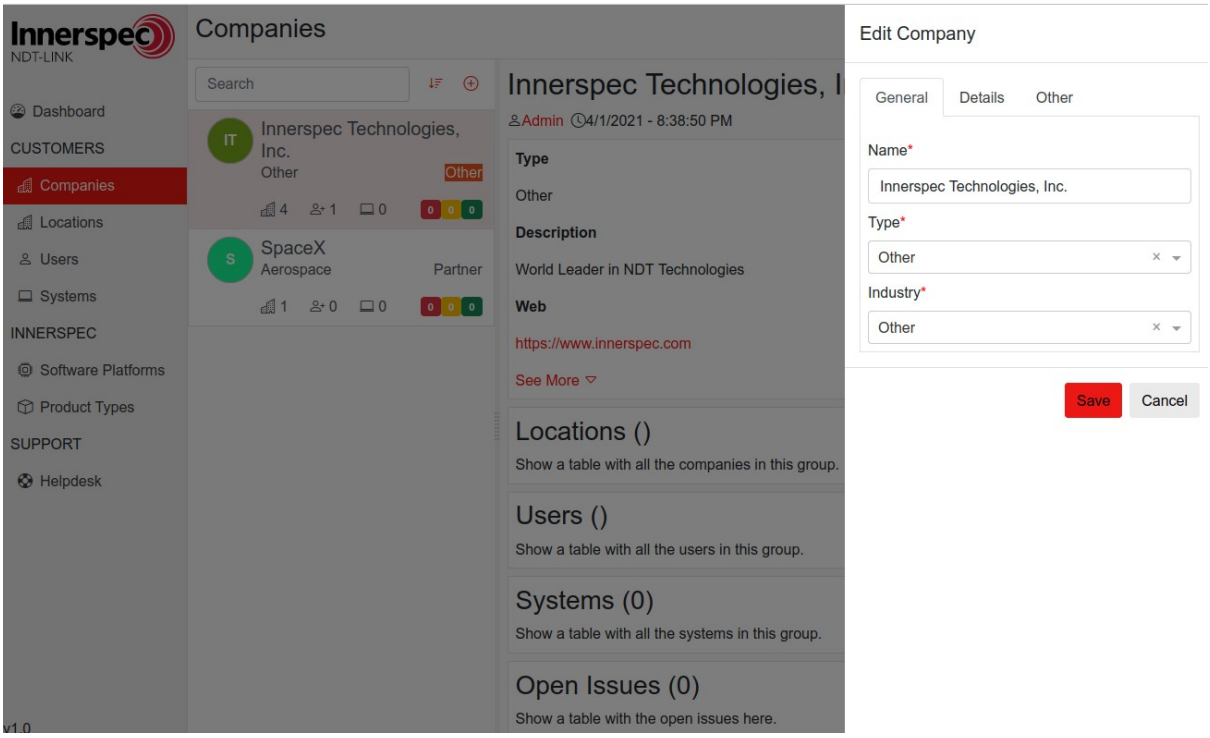
The designs were made only for the Innerspec profile and their usability was tested using *First Click* technique on the *Optimal Workshop* platform. The testers were the same people who took the Tree Test for the preliminary Information Architecture made for Innerspec profiles. The First Click test was done at the same time as the Test Tree in order to optimize the time dedication of the participants.

The resultant success rate was high, around 80%. The reasons for the failures were related in most of the occasions to not identifying the meaning of some icons, or mistaken static texts for links. These two issues will be fixed by showing tool-tips when hovering the icons and by styling the links differently from the rest of the text.

6.2. Hi-Fi

The following figures show the final prototypes for the main view and the create/edit form, based on the results obtained from the Tree testing and First Click test results.




Innerspec
NDT-LINK
Companies
Search
Dashboard
CUSTOMERS
Companies
Locations
Users
Systems
INNERSPEC
Software Platforms
Product Types
SUPPORT
Helpdesk
IT
Innerspec Technologies, Inc.
Other
Other
4 1 0
S
SpaceX
Aerospace
Partner
1 0 0
Innerspec Technologies, I
Admin 4/1/2021 - 8:38:50 PM
Type
Other
Description
World Leader in NDT Technologies
Web
https://www.innerspec.com
See More
Locations ()
Show a table with all the companies in this group.
Users ()
Show a table with all the users in this group.
Systems (0)
Show a table with all the systems in this group.
Open Issues (0)
Show a table with the open issues here.
v1.0
Edit Company
General
Details
Other
Name*
Innerspec Technologies, Inc.
Type*
Other
Industry*
Other
Save
Cancel

7. Development Platform

The following are the main technologies selected to implement NDT-Link:

1. **Front-End***:*
    1. HTML5/CSS3.
    2. Bootstrap with custom SASS following H.Roberts[1] style guide.
    3. Angular as web application framework with NgRx[8] for state management:
        - The reason to select this framework over other options such as React or Vue is because Angular contains all the necessary tools to build a business class application out of the box, reducing dependencies, and it provides an opinionated structure that ease sharing the project with other developers who also know Angular[2].
    4. Unit Testing and E2E testing with Jasmine and Karma.
2. **Back-End***:*
    1. Node.js as server side platform with TypeScript[9]:
        - The main reason to select this back-end technology over PHP, C# or Java is to accelerate development by using the same language on both the front-end and back-end, which will ease the development process done by one single person (the author of this thesis)[3].
        - Another reason is that the Software team at Innerspec Technologies is very experienced with JavaScript and Node.js. Using the same technologies in NDT-Link will ensure that others from the team can expand the project in the future with a small learning curve.
    2. NestJS server side framework based on Express.js.
    3. Database connection and queries:
        1. TypeORM with raw SQL queries in certain cases to improve performance.
    4. Authentication/Authorization:
        1. Passport.js middleware with local and JWT[4] token based authentication strategies.
    5. AWS SDK for transactional emails.
3. *Database Service:*
    1. MySQL server.
4. *Hosting:*
    1. Amazon Web Services have been selected to host NDT-Link. The following sub-services operate behind an Elastic Load Balancer and CloudFront CDN[10] in a multi-zone configuration to optimize access to all users around the globe:
        1. Front-End: AWS S3.
        2. Back-end: AWS Beanstalk (EC2 instance).
        3. Database Service: AWS RDS.
5. *Transactional Email Service:*

8 https://ngrx.io/

9 https://www.typescriptlang.org/

10 Content Delivery Network

    1. Amazon SES (Simple Email Service).
6. *Other*:
    1. MapBox for maps and address geocoding:
        1. *Mapbox* is a cloud based location platform that provides maps and geolocation services similar to Google Maps. Mapbox provides an API with several endpoints that handle different types of requests for fetching maps or for parsing addresses to generate geographic coordinates. The main advantage of Mapbox over Google Maps is that Mapbox offers 50.000 map loads at no additional cost.
    2. PayPal as gateway for payments.

8. Planning

8.1. Implementation Stages

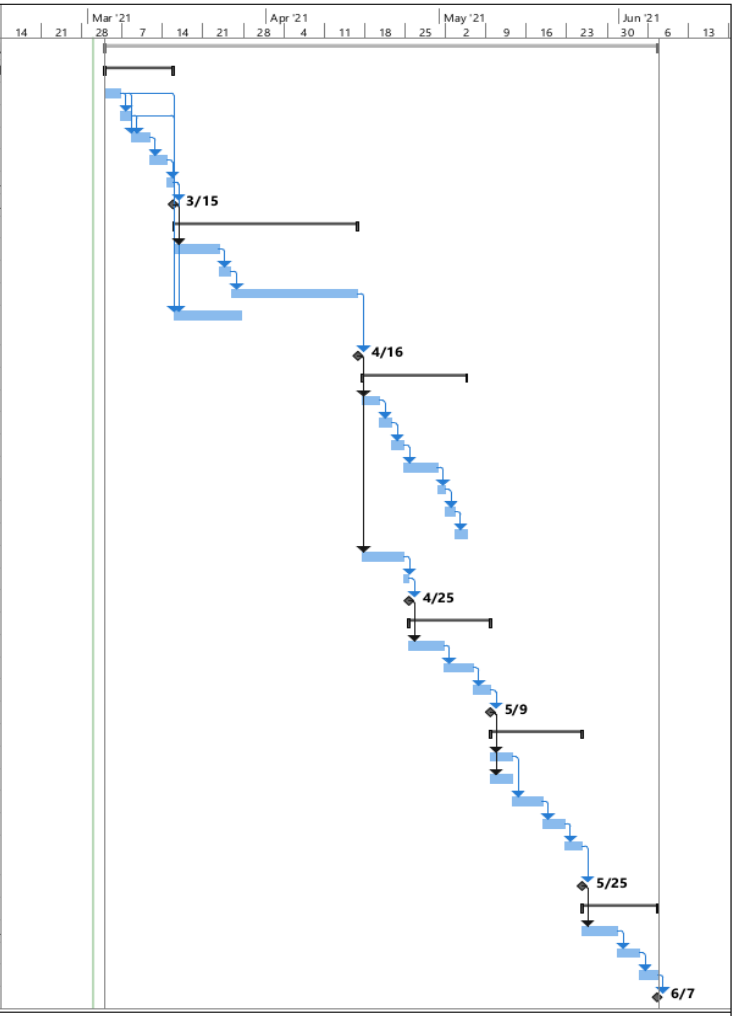
NDT-Link will be implemented in six consecutive stages:

1. *Product Design:*
    - The goal of this stage is to create specifications needed for the development of the main components that conform NDT-Link. These specifications will contain UML diagrams, model descriptions of the data entities to use, selected software stack for development, design mock-ups for the front-end, and a detailed description list of the different endpoints served by the back-end.
2. *Walking Skeleton:*
    - The goal of this stage is to implement prototype versions for the front-end, back-end and Product Connector based on the specifications created in the design stage. These prototypes will use mock data and will have minimal interactivity between them.
    - Once finished, the prototypes will be shared with personnel from UOC and Innerspec for review and feedback.
3. *Minimum Viable Product:*
    - In this stage, the prototypes will be expanded with all the required functionality to produce the first fully functional beta version of NDT-Link. This will include the implementation of all the must-to-have functionality in each component, as well as full connectivity front-end/back-end, and back-end/Product Connector.
4. *Minimum Marketable Product:*
    - This stage will expand the beta version with need-to-have functionality before it can be released to Innerspec personnel for full usage. This stage will include a finalized Product Connector that can upload inspection data, and updated back-end/front-end that can process and display it.
    - In this stage, End-to-End tests will be performed too in order to automate and accelerate some of the testing and debugging.
5. *Minimum Delightful Product:*
    - The main purpose of this stage is to enhance the beta version of NDT-Link with a custom or improved CSS theme, as well as extra functionality such as maps and geolocation, user to user messaging, notifications and transactional mailing. Also, in this stage, the existing End-To-End tests will be expanded in order to cover more test cases.
    - Once finished, this version of NDT-Link will be released to Innerspec customers.
6. *Documentation:*
    - This final stage is dedicated to the formalization of the project documentation and to the creation of media that will be used to showcase NDT-Link.

8.2. Gantt Chart

The following chart shows a preliminary list of tasks for each one of the six implementation stages of the project, including their estimated duration and expected milestones in red. The chart also contains a timeline that shows each task positioned in time in relation to its dependencies.

ID	Task Mode	Task Name	Duration	Predecessors	Start	Finish
0		**NDT-Link**	**87 days**		**Thu 3/4/21**	**Mon 6/7/21**
1		**Product Design**	**11 days**		**Thu 3/4/21**	**Mon 3/15/21**
2		UML Diagrams & Data Modeling	2 days		Thu 3/4/21	Sat 3/6/21
3		Application Architecture	2 days	2	Sat 3/6/21	Mon 3/8/21
4		Front-End: Information Architecture & Usability Study	3 days	2,3	Mon 3/8/21	Thu 3/11/21
5		Front-End: Prototypes	3 days	4	Thu 3/11/21	Sun 3/14/21
6		Back-End: Definition of End-Points	1 day	5	Sun 3/14/21	Mon 3/15/21
7		**Design specifications ready**	0 days	6	Mon 3/15/21	Mon 3/15/21
8		**Walking Skeleton**	**29 days**		**Tue 3/16/21**	**Fri 4/16/21**
9		Back-End: CRUD endpoints with mock data	7 days	7	Tue 3/16/21	Tue 3/23/21
10		Back-End: Authentication and authorization endpoints with mock data	2 days	9	Tue 3/23/21	Thu 3/25/21
11		Front-End: CRUD pages	20 days	10	Fri 3/26/21	Fri 4/16/21
12		Product Connector: Authentication and Authorization with back-end, and basic communications with product's firmware	2 wks	2,3,6	Tue 3/16/21	Sat 3/27/21
13		**Prototype finished**	0 days	11	Fri 4/16/21	Fri 4/16/21
14		**Minimum Viable Product**	**17 days**		**Sat 4/17/21**	**Wed 5/5/21**
15		Back-End: Database persistence for all endpoints	3 days	13	Sat 4/17/21	Tue 4/20/21
16		Back-End: Endpoints for recovery of user accounts	2 days	15	Tue 4/20/21	Thu 4/22/21
17		Back-End: Simple management of transactional emails	2 days	16	Thu 4/22/21	Sat 4/24/21
18		Back-End: Connection with credit card processing service	5 days	17	Sat 4/24/21	Fri 4/30/21
19		Front-End: Dashboard page	2 days	18	Fri 4/30/21	Sun 5/2/21
20		Front-End: User profile page	1 day	19	Sun 5/2/21	Mon 5/3/21
21		Front-End: pages for recovery of user accounts	2 days	20	Mon 5/3/21	Wed 5/5/21
22		Product Connector: Synchronization of users and status information with back-end	7 days	13	Sat 4/17/21	Sat 4/24/21
23		Create show video showcasing the product	1 day	22	Sat 4/24/21	Sun 4/25/21
24		**Beta version finished**	0 days	23	Sun 4/25/21	Sun 4/25/21
25		**Minimum Marketable Product**	**13 days**		**Sun 4/25/21**	**Sun 5/9/21**
26		Product-Connector: Upload of inspection data to back-end	1 wk	24	Sun 4/25/21	Sat 5/1/21
27		Front-End: Plot charts with inspection data uploaded from products	1 wk	26	Sat 5/1/21	Thu 5/6/21
28		E2E Testing	3 days	27	Thu 5/6/21	Sun 5/9/21
29		**Publish product internally for Innerspec personnel**	0 days	28	Sun 5/9/21	Sun 5/9/21
30		**Minimum Delightful Product**	**14 days**		**Sun 5/9/21**	**Tue 5/25/21**
31		Front-End: Improved custom CSS theme	3 days	29	Sun 5/9/21	Thu 5/13/21
32		Product-Connector: Upload product geolocation to back-end	3 days	29	Sun 5/9/21	Thu 5/13/21
33		Front-End: Maps showing current and historic geolocation of products	5 days	31	Thu 5/13/21	Tue 5/18/21
34		Front-End: User to user messaging and events notifications	3 days	33	Tue 5/18/21	Sat 5/22/21
35		Transactional Emails: Implement separate Mail Service and update back-end email management	3 days	34	Sat 5/22/21	Tue 5/25/21
36		**Publish product for Innerspec customers**	0 days	35	Tue 5/25/21	Tue 5/25/21
37		**Product Documentation**	**12 days**		**Tue 5/25/21**	**Mon 6/7/21**
38		Finish Thesis	6 days	36	Tue 5/25/21	Mon 5/31/21
39		Create written presentation of the product for public audience	3 days	38	Mon 5/31/21	Fri 6/4/21
40		Create video showcasing the product	3 days	39	Fri 6/4/21	Mon 6/7/21
41		**Final product documentation ready**	0 days	40	Mon 6/7/21	Mon 6/7/21



Page 1

Figure 13: Gantt chart for estimated implementation planning

9. Implementation

9.1. Back-End

A RESTful[11] API[12] of the applications has been developed with *NestJS* and *TypeORM* using *Data Transfer Objects*[13] to map database entities to responses sent to the front-end and to receive payloads from it for create, update and delete operations. DTO mapping is achieved by using the *class-transformer*[14] package, and the validation of payloads sent with POST and PATCH requests is done with the the package *class-validator*[15].

The API also contains a module for authentication that uses *Passport.js* to implement username/password and JWT based authentication strategies. The JWT token is issued after a successful login and sent in a 200 HTTP response to the front-end, who then must include the token in any consecutive request as a *Bearer* token in the *Authorization* HTTP header. This is achieved by a NestJS Interceptor and two guards that have been also implemented as part of the authentication module.

In order to deploy the database for the first time, the script *seed.ts* has been implemented to populate the database with company Innerspec Technologies, its locations, and the superuser that can be used to access the application for the first time to create more content. This script is executed every time the application starts, and will not perform any operations if the the database contains data.

In order to protect the application from malicious activities, CORS and rate limit policies have been applied among others in the NestJS entry script as basic security measures.

With the exception of the authentication and shared modules, the rest of the application source code is organized in modules containing domain functionality (Companies, Locations, etc..). Shared functionality is organized in modules stored under folder *shared*. Absolute paths to the modules using the prefix @ has been configured in the *tsconfig.json* file in order to facilitate the importation of functionality from any file in the source tree.

The configuration of the application is handled from the *config.ts* file, responsible for pulling certain values from environmental variables when available based on the current execution environment (development or production).

11 Conforms to the constraints of REST architectural style

12 Application Programming Interface

13 https://en.wikipedia.org/wiki/Data_transfer_object

14 https://github.com/typestack/class-transformer

15 https://github.com/typestack/class-validator

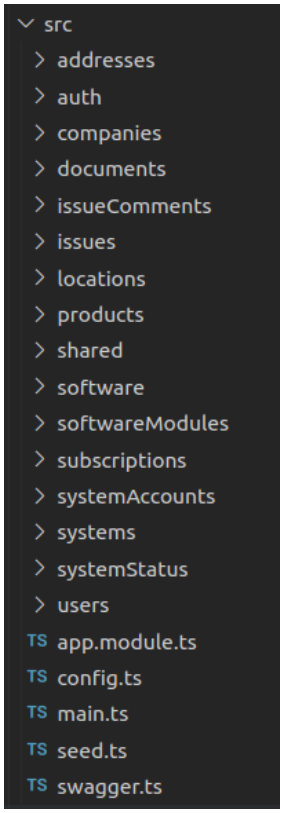


Figure 14: Implementation - Back-end file structure

Each domain module is organized in the same manner and includes a barrel export file:

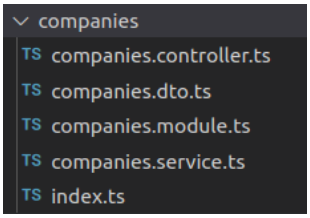


Figure 15: Implementation - Back-end module organization

The shared modules provide the following functionality:

1. *crud: generic CRUD controller and service that are extended by other controllers and services.*
2. *domain*: contains all the entities used by the database ORM.
3. *geocoding*: contains a service that parses addresses into geographical coordinates using the Mapbox API, described in detail in the section Used APIs.
4. *hmtl-template*: module that provides a service that parses HTML templates replacing tags in them with values.
5. *mail*: this module provides a service in charge of sending transactional emails. The Amazon Simple Email Service (ASES) section describes in detail how this module and the *html-template* one work.

6. *persistence*: configures the database ORM and provides it as a connection to the rest of the application.
7. *infrastructure*: provides common functionality that can be used from the rest of the modules.
8. *payment*: this module has been implemented to process payments with the PayPal gateway. Section PayPal describes in detail its implementation, and section Sequence Diagrams describes all the stages involved with several UML diagrams.
9. *pipes* and *validators*: contain NestJS custom pipes and validators shared by other modules.

Figure 16: Implementation - Back-end shared modules

In order to start the application, the file README.MD provides instructions for both, development and production environments. Once the application starts, it will attempt to connect to a database that must have been previously created. The name of the database, its credentials, and the URL of the server that hosts it, are defined in the *config.ts* file:

```
const config = () => ({
  site: {
    name: 'NDT-Link',
    description: 'NDT-Link API',
    // URL from where the application will be listening.
    url: isProduction() ? 'http://api.ndt-link.com' : 'http://localhost',
    port: +process.env.PORT || 3000,
    // Credentials for the superuser created by the seeder.
    superEmail: process.env.APP_ADMIN_EMAIL || 'admin@innerspec.com',
    superPassword: process.env.APP_ADMIN_PASSWORD || 'Temate#1',
  },
  client: {
    // URL where the front-end is hosted, used for CORS.
    url: isProduction() ? 'http://www.ndt-link.com' : 'http://localhost:4200',
  },
  // Credentials for the MySQL server.
  database: {
    server: process.env.DB_SERVER || 'localhost',
    port: +process.env.DB_PORT || 3306,
    name: process.env.DB_NAME || 'ndtlink_nest',
    user: process.env.DB_USER || 'root',
    password: process.env.DB_PASSWORD || 'temate',
  },
  // Configuration for JSON Web Tokens used for authentication.
  jwt: {
    secret: process.env.JWT_SECRET || 'SomethingReallySuperWeird',
    signOptions: {
      // All tokens will expire one hour after they are issued.
      expiresIn: process.env.JWT_DURATION || '3600s',
    },
    ignoreExpiration: false,
  },
});
```

Figure 17: Implementation - Back-end configuration file

9.2. Product Connector

ANNEX 10 describes in detail the endpoints implemented by the back-end to communicate with the Product Connector service installed on Innerspec systems. This service has been implemented by other members of the Software team and integrated in another custom tool called *System Manager*, a service responsible for configuring several aspects of the systems where it runs.

9.3. Front-End

The *Angular* application has been implemented using *DTOs* to map responses received from the back-end and payloads sent to it for creation and update operations. The application state is managed with NgRx Store, NgRx Effects, and NgRx Entity. For HTTP requests to the back-end, Angular HttpClient is used from the NgRx Effects.

Similarly to the back-end and with the exception of authorization, the organization of the source code in the front-end is based on domain functionality. Each module is structured the same way:

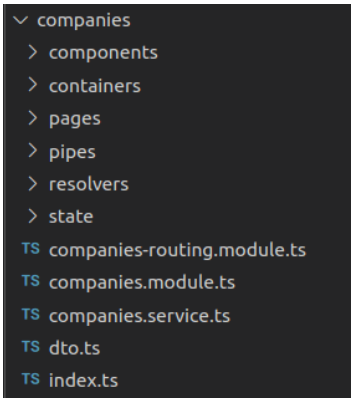


Figure 18: Implementation - Front-end module structure

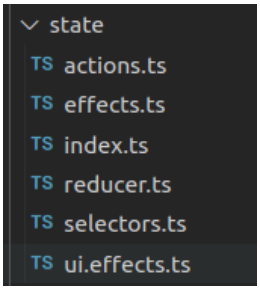
Each module provides its own routes defined in **-routing.module.ts* files which are imported from the *AppModule* router.

In terms of components, each module defines 3 different types:

1. *Components*: with well defined, one single concern, not-state-aware functionality.
2. *Containers*: are aware of the application state and use NgRx Store/Actions to fetch data that they pass to the components.
3. *Pages*: merely for composition, they are the “landing” pages defined for each route. They use the containers to create the desired markup.

For the detail pages that show content for a certain record in the database, the *resolvers* folder contains *Angular Resolvers* are implemented to pull the information from the back-end using NgRx Effects before the route is completed. Also, some modules have a *pipes* folder that contains *Angular Pipes* used to format Enums in the markup.

Finally, the state folder contains all the NgRx related functionality:



The effects are split into two files, one for domain related functionality (fetch all, fetch one, create, update, etc..), and another one just for UI responses to changes in state (i.e. show notification when a new item is created).

For authentication, the *auth* module implements an interceptor that injects the token in the HTTP requests, as well as guards to check if the user is authenticated and the role they have. Section Security describes in detail this process and others related to security.

The *site* module implements generic pages not related to any domain in particular. It also implements an interceptor to handle errors.

The shared folder contains several modules, the mos important ones are:

1. *config*:
    1. A self-made module to handle and inject the application configuration (pulled from the environmental variables) from any module.
2. *ui*:
    1. A self-made module that implements many of components that are used by the domain modules to compose their components.
    2. This module also includes directives and pipes that can be used by other modules.
3. *modal*:
    1. A module implemented to handle modal dialogs using Bootstrap CSS.
4. *modal-form*:
    1. This module was implemented to handle forms in modal dialogs using Bootstrap CSS.
5. *notifications*:
    1. A self-made module to handle emission of “Toast” notifications using Bootstrap CSS.
6. *documents*:
    1. A module implemented to handle file uploads to the back-end.
7. *model-selector*:
    1. This module provides drop down boxes for different domains (User, Company, etc..) that perform queries to the back-end to provide asynchronous search results when the user types in them. The reason for not having these components in the domain modules is to avoid circular dependency between modules (i.e., Locations imports User, but creating a new User, the application must also present a component from the Locations module).

8. *classes*:
    1. Contains some self-made classes that are reused to extend functionality when implementing new components and services. The two most relevant ones are:
        1. *FormComponent*:
            1. Implements common functionality for Angular Reactive forms that is repetitive. All forms in the application extend from it.
        2. FormInputControl:
            1. Implements common functionality for Angular Reactive form elements that is repetitive, such as error management. All form components defined in the application extend from it.
        3. *CRUDService*:
            1. Implements a basic service that uses the *Angular HttpClient* to perform CRUD operations with the back-end.
9. *pipes*:
    1. Contains custom Angular pipes shared by different modules.
10. *static*:
    1. Contains JSON data used by certain components (i.e., name of countries).

The application configuration is stored under *src/environments* folder following Angular standards. Also, the folder *src/theme* contains SASS files used to customize some aspects of Bootstrap.

The file README.MD provides instructions on how to launch the application in development mode. Also, section Installation/Deployment Instructions describes this topic in more detail.

10. Used APIs

10.1. Mapbox

NDT-Link uses *Mapbox* to show maps on the front-end containing markers that indicate the position of Locations and status updates sent from customers' systems deployed in the field. The implementation of *Mapbox* functionality is done on the front-end and back-end.

Front-End

The component *UiMapComponent* was created as part of the *UiModule* to show a map with markers passed as inputs. This component is used on the site dashboard page, company page, location page, and system page.

```
<ng-container *ngIf="{loading: isLoading$ | async, items: items$ | async} as data">

    <ng-container *ngIf="data.loading; else map">
        Loading...
    </ng-container>

    <ng-template #map>
        <ui-map [name]="name" [markers]="markers" [zoom]="1"></ui-map>
    </ng-template>

</ng-container>
```

Figure 19: Mapbox - UiMapComponent used to compose the dashboard system map

Back-End

The module *GeocodingModule* was created to parse addresses into geographic coordinates. It provides service *GeocodingService* that sends a request to *Mapbox* geocoding endpoint containing the address to parse, and it receives as response the latitude and longitude of the address. This service is used by the module *LocationsModule to geocode* the locations' addresses before they are updated in the database.

```
async updateOne(
  id: string,
  changes: LocationUpdateDto,
): Promise<LocationDto> {
  const foundItem = await this.repo.findOne(id);
  if (foundItem) {
    // Before saving the new location, we need to geocode its address.
    const coords = await this._geocoding
      .codeAddress({
        street: changes.street,
        street2: changes.street2,
        city: changes.city,
        state: changes.state,
        zip: changes.zip,
        country: changes.country,
      })
      .toPromise();

    const udpatedItem = Object.assign(foundItem, changes);
    udpatedItem.lng = coords[0];
    udpatedItem.lat = coords[1];

    const p = await this.repo.save(udpatedItem);
    return this.findOne(p.id);
  }
```

Figure 20: Mapbox - GeocodeService called from LocationsService when updating a location

10.2. PayPal

In order to decouple payments from the rest of the application, the back-end module *PaymentModule* was developed to handle communications with the *PayPal* gateway. This module provides service *PaymentService* that uses *PayPal's Checkout* JavaScript SDK. This decoupling will simplify the change to a different payment gateway in the future when needed.

The *PaymentService* provides two methods for handling the creation and execution of a purchase order when called from the *SubscriptionsService*. These process is described in detail in the section UML Sequence Diagrams.

```
async createPayment(payment: PaymentWriteDto, returnUrl: string): Promise<string> {…
}

async executePayment(token: string): Promise<string> {…
}
```

Figure 21: PaymentService - Methods handling communications with PayPal gateway

```
async processPurchase(token: string): Promise<any> {
  const paymentId = await this._paymentService.executePayment(token);

  const payment = await this.paymentRepo.findOne({
    where: { id: paymentId },
    relations: ['system', 'softwareModule'],
  });
  if (!payment) {
    throw new HttpException(
      'Payment information not found after it was processed',
      HttpStatus.INTERNAL_SERVER_ERROR,
    );
  }

  const system = payment.system;
  const softwareModule = payment.softwareModule;

  const newSubscription = new SubscriptionCreateDto();
  newSubscription.system = system.id;
  newSubscription.softwareModule = softwareModule.id;
  newSubscription.type = 'weekly';

  return await this.addOne(newSubscription);
}
```

Figure 22: PayPal - SubscriptionService calling PaymentService method to process a purchase

10.3. Amazon Simple Email Service (ASES)

NDT-Link uses two modules to handle transactional emails that are sent from the back-end in response to certain user actions:

1. *MailModule*:
    1. Communicates with *Amazon Simple Mail Service* API through the AWS JavaScript SDK to send emails.
    2. Provides service *MailService* with method *SendMail* that is called from other back-end services when they need to send an email. This method expects an email address for the recipient, a subject and content for the body of the message.
2. *HtmlTemplateModule*:
    1. Loads predefined HTML files and replaces the tags in them with values passed in a flat object and returning the resultant HTML as a string.
    2. Provides service *HtmlMailService* with method *toString* that accepts the absolute path of an HTML file and a flat object with the values to use for the tag replacement. This service is called by other services when they need to parse their custom email HTML templates.

```
const newUser = Object.assign(new User(), { ...item, password, profile });
const savedUser = await this.repo.save(newUser);

// Send email
const msg = this._htmlTemplate.toString('users/mail/registration.html', {
  email: savedUser.email,
  password,
});
this._mail.sendEmail(savedUser.email, 'Welcome to NDT-Link', msg);
```

Figure 23: ASES - UsersService sending an email in response to a new account registration

The process of sending an email from a back-end service is a follows:

1. The service calls *HtmlTemplateService.toString* with the path to the HTML file and a dictionary of values.
2. The service calls *MailService.SendMail* with the email address for the recipient, the subject and the result from *HtmlTemplateService.toString* as content.
3. The call to *MailService.SendMail* is asynchronous and the caller service should not wait for it s resolution since Amazon can process the transactional email at any time.
4. The HTML templates are stored under the folders of each domain model since they are custom to the actions performed by the domain models.
5. The tags in the HTML templates are delimited by double curly braces.

This decoupling and modularization of the mail process was implemented to ensure an easy transition to alternative mail services and HTML template parsing libraries in the future if needed.

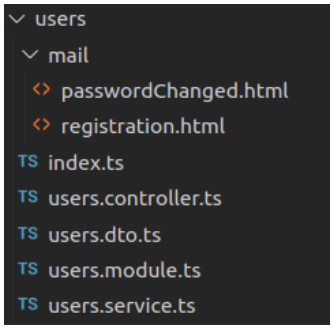


Figure 24: ASES - Location of email HTML templates used by the Users Module

```
<tr
    style="font-family: 'Helvetica Neue',Helvetica,Arial,sans-serif; box-sizing: border-box; font-size: 14
    <td class="content-block"
        style="font-family: 'Helvetica Neue',Helvetica,Arial,sans-serif; box-sizing: border-box; font-size
        valign="top">
        <p>Hello</p>
        <p>You have been invited to join NDT-Link.</p>
        <p>An account has been created for you with the following credentials:</p>
        <ul>
            <li>Username: <b>{{email}}</b></li>
            <li>Password: <b>{{password}}</b></li>
        </ul>
        <p>You can sign in NDT-Link <a href="{{baseUrl}}" title="Link to access NDT-Link">here</a>.</p>
    </td>
</tr>
```

Figure 25: ASES – Part of the HTML template used to send confirmation email for new accounts

11. UML Diagrams

11.1. Application Domain

NDT-Link has been implemented around a domain organized in five different categories of models:

1. **Contact Management**:
   - Responsible of all the operations related to the creation and maintenance of companies, locations and user accounts.
2. **Innerspec Product Management**:
   - Responsible of all the operations related to the creation and maintenance of Innerspec software platforms, software modules and available products.
3. **User Assets**:
   - Responsible of all the operations related to systems owned by customers.
4. **Help Desk**:
   - Handles all the operations related to the problems found by users when using their systems.
5. **Support**:
   - Provides functionality to the other domain categories.

Each domain category contains several models:

1. **Contact Management**:
   - Company
   - Location
   - User.
2. **Innerspec Product Management**:
   - Software
   - SoftwareModule
   - Product
3. **User Assets**:
   - System
   - SystemAccount
   - SystemStatus
   - Subscription
   - Payment
4. **Helpdesk**:
   - Issue
   - IssueComment
5. **Support**:
   - BaseEntity
   - Session
   - Document

The following diagrams show the relations between all the domain models organized by their categories.

High-Level Domain Diagram

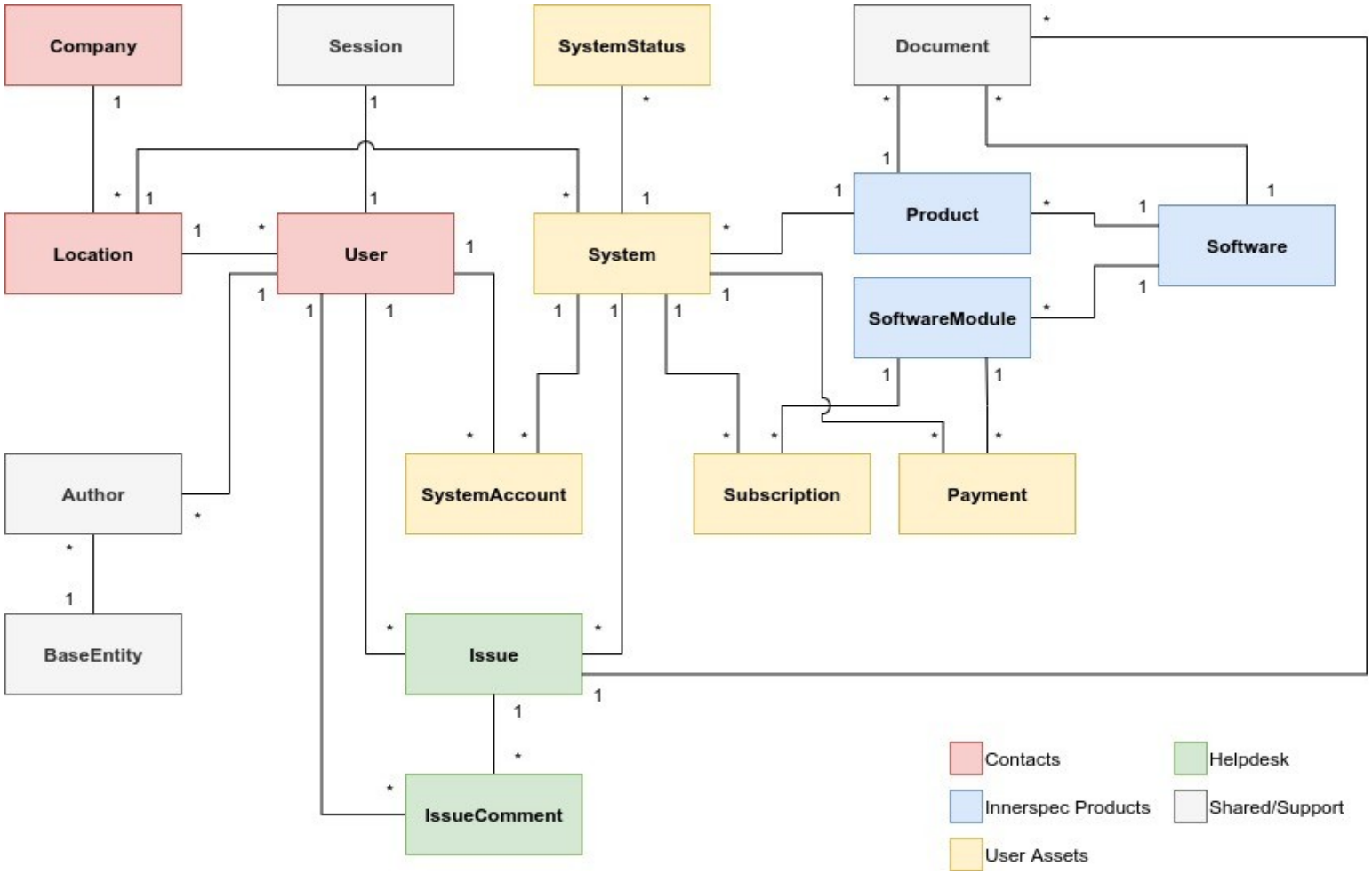


Figure 26: Application Domain UML diagrams – High Level

Contact Management

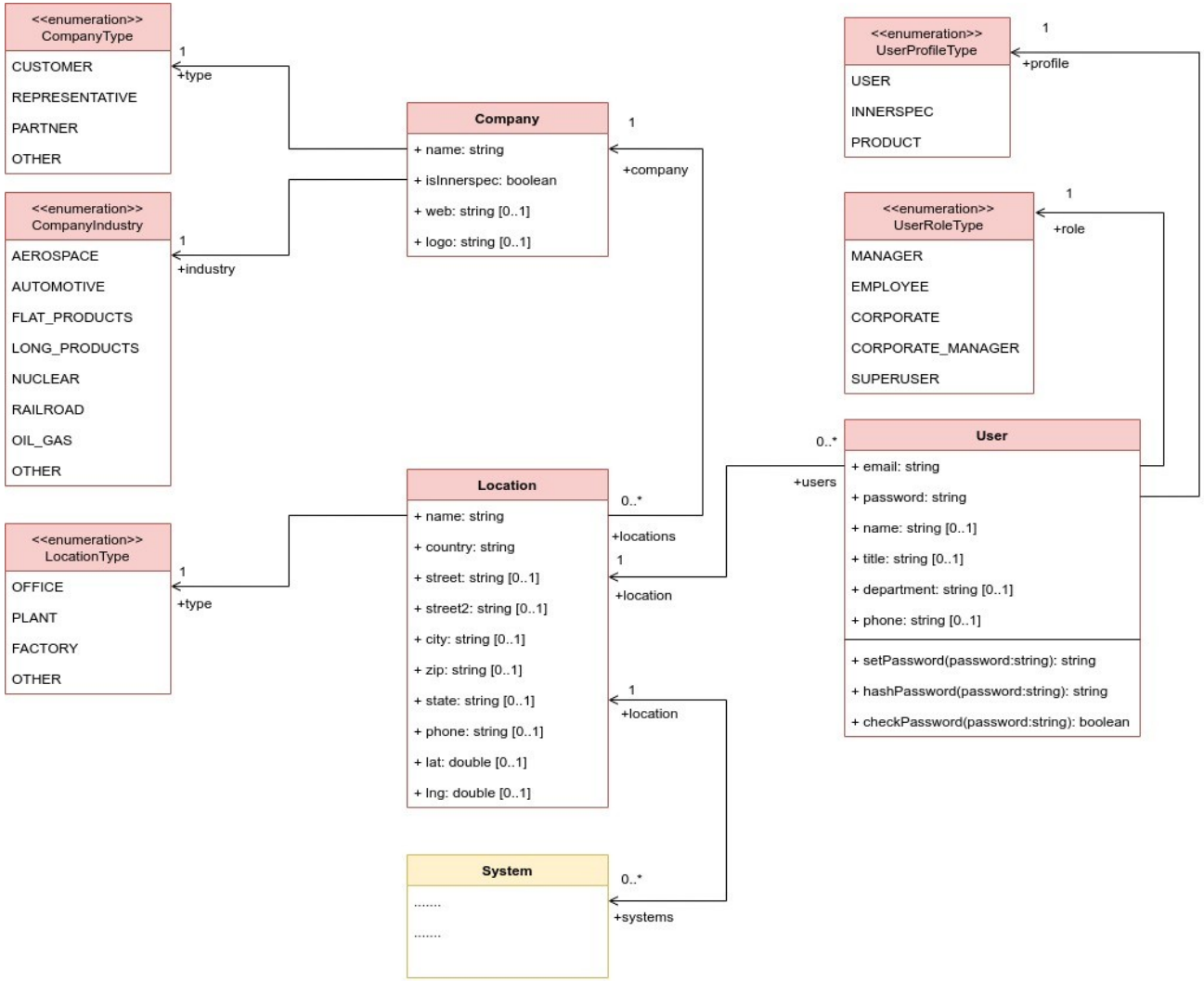


Figure 27: Application Domain UML diagrams - Contact Management

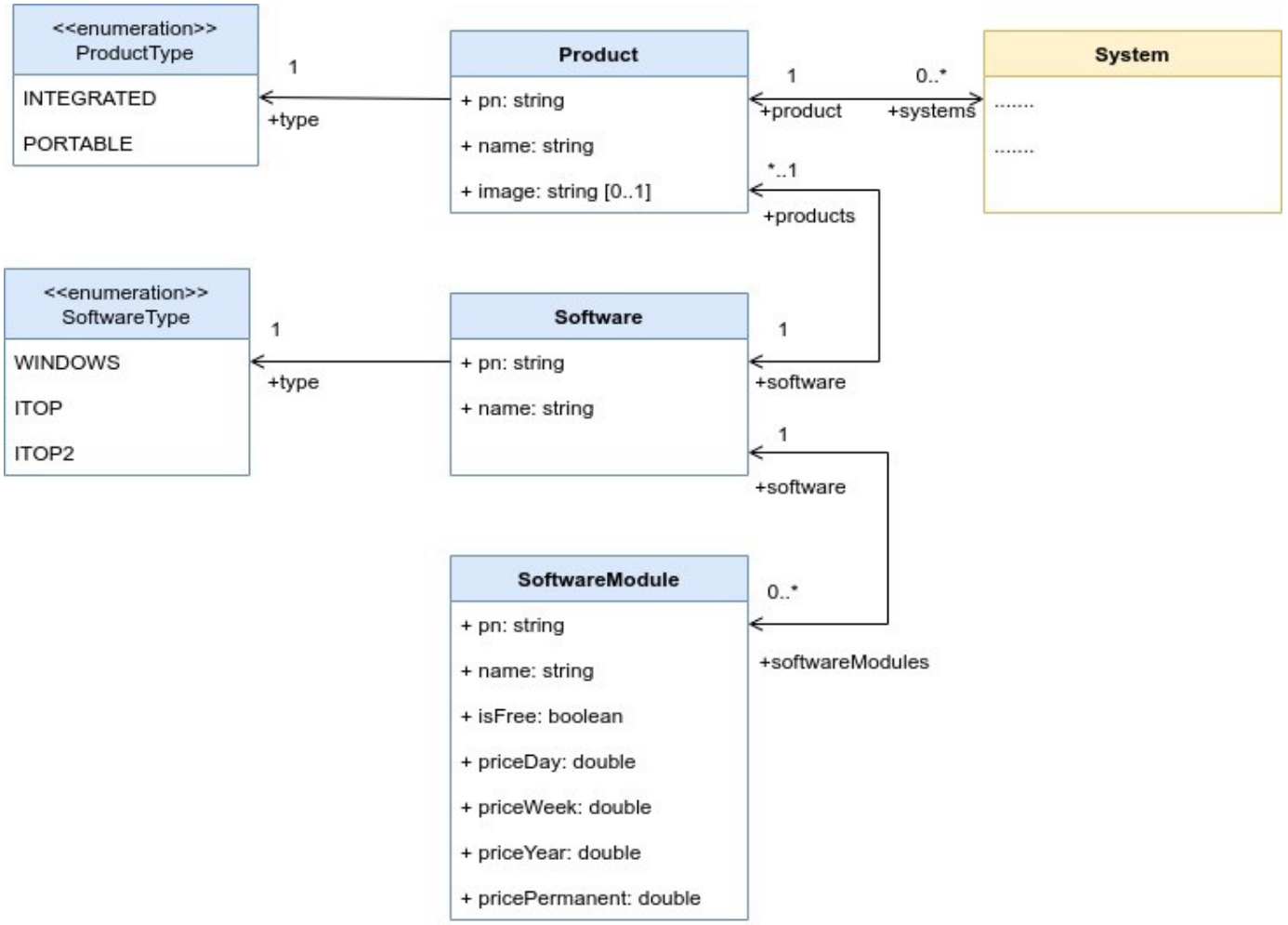


Figure 28: Application Domain UML diagrams - Innerspec Product Management

User Assets

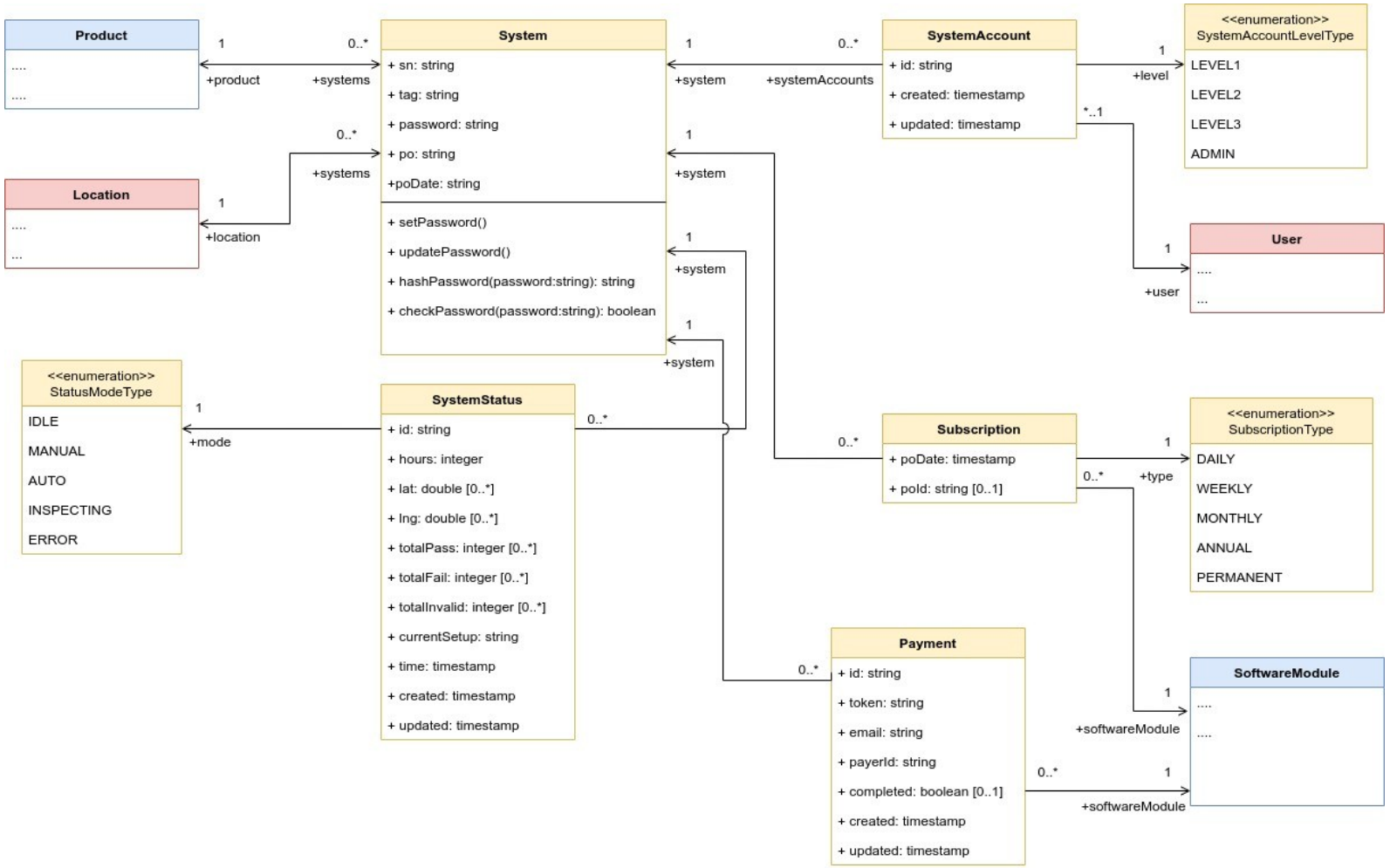


Figure 29: Application Domain UML diagrams - User Assets

Help Desk

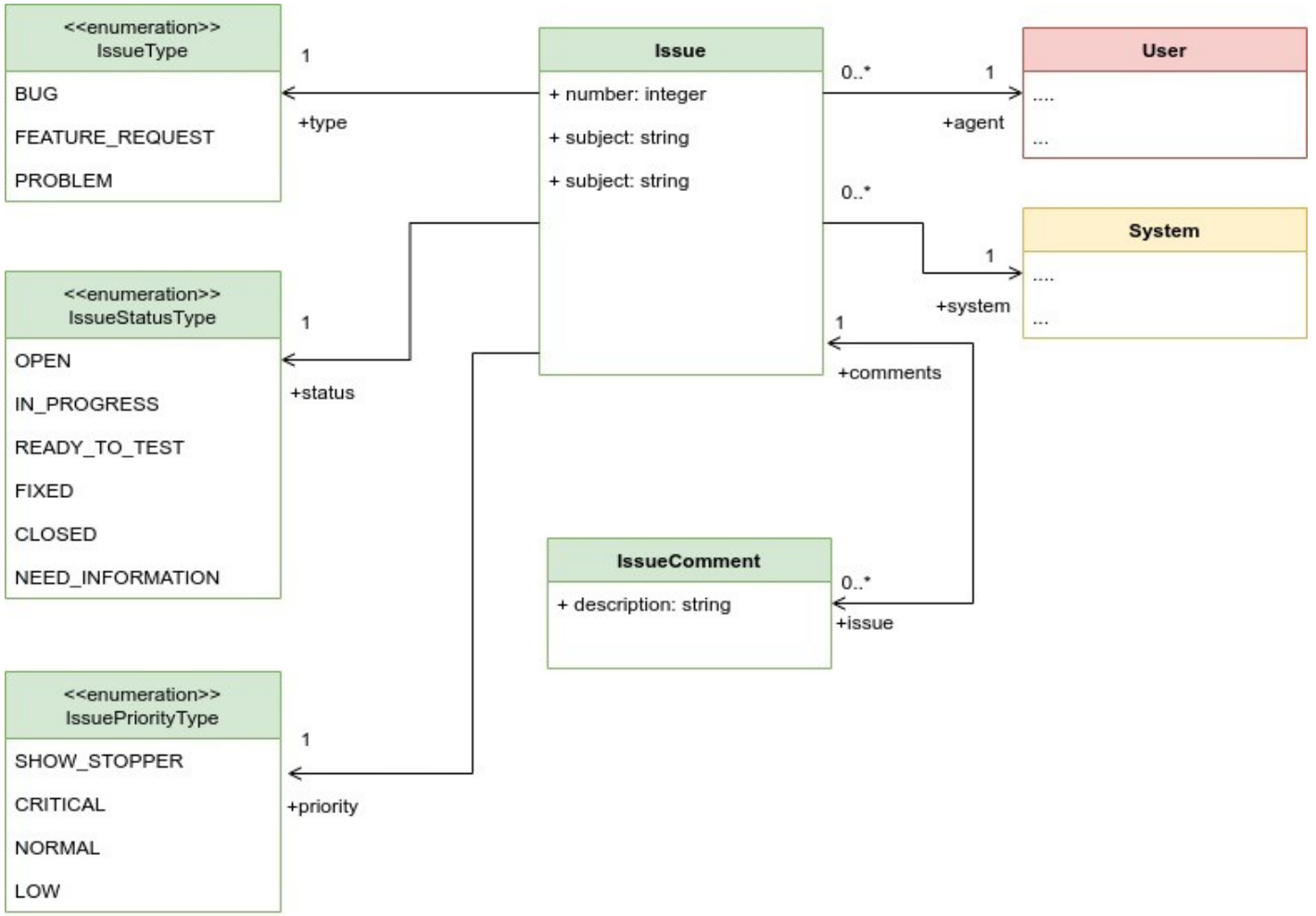


Figure 30: Application Domain UML diagrams – Help Desk

Support

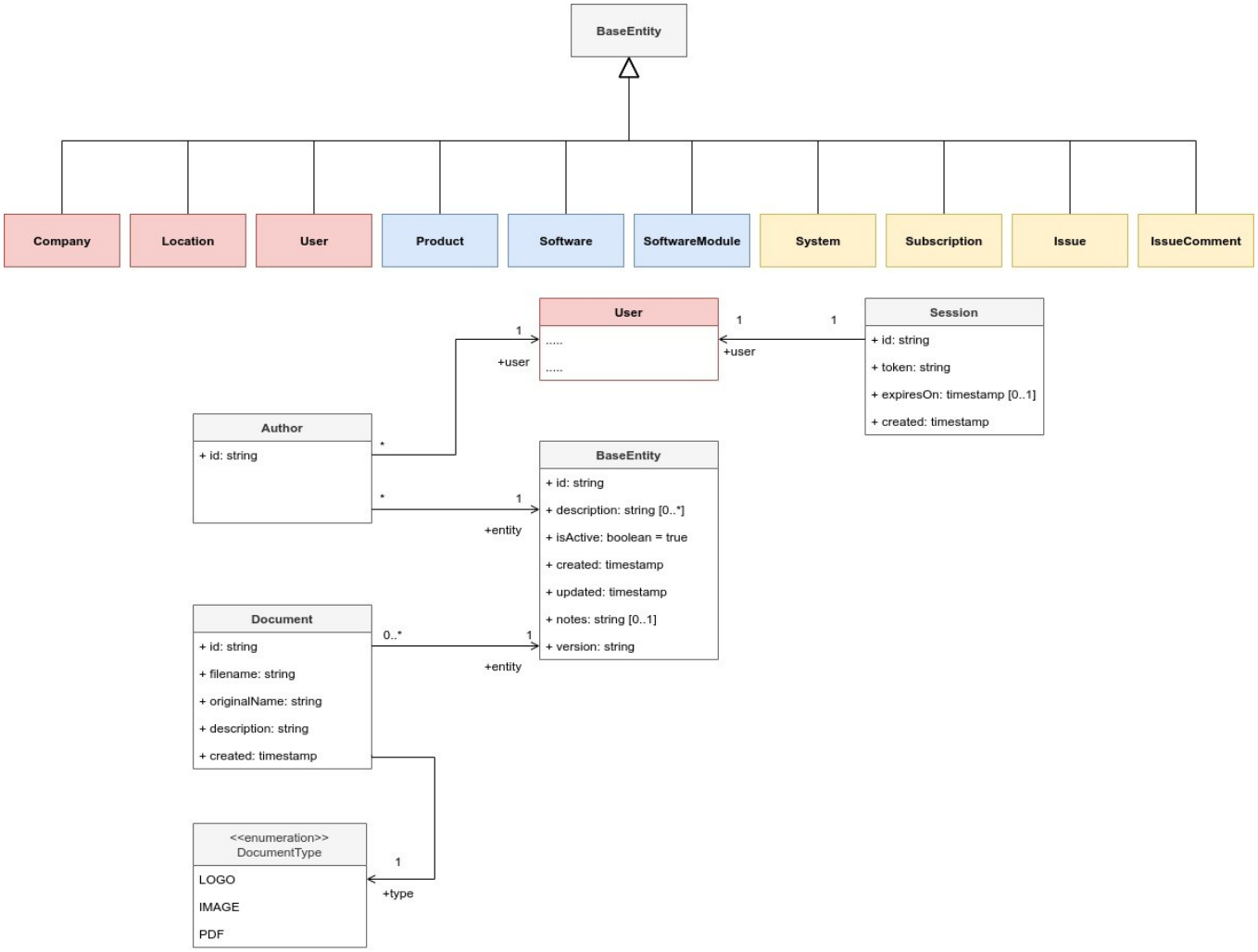


Figure 31: Application Domain UML diagrams - Support

11.2. Activities

There are many different operations users can perform on NDT-Link. This section will focus on describing the ones that are fundamental to providing connectivity between NDT-Link and systems:

1. **User Registration**:
    1. Create new NDT-Link accounts for users (User and Innerspec profiles).
2. **System Registration**:
    1. When a system is built, Innerspec's Production team creates a new record on NDT-Link for the system and customer's location.
3. **Reset System's Asset Password**:
    1. In order to communicate with NDT-Link, all Innerspec systems need to authenticate by providing their Asset Number and Asset Password. This operation allows changing the Asset Password for a system when needed.
4. **Add System Account**:
    1. Add an existing NDT-Link user to a system so they can authenticate on it with their NDT-Link credentials.
5. **Purchase Subscription**:
    1. Buy a subscription to a software module for a system.

The following figures show Activity Diagrams for the operations described above.

User Registration

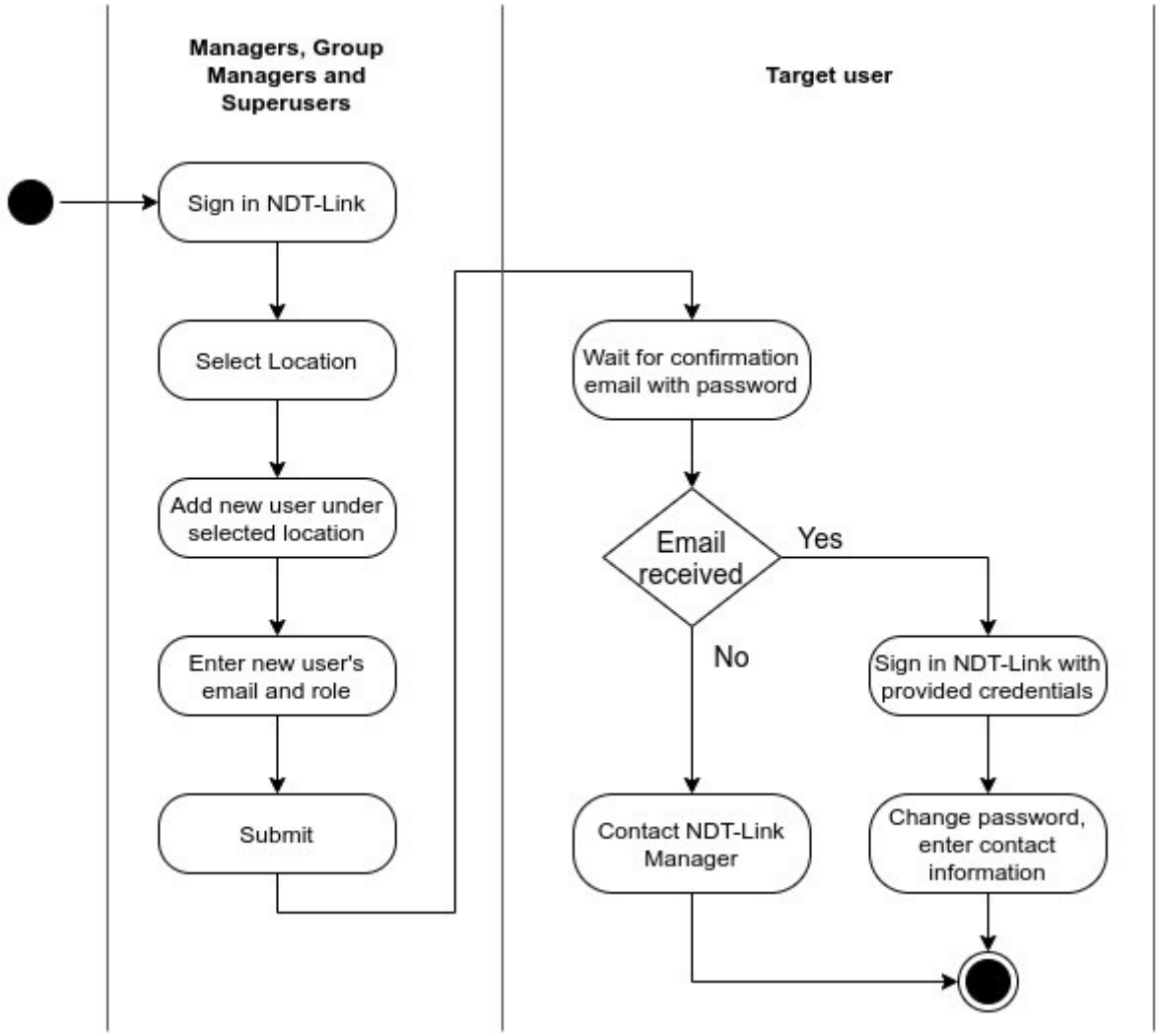


Figure 32: Activity UML Diagrams - User Registration

System Registration

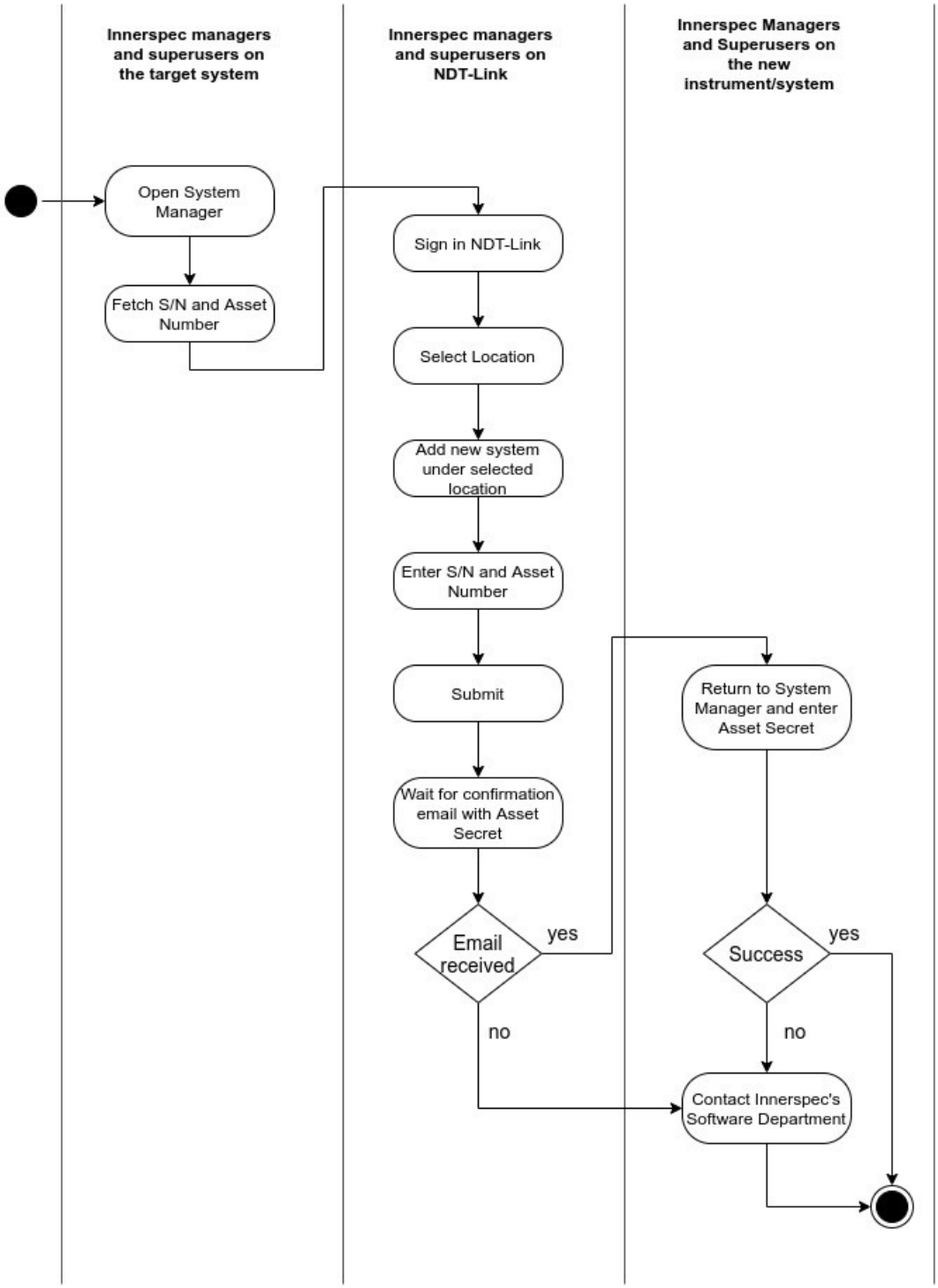


Figure 33: Activity UML Diagrams - System Registration

Reset System's Asset Secret

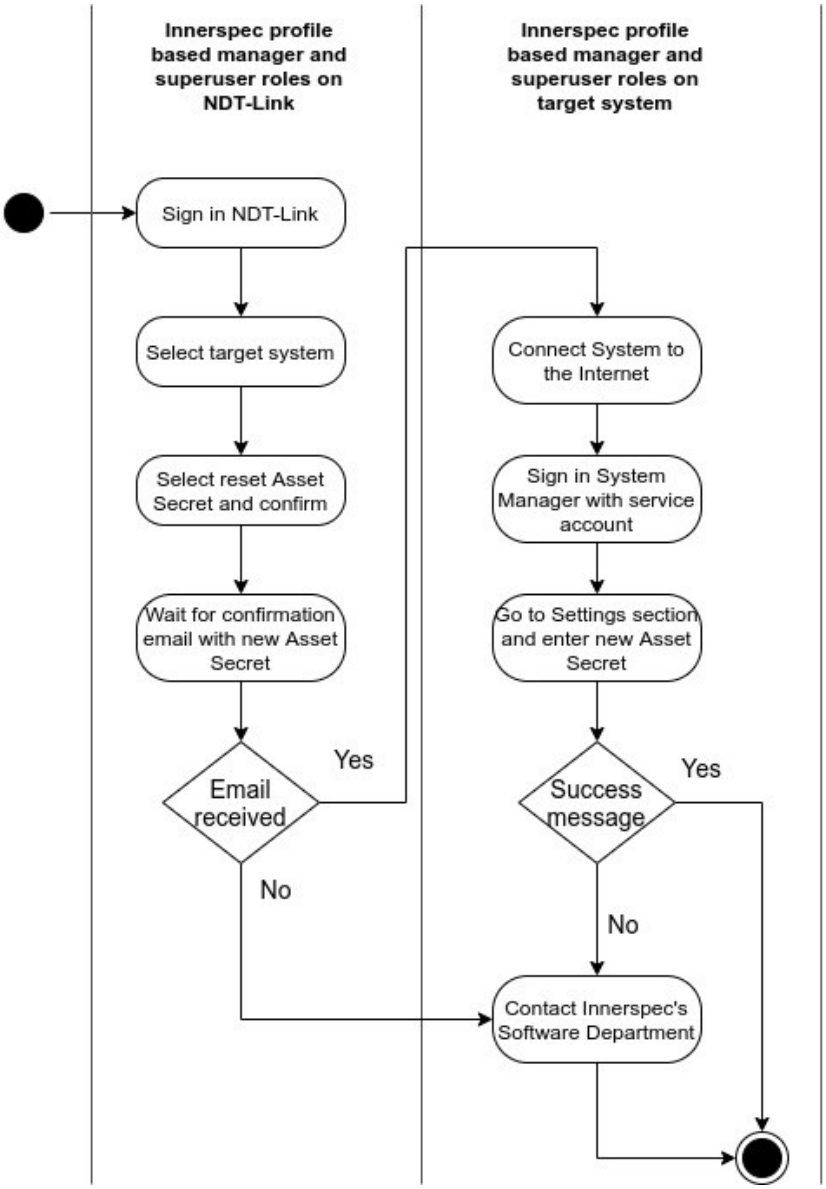


Figure 34: Activity UML Diagrams - Reset System's Asset Secret

Add System Account

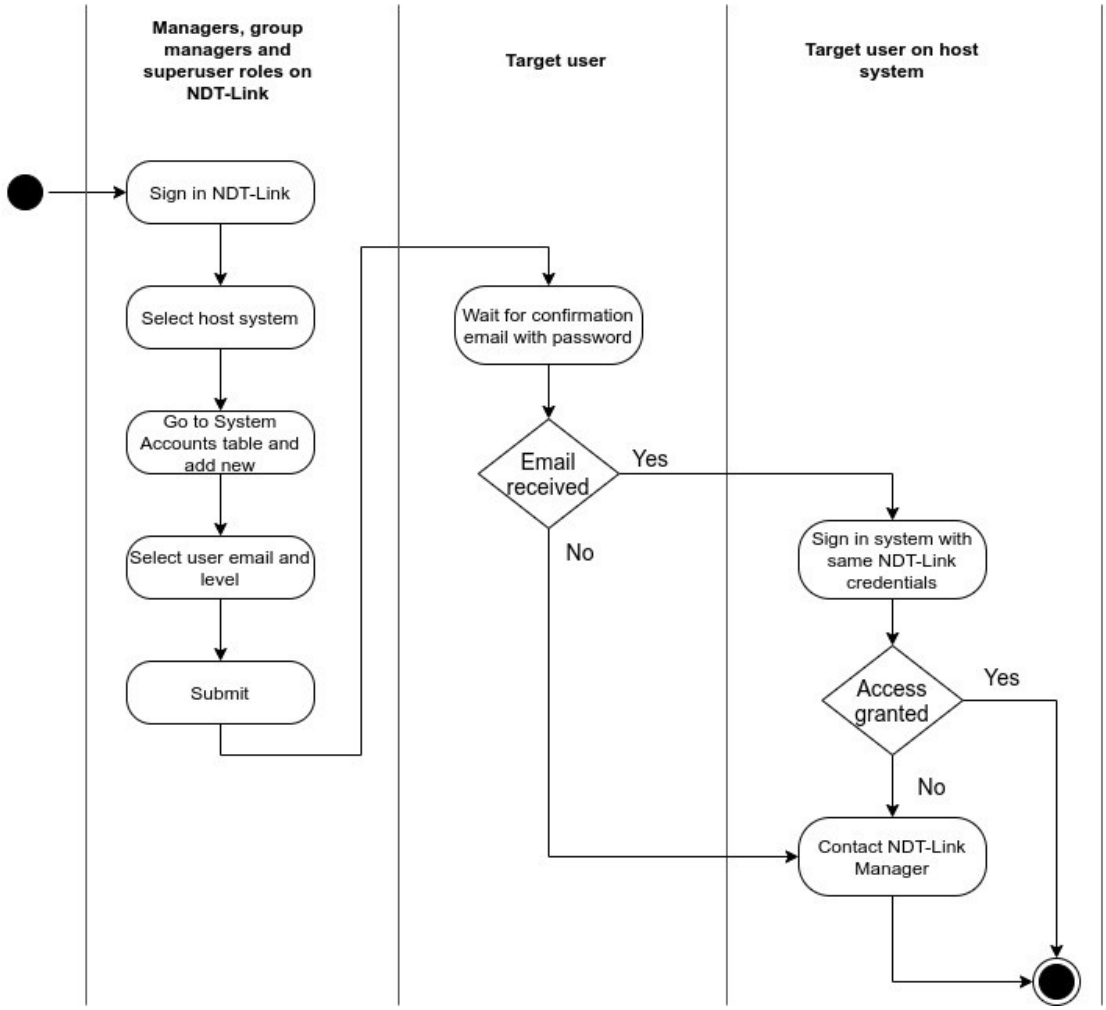


Figure 35: Activity UML Diagrams - Add System Account

Purchase Subscription

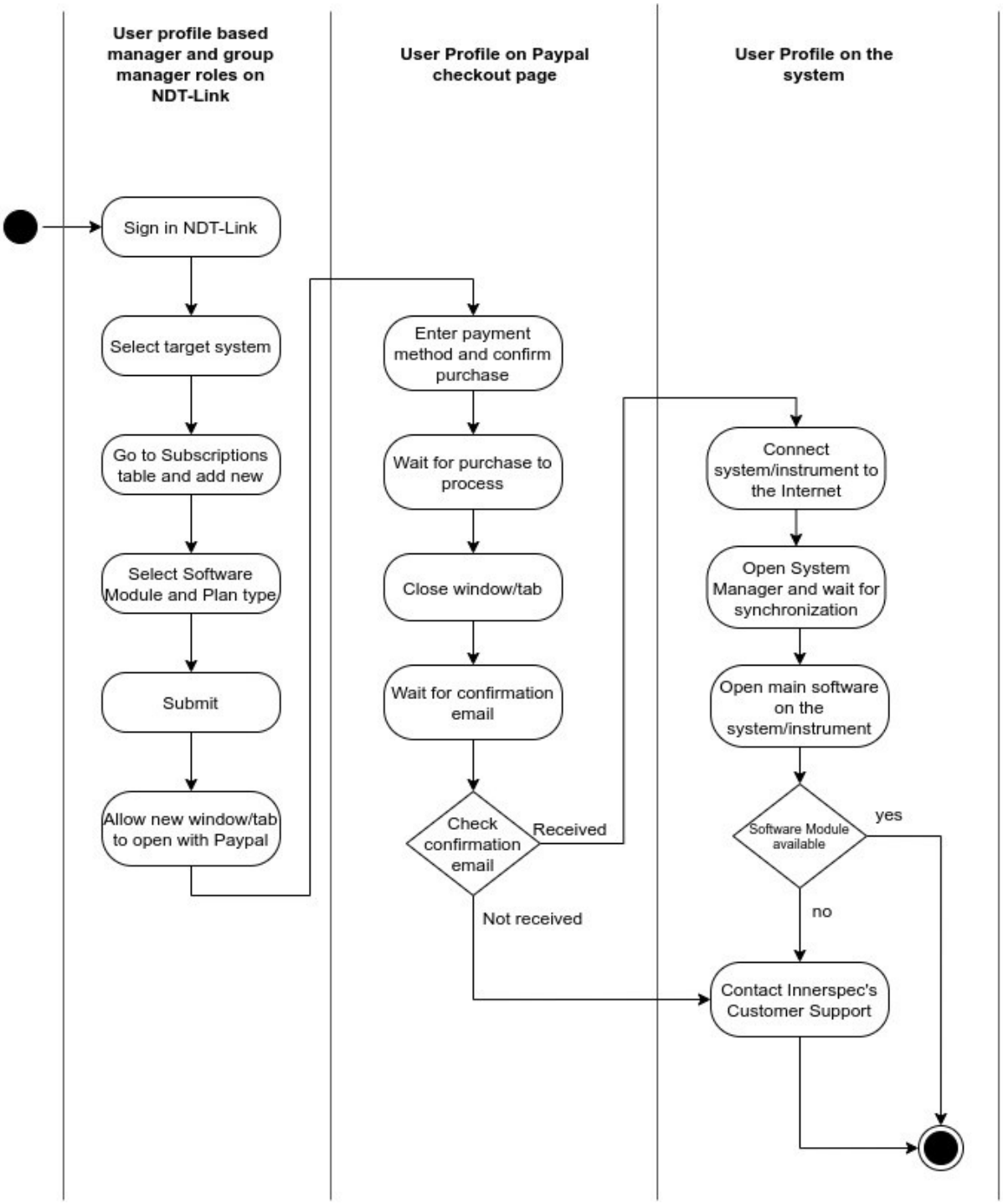


Figure 36: Activity UML Diagrams - Purchase Subscription

11.3. Sequence Diagrams

There are many operations and interactions occurring between the different blocks that compose NDT-Link. The most complex one is the purchase of new subscriptions, because it involves several components and services from both the front-end and back-end, as well as a third-party library, the PayPal payment gateway and multiple asynchronous HTTP requests. Due to its complexity, this section will focus on describing this process only.

Summary

The purchase of a new subscription for a system consists of three stages:

1. **Order Confirmation**:
    1. The back-end creates a new *PayPal* order object with the subscription options selected by the user on the front-end. This order also includes a return URL where the user should be redirected from *PayPal* after processing the payment. The return URL points to a component on the front-end (New Purchase Component) that will show a success message after the purchase is finalized.
    2. The back-end sends the order to *PayPal's* gateway through a third party library.
    3. *PayPal* verifies the order request and responds with a unique token assigned to the transaction, and a URL to the checkout page where the user reviews the order, enters the selected payment method, and accepts to be charged.
    4. The back-end records in the database a new payment with the transaction token received from *PayPal's* gateway, as well as the system and software module the purchase is being made for.
    5. The back-end responds to the front-end with the return URL.
2. **Payment Confirmation**:
    1. The front-end informs the user that he will be redirected to *PayPal* to finalize the purchase.
    2. The front-end opens a new window that navigates to the return URL, which is *PayPal's* checkout page.
    3. The user selects payment method, enters its details (i.e., credit card number) and accepts the purchase.
    4. *PayPal* confirms the entered payment information is correct and redirects the user to the return URL with the transaction token as a query parameter.
3. **Payment Execution**:
    1. The New Purchase component on the front-end extracts the transaction token from the browser route and informs the user to wait while the order is processing. The front-end sends the token to the back-end for execution.
    2. The back-end sends an execution request to *PayPal's* gateway for the provided transaction token.
    3. *PayPal* executes the order (charges the user) and responds with a summary of the transaction, which includes the transaction token and its status.

4. The back-end checks that transaction status is *COMPLETED* and updates the payment record in the database accordingly. It's at this point when the back-end creates and assigns the purchased subscription to the system and sends a confirmation email to the user.
5. The back-end responds to the front-end with a success response, and the New Purchase component on the front-end shows a message informing that the order has been processed. The user now can close the browser window with the New Purchase component.

The size of the full sequence diagram showing the stages described above is too big to fit in this document. Instead, the three stages have been extracted in separate figures presented in the following sections. The full sequence diagram is provided with this document as ANNEX 9.

Order Confirmation

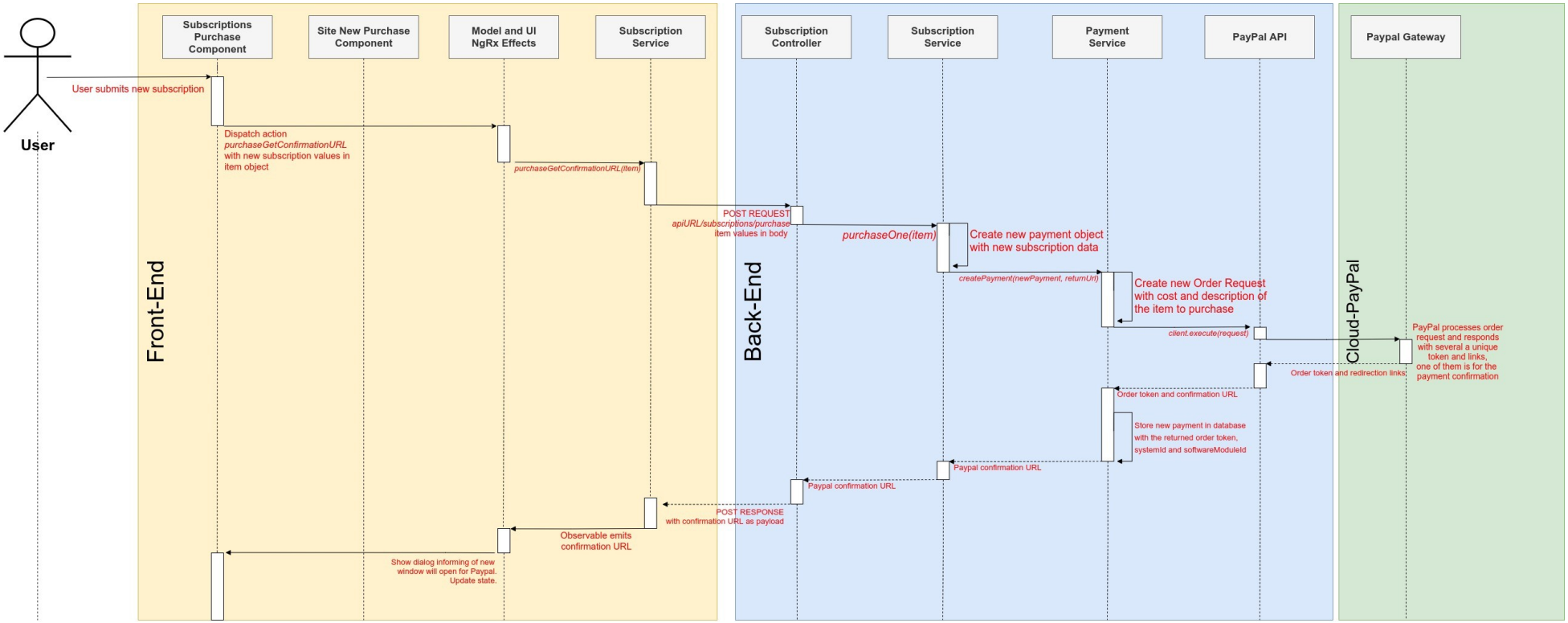


Figure 37: UML Sequence Diagrams - Subscription Order Confirmation

Payment Confirmation

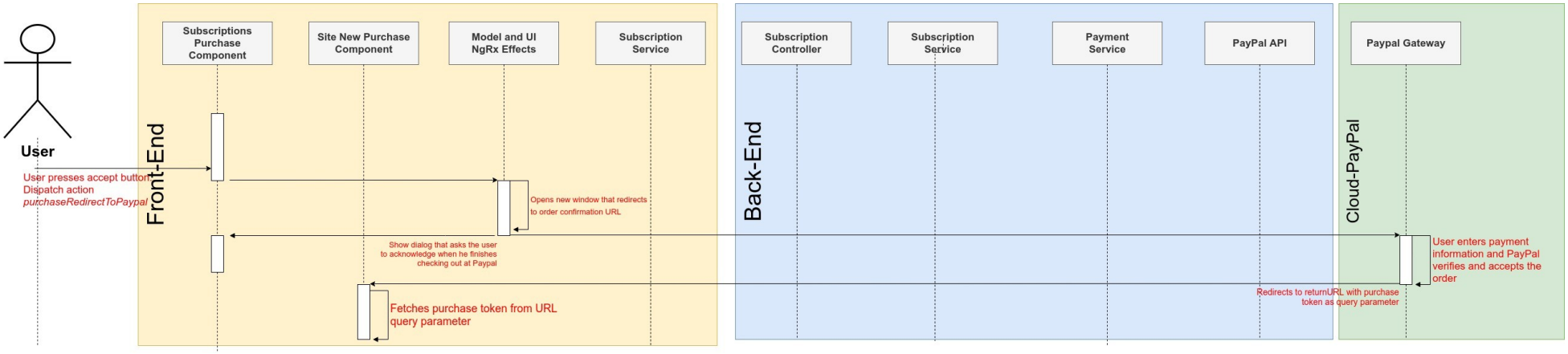


Figure 38: UML Sequence Diagrams - Subscription Payment Confirmation

Payment Execution

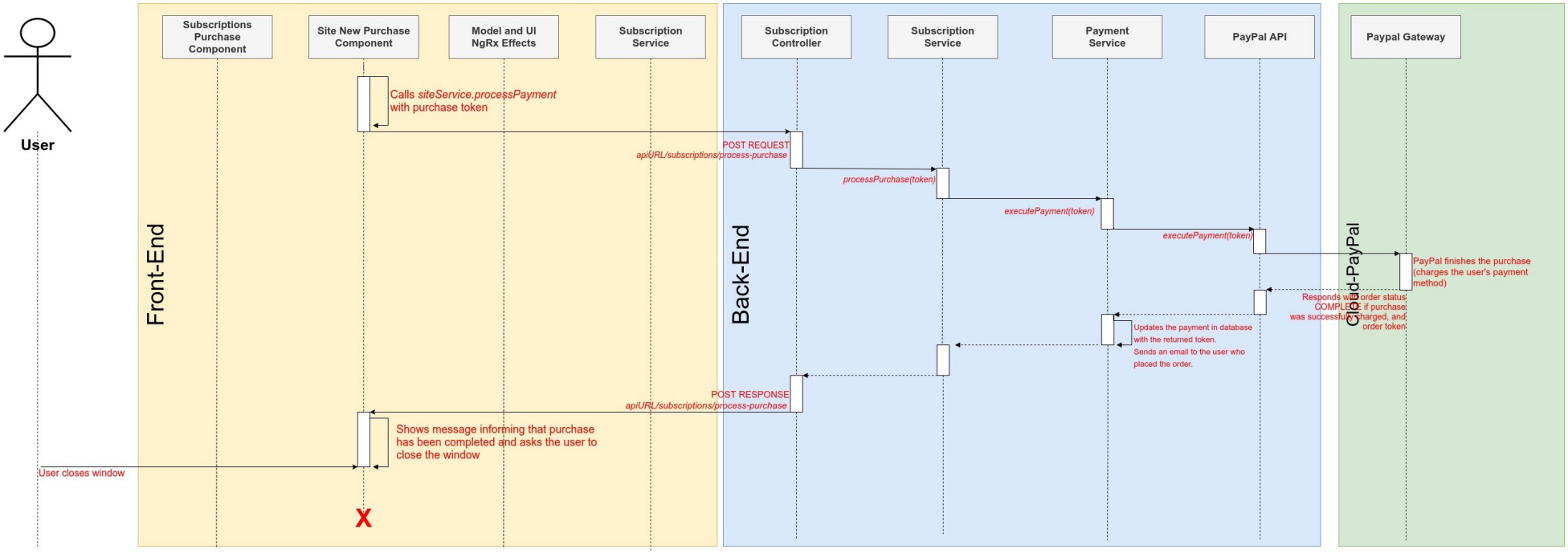


Figure 39: UML Sequence Diagrams - Subscription Payment Execution

12. Security

There are several layers of security measures implemented in NDT-Link:

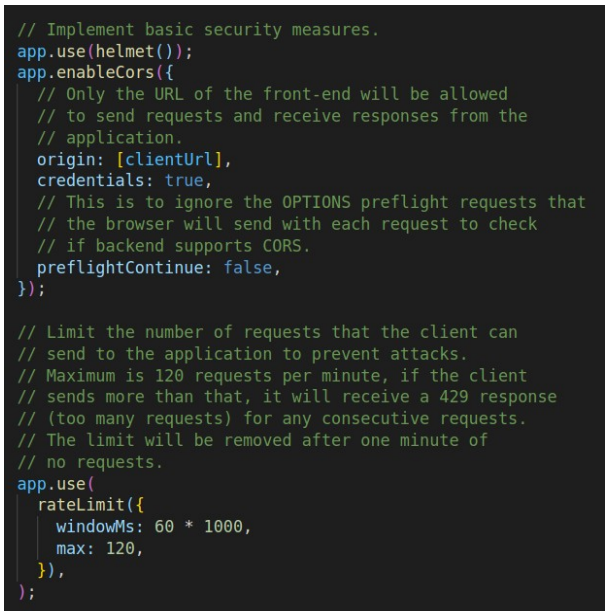
1. Back-End:
    1. CORS, rate-limiting and HTTP header optimizations.
    2. User authentication with *JSON Web Tokens*.
    3. User authorization to endpoints restricted with *NestJS* guards.
2. Front-End:
    1. User authentication with Angular HTTP interceptor and *JSON Web tokens* in HTTP header for all requests.
    2. User Authorization with Angular route Guards.
3. Hosting:
    1. HTTPS with certificate issued by *Amazon Web Services*.

12.1. Back-End

In order to allow requests from the front-end running on a different host machine, the back-end uses the *CORS* middleware from *Express* that enables and configures *Cross-Origin Resource Sharing*.

Another package used by the back-end is *express-rate-limit,* used to block repeated requests sent to the exposed endpoints when they exceed a certain configured limit. When this happens, the back-end will respond to the request with HTTP response code 429 (too many requests) and the client's IP will remain blocked for a certain amount of time.

The back-end also uses the package *Helmet* that injects more than a dozen of small middleware functions into the *NestJS* application to set security-related HTTP response headers to help prevent cross-site scripting attacks and other cross-site injections.



```
// Implement basic security measures.
app.use(helmet());
app.enableCors({
  // Only the URL of the front-end will be allowed
  // to send requests and receive responses from the
  // application.
  origin: [clientUrl],
  credentials: true,
  // This is to ignore the OPTIONS preflight requests that
  // the browser will send with each request to check
  // if backend supports CORS.
  preflightContinue: false,
});

// Limit the number of requests that the client can
// send to the application to prevent attacks.
// Maximum is 120 requests per minute, if the client
// sends more than that, it will receive a 429 response
// (too many requests) for any consecutive requests.
// The limit will be removed after one minute of
// no requests.
app.use(
  rateLimit({
    windowMs: 60 * 1000,
    max: 120,
  }),
);
```

Figure 40: Security - CORS, Helmet and Rate-Limit in NestJS main.ts

User Authentication and Authorization

The back-end module *AuthModule* handles all operations related to authentication and authorization. It uses the packages *Passport.js* and *@nestj/jwt* to implement two authentication strategies, one to validate credentials using the user's email and password stored locally in the database server, and another to generate and sign new *JWT* tokens as a response to a log in request, or to validate the ones received in the *Authorization* header of the HTTP requests when the client is trying to access one of the exposed endpoints.

The *AuthModule* provides service *AuthService* with three methods used to log in and log out users and systems which. The method *loginSystem* is used by the systems deployed in the field.

For Authorization, two *NestJS* guards have been implemented:

1. *AuthJwtGuard*:
    1. Extracts and validates the JWT token from the *Authorization* header of the requests received from the clients. If the validation fails, the back-end will respond with HTTP error code 401 (Unauthorized) if the token is missing (user or system not logged in), or 403 (Forbidden) if the the token is expired or malformed.
    2. It also extracts from the token the id, profile and role of the user or system sending the request, and injects them into the HTTP request context so they can be accessed by other services and modules of the application.
2. *AuthGuard*:
    1. It will block the controller route in which is used if the user extracted from the request JWT token doesn't have the required profile and role. In this case the back-end will respond with HTTP error 401 (Unauthorized).

2. In order to reuse this guard with different combinations or profiles and roles, custom *NestJS* decorators *@AuthProfiles* and @AuthRoles have been implemented to pass multiple profiles and roles as parameters to the guard.

The following figure shows part of the *SubscriptionsController* in which a route to create subscriptions bypassing the *PayPal* purchase process is limited only to Innerspec users with roles manager and superuser, while the one that does require a *PayPal* purchase is limited to users with roles manager and corporate manager.

```
@Post()
@AuthProfiles(UserProfileType.INNERSPEC)
@AuthRoles(UserRoleType.MANAGER, UserRoleType.SUPERUSER)
@UseGuards(AuthGuard)
create(@Body() item: SubscriptionCreateDto) {
  return this.service.addOne(item);
}

@Post('purchase')
@AuthProfiles(UserProfileType.USER)
@AuthRoles(UserRoleType.MANAGER, UserRoleType.CORPORATE_MANAGER)
@UseGuards(AuthGuard)
purchase(@Body() item: SubscriptionCreateDto) {
  return this.service.purchaseOne(item);
}
```

Figure 41: Security - Authorization guards and decorators

12.2. Front-End

All the authentication and authorization operations on the front-end are handled by the *AuthModule* which provides the following:

1. *Angular service AuthService.*
2. *NgRx actions and state.*
3. *Angular HTTP interceptor AuthTokenInterceptor*
4. *Angular Guards AuthPublic*, *AuthPrivate* and *AuthHasCredentials*.

AuthService provides methods *logIn* and *logOut* that are called by *NgRx* actions sent to the store from the *UsersModule*. The *logIn* method sends the provided username and password to the back-end authentication endpoint and receives the JWT token in the response's payload if the authentication succeeds, then it stores the token in browser's *LocalStorage*. The *AuthService* also provides method *loadUserData* to read the token stored in *LocalStorage.*

AuthTokenInterceptor fetches the JWT token from *AuthModule* store if a user is authenticated and injects it as a *Bearer* token into the *Authorization* HTTP header of all requests sent to the back-end.

AuthPublicGuard blocks the navigation to certain routes if the user is authenticated (i.e., users already logged in shouldn't be able to navigate to the *UsersLoginComponent* component that shows the log in

form). *AuthPrivateGuard* blocks the navigation to certain routes if the user is not authenticated, and instead, it redirects the user to the *UsersLoginComponent*. Lastly, the Guard *AuthHasCredentials* prevents navigation to a route if the logged in user doesn't have the required profile and role.

The guards are used from the routing modules of each domain module. The following figure shows and example of this use from the *UsersRoutingModule*.

```
const routes: Routes = [
  // Public paths, only available if the user is
  // not logged in.
  {
    path: 'login',
    component: UsersLoginPage,
    canActivate: [AuthPublic],
  },
  {
    path: 'reset-password',
    component: UsersResetPasswordPage,
    canActivate: [AuthPublic],
  },
  // Private paths.
  {
    path: '',
    canActivate: [AuthPrivate],
    children: [
      {
        path: ':id',
        component: UsersViewPage,
        resolve: { location: UserResolver },
      },
      { path: '', component: UsersIndexPage, pathMatch: 'full' },
    ],
  },
];
```

Figure 42: Security - Angular route definition with guards for the Users module

12.3. Hosting

The HTTPS protocol has been enabled on both the front-end and back-end. For the front-end to provide encrypted communications between the user's browser and *Amazon S3* service where the *Angular* application is hosted. For the back-end to encrypt the data exchanged between the *Angular* application running on the user's browser and the *NestJS* back-end application running on an *Amazon* EC2 instance managed through *Amazon Beanstalk* service.

The HTTPS restriction for the *Angular* application hosted in the *Amazon S3* service is provided through a distribution of *Amazon CloudFront* launched in the regions of Canada, USA and Europe that redirects all HTTP connections to HTTPS.

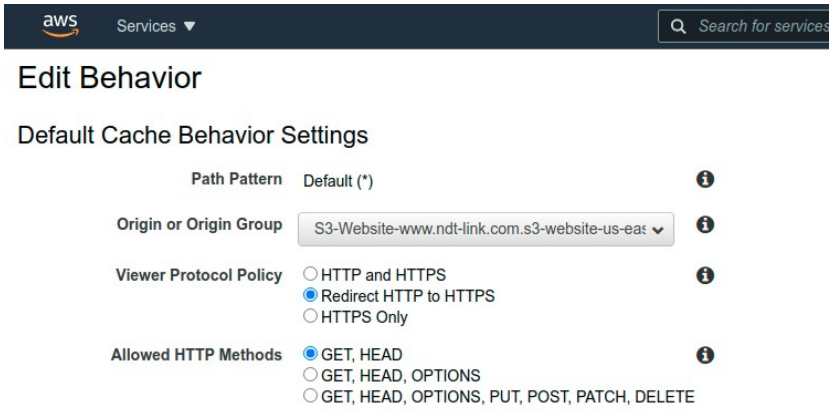


Figure 43: Security - HTTPS configuration for front-end on Amazon CloudFront

The HTTPS restriction for the back-end *NestJS* application hosted on the *Amazon EC2* instance is configured through a *Load Balancer* that only accepts connections to the 443 port.

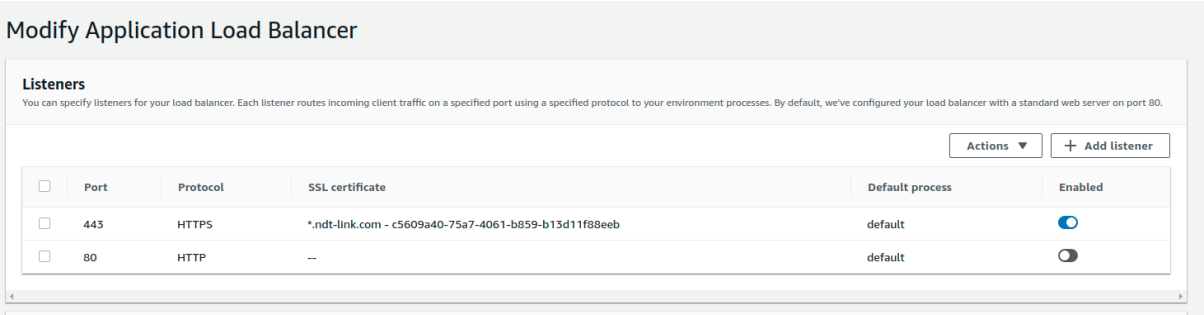


Figure 44: Security - HTTPS configuration for back-end on Amazon EC2 instance with Load Balancer

Also, the communications between the *Amazon EC2* instance and the *MySQL* server hosted on an *Amazon RDS* instance are encrypted too. All external ports on the *EC2* instance have been closed with the exception of port 443 (HTTPS), while on the *RDS* instance, only communications within the same security group where the two instances belong are allowed (direct external communications to the RDS instance and blocked).

13. Installation/Deployment Instructions

13.1. Installation for Development

In order to launch the front-end and back-end applications for development, the following requirements must be met on the development computer:

1. *Node.js* version 14 or above.
2. Angular version 11 or above.
3. *MySQL* server version 8 or above.
4. An empty database named *ndtlink_nest*, with access granted to user *root* with password *temate*. If different values are used, they must be set in their correspondent environmental variables defined in the back-end file *config.ts*.

Steps to follow to download and start the front-end and back-end applications for development:

1. **Front-end**:
    1. Clone repository:
        - *git clone* https://github.com/InnerspecTechnologies/ndtlink-client
    2. Enter the project directory:
        - *cd ndtlink-client*
    3. Install dependencies:
        - *npm install*
    4. Modify the files *environment.ts* as needed.
    5. Start Angular project:
        - *ng serve*
2. **Back-end**:
    1. Clone repository:
        - *git clone* https://github.com/InnerspecTechnologies/ndtlink-backend
    2. Enter the project directory:
        - *cd ndtlink-backend*
    3. Install dependencies:
        - *npm install*
    4. Set environmental variables as needed to override default configuration set in *config.ts*.
    5. Start *NestJS*:
        - *npm run start:dev*

13.2. Build and Deploy

General Arrangement

The *Angular* front-end application is hosted on the *Amazon S3* service, and the back-end *NestJS* application is hosted on an *Amazon EC2* instance which communicates with a *MySQL* server hosted by the *Amazon RDS* service. Both the EC2 and MySQL instances are managed from the *Amazon Beanstalk* service.

The front-end application is accessed at https://www.ndt-link.com through a CDN provided by the *Amazon CloudFront* service launched in the USA, Canada and Europe regions. The *Amazon EC2* instance hosting the back-end application is part of a secure *Amazon Virtual Private Cloud* that restricts direct external access to the database. The back-end application responds to requests sent to https://api.ndt-link.com. These two URL addresses are managed are resolved by *Amazon Route 53 DNS* service.

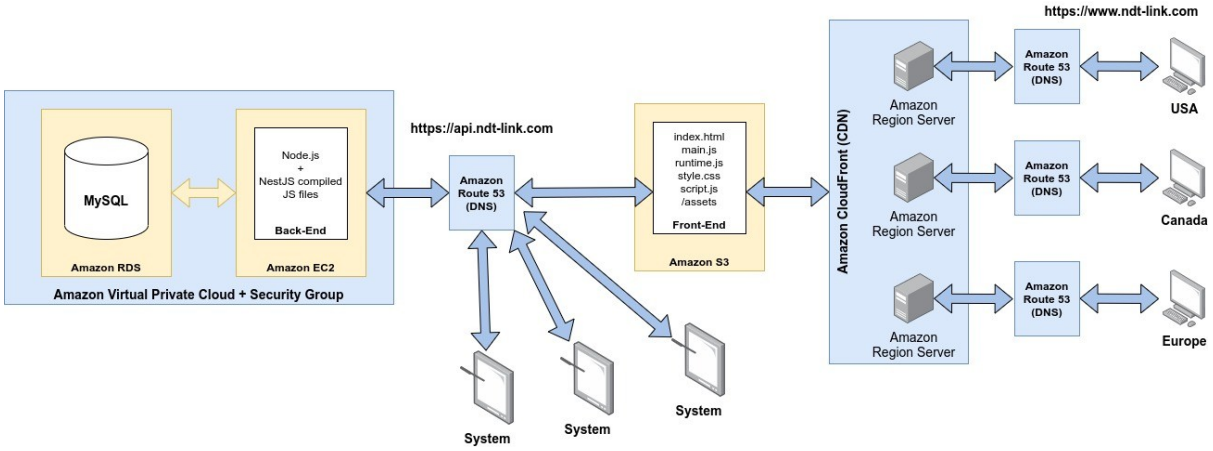


Figure 45: Front-end and back-end deployment on Amazon Web Services

For security reason, specific details on how these services have been configured will not be described in this document since they can be shared only with Innerspec personnel.

Continuous Delivery with Github Actions

The *Github* repositories for the front-end and back-end applications contain a *master* and a *release* branch. Commits to the *release* branch will trigger a *Github* action that builds the source code and publishes the result to *Amazon S3* and Amazon *Beanstalk* services respectively.

These *Github* actions are defined in the file *.github/workflows/build_and_deploy.yml* present in each repository.

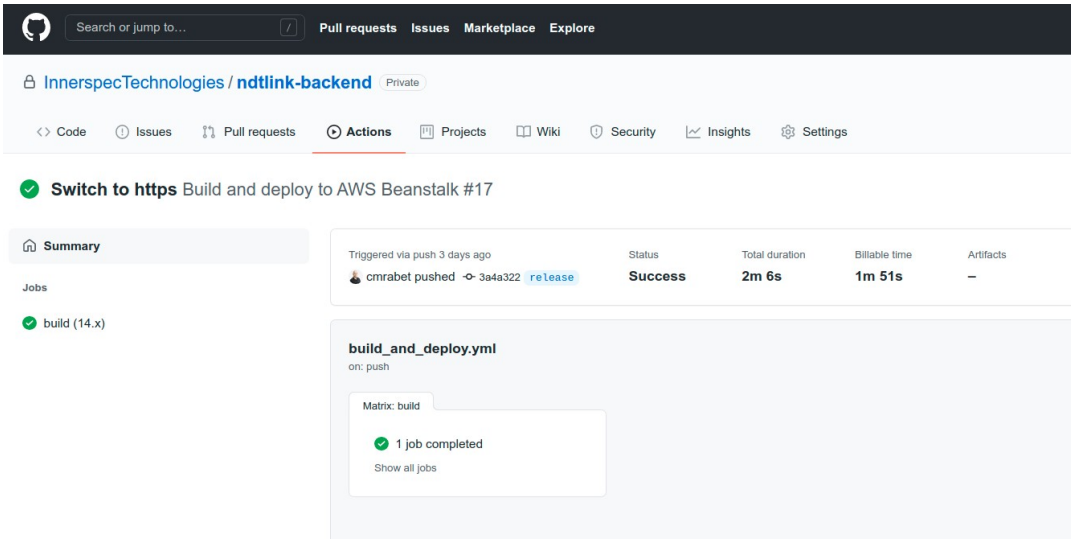


Figure 46: Github Action that completed building and deploying the back-end after new commit

13.3. Accounts for evaluation tests

The following accounts are available for testing NDT-Link for the purpose of evaluating what has been implemented for this thesis:

1. Superuser account:
    1. Username: admin@innerspec.com
    2. Password: Temate#1
2. Innerspec profile-based account:
    1. Username: cmrabet@innerspec.com
    2. Password: Temate#1
3. User profile-based account:
    1. Username: cmrabet@gmail.com
    2. Password: Temate#1

14. Future Improvements/Roadmap

There are several objectives from the planning stage that have not been implemented due to lack of time. They will be implemented before releasing NDT-Link to customers:

1. Primary Objectives:
    1. Finish applying role-based guards for all routes and operations on the front-end and back-end. At the time of the writing of this document, guards are only applied to enforce profile based restrictions but not role-based ones (this means that any user with the profile Innerspec or User can perform any of the available operations for his profile with no restrictions).
    2. Pagination for the list views and tables.
    3. Add recurrent CRON job in the back-end that deactivates the expired subscription.
    4. User profile page.
    5. Unit Tests and E2E tests.
2. Secondary Objectives:
    1. Reception, organization, and display of inspection data sent from the systems.
    2. Messaging between users.

Once the pending objectives are implemented, and NDT-Link has been in use by customers for a certain amount of time, the following features will be implement:

1. Database migrations, as a replacement to the *seeds.js* script that runs every time the back-end starts.
2. Expand data sharing between products and NDT-Link by synchronizing system setups and configurations.
3. Integrate tests with Git Actions, and expand E2E tests.

15. Conclusions

15.1. General

The final effort required to implement NDT-Link was more than originally planned during the design stage. There have been several deviations from the timing diagram presented in section 8.2. as well as delays. One of the tasks that impacted negatively the progress of the project has been the study of the Information Architecture, which has required a considerable amount of time, even though the participation was low with minimal results.

The implementation of the Product Connector has forced to work on tasks that were planned for much later, changing the plan considerably for some milestones. This happened because the team responsible for the development of the Product Connector depended on the availability of the endpoints from NDT-Link in order to continue making progress. Other tasks had to put on hold in order to implement all the required endpoints.

15.2. Front-End

Previous knowledge of the Redux pattern helped to quickly understand and integrate NgRx. This library allowed the implementation of a cleaner and more organized project structure than one using just services and Observables. Using this library was fun and pleasant, even though the amount of boilerplate generated was a concern in the beginning of the implementation stage.

The implementation of the UiModule took an important amount of time from the development of the actual functionality needed by the application. In order to keep the application as much independent as possible from any dependencies, the implementation of basic repetitive controls such as input boxes and modal dialogs was necessary. The resultant decoupling is acceptable, however for faster development and focus on the business of the application, a better approach like Angular UI Framework would have been a better approach.

The most complicated task has been creating a CRUD structure that implements the list view, detail view and creation and edition forms for each model. The complexity resided mostly in managing the forms to work correctly with the state.

In general, developing with Angular has been fun and pleasant. Using it for a project as big as NDT-Link has showed where this framework shines; organization, structure, robustness and reduction of third party dependencies.

15.3. Back-End

Learning NestJS took a considerable amount of time, more than originally estimated, even though node.js and Express.js were very familiar. After overcoming the learning curve, the development with NestJS has been fast and enjoyable.

Evaluating different options for database interaction took also an important amount of time that was not considered in the initial planning. After trying different ORMs, query-builders and drivers for direct access to MySQL, it was decided to use TypeORM. This ORM has many problems, and getting help from the maintainers was slow. This is the reason the most complex queries were implemented with raw SQL, and just the simple ones were done with TypeORM methods.

The most complicated part to implement in the back-end has been the processing of payments with PayPal. The developer documentation available from PayPal is very confusing and poorly organized. Most of the time dedicated to this task was spent in reading documentation and trying to understand the API and how the different stages of the payment process work. Once this was clear, the implementation was very straightforward.

Finally, even though working with NestJS has been very enjoyable, the options available for interacting with relational databases in Node.js are poor. This strongly forces to reconsider keeping Node.js as the back-end platform for next versions of NDT-Link or instead, replace it with other technologies such as .NET with EntityFramework.

15.4. Deployment

The implementation of Git actions has been very straightforward. There is plenty of information on GitHub.com, as well as hundreds of third-party actions that can be incorporated to perform all kind of tasks for CI and CD.

Configuring Amazon S3 to deploy the front-end was an easy task. However, setting up Amazon Beanstalk to deploy the back-end was more complicated than expected. The reason is that Amazon Beanstalk is a facade to several other services (EC2, Security Groups, Load Balancer, etc.), which need to be understood and configured individually.

Finally, the only difficulty encountered when deploying MySQL in Amazon RDS was to correctly configure the Amazon Security Groups and Virtual Private Cloud so the EC2 machine hosting the back-end application could reach the MySQL instance. Also, the configuration of Route 53 and CloudFront to enable HTTPS was surprisingly very straightforward. Using an Amazon SSL certificate required no changes at the NestJS level.

Annex 1. Project Deliverables

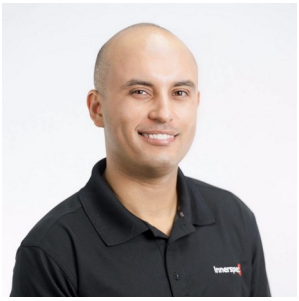
Name/Title	ANNEX
UX Test results	**4A & 4B**
UX First Click test results	**5A & 5B**
Front-End source code	**6**
Back-End source code	**7**
UML Sequence Diagram for PayPal purchase process	**8**
Product Connector Specifications	**9**
Masters	**10**
Presentation	**11**

Table 1: Project Deliverables

Annex 2. Bibliography

1: Harry Roberts, High-level advice and guidelines for writing sane, manageable, scalable CSS, https://cssguidelin.es/
2: Stephen Fluin, Why Developers and Companies Choose Angular, https://medium.com/angular-japan-user-group/why-developers-and-companies-choose-angular-4c9ba6098e1c
3: IhorFeoktistov, Why and When to Use Node.js?, https://relevant.software/blog/why-and-when-to-use-node-js/
4: Auth0 Group, Introduction to JSON Web Tokens, https://jwt.io/introduction

Annex 3. Vita



Chakir Mrabet was born in Tangiers (Morocco) and was raised in Spain, where he graduated from *Universitat Politècnica de Catalunya* in Barcelona with a Bachelor's Degree in Industrial Engineering, major in Industrial Electronics and minor in Informatics and Telematics. After working for several years as a Software developer, Chakir joined the Automotive sector as an R&D Engineer for *Gestamp Automoción*, an international group headquartered in Spain dedicated to the design, development and manufacture of metal automotive components. At *Gestamp Automoción*, Chakir was responsible for the application and innovation of ultrasonic inspection systems used for the detection of defects created by welding processes of automotive parts. This area brought Chakir close to *Innerspec Technologies*, a US based company world leader in NDT[16] solutions specialized in EMAT[17] systems and instruments. Chakir joined *Innerspec Technologies* in 2007 as the manager of the European office in Spain, and in 2012 as the Director of Software Engineering in Lynchburg, Virginia (USA).

16 Non-Destructive Tests

17 Electro-Magnetic Acoustic Transducers