

College of Science and Technology

School of Science and Technology

# SOFT30121: Advanced Analysis and Design

# Systems Analysis Design and Implementation

*By*

*N0683858: Hamid Mujtaba*

*N0688119: Elliot Harding*

*N0756079: Steffan Walker*

*N0704050: Michael Abraham*

*N0772082: Hamed Alshebli*

*T0084963: Haowei Goh*



NTU Stores Management System

￼

SOFT30121: Advanced Analysis and Design i

Systems Analysis Design and Implementation i

1. Introduction To Our System 3

1.1. Introduction 3

1.2. Architectural Pattern 3

1.3. Design Patterns 3/4

2. Design Documentation 4

1.1. Architecture Diagram 3

1.2. Deployment Diagram 3/4

1.3. Process Diagram 3

1.4. Structure Diagram 3

1.5. Component Diagram 3

3. User Help documentation 3

4. Variable Naming Conventions 3

5. Acceptance test plan 4

3.1. Functional tests 4

3.2. Performance requirements 4

3.3. Unit tests 4

6. Conclusion 3

3.2. References 4

# Introduction to Our System

## Introduction

//TO DO

## Architectural Pattern

Our system uses MVC architecture. This architecture comes with its share of advantages and disadvantages…

**ADVANTAGES:**

### Supports Collaboration

One benefit of using an MVC architecture is the division of the system into the three separate components. The Model, View and Controller are loosely coupled which means there is a reduced chance that a change in one will affect the other. One instance being if the View of the system changes there will be no need to change the Model. This loose coupling allows multiple developers to work on the system without the fear of interfering with another’s work.

### Faster Development

An additional benefit of the previously mentioned loose coupling is the hastened development. By having components where change in one is less likely to affect the other, developers can spend less time waiting for others to complete a feature and can focus more on developing other components.

### Easy Modification

MVC has a good capacity for future expansion/modification due to the reduced dependencies between components. This allows for additions to be made to components without having to change a great amount in others. For instance, if a client requests additional features after development has already begun, their implementation would be easier due to the loose coupling. This makes MVC useful for developing systems where requirements may change frequently.

### Multiple Views

Since the View component of the system is separated from the business logic, it is possible to support multiple views. This is useful for systems with multiple types of users that could require their own individual view of the system. For instance, in a business system the administrators would have greater privileges than regular staff and as such would need a different system view to reflect this.

### Code Reusability

Since the business logic and the view of the system are separate entities, the code has an improved reusability since it is not closely tied to any system. This provides a large advantage to organisations that develop a multitude of programs since they can reuse parts of the program in other software they are developing.

DISADVANTAGES:

### Complexity

Separating the system into multiple components can introduce complexity in relation to how control flows between the view, controller and model. Due to this, a good level of understanding for the architecture is needed in order to design a system that makes proper use of MVC’s loose coupling. A poor understanding of the architecture can lead to difficulties debugging due to the complicated flow of control.

### Unsuitable for Small Applications

The divided workload and loose coupling of components is only advantageous to large applications with numerous developers. For small applications with fewer developers and a reduced need for complexity, the MVC architecture can be more work to utilise than is worth. Due to the greater complexity of the architecture, a small team may end up over engineering their program and creating a greater workload for themselves than was needed.

## Design Patterns

Below are the various design patterns used in the system.

### Singleton

A singleton is a class with only one instance during runtime and a global point of access. This pattern is used in the design of the ‘DatabaseComms’ class. While the ‘DatabaseComms’ class is designed as a static class, the closest design pattern to it is that of the singleton.

The singleton design pattern was used for the ‘DatabaseComms’ class due to its widespread use throughout the system. It made sense to not waste the resources required to instantiate and destruct an object of a class which would be used in every page of the system.

Lazy initialization; the delaying of the creation of an object until it is required could have been an alternative pattern for the ‘DatabaseComms’ class. This would have meant that resources of the class would have only been added to memory if used, something which would have proved highly effective as not all features of the class are used within all use cases of the system.

### Module

The module design pattern groups related elements, such as methods or classes into a single theoretical entity. This design pattern was also used for the ‘DatabaseComms’ class as a way of grouping together all database networking methods.

### Publish/Subscribe

This describes behavioural design pattern, in which a one-to-many dependency among objects change for one object resulting in all its dependents being notified. This design pattern is used throughout the system, an example being the deleting of a ‘TransactionRow’ control causing all other invoice rows to update their position to maintain a consistent series of ‘TransactionRow’ controls.

An alternative publish/subscribe to consider for the ‘TransactionRow’ controls is the servant behavioural pattern. This defines common functionality for a group of classes, i.e. a helper class. The helper class would manage the ‘TransactionRow’ controls.

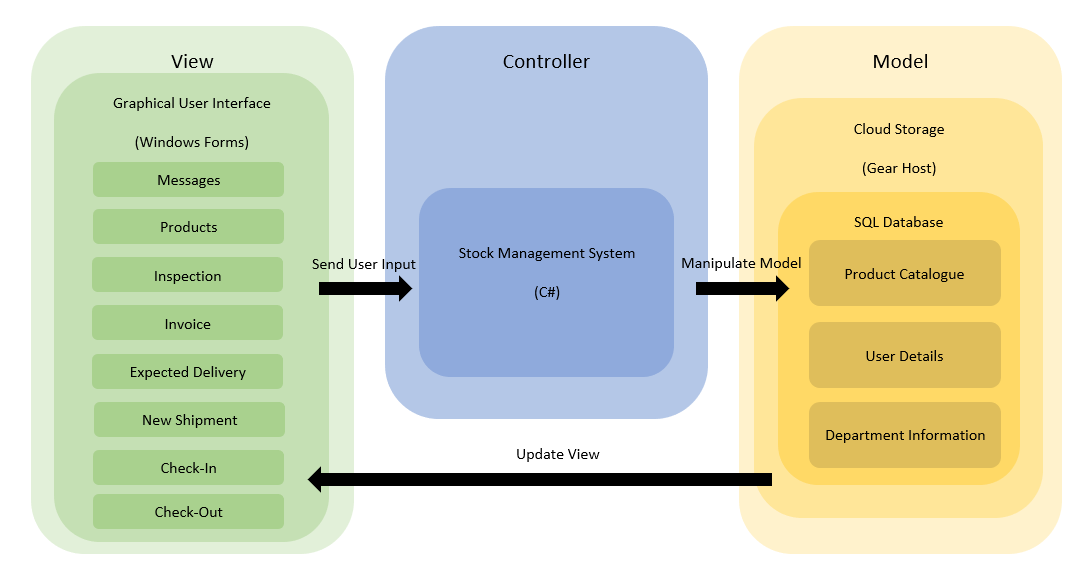
### Façade

A façade design pattern provides an interface to a more complex subsystem. Each of the page classes of the system utilize this design pattern, to make the system easier to use.

# Design Documentation

## Architecture Diagram

### Diagram



### Purpose of Diagram

//DESCRIPTION OF ARCHITECTURE

### How System Relates to Architecture Diagram?

The architecture our group has chosen for this project is MVC or Model-View-Controller. This architecture was deemed most appropriate based on the project’s requirements. The group consisted of 6 members meaning the Model, View and Controller workload could be split up to allow quicker development. Furthermore, since MVC is flexible, it would allow changes to the interface (view) to be made by one member without affecting any progress made to the model by another member and thus reducing wasted development hours. The provided architecture diagram should provide a clear view of the subcomponents that make up the system in addition to the interactions between them.

The View consists of a graphical user interface the user can utilise to interact with the system. The interface was designed using Windows Forms which is included with Visual Studio. This allowed the View and Controller parts to be developed using the same IDE which allowed quicker and simpler development of the application. Windows Forms is widely used among developers to create well designed interfaces with ease. Furthermore, it is developed by the industry leading Microsoft who are known to develop efficient, robust and powerful software.

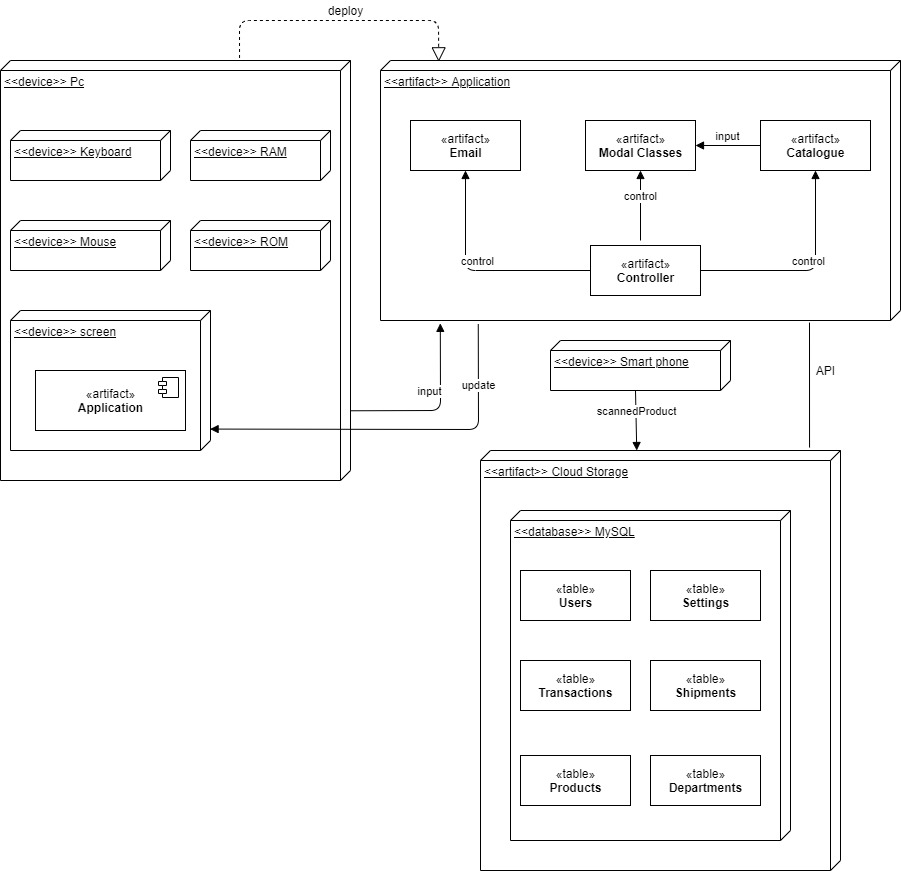
The Controller is responsible for receiving and validating the user input from the view before sending the relevant instructions to the model. Validation ensures the input data is not erroneous and if so, prevents it from being stored within the model. The controller uses listeners to wait and respond to specified events occurring in the view such as a button being clicked. The data from the input fields is then sent to the model for processing after it has been validated. The controller is programmed using C# which is supported by Visual Studio an allows for easy data retrieval from the view.

The model is responsible for storing the systems data and processing it depending on the instructions it receives from the controller. The interaction is done through the ‘DatabaseComms’ class which is written in C# and the data is stored in an SQL database on a cloud server. After the model has processed instructions from the controller and updated the stored data, it will then update the view.

An example of the system from a Model-View-Controller perspective would be the user clicking on an item from the catalogue. The view would recognise which item has been clicked and sends that data to the controller. The controller would then make a request to the model for the products page and the information regarding the product. The model would then update the view with the desired product page along with information associated with said product.

## Deployment Diagram

### Diagram



### Purpose of Diagram

The deployment diagram is used to show the underlying hardware used in the system and the software that runs on this hardware. The diagram communicates how the software system and the hardware work together to execute different task done.

### Why Use It?

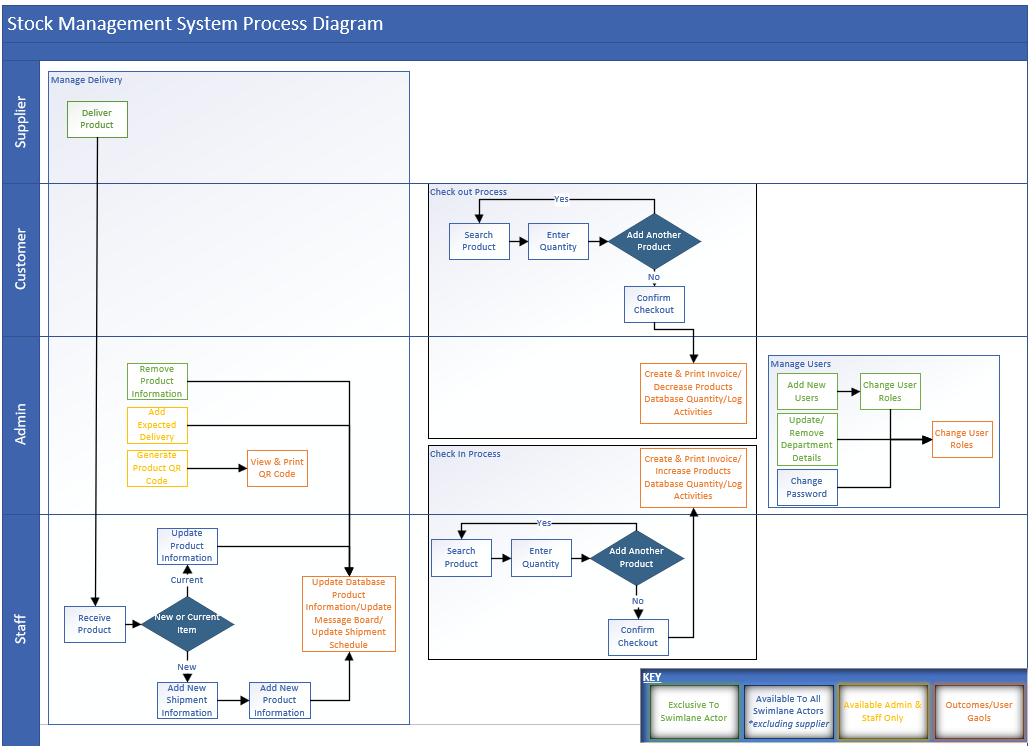
Where deployment shows hardware and software components of a system, it also acts a guideline to setup the proposed system. It is for this reason, a deployment diagram was used, so any organization/person who wishes to implement the system, knows the resources required and has the knowledge of what to expect from the system.

### How System Relates to Deployment Diagram

The deployment diagram shows the fundamental resources required to carry out the tasks of the system. It shows a computer system that has input devices to feed data to the system and internal devices that run the application; which is also shown and is further elaborated through mentioned modules of modal classes, controller class and email (which are the underlying software components to run the system). A database is connected to the software system and tables shown that keep records required by the system and a smart phone is present to scan the products.

## Process Diagram

### Diagram



### Purpose of Diagram

Process Diagrams give a visual representation, in lineal order, of the interactions between an actor and the system. Separated by swimlanes to show accountability between each connected event, the diagram denotes what needs to be accomplished to progress through each process’s timeline.

### Why Use It?

We have used a process diagram to clarify every activity occurring in each process and illustrate a logical path for the collection of flows that form our stock control management programme.

### How System Relates to Process Diagram?

Our diagram divides the system into several major processes that transpires across multiple pages of the programme:

* **Manage Delivery:**

This process represents the procedure that unfolds when a new shipment of products is supplied and documented into the system. After the transferal of goods between the supplier and the staff member, this diagram highlights each event (carried out in the systems ‘Shipment’ flow) required to update or add new stock information to the database.

* **Check out/Check in Process:**

Both these processes occur within the same concept of a ‘Basket’ flow, sharing the same functionality to append products to a list, however the outcome of each process delivers altered results; either increasing or decreasing the quantity associated to the product within the database whilst creating a log of these activities. Additionally, the ‘Check out’ process composes an invoice ready to be sent to the appropriate body.

* **Manage User:**

This highlights the administrative capabilities the system offers. Access through different actors invoke separate functionalities; admin possess the highest clearance of authorisation, permitting access to additional features.

As our system involves user roles with overlapping functionalities, provided below is a key to illustrate which events and activities are also available to o

## Structure Diagram

//REVIEW

Include your structure diagram here and explain why you have used it and what it shows –make sure to talk about the design pattern used.

## Component Diagram

//REVIEW

# User Help documentation

//TO DO

Explain how to run and use your system. Provide installation instructions including any necessary dependencies.

For a first class, this should also include evidence that this user help documentation is useful.

# Code Naming Conventions

For the systems code variable and function naming conventions were outlined. Reasons for using conventions included:

* Reducing effort needed to read and understand source code
* To allow for easier debugging of source code, such as knowing the scope of member variables.
* Code reviewers can focus on issues other than syntax

The naming conventions are as follows:

|  |  |
| --- | --- |
| **Code element** | **Rule** |
| Functions | CamelCase |
| Variables | All variables are in CamelCase   * Class member variables start with m\_ * Static public member variables start with an upper-case letter * Constant variables start with a c\_ |
| Controls | Control variable names start with the type of control they are followed by an underscore. For example; ‘txt\_name’ is a text control |
| Dialogs | Dialog classes contain the word Dialog within their name. |

## Acceptance test plan

## Functional tests

### User: Consumer

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Procedure | Expected Outcome | Actual Outcome |
| Login/Sign up | | | |
| Sign up | Run the application.  Click ‘Signup’.  Type in username, password and choose department. If the data is incorrect, an error will be displayed.  Based on if you’re a student, staff or admin, you have different level of authorization over the system. | - User gets created/added to the database and directed to the home page (products page)  - Should be able to use | - Correctly creates a new user and directs them to the home page (products page).  - You can use the new account to login from the ‘Login’ page. |
| Login | Fill in the login details (N number + password). If user details are incorrect, an error will be displayed, and user is prompted to enter credentials again or create a new account. Click “login” to login. | - Validate username and password from database when attempting to login  - Display an error message if user details incorrect or they don’t exist in database | - The user login in successfully if they type in their details correctly  - No way to tell if account already exists (login page)  - User displayed with an error message when wrong details used to login |
| Forgotten Password | When you click on the “forgotten password” button, it will ask the user for your N number. After typing their N number and confirming it will email the user their N number. | - Send an email to the user’s NTU email with their password | - Doesn’t tell you the N number doesn’t exist if user types in wrong number  - Sends email to the users NTU email |
| Products Page | | | |
| Browse products/ catalogue | The user will be directed to the products page once they successfully login. You can click on any product to display more information regarding the product or browse the catalogue and view more products by scrolling down. | - Displays all the products with their pictures and name  - Can scroll up/down to browse all the products  - When you click on the product, displays all the information regarding that product (Single products page) | - Sometimes takes a very long time for the products to load  - Images can be a little glitchy when scrolling up/down  - Products displayed correctly with pictures  - Can browse products perfectly (scroll up and down)  - When clicked on a product, it displayed detailed information about the product |
| Search Function | On the ‘Products page’, there is a search bar at the top which lets you search for specific products. If you type in the product you want to find, the system will help you by filtering them | - Filters all the products in the catalogue and displays all the correct products  - Only displays the products that are related (e.g. if user searches for product 12, it will only display products that have ‘12’ in their name) | - The products are filtered correctly depending on what product you try searching for. E.g. If you search for product 12, every product with the number 12 will be displayed (12, 121 etc)  - Few products that are not related are also displayed (10, 11, 12) as it finds all products which contain any of the 2 digits (1 and 2) as well. |
| Display single product + Information | If you click on any product in the catalogue (products page), it will take you to the single products page where all the information about that specific product will be displayed. | - Should display the product image, product name, current stock, price, VAT, supplier id and product description  - It should also let you see the location of the product marked on a map | - Product displayed with a picture, and all the correct information (price, stock, description etc)  - Shows the location where the item is located in the stores |
| Add product to basket (Check In/Out) | If you enter the quantity and click ‘Check In/Out’, it will add the product to the basket | - After use types in quantity and clicks on ‘Check In/Out’ a popup box should appear asking if they would like to view their basket or continue shopping  - The products will be added to the basket | - Lets you add more quantity for a product in the basket than there is available (current stock)  - Successfully adds product to the basket |
| Basket | | | |
| Check Out | After adding the products to the basket, you can navigate to the basket page and click on ‘Check Items Out’ to check out. | - Updates stock and removes them from basket | - Checks out the items and updates the stock amount in the database  - Once products checked out, removes them from basket |
| Clear Basket/Remove product | Users can remove all products from their basket by clicking the ‘Clear All’ button or remove a specific product from the basket by clicked the ‘X’ next to the products. | - Should remove all the products from the basket  - Doesn’t change the stock amount  When ‘X’ pressed, only that product is removed | - Removes all the products from the basket  - Stock amount doesn’t change  - Removed the products if you click ‘X’ |
| Return Items | If the user wants to return an item which they have checked out, they should be able to do so by scanning the item back in using the mobile application and simply clicking ‘Return Items’ | - Update the database and the current stock amount for the right product  - An error message if there are no products in the basket | - Lets the user return items, which updates the stock amount currently in the system  - Error message was displayed when trying to return items with an empty basket  - User added products to the basket from the catalogue and clicked the ‘Return Items’ accidently. The system assumed they are trying to return products which have been previously checked out and updated the database. |
| Settings & Messages Page | | | |
| Change Password | If the user wishes to change their password, they can navigate to the ‘Settings’ page where they will be able to choose a new password | - After typing in the new password, the user | - Lets you successfully change your password |
| Messages | Users can see the messages which will remind them if they have taken any equipment which needs to be returned | - Display messages reminding the user when the items need to be returned by. | - Functionality doesn’t work as we didn’t implement it due to time restrictions |
| Mobile Application | | | |
| Login | Login the application by typing your N number. No password needed. | - User logs in when they type in their N number and are directed to the home screen of the application  - If user doesn’t type in the correct N number or user doesn’t exist, an error message will be displayed | - Login successfully and gives an error if user doesn’t type in any username  - User can login even if they don’t have an account on the system or without the N number (No validation being made) |
| Scan Products (QR Code) | Click on the scanner in the app which will open the scanner, allowing users to scan multiple products at once | - Scan the QR code and update the basket  - The basket should display the correct product  - If same product is scanned, it should update the quantity | - Successfully scan the products and add them to the basket  - If same product is scanned multiple times, it will update the quantity of the product in the basket |

### User: Admin & Employee (Staff)

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Procedure | Expected Outcome | Actual Outcome |
| Login | Admin and staff will have to use the same login page as regular users | - If you type in admin/staff login details, you will be logged into an admin/staff account where you will have extra UI options (on navigation menu) and more privileges | - Logged into a staff account successfully  - Extra options available on the navigation menu |
| Products (Single product information) | | | |
| Change stock (amount) | Staff/Admin users can change the stock amount if they click on a product from the products page, and then click on the ‘Current Stock’. | - When you click on the ‘Current Stock, a popup box will appear asking you to enter new stock amount  - Will GUI with the new stock amount | - Updated the database and the GUI with the new stock amount |
| Change VAT | VAT can also be changed by clicking on the ‘VAT’ on the ‘Single Products page’. | - When you click on the VAT, popup box will appear asking you to enter new VAT  - Will update the VAT from GUI | - Updated the VAT in GUI |
| Change supplier ID | Click on ‘Supplier Id’ to change the id | - When supplier id clicked, asks user to add new id  - Updates GUI | - Successfully changed the supplier Id + updated GUI |
| Edit product information | Change the product description by clicking on the description box next to the product. | - A box will appear with the product description when clicked  - The text box should contain the current description so you can modify it however you want. Or add a completely new description | - Updated the product information successfully |
| View QR Code | View the QR code for the product | - will display the QR code for that specific product | - Displays the QR code for the product |
| Save QR Code | After viewing the QR code you can save it to the local storage by clicking the ‘Save’ button | - Saves the QR code  - | - Saves the QR code on the local storage |
| Edit Map Location | You can change the map location of a product by simply clicking ‘Edit Map Location’ which will allow you to click on the map to indicate the new location. | - When you click on the new location on the map, it will be marked with a big red ‘X’. | - Lets you update the location of the product on the map |
| Remove Product |  |  | - Removes product from system and updates database |
| Save Changes (Made to the product details) | After making all the changes to a single product, click on the ‘Save Changes’ button to save the changes | - Updates the database and add the new information | - Updates the system and the database with the new information |
| Invoices | | | |
| Display invoices for different departments | If the user navigates to the ‘Invoices’ page, they can choose to display the transaction history for departments. They will need to first choose a department, choose the dates for which they want the transactions for and then click ‘Fill Form Log’ (select ‘All Departments’ to view transactions for all departments). | - Display all the transaction history for each department between chosen dates  - Display all the transactions for all the departments between certain dates  - Displays the product id, price and the total amount the department owes. | - Displays all the transactions between certain dates made by staff from each department separately (display transition for a specific department)  - Displays all the transactions made by all departments between certain dates  - Displays all the data correctly  - Gives an error message when you request information for a department which doesn’t have any transactions |
| Clear (displayed) Invoice data | To clear the data from the GUI, click the ‘Clear’ button. | - Should clear all the data displayed in the GUI | - Cleared all the data that has been displayed on the GUI |
| Add Row | If user feels like they need to manually log in a transaction, they can do this by adding a row to the displayed data by clicking the ‘Add Row’ button. | - Should create a new row of data which user is able to fill in  - Manually added row also gets saved in the transactions log when you create the invoice | - Created a new row under the rest of the displayed invoice data  - New data was saved when invoice created |
| Create & Send Invoice | To create an invoice after displaying the data the user wanted, they can click on the ‘Create and Send Invoice’ button which will save the data in an excel format. | - Create an excel spreadsheet with all the coulombs and rows labelled correctly.  - Has total amount of the equipment the department (staff) checked out | - Creates an excel document with all the information the user requested for and saves it on the local storage  - Creates an excel document even if you didn’t request any information (empty document)  - Has total amount at bottom of the spreadsheet |
| Shipments | | | |
| Display incoming shipments | The user check if there are any incoming shipments by navigating to the ‘Shipments’ page and then choose the dates and click ‘View’ | - Should display supplier name, site name, order number and requested date  - Should only display shipments between the dates chosen | - Displayed the shipments between the chosen dates  - Displayed the data correctly  - Showed an error message when requested data for when there were no incoming shipments |
| Add Expected Delivery | User can manually add new shipments by clicking on the ‘Add Expected Delivery’ button on the ‘Shipment’ page and then fill in all the text fields with the correct data. | - Will add all the data to the database  - Will display a reminder of the expected delivery on the messages page | - Added new expected delivery even thought all the text fields were empty  - Displayed a reminder on the messages page for staff/admin |
| Add New Shipments | If you click on the ‘Add new shipment’ it will take you to another page which will allow you to add new product | - Should add the new product in the database  - New product should be displayed in the product catalogue | - New product successfully added and is displayed in the product catalogue |
| Settings | | | |
| Change Password | If the user wishes to change the password, they can navigate to the ‘Settings’ page where they will have the option to choose a new password | - After typing the new password, will update it in the database | - Lets you successfully change your password |
| Create New User | Admin will be able to add new (employee) staff/admin users. They can also choose what level of privilege they will receive (admin or employee staff).  Only the admin user has this privilege | - New user will get created in the database  - Should be able to login with the new user details  - Admin user should have more access and (employee) staff should have less | - Can login with the new user details  - Admin user had access to more features |
| Add/Remove Departments | The user can add or remove departments. These departments will appear when a new (consumer) user wishes to sign up. If they click on the ‘+’ button, they can type in a new department, or click ‘X’ next to an existing one to remove it. | - Should update the GUI and remove the department when ‘X’ clicked  - A popup text box should appear asking you to type a new department when ‘+’ clicked  - Should update the GUI and add a new row with the new department | - Removed the department from the GUI when ‘X’ clicked  - Added new department when ‘+’ pressed & a popup box appeared asking to type new department |
| Update Departments | After user has finished modifying the departments and wish to save the changes to update the database, they will need to click ‘Update Departments’. | - Updates the database with the new information  - Will display the new departments when the department list is viewed in the invoices page or signup page (for consumer staff) | - Updated the database  - Displayed the information correctly on other pages |
| Messages | The message feature will have reminders for the staff if there is a new shipment/delivery expected soon | - Will display the messages in the ‘Messages’ page when a new ‘expected delivery’ is added  - Only display this information for the staff/admin users  - Should only display the reminder once on the screen | - Displays the messages for expected delivery when new ‘expected delivery’ added  - Displays the message (reminder) more than once (repeats the same message) |

## Performance requirements

//TO DO

## Unit tests

//DOING

## Conclusion

//TO DO

## References

<https://en.wikipedia.org/wiki/Software_design_pattern>

Oliver, L. & Luukkala, V. (2016). On UML’s Composite Structure Diagram.

<https://www.researchgate.net/profile/Ian_Oliver9/publication/268374023_On_UML's_Composite_Structure_Diagram/links/54f60d230cf27d8ed71d4af3.pdf>

Bell, D. (2004). UML basics: The component diagram. <https://www.softwareresearch.net/fileadmin/src/docs/teaching/WS13/SE/UML_basics-_The_component_diagram.pdf>

Genero, M. & Manso, T. & Visaggio, A. & Canfora, G. & Piattini, M. (2007). Building measure-based prediction models for UML class diagram maintainability.

<https://link.springer.com/article/10.1007/s10664-007-9038-4>