# Functional Requirements

These should be described in enough detail for the designers to design a system satisfying the requirements. These should also be testable.

They should also be organised appropriately (could be by feature) and prioritized (could be by MoSCoW).

A good requirement is:

* Correct
* Unambiguous
* Complete
* Consistent
* Ranked for importance
* Verifiable
* Modifiable
* Doesn’t specify any particular design
* Traceable

Not all of the requirements above follow this. Also need ratings added to them.

## Usability requirements

Must

**2.1.1: Store the inventory of the store’s stock**

The system must utilise a database to store information on the products being sold, their quantities and the alpha codes used in store to uniquely identify products. This database should update in real time when products are removed from or returned to the store. Keeping a digital count of stock is more accurate and robust than written logs.

**2.1.2: Display currently available stock**

The system must display, in a catalogue style, the stock currently available in store. The information should include the products name, the products alpha code, a picture of the product and the quantity available. Since the consumer can find whether the product they require is stocked immediately, there is less time wasted searching and asking staff members if the product is available.

**2.1.3: Use a virtual basket when buying products**

When consumers go to remove an item, it should be added to a virtual basket along with any other products they might want to remove. This basket can then be “Checked out” when the consumer leaves the store and the products removed be registered in the system. Before checking out, the consumer should be able to add and remove products they have in their basket. Virtual baskets are used by a majority of online retail sites which makes the system more relatable to consumers and is also a quick and clear indicator of what a consumer intends to take from the store.

**2.1.4: Allow products to be returned**

The system must allow products to be returned that have previously been removed from the store. When a product is returned it will be inspected by a staff member before the return is registered which in turn updates the stock, the log and the invoice. Consumers may find they no longer need the product that they removed from store or that it was inspected incorrectly by staff. For this reason, a feature to return any product that was removed from the store must be implemented.

**2.1.5: Allow catalogue of products to be changed**

The admin users must be allowed to add new products and remove old ones from the stores catalogue of available products. Since the needs of consumers inevitably changes as time goes on, the ability to add new products to sell and remove old ones will be implemented to keep the store adapting to their customer’s needs.

**2.1.6 Create invoices**

The system must create an invoice based on information of what products have been removed, the price of the products and the department which removed the product. The invoice should be created at the end of the month. By generating invoices automatically, there is less workload on staff and a decreased chance of invoices being completed erroneously.

**2.1.7: Allow users to login to the system**

The system must facilitate logging in for three types of users. These users include:

* **Admin:**

All the privileges available to staff in addition to control over the catalogue

* **Staff:**

With ability to receive shipments and inspect returned items/shipments

* **Consumer:**

Only allowed to remove and return products from the store

Should

**2.1.8: Convert supplier units to consumable units**

The system should convert the unit a product is bought and received in, to the unit that the product is consumed in at the store.

E.g. 1 Box of pens should be converted into the number of individual pens a box contains as pens will be removed from store individually.

Performing this conversion through the system reduces erroneous input and time spent processing deliveries.

**2.1.9: Register received shipments**

The system should allow staff members to input shipment data, such as supplier codes and quantity, so that the store’s inventory updates with the stock contained in the shipment. Allowing the stock to automatically update with the contents of shipments reduces the workload of staff and allows shelves to be refilled quicker.

**2.1.10: Allow stock inspection**

Stock that has arrived from shipments or been returned by a consumer must be stored in an inspection database separate to general stock. When stock has been inspected by a staff member it is then moved to the general stock database. Since products can be damaged upon delivery to the store or by consumers trying to make a return, it is important to have some method of inspecting products before they are re-introduced to the stores stock.

**2.1.11: Display tags for important information**

The system should display, on the product page, any important information relating to the product in question. Products which are toxic, explosive, radioactive or pose any kind of health risk should have this information displayed clearly and readily visible at the top of the product page. While products that pose health risks are usually labelled accordingly, it makes sense to include this information on the product page before the consumer attempts to handle a potentially dangerous item.

**2.1.12: Track items that expire**

Products that can only remain on store shelves for so long should be tracked through the system so that when expiry dates are close, the staff members are notified. Implementing this feature is important for product rotation whereby products with soonest expiry dates are placed closer to the front of a shelf and vice-versa.

Could

**2.1.13: Provide directions to products**

The system could display a map of the store with a marker indicating the location of the desired product within the store. This feature will allow store management to adjust product placement within the store and still provide a means for consumers to find their desired product.

**2.1.14: Remind staff about deliveries**

The system could remind staff members with notifications about upcoming shipments/deliveries. The notifications would include the delivery’s time, products being delivered and any additional information such as whether special handling equipment is needed. Notifying staff of upcoming deliveries allows time to prepare for substances like gases that could require special handling procedures.

**2.1.15: Scan barcodes of products**

The system could allow users to scan barcodes of products to remove the need for manual input of data into the system. Barcodes would be scanned when consumers add products to the basket and when staff need to register received shipments as part of the store’s stock. Scanning barcodes would reduce the amount of data input work done by both staff and consumers and reduce accidental adding of products to the basket.

## Reliability requirements

Must

**2.2.1: Reliable database language**

The system should use SQL for the databases it uses. SQL is an industry standard language designed for building reliable and robust databases which makes it a solid choice for the system. Usage of a good language will help make a database that is expandable and resistant to incorrect data input.

**2.2.2: Log the removal and return of products**

The system must keep a log of what products have been removed/returned by storing the products name, product alpha code, date and time of occurrence, the individual involved and their department name. Keeping a log of the removed products will provide something to calculate invoices from.

Should

## 2.2.3: Backup data externally

## The system should backup any data it is storing locally, into an external storage location to prevent data loss. The data should be backed up to a cloud-based drive like OneDrive to ensure there is an existing backup in the event of data corruption or accidental deletion.

**2.2.4: Provide low stock warnings**

The system should notify staff members when the stock of a particular product is low so that more can be ordered in time. By promptly informing the staff of low stock, there is a lessened chance that the product a consumer requires will not be available which improves the stores reliability.

**2.2.5: Log received shipments**

When shipments are received and registered through the system, they should be logged in a database. The log should include the shipment’s Order Head with details about the shipment itself and the Order Line with details on the products and quantities included in the shipment. Logging the shipments received will give management a clear view of what products are being re-ordered frequently which is very useful information.

Could

**2.2.6: Offline Buffer**

The system could include a buffer by means of storing a local copy of the product catalogue so that products could still be taken out if the system was not connected to the internet. When the system regains an internet connection it would then update the stock level and product removal log. This would prove useful as consumers would still require products from the store if the universities internet was disrupted.

## Performance requirements

Must

**2.3.1: Memory efficient language**

The system should be written in C# due to the languages strong memory management capabilities. This will improve the systems performance and reduce the users waiting time for operations to happen.

**2.3.2: Real time stock update**

The systems should update the stock database as soon as a product is checked out of the store. Updating in real time provides the staff with information that is current and provides a more accurate representation of the stores current stock level.

Should

**2.3.3: Responsive UI**

The graphical user interface should be responsive and receptive to the action’s users make. Interactions such as clicking the increment/decrement quantity buttons should quickly update the number of products in the basket. By making the interface respond swiftly, there is a lessened chance that users will make an erroneous click in anticipation of the interface updating.

Could

**2.3.4: Multi-threading**

The system could make use of multi-threading to process multiple actions concurrently. Considering there will be multiple users making use of the system, a multi-threaded program would be able to handle user request quicker and more efficiently.