Project Report

Computer Science DMAI-0914 – Group 5

University College of Northern Denmark

Adrian Frunza Alex Muriuki Njogu Kristupas Sakalius

Monika Vyšniauskaitė Stefan Patatu

University College of Northern Denmark

Workshop Persistence Project

Technology and Business Computer Science AP Degree DMAI-0914

Project participants

Adrian Frunza	Signature: ______________________________
Alex Muriuki Njogu	Signature: ______________________________
Kristupas Sakalius	Signature: ______________________________
Monika Vyšniauskaitė	Signature: ______________________________
Stefan Patatu	Signature: ______________________________

Supervisors

Istvan Knoll

Ann Francke

Henrik. V

Submission date

20.03.2015

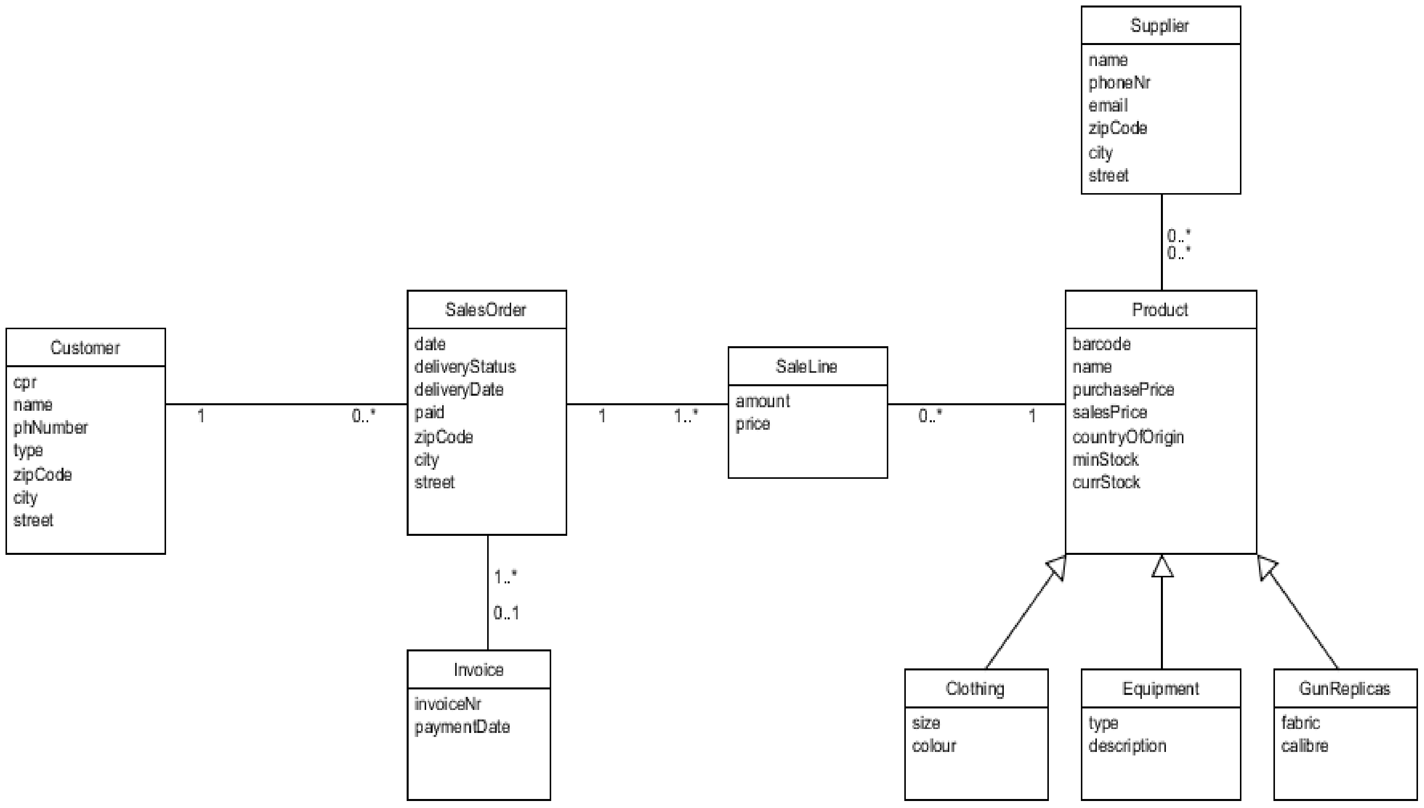
System development

Domain model

This is the domain model. All the classes, their respective attributes, connections, multiplicities can be seen there.

The first thing that was changed in the domain model was to add a saleLine between the product and the SalesOrder which would contain the amount of items purchased of that type and the price of a single unit.

The SalesOrder at first always had an invoice. However, later on it was found out that the order comes first and only then the invoice. This means the multiplicity in the domain model had to be changed from 1 to 0..1 .



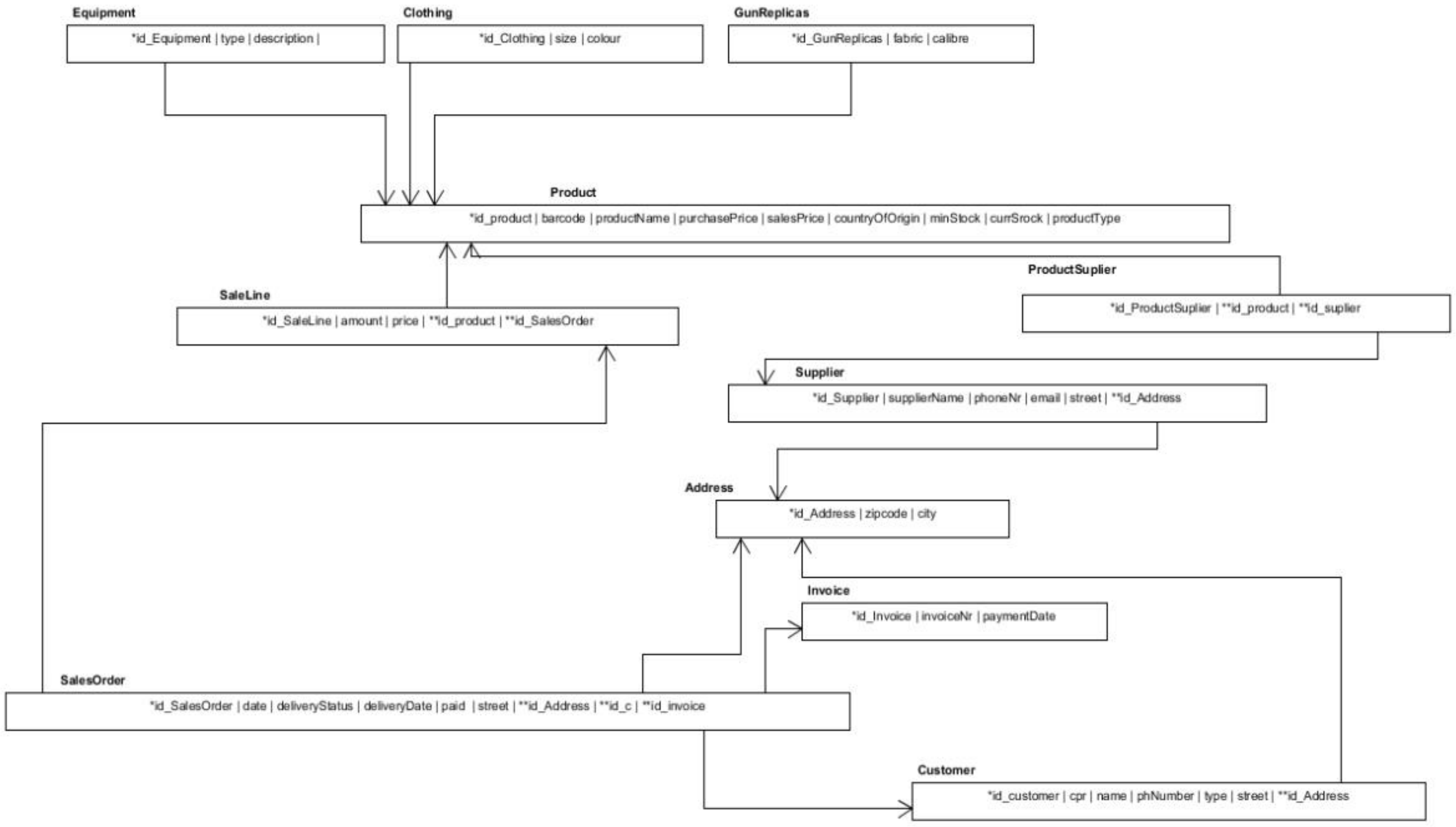
Relational model

The relational model shows how the tables are connected in this database. It was designed based on the domain model.

One of the things discussed was about the way to connect the customer to the order. It was decided to connect “customer” table directly to the “order” table. The other proposal was to connect the customer to the “invoice” table and every time there is an order it is connected to the invoice. The decision is based on the most efficient way of accessing the orders of a customer.

Also, the relational model was normalized by creating the “address” table.

The relation “many to many” between product and supplier was solved by creating an additional table where the two tables are connected.


Equipment
*id_Equipment | type | description |
Clothing
*id_Clothing | size | colour
GunReplicas
*id_GunReplicas | fabric | calibre
Product
*id_product | barcode | productName | purchasePrice | salesPrice | countryOfOrigin | minStock | currSrock | productType
ProductSuplier
*id_ProductSuplier | **id_product | **id_suplier
SaleLine
*id_SaleLine | amount | price | **id_product | **id_SalesOrder
Supplier
*id_Supplier | supplierName | phoneNr | email | street | **id_Address
Address
*id_Address | zipcode | city
Invoice
*id_Invoice | invoiceNr | paymentDate
SalesOrder
*id_SalesOrder | date | deliveryStatus | deliveryDate | paid | street | **id_Address | **id_c | **id_invoice
Customer
*id_customer | cpr | name | phNumber | type | street | **id_Address

Fully dressed use case

Use case name	createSale	
Actors	Cashier	
Pre-conditions	Customer & Product(s) has to exist in the system	
Post-conditions	Sale is registered in the system	
Frequency	As many times as needed	
Main Success Scenario	1. Cashier types in customer’s cpr to start the sale.	2. The system asks for a barcode and quantity.
	3. Cashier types in the product’s barcode, quantity and the same attributes of all the products the customer wants to buy.	4. The system shows the price of each product purchased and updates total amount.
	5. Cashier clicks that all items were added	6. The system asks for the payment and delivery type
	7. Cashier chooses the up-front payment option and receives cash from customer.	8 The system shows that the sale has ended and prints out receipt.
Alternate Flows	1a. The customer doesn’t exist in the system 3a. The product with that barcode doesn’t exist 7a. Customer wants to pay at the end of the month.	

Fully Dressed Use Case

During design of the fully dressed use case two trivial issues were encountered. The first one was how to uniquely identify the customer and the second one was determining which was the main success scenario between up-front payment and monthly payment.

To uniquely identify a customer, it was decided that the cashier keys in the customer's CPR and the main success scenario is when a customer pays upfront as it is the most common in most business cases.

The fully dressed use case implemented is in two-column or conversational format. This is because of its clear visual representation between the actor and the system as opposed to the one-column style which is more compact. [Applying UML and Patterns – Craig Larman]

Main use case is createSale for creation of sales order because this is where the business makes their revenue from. The primary actor here is the cashier as he or she calls upon the services of the system in order to create a sale. Customer and products have to exist in the system forehand.

Main Success Scenario

The trigger event that starts the scenario in this case is the customer walking up to the cashier with items to purchase. The cashier keys in the customer's CPR to start the sale. The system then asks for the barcode and the quantity of item(s) to be purchased so as to countercheck with the current stock. After the cashier keys in the barcode and quantity, the system shows price of product and updates the total amount up to that point. Cashier asserts that all the purchased items were added so that the system asks for mode of payment and delivery type. Cashier chooses up-front mode of payment and receives cash for the purchased items from customer. System signals that the sale has ended and prints out a receipt.

Alternate Flows

Alternate flows are branches from the main success scenario and they are mostly a mix of "happy" and "unhappy" scenarios.

In this case, we have three alternate flows.

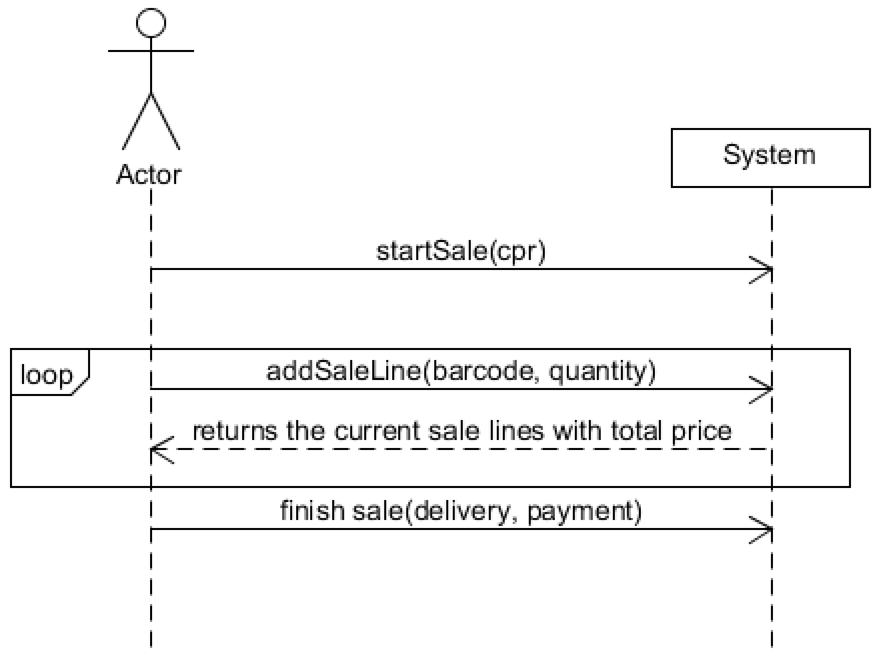
1. The customer does not exist in the system. A customer has to exist in the system for the sale to occur. Customer is created to handle this.
2. The product with that barcode does not exist. A product also has to exist in the system for the sale to occur.

Product is created to handle this.

3. The customer wants to pay for the purchased items at the end of the month and not up-front. The cashier chooses the monthly payment option in order to bill the customer an invoice for the goods purchased at the end of the month.

Sequence diagram

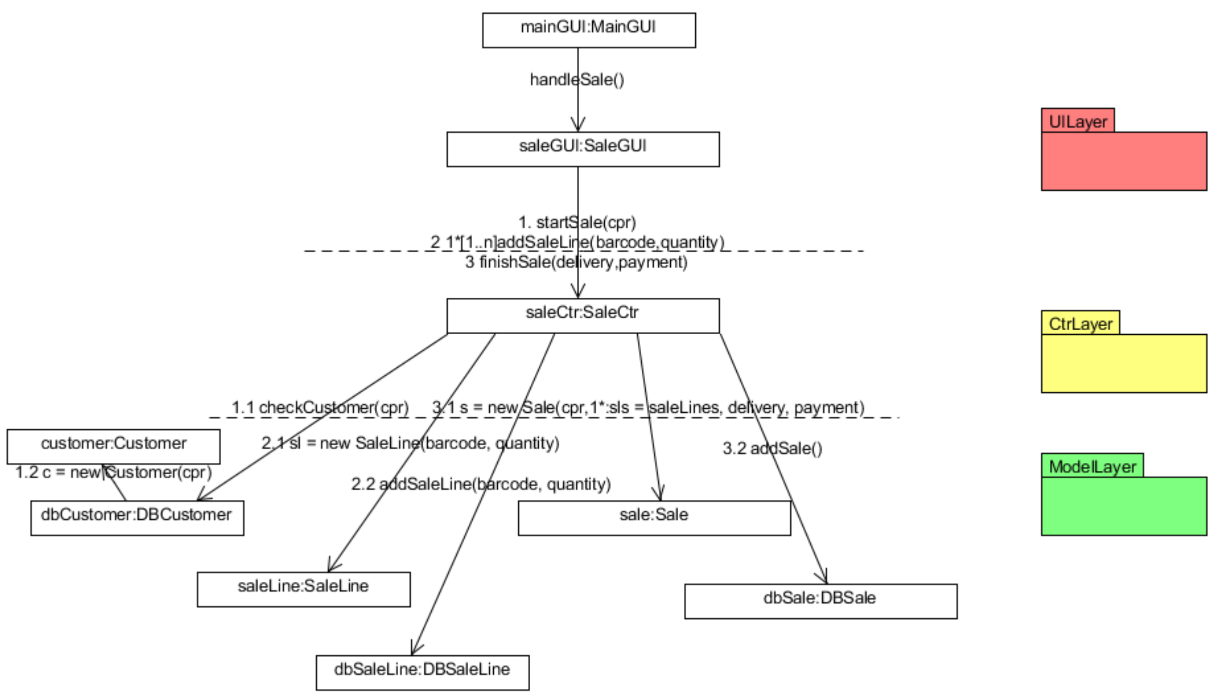
This is the sequence diagram. You can see how actor (cashier) interacts with the system and how it responds.


Actor
System
startSale(cpr)
loop
addSaleLine(barcode, quantity)
returns the current sale lines with total price
finish sale(delivery, payment)

Communication Diagram

Below is the communication diagram. It represents the deeper look into our system and how the methods should work.

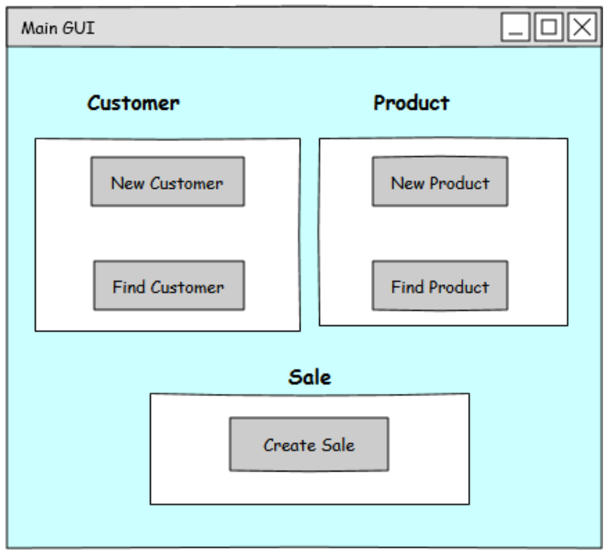
The first problem we had with making it was deciding on how the methods should work. This was solved by choosing a single huge method in the UI layer which then was separated into many smaller ones in the lower layers. A couple of numbering issues were also encountered and solved with the teacher's aid.

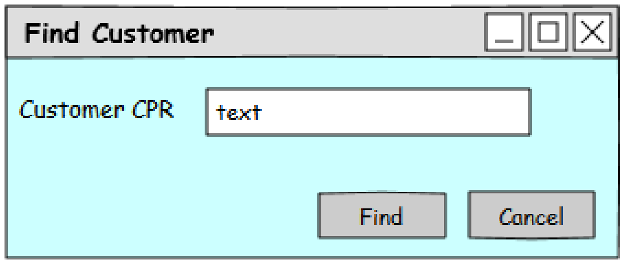


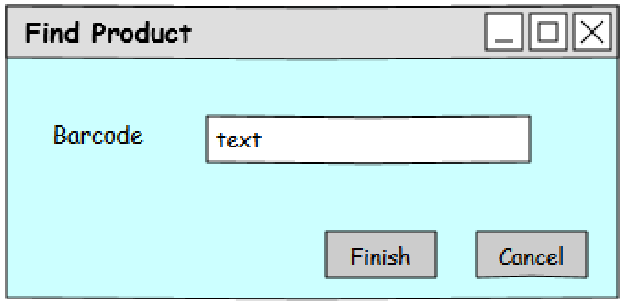
Mock Ups(s)

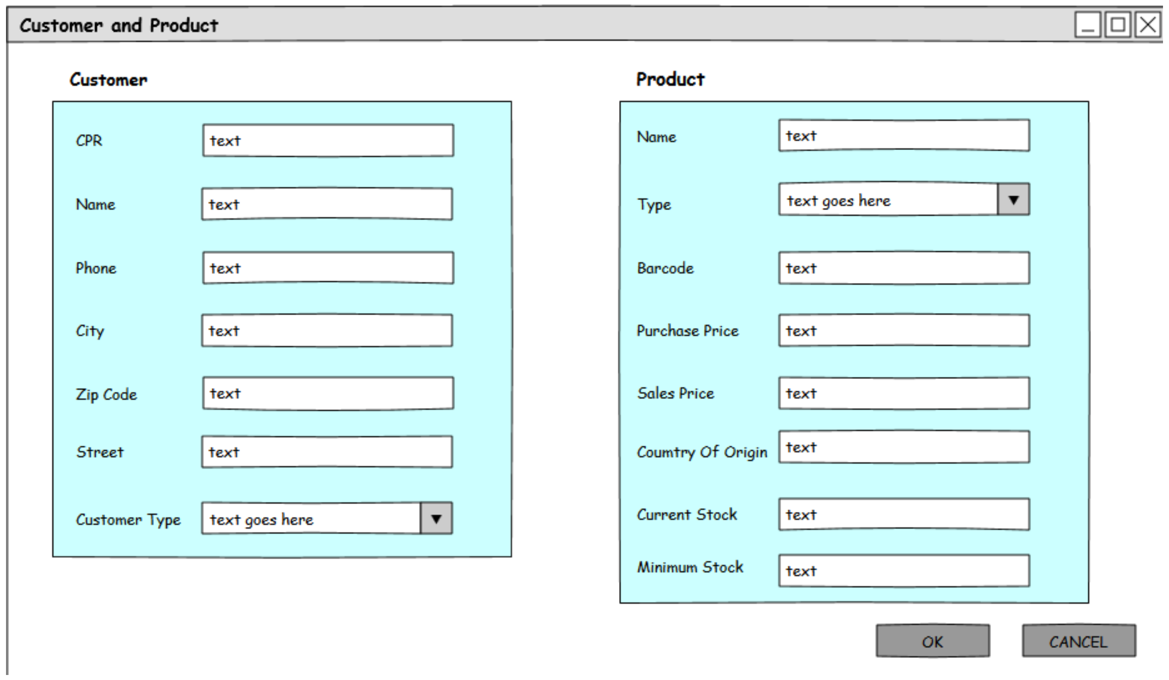
The design of the mock up(s) is based on virtual windows method by using the domain model and use cases as a starting point on ideas and finally implementation in the GUI. The initial ideas were a bit rough. For example there was too much clustering of information on one interface.

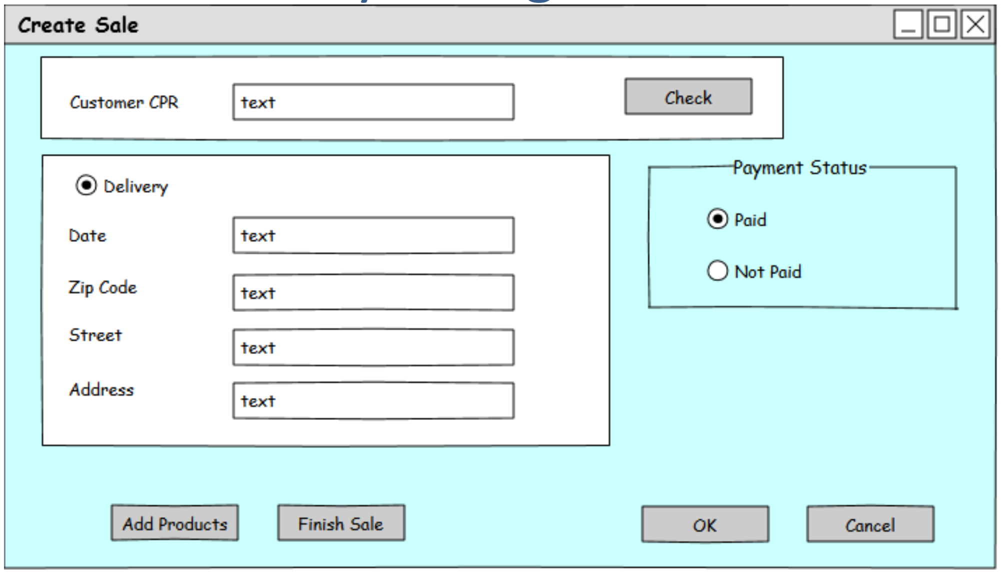
The final design agreed upon is based on simplicity, reducing complexity and adaptability to a small business or company. [User Interface Design – Soren Lauesen]

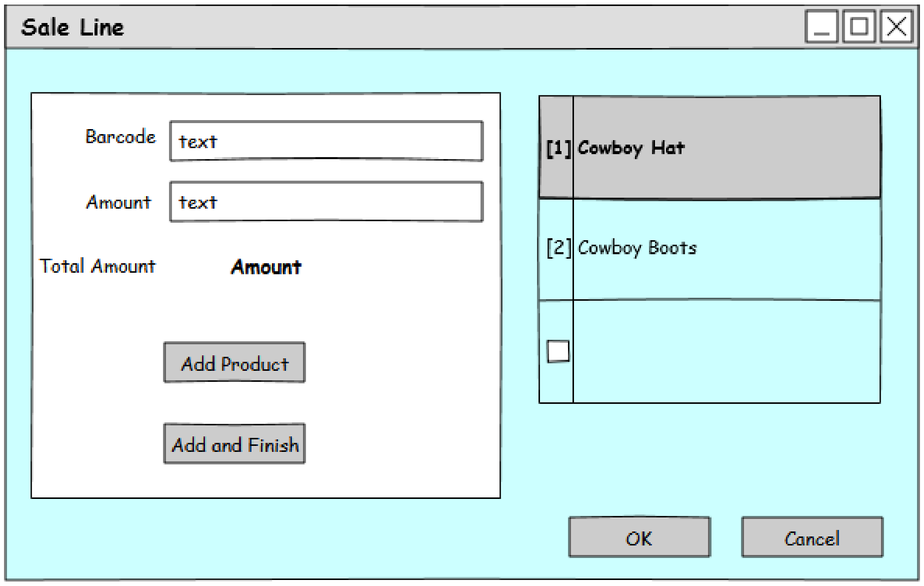









Create Sale
Customer CPR
text
Check
Delivery
Date
text
Zip Code
text
Street
text
Address
text
Payment Status
Paid
Not Paid
Add Products
Finish Sale
OK
Cancel


Sale Line
Barcode
text
Amount
text
Total Amount
Amount
Add Product
Add and Finish
[1] Cowboy Hat
[2] Cowboy Boots
OK
Cancel