Software Design F28SD

Kerr and Brown – Shoe Shop

An Exercise in Systems Design

Halima Ware : H00178418
Cassandra Ann Fernandes : H00200702

Contents

T1.1: Requirements

ID	Details	Type	Priority
R1	**The system shall display a list of all products offered by the company**	• **Products** • **Functional**	**Must Have**
R2	**The system shall organize the list of products by shoes or belts**	• **Products** • **Functional**	**Must Have**
R3	**The system shall display detailed product description with name, picture and price on click of quick view.**	• **Product** • **User Interface** • **Functional**	**Must Have**
R4	**The system shall display the number of products in the cart in each page of the catalogue**	• **Product** • **User Interface** • **Functional**	**Should Have**
R5	**The system shall have a separate page for return/repair orders with name, receipt and product name.**	• **Product** • **Order** • **Functional**	**Must Have**
R6	**The system should be able to search through all of the companies products by name, type and price.**	• **Product** • **User Interface** • **Functional**	**Must Have**

R7	**The system shall send an alert to admin when stock is over/insufficient**	• **Product** • **Order** • **Functional**	**Must Have**
R8	**The system shall redirect the customer to registration page if not logged in**	• **User Interface** • **Functional**	**Should Have**
R9	**The system will validate orders with 2 products minimum before purchase can be made.**	• **Orders** • **Functional**	**Must Have**
R10	**The system shall display a choice between shoes or belt at start**	• **Products** • **Functional**	**Should Have**
R11	**The system shall display customer details when checking out**	• **Payment** • **Functional**	**Must Have**
R12	**The fixed rate of 25 shall be added to checkout as shipping expenses by the system**	• **Payment** • **Functional**	**Must Have**
R13	**The system can add multiple orders of the same product**	• **Orders** • **Product** • **Functional**	**Must Have**
R15	**The system shall deduct 10% from the first order placed by the customer**	• **Payment** • **Functional**	**Must Have**
R15	**The system shall display a newsletter sign up at the end of**	• **User Interface** • **Functional**	**Must Have**

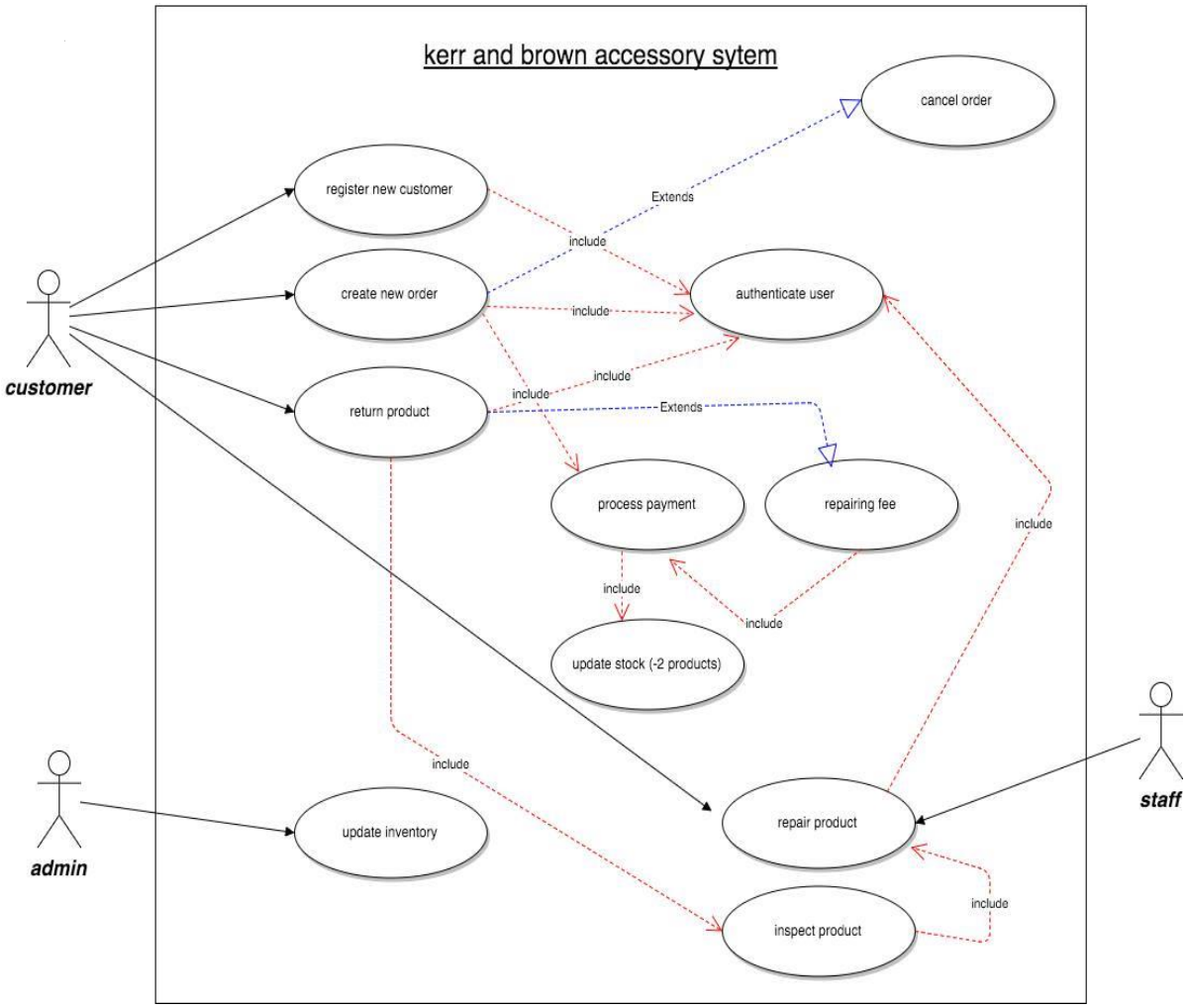
	a purchase		
R16	**The system shall display multiple addresses depending on the customer i.e home, office etc.**	• **Orders** • **Functional**	**Must Have**
R17	**The system shall display products according to type and season**	• **Products** • **Functional**	**Could Have**
R18	**Orders will be displayed according to time ordered, estimated waiting time and if completed**	• **Orders** • **Functional**	**Should Have**
R19	**The system will display a list of products to be updated with relevant information such as stock**	• **Products** • **Functional**	**Must Have**

Non Functional Requirements

ID	Details	Type	Priority
R20	**The system shall use a browser as it's user interface**	• **Performance** • **Non-Functional**	**Must Have**
R21	**The system shall authenticate all users browsing the site who are not customers**	• **Security** • **Non Functional**	**Must Have**
R22	**The system shall log in a customer within 3 seconds**	• **Performance** • **Non-Functional**	**Should Have**
R23	**The system shall store sales transaction data**	• **Availability** • **Non-Functional**	**Must Have**
R24	**The system shall be available 24 hours, 365 days per year**	• **Availability** • **Non-Functional**	**Must Have**
R25	**The system shall support 100,000 transactions per day**	• **Capacity** • **Non-Functional**	**Must Have**
R26	**The system shall ensure the customer does not have more than 1 account**	• **Security** • **Availability** • **Non-Functional**	**Must Have**
R27	**The system shall support a peak of transaction rate of 10 transactions**	• **Capacity** • **Non-Functional**	**Should Have**

	per second		
R28	**The system shall display different window for return-repair orders**	• **Performance** • **Non-Functional**	**Should Have**

T1.2 : Use Case Diagram : Kerr And Brown Accessory System



Textual description for base Use case : ReturnProduct

Use case: ReturnProduct

ID: 1

Brief description: Customer returns product

Primary actors: Customer

Secondary actors:

Preconditions :

1. The user is logged into the system
2. Customer should have a valid order number.

Main Flow:

1.Include authenticate user.
2.The system displays the user's details including the product bought
3. Include inspect item

Extension point : Repairing Fee

4. Product is returned.

Postconditions :

1. Customer is refunded

Alternative flows: none.

Textual description for base Use case : Create new Order

Use case: Create new Order

ID: 2

Brief description: Customer places new order for products

Primary actors: Customer

Secondary actors: Admin

Preconditions :

1. The user is logged into the system
2. Customer should be registered.
3. Products should have sufficient stock
4. Cart should contain 2 items.

Main Flow:

1.Include authenticate user.
2.Display all products that met the search criteria.
3. Customer adds 2 products to the cart.
4. Include process payment.
5. Sends the customer the confirmation email with copy of receipt.

Postconditions :

1. Product is bought.
2. Customer and admin notified of purchase.

Alternative flows: none.

Textual description for base Use case : Update Inventory

Use case: Update Inventory

ID: 3

Brief description: Admin updates the inventory after purchase by customer

Primary actors: Admin

Secondary actors: Staff

Preconditions :
1. The admin is logged into the system

Main Flow:
1.The system notifies the admin about the insufficient stock
2.Display all products without stock.
3. Admin sends the list to staff to prepare the stock.

Postconditions :
1. Product stock is updated.

Alternative flows: none.

Extension Use case: Repairing Fee

ID: E1

Brief description: Product is repaired

Primary actors: Customer, staff

Secondary actors: none

Segment 1 Preconditions :

1. Product is damaged

Segment 1 Flow:

1.Damage is priced.
2. Bill is calculated and added to customers account.
3. Customer pays the fee.
4. Product is returned.

Postconditions :

1. The product is repaired.

Alternative flows: none.

Extension Use case: Cancel Order

ID: E2

Brief description: Order is cancelled

Primary actors: Customer

Secondary actors: none

Segment 1 Preconditions :

1. Customer needs to be logged in.
2. Product should have valid order number.
3. The cancel button should be clicked by the user.

Segment 1 Flow:

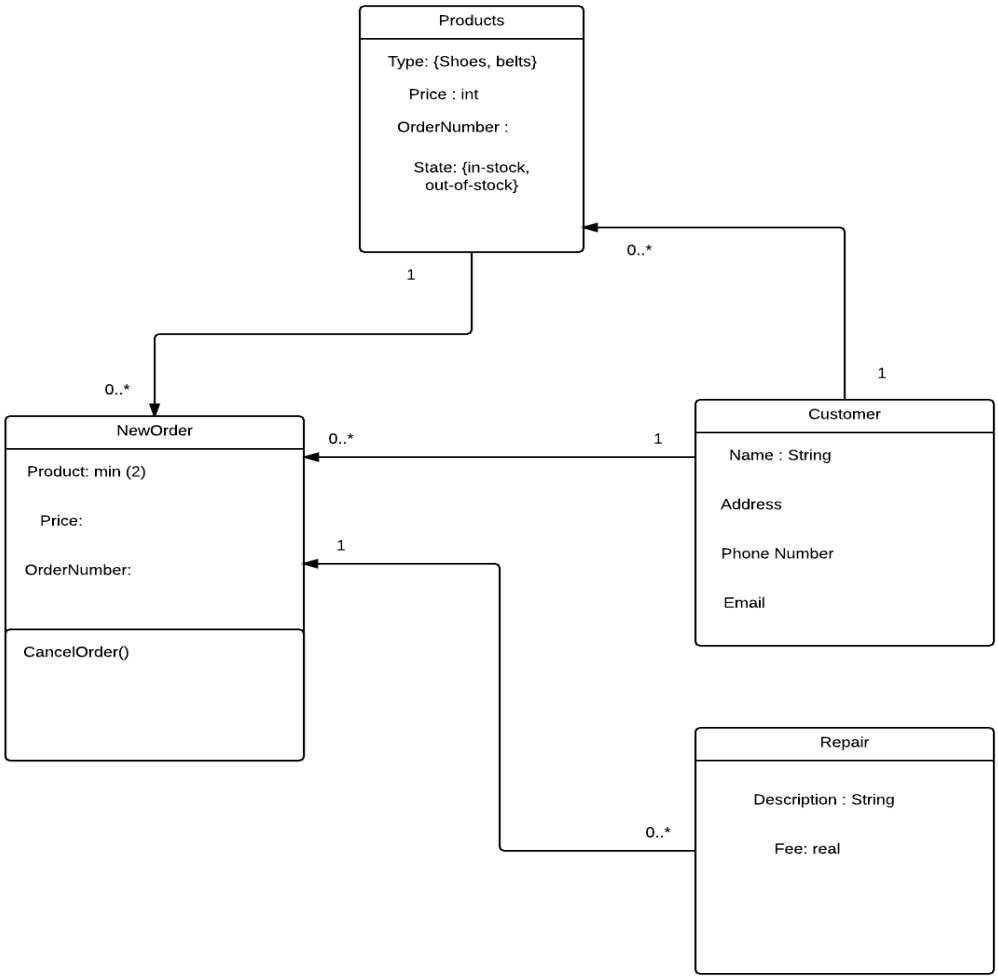
1. Remove the product from the shopping basket.
2. Product is added back into the stock.

Postconditions :

1. The change has been recorded in the system.
2. Customer has been refunded the payment.

Alternative flows: none.

T2. : Class Diagram: Create New Product

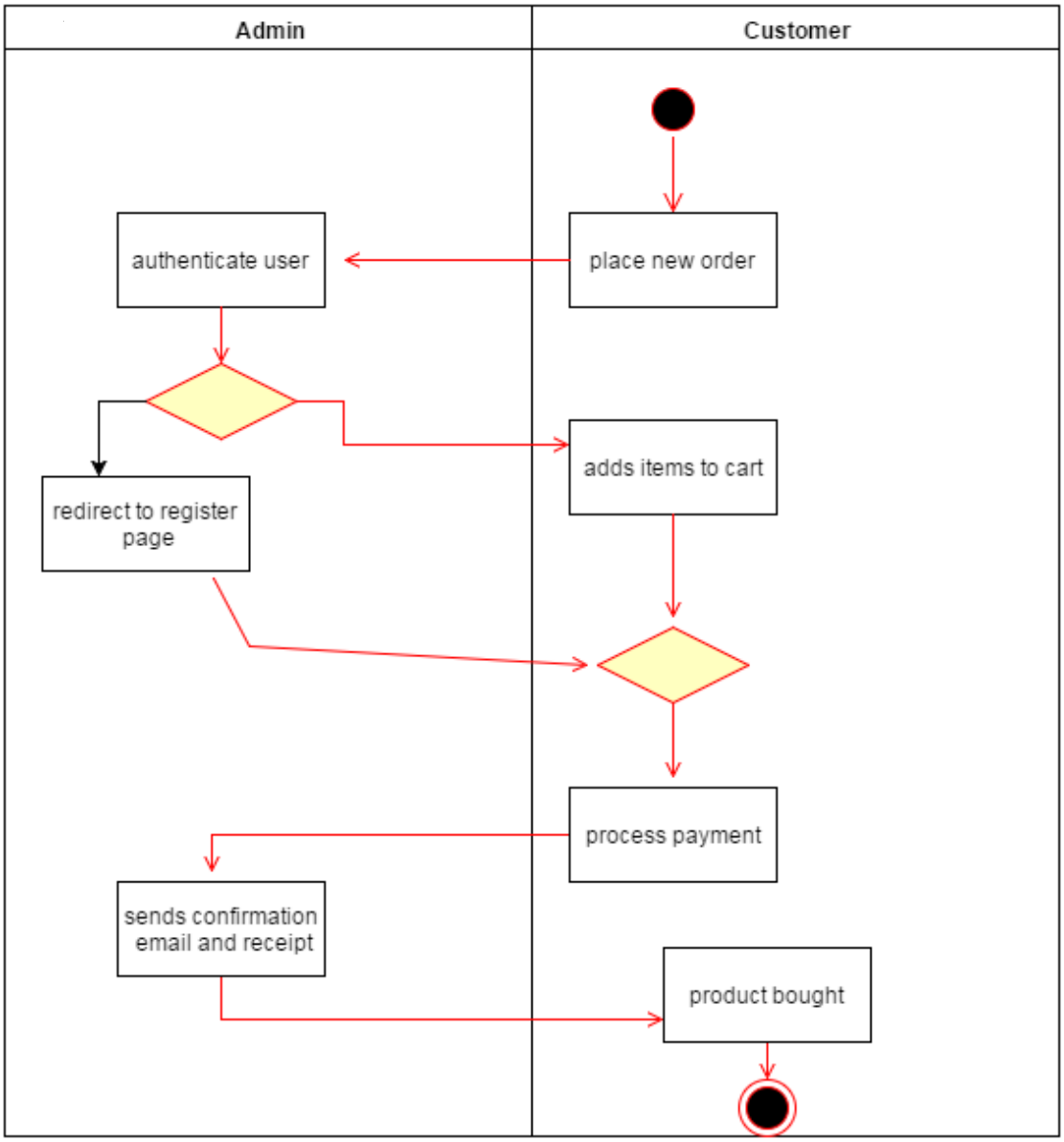


D1. Kerr and Brown Shoe Shop

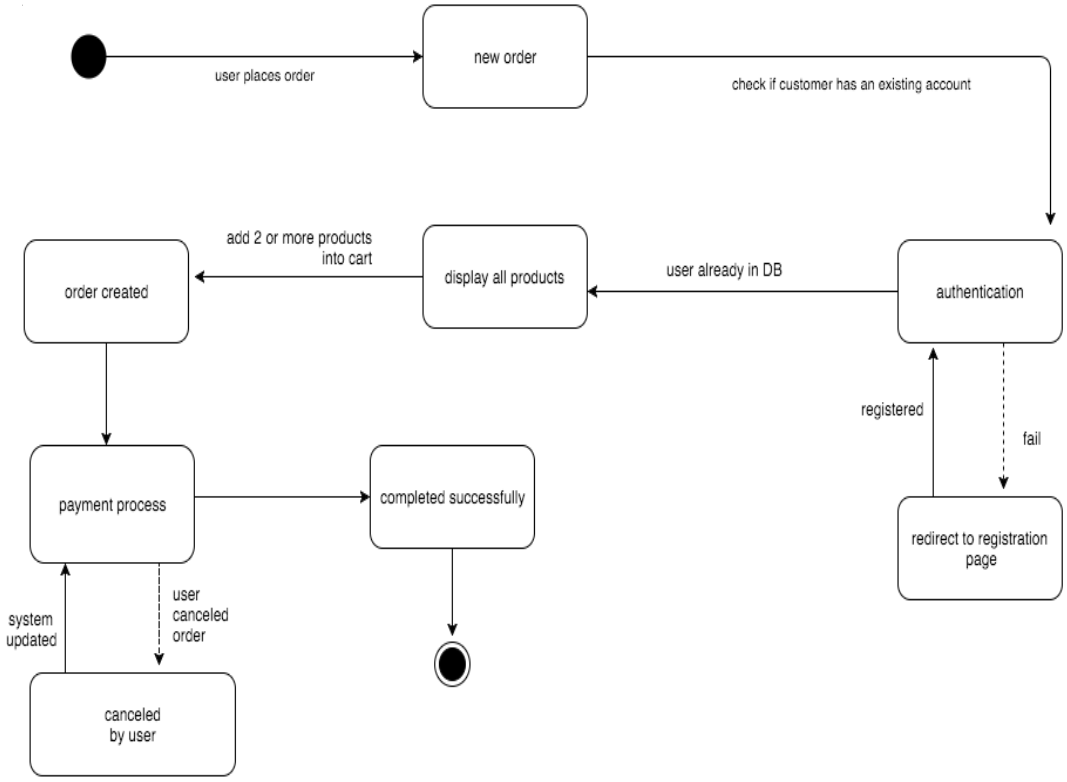
1. **Customers, staff and admins represent the actors.**
2. **Products would be out class, and a simple type attribute is associated with it, i.e. shoes, belts. A product can either be in stock or out-of-stock this suggest a status attribute for the products class**
3. **Given that a product may require repairs i.e the customers can return a product and can be either refunded or their product can be repaired for a fee. Hence the repair class.**
4. **Customers need to be registered and so we need a customers class, where each customer has a unique instance of the customer class. The class contains basic info of the customer.**

D2:

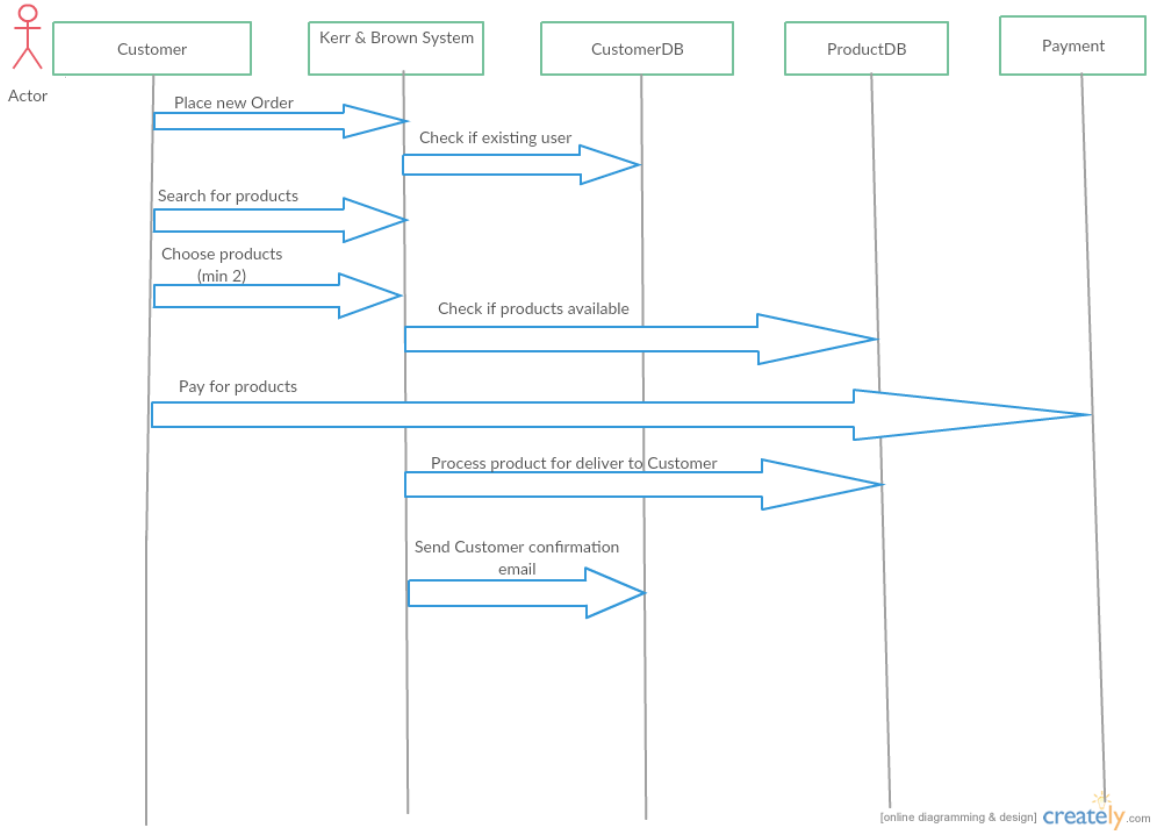
T3.1: Activity Diagram : Create New Product



T3.3 : State Machine Diagram : Create New Product



T3.2 : Sequence Diagram : Create New Product



D3. : Strengths and weaknesses

With the use of UML in this coursework, we have understood the idea of visualising a design for better communication and understanding. We aimed for a well-designed, robust and maintainable object oriented system which meets the users' requirements.

With the notations, use case and textual descriptions we are able to map out the requirements of the system well before actual implementation is thought of.

The strength with this is we could figure out what we might be doing wrong at an early stage and divide the work in stages.

A disadvantage we faced on this was that we could not really proceed to the next step without the completion of this stage.
After clearly defining our system boundary and its requirements, we could move on to the next step.

With the class diagram, we could easily see what methods and functions we needed because we had already mapped out the requirements earlier.

After that, the entire task became relatively easier as we just had to create the activity, sequence and state diagrams, with the use of our use cases, where we basically drew out our system in processes. This helped tremendously as it helped us see visually how our system would work.