# Project Deliverable 2 – Documentation and Coding

| | |
|---|---|
| **Faculty Name:** | Information Technology |
| **Module Code:** | ITECA3-B12 |
| **Module Name:** | Web Development and e-Commerce |
| **Content Writer:** | Sandile Dlamini |
| **Copy Editor:** | Mr Kyle Keens |
| **Submission Date:** | Block 2 Week 4 |

| | |
|---|---|
| **Student Name** | Otsile Modiselle |
| **Student Number** | HQ9SJ1J57 |
| **Project Title** | Bride & Joy Deliverable 2 |
| **Submission Date** | 10 June 2024 |

# Table of Contents

## 2.1 Introduction

Three months ago, the team undertook a task from client Sandy Hurynarin to build an eCommerce website for their wedding dress shop, Bride & Joy. The site had to allow for browsing a catalog of garments across different categories and dress sizes, adding products to wish lists and carts, and placing orders for payment. Additionally, Sandy wanted to showcase trustworthiness by enabling easy returns logging through the site. These and other requirements were thoroughly documented in the first deliverable.

This second installment details the software development lifecycle journey to transform the requirements into a functioning program. Having investigated and analyzed the requirements in Deliverable 1, the process here begins with the design phase. Several UML diagrams, including a Use Case diagram, Context diagram, and Enhanced Entity Relationship diagram, were drafted.

Construction of the database and prototypes followed. The team adopted a different understanding of prototyping during the project. Instead of creating two functional sites with divergent interfaces, they created non-functional and functional prototypes. Static HTML pages were manually written for each relevant section of the website as a non-functional prototype. Early iterations included slight styling variations. Once completed, back-end scripting was written, decomposing, reshuffling, and converting the HTML into PHP for a functional prototype.

With PHP bringing life to the locally hosted static pages, the database was integrated and tested for bugs. Further database normalization was required, along with creating numerous support tables to achieve some necessary functionality.

After satisfactory testing, site integration proceeded. Multiple unsuccessful attempts were made at affordable cloud solutions hosting, with the technical overhead proving impractical. The team eventually settled on a regular web hosting vendor, 00Webhost, achieving the desired results. The client was happy with the outcome, signing off on the project and eager to work on the site's continued maintenance.

## 2.2 Prototyping

Despite completing the prescribed textbook cover to cover, the content writer harbored doubts in their abilities as a novice web designer. Hence, additional training was sought through a 150-video lecture Udemy course explaining the intricacies of web design. In the interest of time, mainly the early lessons were covered to seek out important techniques necessary throughout the project. This included a thorough explainer of the box model and how to align elements side by side using the display: float CSS property. With these in hand, the team set out to complete the project on time, hoping to return to the course later for a broader retrospective.

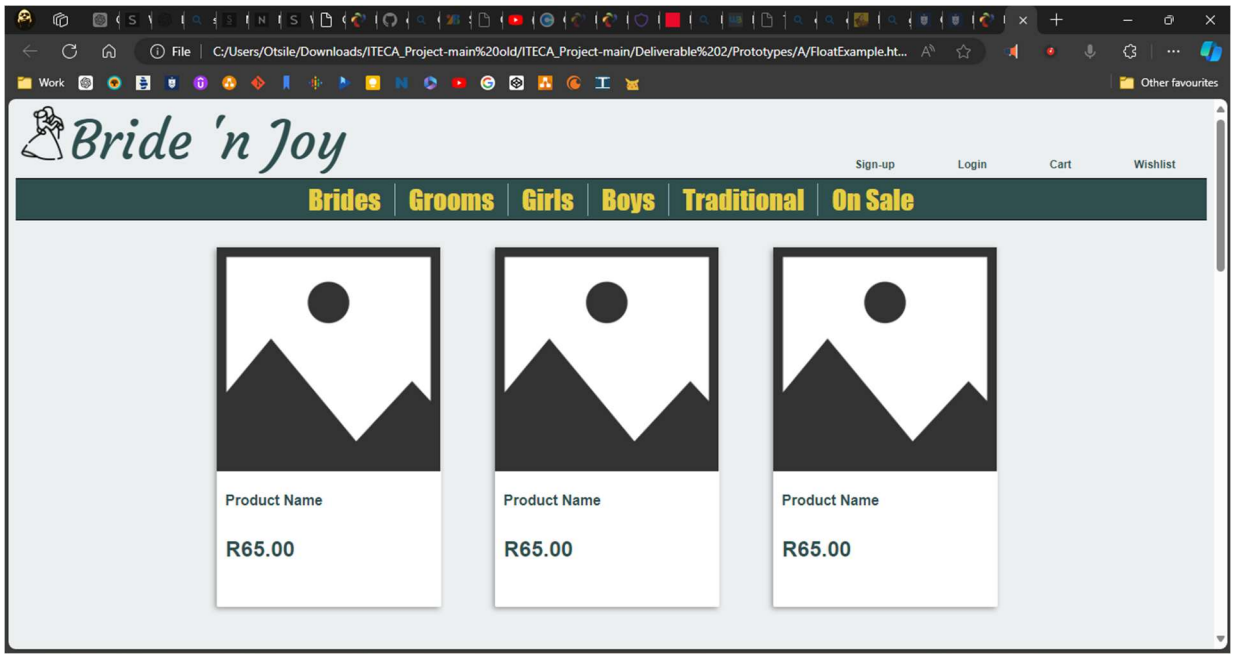
By the time the prototype was ready and hosted, mere weeks from the deadline (which was pushed a few times), a sense of satisfaction and fatigue had set in. The project ticked most of the boxes while leaving a few out. Upon returning to the Udemy course, more modern and advanced properties for aligning elements were introduced, such as flexbox and CSS grid. These were easier and more useful to implement than the outdated float technique used initially.

Despite all the planning and sequential working, the team adopted an agile mindset by prioritizing code and a working prototype by the deadline. This approach, brought on by the lack of knowledge and anxiety, inadvertently introduced technical debt that now needed refactoring.

Upon learning about responsive prototypes, the team pushed through the Udemy course to cover media queries. Media queries work with flexbox and CSS grid to make sites more responsive to the viewport. They involve customizing CSS properties for different viewport sizes, such as smartphones, tablets, and large screens. By the time this knowledge was fully understood, reworking the “finished project” from the start proved too daunting. Instead, efforts were made to salvage some elements of the site, the database, and content like the carousel and images, into a third working prototype meeting all the requirements of responsiveness. Even now, prototype 3 remains unfinished, but there is an intent to complete it for portfolio experience purposes.

### a. Prototype 1.

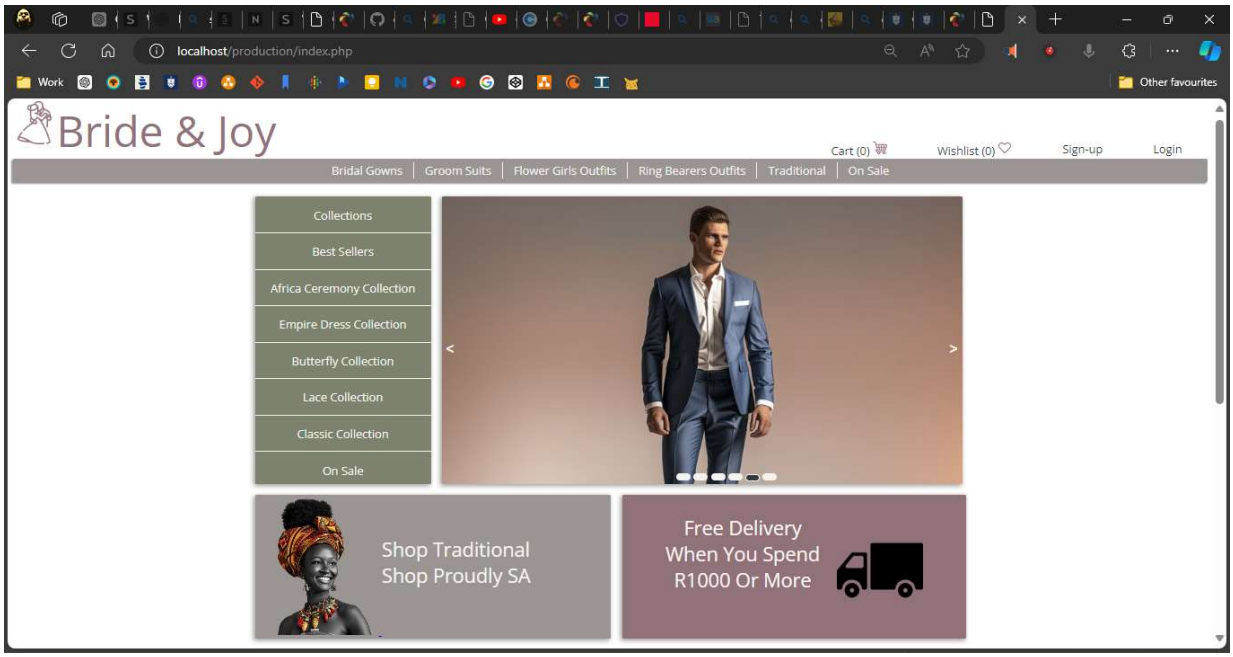
As explained in the introduction, the initial understanding of prototyping did not align with the stipulated answer sheet template. Static HTML pages for the Bride & Joy site were mocked up as non-functional prototypes. Figure 1 shows one of the first iterations. Over time, understanding of color theory and website personalities advanced beyond the initial curved font logo design.

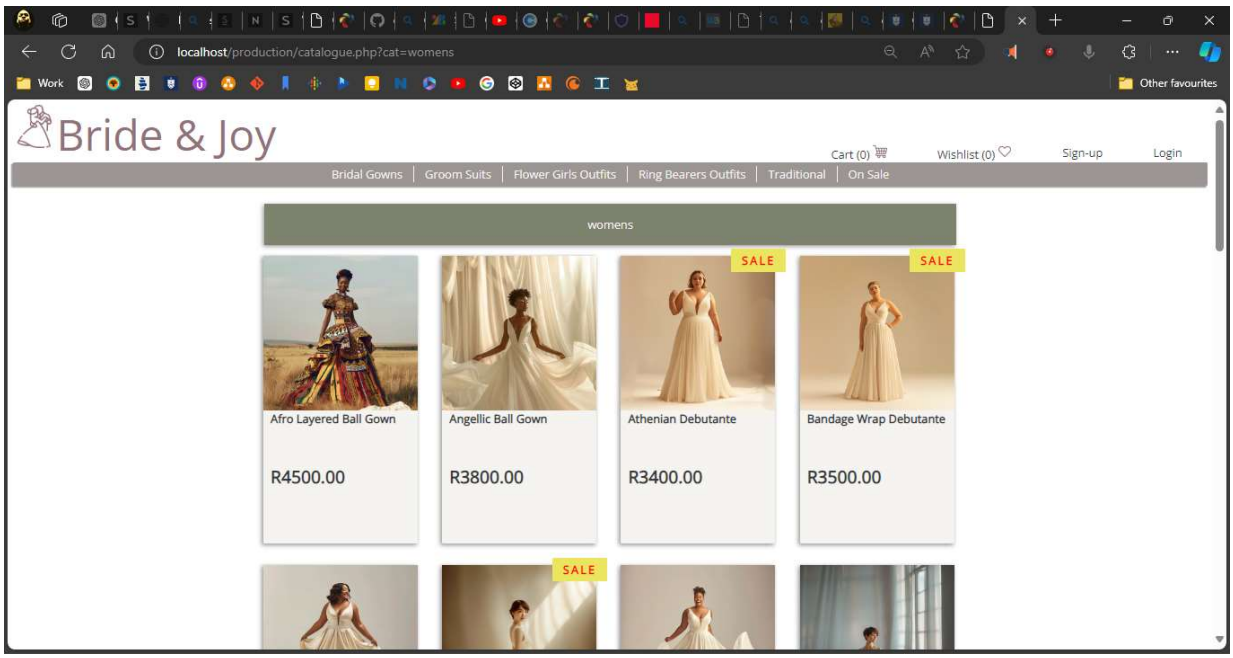



Lack of color theory knowledge led to the belief that all shades of green symbolized growth and new life related to weddings. Instead, the dark shade used proved unsuitable. A bold and clear font was chosen, in the Compact font family, but this too was abandoned.

At this stage, a clear idea of an index page had not been settled on. The team knew that an eCommerce site must sell goods and prepared a catalog page that left much to be desired. It was a valiant effort for first-time builders with no idea of how to place two div elements side by side. The float technique was used to place three product cards in a row, a method used throughout the final deliverable at the expense of responsiveness. Unfinished prototype 3 works to ameliorate this.

**Prototype 2.**



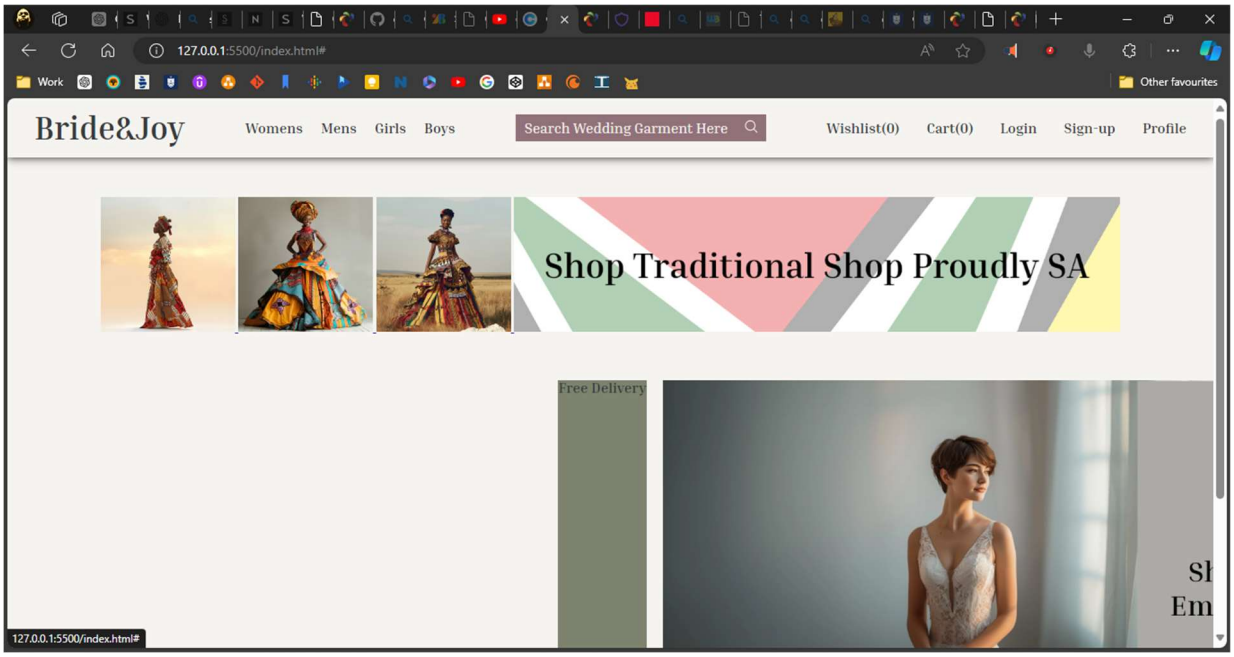




The working prototype screenshots show significant design progress. A more muted pastel color palette and simpler Open Sans font family were chosen for consistency. Images and product information were pulled from the database, and a suitable index page was created.

The web design course taught the team to incorporate a working product carousel (or image slider) powered by JavaScript, as is characteristic of online clothing retailers. The float

technique was used to place index page blocks side by side and many other elements throughout the project. Again, this was to the detriment of responsiveness, which would have been better served by flexbox and CSS Grid. Refactoring the existing project's CSS to include media queries for each page would have been a mammoth task, necessitating a total rebuild that is currently under construction.

The third prototype below achieved a more modern-looking navigation panel, justifying a logo, links, a search input box, button, and more links side by side using flexbox. Root em (rem) units were used to size elements in CSS instead of pixels (px) for better scalability and responsiveness.




## 2.3 Designing

Please refer to the appendix for all diagrams listed and discussed below.

**a. Class Responsibility Collaborator (CRC) cards**

CRC cards were created for classes: customer, product, order, courier, collection, and return. While the database was to incorporate many more objects, the list was kept short for brevity and similar objects were generalized where possible.

A brief explanation of CRC card 5 for collections follows. The collections class generalizes the lists customers can make in the form of a cart list or a wish list of products. It is responsible for organizing user desires with the eventual goal of populating a shopping order, standard fare for an eCommerce site. The class collaborates with customer, order, and product classes among others. The collection class is an aggregation (make up) of wishlist and cart subclasses.

**b. Enhanced Entity Relationship Diagram (EERD)**

The EERD maps out the database design for Bride & Joy as a business. At the top is the stakeholders supergroup, divided into juristic (or legal) stakeholders and non-juristic stakeholders, namely customers. The direct supergroup of non-juristic stakeholders was omitted for brevity, but this could have also included employees together with customers. The double line from a superclass entity leading to the encircled ‘d’ symbols denotes the disjoint constraint, meaning that each member of the stakeholders superclass can only be a member of one subclass. All subclass members inherit the attributes and relationships of their superclass entities. The U symbols leading to the customer and juristic entities indicate that they are subsets of the stakeholders superclass entity.

In the remainder of the diagram, a customer can curate collections of both or either wishlists and carts. Wishlists and stock records detail listed products sold by Bride & Joy. Cart entities try to deplete available stock records by creating a workitem entity called an order (the stock is moved to the order entity). Workitems are created and fulfilled by suppliers and couriers. The stock entity is subdivided into subclass entities such as mens, womens, etc.

**c. Context Diagram**

The context diagram depicts the general functioning of the eCommerce website. At its simplest, the site conveys information about available products to customers who populate shopping carts for purchase. Additional functionality such as sending product queries to the site admin and store owner was originally envisioned. Customers provide their delivery details on the site before receiving an order confirmation.

The site admin was to be able to receive product queries through the site and attend to them, but this has not been implemented. Through the site's backend, the admin can monitor inventory levels and make relevant adjustments to the database stock table.

The site's backend was envisioned with a relay mechanism to couriers for automated creation of order delivery requests. However, while the project is a working prototype, the necessary advanced API technologies have yet to be implemented.

**d. Data Flow Diagram (DFD)**

The DFD depicts eleven processes stemming from either a customer or site admin/owner interaction. Customer login or sign-up data will interplay with user data on the database. Normal eCommerce interactions (processes), such as browsing product catalogs, adding items to lists, making a purchase or return, etc., will trigger necessary data interplaying with the inventory data (product and stock tables) of the database. Couriers would eventually receive information from the backend about required order delivery details. The owner will update the database tables with stock updates.

**e. Use Case Diagram**

The use case diagram portrays all the ways various users will interact with the site. Customers are distinguished as either registered or unregistered. All customers can browse for products. Unregistered users can attempt to log in before being prompted to sign up. Additionally, they can add items to a Wishlist as session-cached information, but they will need to sign up to save the Wishlist or add items to the cart and check out.

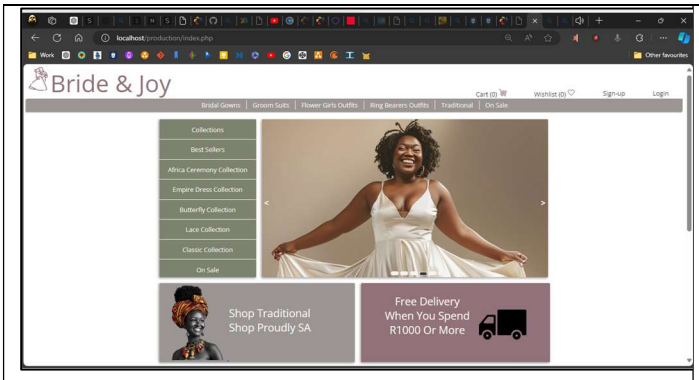
Once signed up and logged in, functionality extends to finalizing purchases, arranging deliveries, (and eventually reviewing products and logging returns). The owner will be able to monitor the backend database to manage stock levels.

**f. Database Design**

Please refer to appendix.

## 2.4 Coding

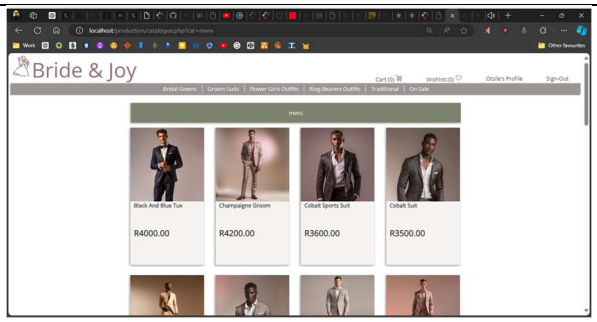
### a. Screenshots



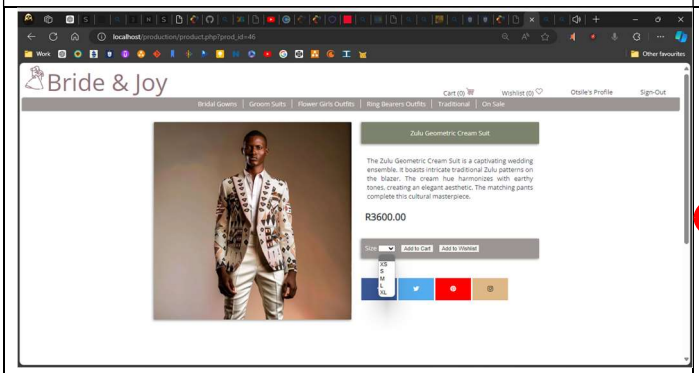

1. Arrive at index page and either browse or login.


2. Input login credentials.


3. Receive login confirmation by name in the navigation bar.



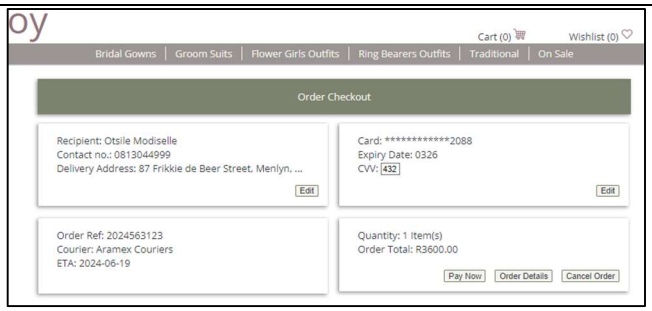

4. Browse products.




5. Click a product to review information and select size before the add to cart button is clickable.

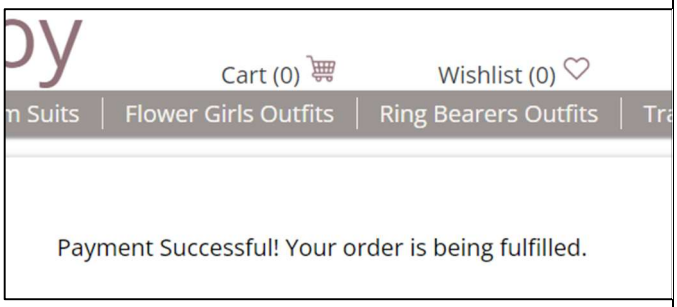

6. Receive add to cart confirmation in the navigation indicator (notice the icon changes from an empty cart to a full one).


7. Click on the cart navigation menu item to view the list of items added to the cart. Items can be deleted, or the total can be reviewed before clicking checkout.




8. Arrive at the order checkout screen and review information. New users will be prompted to update information before the pay now button is clickable. Otherwise, they can review details, cancel the order, or pay now.


9. If customers choose to edit information, JavaScript audits inputs before they can be saved.




10. Upon successful payment, an appropriate message is displayed, the database stock table is updated, and the navigation cart count updates.

## b. Sample PHP Code

```
35          if (password_verify(str_replace("'", "", $inputPassword), $retrieved_pw))
36          {
37            session_start();
38
39            $_SESSION['customer_id'] = $retrieved_customer_id;
40            $_SESSION['forename'] = $retrieved_name;
41            $_SESSION['surname'] = $retrieved_surname;
42
43            if (isset($_SESSION['cart']))
44            {
45              foreach($_SESSION['cart'] as $temp_cart_item)
46              {
47                  $query = "SELECT prod_id
48                            FROM stock
49                            WHERE stock_id = $temp_cart_item;";
50
51                  $result = queryMysql($query);
52
53                  if ($result->rowCount())
54                  {
55                      $row = $result->fetch();
56                      $prod_id = $row['prod_id'];
57
58                      add_collection($pdo, $retrieved_customer_id, $prod_id);
59                      $coll_id = latestPrimaryKey();
60                      add_cart($pdo, $coll_id, $temp_cart_item);
61                  }
```

Above is a snippet of code from script login.php. The Bride & Joy site emulates the convenience of most modern sites in that it caters for customers browsing and adding items to a temporary cached list, be it wishlist or cart. Once a user logs in, the site checks whether their email address exists in our stakeholders table. If email does not exist, they are prompted to go sign up. If it does exist, their customer ID is retrieved and used to further retrieve their stored password hash into variable. retrieved_pw. All this happens in lines 1-34.

The code snippet details the password checking via the password_verify inbuilt PHP function. A successful verification returns true, triggering the start of a session. User data is populated for page to page carrying, see lines 37-41. The temporary session cart is then emptied to the database should it have items for, see lines 42-61.

## c. Sample HTML Code

```
41      echo "<a href=\"product.php?prod_id=$cat_prod_id\">";
42
43      $queryOnSale = "SELECT prod_id FROM sales WHERE prod_id = $cat_prod_id;";
44      $saleResult = queryMysql($queryOnSale);
45      if($saleResult->rowCount())
46      {
47        echo"<div class='card on-sale'>";
48      }
49      else
50      {
51        echo"<div class='card'>";
52      }
53
54        echo <<<_END
55            <img src="images/img/$cat_prod_img" alt="" class="thumbnail">
56            <p class="prod-name card-detail"><strong>$cat_prod_name</strong></p>
57            <p class="price card-detail"><strong>R$cat_price</strong></p>
58        </div>
59      </a>
60      _END;
61    }
```

We employed query strings to pass along user browsing intentions. So, when they click on the *Groom Suits* catalogue link in the nav to view suits, for example, the href would appear as follows:

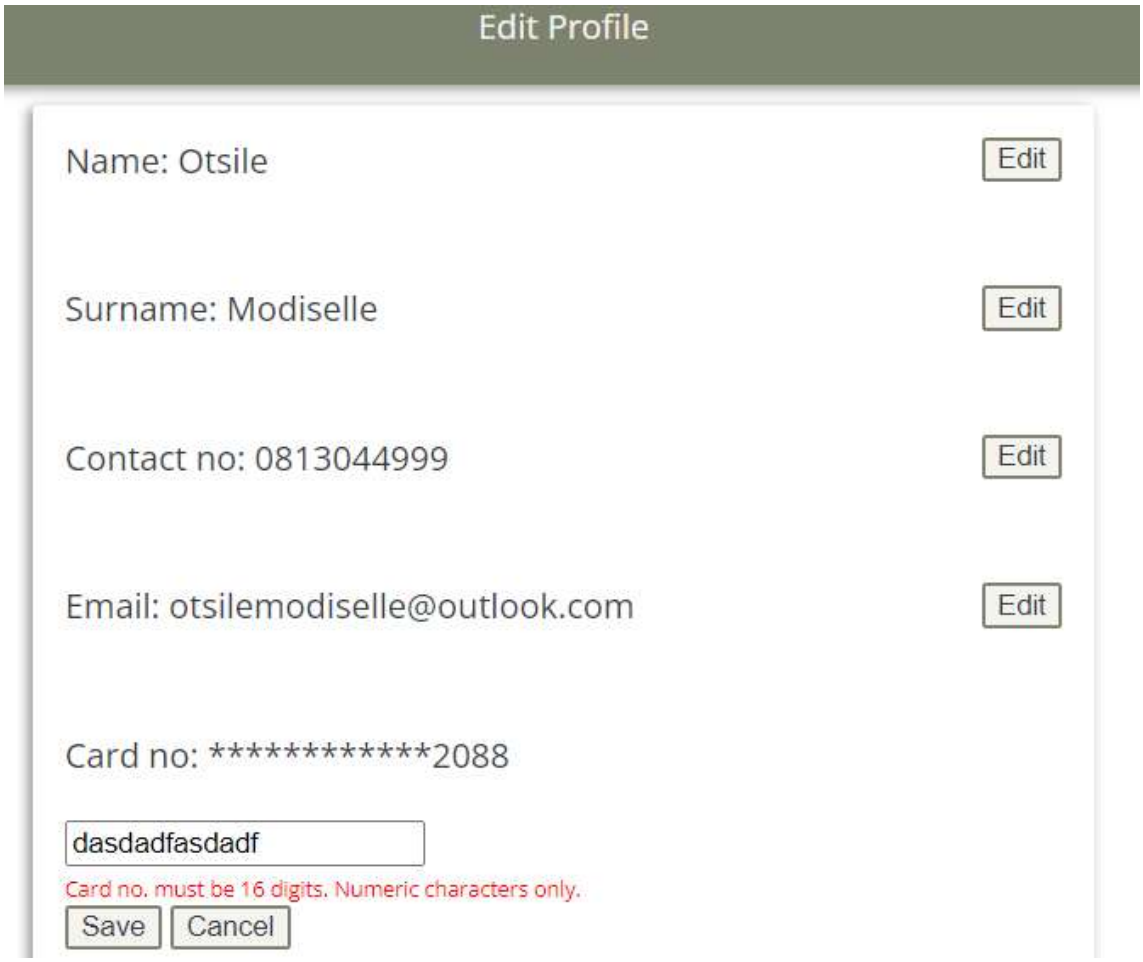
catalogue.php?catalogue=mens

The *mens* at the end (query string) would trigger a query to the database to fetch product information from the mens table. That information would then be held in variables $cat_prod_id, for the product ID, $cat_prod_img, for the product image, $cat_prod_name, for its name, $cat_prod_price, for its price. A loop fetches and stores this data for us to manipulate. All this happens in lines 1-42 of file catalogue.php.

PHP's powerful DOM manipulation through careful planning and outputting of HTML is on display in the above code snippet. Line 41 outputs an opening anchor tag that will make our product card clickable as a link. Lines 43-48 check the product ID in the fetch loop against products listed in the sale table. If the current product being iterated on is found in the sale table, we echo a product card (div element) with a class of "on-sale", see line 47. If it is not found, we echo a product card without such a class, see line 51.

This "on-sale" class would prove useful when highlighting products on sale on the catalogue page through CSS below. Lines 54-59 output the remaining HTML cleverly interspersed with key data from the database and necessary for producing the rest of the product card.

## d. Sample JavaScript Code

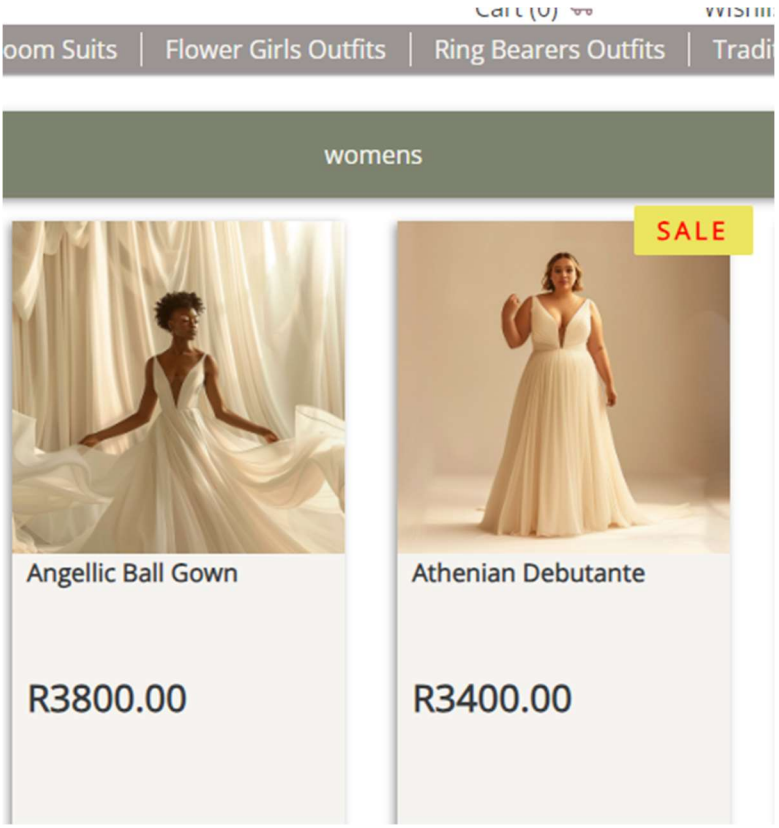



```
265  function editCard()
266  {
267      var card = document.getElementsByClassName("edit-card-input")[0].value;
268      var cardFeedback ="";
269      var numerical = /^[\d/]+$/;
270
271      if (card.length < 16){
272          cardFeedback = "Card no. must be 16 digits. ";}
273      if (!numerical.test(card)){
274          cardFeedback += "\n Numeric characters only."}
275      document.getElementsByClassName("edit-card-verify")[0].textContent = cardFeedback;
276      return cardFeedback;
277  }
```

Multiple plain JavaScript files were written with hundreds of lines of code. One intricate file helped operate the looped carousel of image slides on the index page. The other two almost exclusively worked to verify user inputs and provide them with relevant feedback.

The code snippet features a function that checks the card number input per the above screenshot. Line 265 fetches the value and 269 creates a regular expression of digits only. Lines 271-274 check for the input value's length (element accepts max length 16) and format (against the regex), and 275 returns the feedback.

## e. Sample CSS Code




```
1090  .on-sale::after {
1091      content: "SALE";
1092      color: red;
1093      background-color: rgb(236, 229, 95);
1094      font-family: 'Open Sans', sans-serif;
1095      font-weight: bold;
1096      position: absolute;
1097      top: -10px;
1098      right: -15px;
1099      font-size: 16px;
1100      padding: 5px 15px;
1101      letter-spacing: 2pt;
1102      border-top-left-radius: 2px;
1103      border-top-right-radius: 2px;
1104      border-bottom-left-radius: 2px;
1105      border-bottom-right-radius: 2px;
1106  }
```

Further to the discussion of creating custom classes on the fly via the HTML code snippet in question .c, we use that class to highlight products cards with a yellow sale sticker seen in the screenshot above. The selector .on-sale::after, is a pseudo element, meaning that it does not exist in our HTML. We supply it's content via the appropriate property, line 1091, and further style it with the remaining lines.

## f. Sample MySQL Table Screenshots

```
mysql> SHOW TABLES;
+-------------------------+
| Tables_in_bride_and_joy |
+-------------------------+
| boys                    |
| cart                    |
| collection              |
| courier                 |
| customer                |
| girls                   |
| juristic                |
| mens                    |
| order_                  |
| product                 |
| restocking              |
| return_                 |
| sales                   |
| stakeholders            |
| stock                   |
| supplier                |
| traditional             |
| wishlist                |
| womens                  |
| workitem                |
+-------------------------+
20 rows in set (0.00 sec)
```

```
mysql> SELECT prod_id, supplier_id, prod_img, prod_name, price
    -> FROM product;
+---------+-------------+-----------------------------------------+-----------------------------------+---------+
| prod_id | supplier_id | prod_img                                | prod_name                         | price   |
+---------+-------------+-----------------------------------------+-----------------------------------+---------+
|       1 |         323 | afro_layered_ball_gown.webp             | Afro Layered Ball Gown            | 4500.00 |
|       2 |         322 | angellic_ball_gown.webp                 | Angellic Ball Gown                | 3800.00 |
|       3 |         324 | athenian_debutante.webp                 | Athenian Debutante                | 3400.00 |
|       4 |         324 | bandage_wrap_debutante.webp             | Bandage Wrap Debutante            | 3500.00 |
|       5 |         323 | bantu_ballerina_flower_girl.webp        | Bantu Ballerina Flower Girl       |  600.00 |
|       6 |         325 | black_and_blue_tux.webp                 | Black And Blue Tux                | 4000.00 |
|       7 |         322 | champaigne_monroe.webp                  | Champaigne Monroe                 | 3800.00 |
|       8 |         325 | champaigne_groom.webp                   | Champaigne Groom                  | 4200.00 |
|       9 |         324 | classic_laced_flower_girl.webp          | Classic Laced Flower Girl         |  900.00 |
|      10 |         325 | cobalt_sports_suit.webp                 | Cobalt Sports Suit                | 3600.00 |
|      11 |         325 | cobalt_suit.webp                        | Cobalt Suit                       | 3500.00 |
|      12 |         324 | cross_hatch_lattice.webp                | Cross Hatch Lattice               | 3200.00 |
|      13 |         325 | double_breasted_tangerine_groom.webp    | Double Breasted Tangerine Groom   | 3800.00 |
|      14 |         322 | etched_velvet_ball_gown.webp            | Etched Velvet Ball Gown           | 4200.00 |
|      15 |         325 | executive_groom.webp                    | Executive Groom                   | 3900.00 |
|      16 |         322 | fabric_flower_girl.webp                 | Fabric Flower Girl                |  600.00 |
|      17 |         324 | fairy_debutante.webp                    | Fairy Debutante                   | 3800.00 |
|      18 |         322 | hybrid_velvet_ring_bearer.webp          | Hybrid Velvet Ring Bearer         |  900.00 |
|      19 |         324 | lateral_cascade_empire_dress.webp       | Lateral Cascade Empire Dress      | 3700.00 |
|      20 |         322 | miniature_black_bowler_hat_suit.webp    | Miniature Black Bowler Hat Suit   | 1200.00 |
|      21 |         322 | miniature_double_breasted_cream_suit.webp | Miniature Double Breasted Cream Suit | 1300.00 |
|      22 |         322 | miniature_navy_waistcoat_set.webp       | Miniature Navy Waistcoat Set      | 1000.00 |
|      23 |         322 | miniature_sabbath_suit.webp             | Miniature Sabbath Suit            | 1100.00 |
|      24 |         322 | miniature_silver_suit.webp              | Miniature Silver Suit             | 1300.00 |
|      25 |         322 | miniature_tux.webp                      | Miniature Tux                     | 1500.00 |
|      26 |         322 | muted_butterfly_ball_gown.webp          | Muted Butterfly Ball Gown         | 4300.00 |
|      27 |         323 | navy_traditional_flower_girl.webp       | Navy Traditional Flower Girl      | 1200.00 |
|      28 |         324 | ornate_empire_collared_lace.webp        | Ornate Empire Collared Lace       | 3500.00 |
|      29 |         324 | ornate_empire_traditional_lace.webp     | Ornate Empire Traditional Lace    | 3800.00 |
|      30 |         324 | ornate_empire_waist_fairy_sleeveless.webp | Ornate Empire Waist Fairy Sleeveless | 4000.00 |
|      31 |         325 | powder_silver_groom.webp                | Powder Silver Groom               | 3800.00 |
|      32 |         324 | princess_butterfly_ball_gown.webp       | Princess Butterfly Ball Gown      | 3900.00 |
|      33 |         325 | rose_gold_satin_suit.webp               | Rose Gold Satin Suit              | 3600.00 |
|      34 |         324 | royal_collared_flower_lace_empire.webp  | Royal Collared Flower Lace Empire | 3800.00 |
|      35 |         325 | silver_satin_suit.webp                  | Silver Satin Suit                 | 3700.00 |
|      36 |         325 | steel_blue_suit.webp                    | Steel Blue Suit                   | 3700.00 |
|      37 |         325 | tangerine_groom.webp                    | Tangerine Groom                   | 3700.00 |
|      38 |         322 | traditional_laced_flower_girl.webp      | Traditional Laced Flower Girl     | 1300.00 |
|      39 |         324 | traveler_empire_lace_dress.webp         | Traveler Empire Lace Dress        | 3400.00 |
|      40 |         322 | velvet_ballerina_bride.webp             | Velvet Ballerina Bride            | 3500.00 |
|      41 |         322 | velvet_monroe.webp                      | Velvet Monroe                     | 3500.00 |
|      42 |         325 | velvet_black_executive_groom.webp       | Velvet Black Executive Groom      | 3400.00 |
|      43 |         323 | zulu_geometric_coarse_cream_suit.webp   | Zulu Geometric Coarse Cream Suit  | 3700.00 |
|      44 |         323 | zulu_geometric_coarse_fawn_suit.webp    | Zulu Geometric Coarse Fawn Suit   | 3800.00 |
|      45 |         323 | zulu_geometric_coarse_latte_suit.webp   | Zulu Geometric Coarse Latte Suit  | 3800.00 |
|      46 |         323 | zulu_geometric_cream_suit.webp          | Zulu Geometric Cream Suit         | 3600.00 |
|      47 |         323 | zulu_golden_debutante.webp              | Zulu Golden Debutante             | 3600.00 |
|      48 |         323 | zulu_royal_ring_bearer.webp             | Zulu Royal Ring Bearer            | 1400.00 |
|      49 |         325 | steel_blue_sports_suit.webp             | Steel Blue Sports Suit            | 3700.00 |
+---------+-------------+-----------------------------------------+-----------------------------------+---------+
49 rows in set (0.00 sec)
```

## 2.5 Conclusion

Despite the challenges and requirements that had to be renegotiated and abandoned, Sandy, the product owner, found the current product satisfactory. It has been a long journey with immense learning. Much thanks is extended to the faculty for their support along the way.

## 2.6 Sign-off

| | |
|---|---|
| _____ | _____ |
| Sandy Hurynarin, Project Client | Otsile Modiselle, Project Manager |
| Date: _____ | Date: _____ |

## 2.7 Bibliography


1. Coronel, C., Morris, S. and Rob, P. (2013). *Database principles : fundamentals of design, implementation and management*. London: Course Technology.
2. Dennis, A., Barbara Haley Wixom, David Paul Tegarden and Seeman, E. (2015). System analysis & design : an object-oriented approach with UML. 5th ed. Hoboken, Nj: Wiley.
3. Nixon, R. (2021). *LEARNING PHP, MYSQL & JAVASCRIPT : a step-by-step guide to creating dynamic websites.* S.L.: O'reilly Media.
4. Powers, D. and Springerlink (Online Service (2019). PHP 7 Solutions : Dynamic Web Design Made Easy. Berkeley, Ca: Apress.
5. Ramez Elmasri and Navathe, S.B. (2016). Fundamentals of database systems. Hoboken, New Jersey: Pearson.
6. Schmedtmann, J. (2023). *Build Responsive Real-World Websites with HTML and CSS*. *Udemy*. Available at: https://www.udemy.com/course/design-and-develop-a-killer-website-with-html5-and-css3/?couponCode=KEEPLEARNING [Accessed 20 Mar. 2024].
7. Silberschatz, A., Korth, H.F. and S Sudarshan (2019). Database system concepts. New York, Ny: Mcgraw-Hill Education.
8. Sweat, J.E. (2005). *PHP architect's guide to PHP design patterns : [a practical approach to desing patterns for the PHP 4 and PHP 5 developer]*. Toronto: Marco Tabini & Associates.
9. Wellens, P. (2015). Practical web development : learn CSS, JavaScript, PHP, and more with this vital guide to modern web development. Birmingham, UK: Packt Publishing.
10. Welling, L. and Thomson, L. (2017). PHP and MySQL web development. Hoboken, Nj ; Boston ; Indianapolis ; San Francisco ; New York ; Toronto ; Montreal ; London ; Munich ; Paris ; Madrid ; Cape Town ; Sydney ; Tokyo ; Singapore ; Mexico City Addison-Wesley.

## 2.8 Appendix

Front:

| Class Name: | Customer | ID: | 1 | Type: | Concrete |
|---|---|---|---|---|---|
| Description | A person with an interest in Bride and Joy's products, so much so they have established an account to track and possibly buy some. | | | Associated Use Cases: | 14 |

| Responsibilities: | Collaborators |
|---|---|
| Browse catalogue | Product |
| Build product collections, wishlists, cart | Product |
| Make purchase orders | Order, Courier |
| Make product returns | Orders, Courier |
| Arrange for product movements | Courier |
| Maintain profile data | |

Back

Attributes:

| | |
|---|---|
| Name (Varchar) | Address (Varchar) |
| Surname (Varchar) | Payment Method (Varchar) |
| Email (email) | |
| Password (Varchar) | |
| Contact Number (Varchar) | |

Relationships:

Generalization (a-kind-of): Stakeholder

Aggregation (has-parts): Wishlist, Cart

Other Associations:

Front:

| Class Name: | Product | ID: | 2 | Type: | Concrete |
|---|---|---|---|---|---|
| Description | A singular product that can be viewed and eventually bought by a customer. | | | Associated Use Cases: | 10 |

| Responsibilities: | Collaborators |
|---|---|
| Displayed for browsing | Customer |
| Can be added to a collection, wishlists, carts | Customer, Wishlist, Cart |
| Added to an order | Customer, Cart |
| Helps summarize returns by customer | Customer |
| Helps summarize dispatch movements by courier | Customer, Courier |
| Maintain the details of | |

Back

Attributes:

| | |
|---|---|
| Product Code (Varchar) | Colour (Varchar) |
| Price (Decimal) | Quantity (Integer) |
| Dimensions (Varchar) | |
| Gender (Varchar) | |
| Matrerial (Varchar) | |

Relationships:

Generalization (a-kind-of): Product

Aggregation (has-parts): Supplier

Other Associations:

Front:

| Class Name: | Order | ID: | 3 | Type: | Concrete |
|---|---|---|---|---|---|
| Description | A compilation of order details. | | | Associated Use Cases: | 7 |

| Responsibilities: | Collaborators |
|---|---|
| Summarize items purchased | Customer, Product |
| Summarize delivery details | Customer, Courier |
| Validates return requests | Return |
| Summarize delivery charge | Courier |

Back

Attributes:
Order Total (Decimal)
Date (Datetime)
Order-Descripton (Varchar)
Order Reference (Varchar)

Relationships:
Generalization (a-kind-of): Work-Item

Aggregation (has-parts): Product, Courier

Other Associations:

Front:

| Class Name: | Courier | ID: | 4 | Type: | Concrete |
|---|---|---|---|---|---|
| Description | Legal entity/person entrusted with moving items to the necessary location. | | | Associated Use Cases: | 3 |

| Responsibilities: | Collaborators |
|---|---|
| Confirming receipt of delivery instructions | Customer |
| Confirming receipt of items for delivery | Product |
| Confiirming delivery charge | Order |
| Fulfills various work-items | Order, Return |

Back

Attributes:
Name (Varchar)
Cost (Decimal)
Lead Time (Integer)

Relationships:
Generalization (a-kind-of): Juristic person

Aggregation (has-parts): Banking Details, Trading Name

Other Associations:

Front:

| Class Name: | Collection | ID: | 5 | Type: | Abstract |
|---|---|---|---|---|---|
| Description | A customer assortment of items. Can be classified as either a wishlist or a cart list. | | | Associated Use Cases: | 4 |

| Responsibilities: | Collaborators |
|---|---|
| Organize customer desires while browsing | Customer, Products |
| Populate an order | Order |
| Instantiate as either a wishlist concrete subclass object, or | Products |
| Instantiate as a cart concrete subclass object | Products |

Back

Attributes:
Collection Id (Integer)
Product Id (Integer)
Customer Id (Integer)

Relationships:
Generalization (a-kind-of): Is itself a generalization of the user's curations into either a wishlist, or a cart. Both will contain products but differ in how they implement them.
Aggregation (has-parts): Wishlist and Cart Subclasses
Other Associations:

Front:

| Class Name: | Return | ID: | 6 | Type: | Concrete |
|---|---|---|---|---|---|
| Description | | | | Associated Use Cases: | 4 |

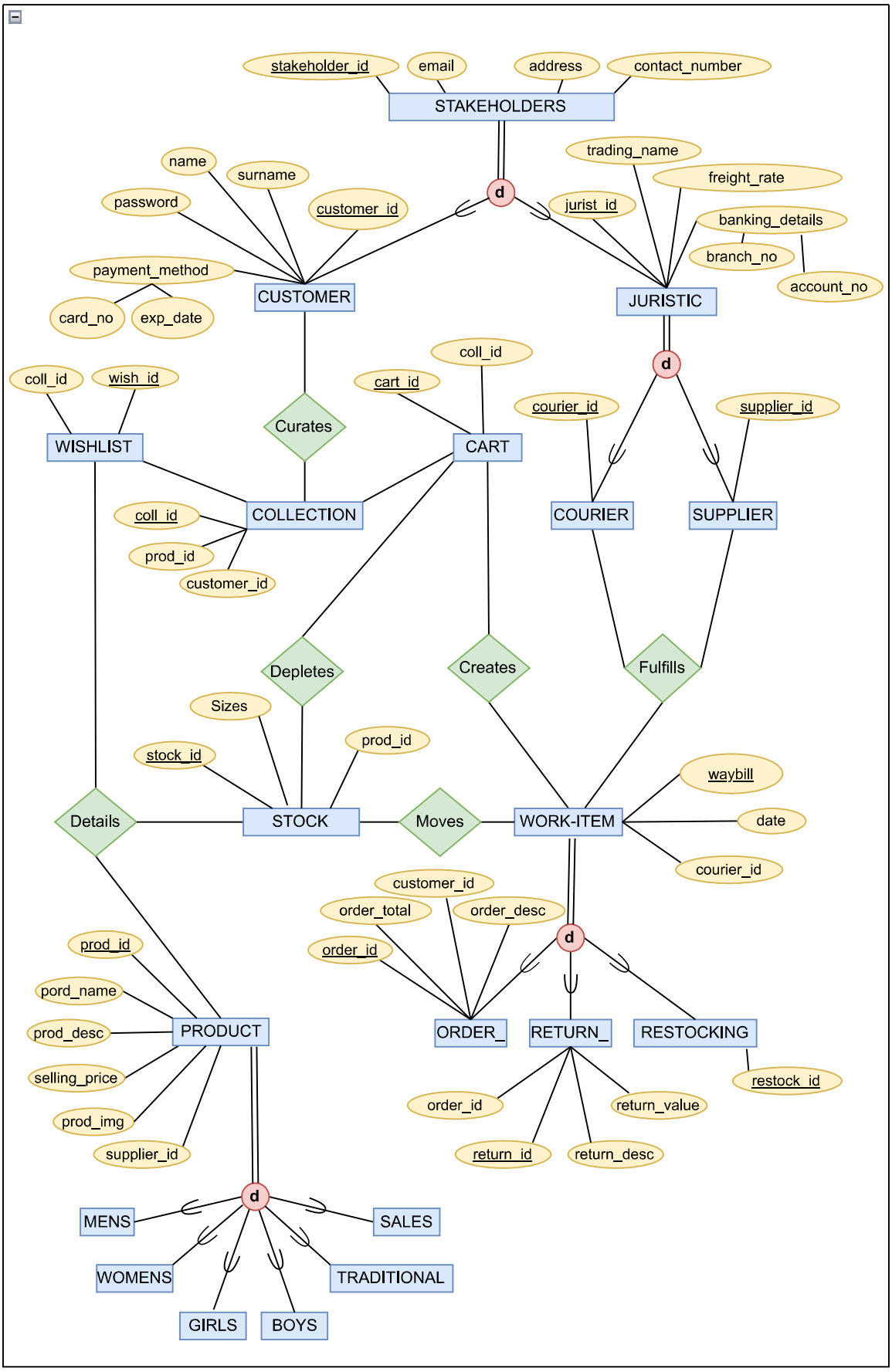
| Responsibilities: | Collaborators |
|---|---|
| Summarizes the request for returning purchased items | Customer, Product |
| Summarizes the request for returning purchased items | Courier, Customer |

Back

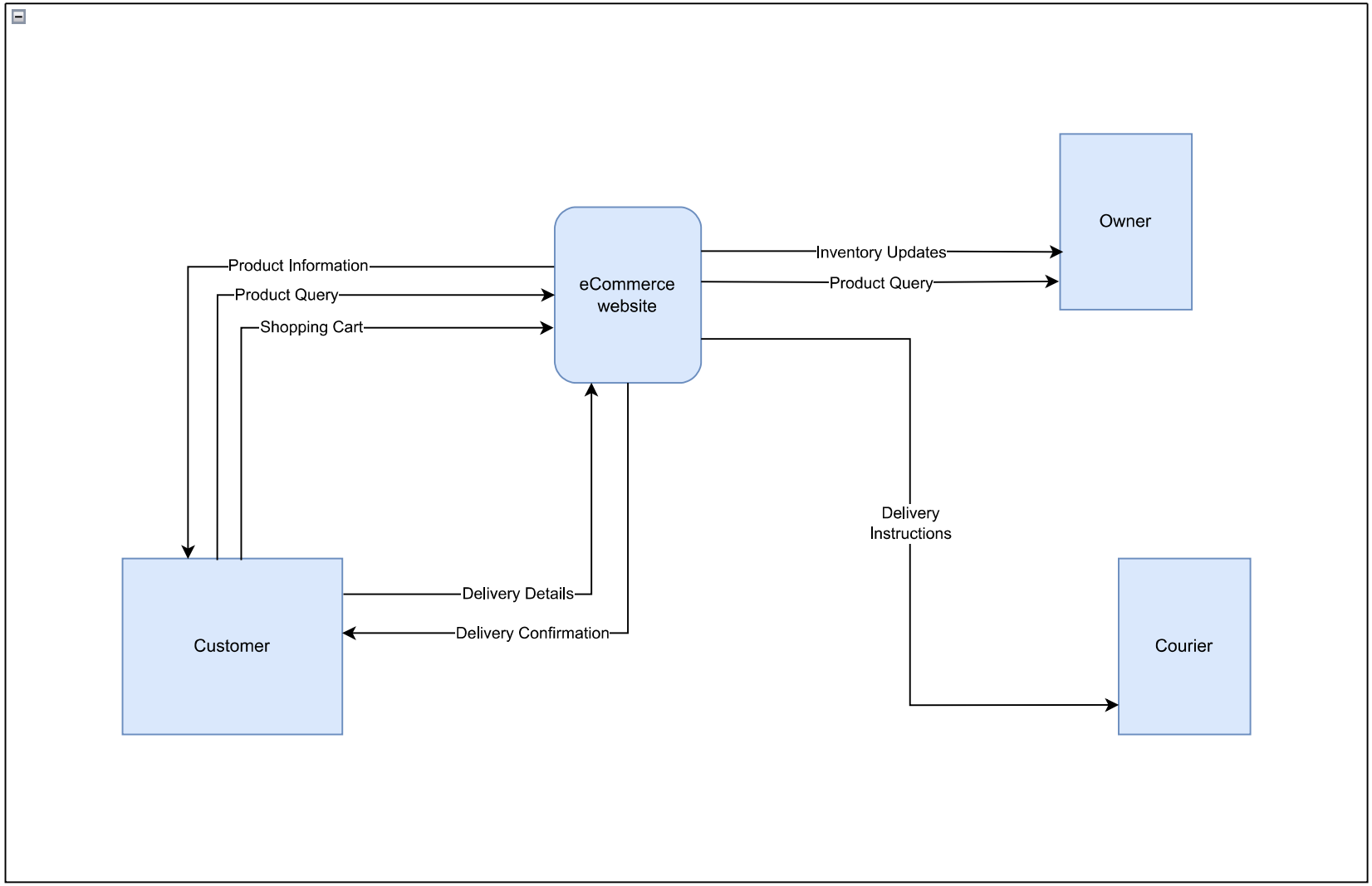
Attributes:
Date (Datetime)
Refund (Decimal)
Return-Description(Varchar)

Relationships:
Generalization (a-kind-of): Work-Item
Aggregation (has-parts): Return-value (Decimal), Return-description (Varchar)
Other Associations:

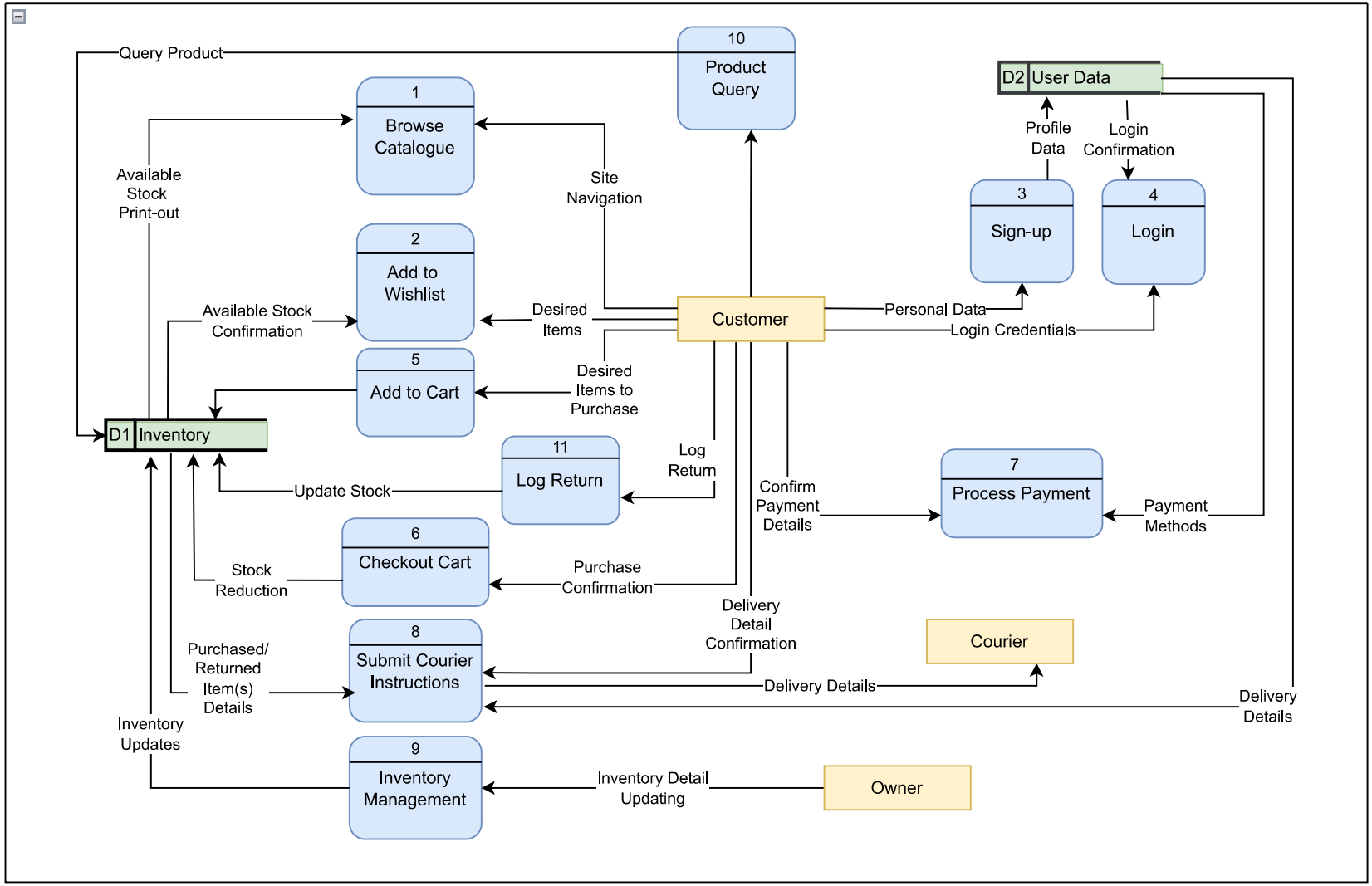
Enhanced Entity Relationship Diagram



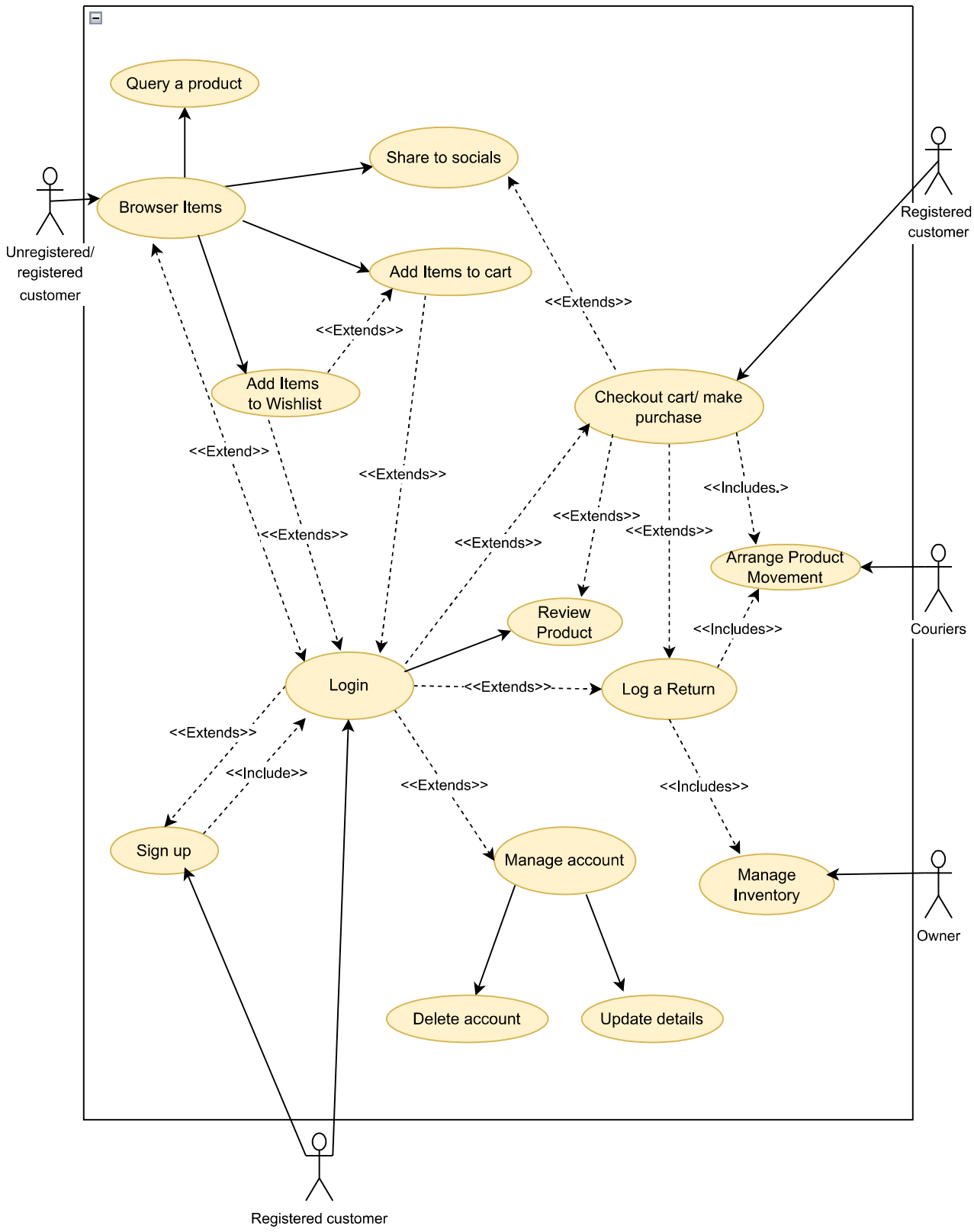
Context Diagram



Data Flow Diagram



Use Case Diagram



# Database Design

After modeling our database onto an EERD, we yield the following tables.

## External_Stakeholders Table

| stakeholder_id | email | address | contact_number |
|---|---|---|---|
| 1 | Johndoe@gmail.com | 1 Skip Street, Pretoria, 0081 | 0831234567 |
| 2 | Info@ramcouriers.co.za | 27 Wrench Rd, Kempton Park, 1600 | 0104948223 |
| 3 | Inventory@mdesigns.co.za | 355 Bullhorn St, Johannesburg, 2215 | 0117375841 |
| 4 | Info@bexexpress.co.za | 120 Loper Ave, Kempton Park, 1619 | 0861239397 |
| 5 | Suzymashaba@gmail.com | 45 Harpoon Lane, Midrand, 0157 | 0723214321 |

## Customer Table

| customer_id | stakeholder_id | name | surname | password | card_no | exp_date |
|---|---|---|---|---|---|---|
| 1 | 1 | John | Doe | Password123! | 4480123456789000 | 03/29 |
| 2 | 5 | Suzy | Mashaba | Secret45### | NULL | NULL |

## Juristic Table

| jurist_id | stakeholder_id | tranding_name | freight_rate | account_no | branch_code |
|---|---|---|---|---|---|
| 1 | 2 | RAM Couriers | R73 | 62000001452 | 250655 |
| 2 | 3 | Marriage Designs | R90 | 401475684 | 632005 |
| 3 | 4 | BEX Express | R59 | 1023657342 | 198765 |

## Courier Table

| courier_id | jurist_id |
|---|---|
| 1 | 1 |
| 2 | 3 |

## Supplier Table

| supplier_id | jurist_id |
|---|---|
| 1 | 2 |

## Collection Table

| coll_id | customer_id | prod_id |
|---|---|---|
| 1 | 1 | 43 |
| 2 | 2 | 14 |

## Wishlist Table

| wish_id | coll_id |
|---|---|
| 1 | 1 |

| | |
|---|---|
| | |

Cart Table

| cart_id | coll_id |
|---|---|
| | |

Once a customer clicks the checkout button, all the records in the cart table will be converted to an invoice description and corresponding total. The records of the cart table will be deleted as the cart is emptied. By contrast, a wish list's contents are emptied when the customer chooses to send them to the cart or merely deletes them.

Whenever the customer clicks the add to wish list or cart button, a new record with their customer id populates the respective table. Should they have not added anything to either table, there will be no records with their customer id and the site will report as such.

Stock Table

| stock_id | prod_id | sizes |
|---|---|---|
| | | |

Product Table

| prod_id | price | quantity | prod_name | prod_desc | prod_img | supplier_id |
|---|---|---|---|---|---|---|
| | | | | | | |

Womens Table

| womens_id | prod_id |
|---|---|
| | |

Mens Table

| mens_id | prod_id |
|---|---|
| | |

Boys Table

| boys_id | prod_id |
|---|---|
| | |

Girls Table

| girls_id | prod_id |
|---|---|

| | |
|---|---|
| | |

## Workitem Table

| waybill | date |
|---|---|
| | |

## Order Table

| order_id | workitem_id | order_desc | customer_id | order_total |
|---|---|---|---|---|
| | | | | |

## Return Table

| return_id | workitem_id | order_id |
|---|---|---|
| | | |

## Restocking Table

| restock_id | workitem_id |
|---|---|
| | |