

D0018E Final Report

Group 12

Simon Malmström Berghem, (bersim-8@student.ltu.se)
Jakob Moregård, (jakmor-8@student.ltu.se)



8 January 2021

Contents

1	Final report	1
1.1	Executive Summary	1
1.2	User Stories	1
1.2.1	Homepage	1
1.2.2	Product page	1
1.2.3	Review page	1
1.2.4	Customer page	1
1.2.5	Signup page	2
1.2.6	Login page	2
1.2.7	Registered customer page	2
1.2.8	Administrator page	2
1.2.9	Cart page	2
1.2.10	Checkout page	3
1.2.11	Roles	3
1.3	System Architecture	3
1.4	Backlog	4
1.5	Database schema	5
1.6	Link to code and website	5
1.7	Test case specifications	6
1.7.1	Demo	6
1.7.2	Checkout	7
1.7.3	Product Page	7
1.7.4	Reviews	7
1.7.5	Cart	8
1.7.6	Login	8
1.7.7	Sign up and Update profile	8
1.7.8	Admin page	9
1.8	Description of the system's limitations and the possibilities for improvements.	9
1.9	Transactions for orders /checkout	9
1.10	Rating and comments on products	10
1.11	References	10
	Appendix A – Images	11
	Appendix B – chronological descriptions	17

1 Final report

1.1 Executive Summary

This report explains the workings of a small e-commerce site with a SQL database that can be accessed through a web browser. On the site customers can add products to their cart and checkout their cart as long as all products are in stock. They can also register to the site and be given a profile which they can customize and access their carts on other devices. The products on the site are easily accessible from the homepage, each with their own product page with detailed information about the product and reviews with ratings from customers. There's also admin accounts that can monitor and maintain the site and the products on the site.

1.2 User Stories

1.2.1 Homepage

Anyone can connect to the site homepage while the website is hosted on a server. Once connected to the website the user will be presented with the homepage as seen in *appendix A*, figure 3. On this page the user will most likely first notice the welcoming image front and center, this image can easily be changed depending on theme, current campaigns, or promotions on the site. Next the user will probably notice the product displays that spans out across the page below the greeting, these product squares are clickable and will redirect the user to that specific products page.

Also on these squares the user can add the product directly to the cart by clicking the buy button. Lastly on the homepage there is a navigation bar at the very top of the page, this navigation bar will be present on all pages of the site to help with navigation, both for seeing what page the user is currently on and for navigation to other pages of interest. To the right in the navigation bar is a status field that will tell the user of their current login status, or lack there of.

1.2.2 Product page

Now for the next page on the site, the product page seen in figure 4 again in *appendix A*, this page is individual and specific to a certain product, and on this page more relevant information about the product will be displayed to the user, such as a short product description, relevant facts, and a better look at the product image.

Located on the right of the image is the purchasing field of the product, the main difference between this field and the purchase button on the previous page is the option to specify the amount of items that the user wants to buy, instead of just adding one item to the cart directly.

Below both of these the review section of the product will be displayed, first a prompt asking if the user wants to write a review of their own, which links to the review submit page. And under the prompt the reviews themselves are displayed in boxes, the header of the box is the score (out of five) the user have given, followed by a optional opinion text field, and lastly to the bottom right the name of the user that submitted the review is displayed.

1.2.3 Review page

This page, as can be seen in figure 5, is very simple, containing only a simple form for submitting a review. First a field for the score, which is mandatory, followed by a large text field for writing the opinion of the product. In the navigation bar the user can now click on '*Product*' which will take the user back to the product page from whence they came.

If the user tries to submit the review while not logged in at this point they will be redirected to the customer page where they will need to submit some basic information, after submitting they will be redirected back to the review submit page and then they can submit a review.

1.2.4 Customer page

Shown in figure 6, the customer page as described in the previous section contains a simple form for submitting some information about the user and after submitting the users role as a *A regular customer* is complete, the users name will be displayed in the navigation bar they and will not need to submit the information again if they want to write more reviews and/or buy products. Though they will be unable to login and/or use the information they submitted again if they initiate a new session.

1.2.5 Signup page

On the subject of submitting information, the sign up page as seen in figure 7, can be accessed by clicking '*Signup*' in the navigation bar if not already logged in as a *registered customer*. This page is also very simple but asks for some more information than the customer page, such as mail and password, once submitted the user have the role of a *registered customer* and is redirected to the homepage and the login status in the navigation bar will be updated.

1.2.6 Login page

Once again this page is very simple as can be seen in figure 8, it is accessed by clicking '*Login*' in the navigation bar if not already logged in as a *registered customer* or a *administrator*, here the user will enter mail and password that they used to sign up with or in the case of a *administrator* login, the credentials they where provided with. In the case of a *registered customer* login the user is redirected to their specific page and the login status is updated. And if the login was for a *administrator* they will be redirected to the administrator page and the login status is updated accordingly.

1.2.7 Registered customer page

On this page as seen in figure 9 the all the current information about the *registered customer* is displayed to the very right, and in the center of the page is a form if the user wishes to change/update some, if not all of the current information, such as a change of delivery address for example.

It can also be observed that the fields of the navigation bar have changed, '*Login*' and '*Signup*' have been replaced with '*Account*' and '*Logout*' respectively. This change will be applied as soon as a user has logged in as either *registered customer* or *administrator* with a slight difference that '*Account*' will say '*Admin*' if the login was as an *administrator*.

1.2.8 Administrator page

Compared to other pages this page as seen in figure 10 looks quite busy in comparison. On the left, just as on the *registered customer* page there is a field with all the credentials of the *administrator*.

In the center top of the page there are three forms that have different functions, the left-most form is for adding new products to the page, all the fields in this form are mandatory and after submitting the new product will assigned a unique product ID and will also be visible on the home page where it's now purchasable. The center form is for manually updating existing products, to update a product a *administrator* will need to enter the products unique ID and new details in the fields they wish to change. The last form to the right is for removing a product from the website, the form only takes the product unique ID and upon submission the product, it's page, and all reviews will also be removed.

Below the forms there is a table, this table displays all fields of the products in columns and all the products themselves as one product per row. this makes it easy for a *administrator* to keep track of all products and details on the website in one place.

1.2.9 Cart page

For this section there are two images in *appendix A*, figures 11 and 12, the first image shows an example of how the cart will look when containing a number of products. And for every product box there are details for the amount of items of that product that are in the cart and what they will cost together. There is also a field for changing the amount of items, if the new item amount is set to lower than items currently in the cart the product will be removed from the cart entirely. If all products are removed from the cart a friendly image will appear to encourage the customer to buy something, this image is also very interchangeable.

In both cases there will be a green button under all other elements that will submit the current cart to purchasing, and the customer will be redirected to the checkout page as a final step in the purchase process.

If a user that has neither the role as *registered customer* or *regular customer* try to add a product to the cart they will be redirected to the customer page in the same manner as for the review page.

1.2.10 Checkout page

The final page of the website seen in figure 13 will be displayed if a customer has committed to a purchase. The page contains a friendly image to thank the customer for the purchase and to the right of the image is a summary of the order with a header to again thank the customer but this time in writing.

Below both of these there is a table that shows all previous orders that that customer has placed, with product name, amount, and pricing. This table is displayed in chronological order.

1.2.11 Roles

A *regular customer* has their own shopping cart which they can access even if it's currently empty. They can add products to the cart from the homepage but they need to enter their contact information before adding anything to their cart. In the cart page they can add or remove products that currently are in the cart or checkout the items currently in the cart if the the amount is in the current stock. Customers can also rate and leave reviews on products on their product pages. They can also sign up to be a registered customer.

A *registered customer* is a customer that has been registered with their unique mail and password with their contact information. Registered customers can also login or logout from the site and freely add products to their cart from the homepage and change the amount in cart page without having to enter their contact information again. Registered customers can also checkout the items currently in their cart if they're in stock. They also has their own profile page where they can see and update their contact information. Like customers, registered customers can also rate and leave reviews on products on their product pages.

An *administrator* can also login or logout to the site and see available products on the homepage. Admins does not have access to a shopping cart and cannot add or remove products to a cart, but still has access to the product pages. Admins however has access to the admin page where they can see all admin users, products and publish, edit or remove products.

1.3 System Architecture

The website is hosted with a public LUDD DUST server that has a local SQL server. The website uses python/Flask to create an app, PyMySQL to access the SQL server and HTML templates as the frontend for the web pages.

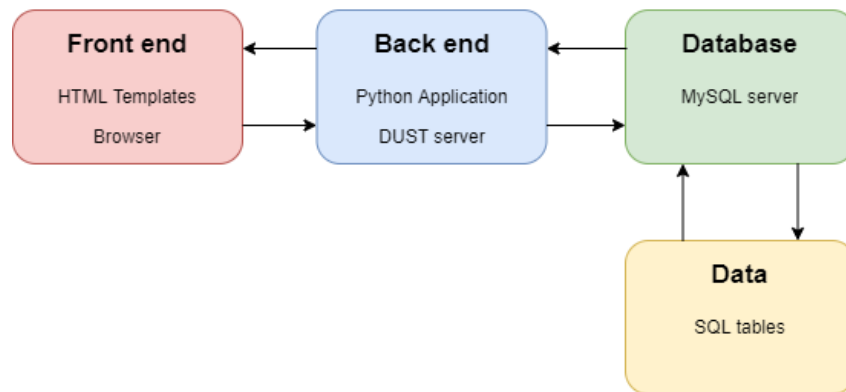


Figure 1: System Architecture graphically represented

1.4 Backlog

ID	Description	Prio	Effort	Sprint	Status
Database setup	Setup relational database for the web page.	3	Medium	1	done
Web page setup	Write simple test web page.	2	Low	1	done
Cooperation	Setup git repository.	2	Low	1	done
Server setup	Setup server for web page.	1	Medium	1	done
W-D communication	Get data from the database to appear on a web page.	3/1	Medium	1/2	done
Expand customer web page	Flesh out the web page interface for the customers	3	Medium	2	done
Publish Products	Publish products on the web page, linked to the database.	1	High	2	done
Edit Products	Edit aspects of current products on web page and database, such as pricing, stock, description, etc.	2	Medium	2	done
Remove Products	Remove products from web page and database.	2	Low	2	done
Exception handling	Handle exceptions to prevent Internal Server Error page.	1	Low	3	done
Cookies	Add cookies that track session data to the web page.	1	High	3	done
Login	Add a login function to the web page.	2	Medium	3	done
Login access	Restrict access to certain web pages based on user login.	2	Medium	3	done
Cart	Add a cart function to the web page.	3	High	3	done
Add to cart	Let customers add products to cart.	1	Medium	3	done
Remove from cart	Let customers remove products from cart	2	Medium	3	done
Buy cart	Let customers proceed to checkout	1	High	4	done
Setup product pages	Implement product pages which shows detailed information about a specific product	2	Medium	4	done
Create reviews	Let customers leave ratings & comments on product pages	3	Medium	4	done
CSS styling	Improve the general appearance of the different pages of the site	3	Medium	4	done

1.5 Database schema



Figure 2: The tables included in the database schema

1.6 Link to code and website

[Github repository](#)

[Our website](#) (must be hosted first).

1.7 Test case specifications

1.7.1 Demo

[Our website](#) (must be hosted first).

Admin tests:

1. Login (top left corner) with admin account, Mail: Lisa@Master.com, Password: VeryCool
2. Can add, remove or update products
3. Changes show up on homepage
4. Cannot access cart or add anything to cart via homepage

Customer tests:

1. Add product from homepage, if incorrect amount error message or page is reloaded
2. Is sent to enter contact information, all chars
3. If contact information entered incorrectly site will reload
4. If contact information entered correctly will resend to cart page with product and amount specified from homepage
5. Can change amount on product, if ≤ 0 , product is removed
6. Can add new products from homepage without entering contact information again

Registered customer test:

1. Sign up on homepage, if incorrect input page is reloaded
2. Can change profile, if incorrect input page is reloaded
3. Add product from homepage, if incorrect amount error message or page is reloaded
4. Sent to cart page and can change amount on product, if ≤ 0 , product is removed

Product page test:

1. Click on product
2. See that information is correct
3. Add to cart

Checkout test:

Preconditions: Two carts with two different customers, with none of them having a previous order.

1. Checkout one of the carts and see that the stock is updated
2. Make a new cart on the customer that just checkout their cart and checkout that cart and see the previous order
3. Make a another cart on the customer that just checked out carts and add the full stock of a product that is in the other customer's cart
4. Checkout the new cart and see the previous orders
5. Try to checkout the other customer's cart, which will result in a reload of the cart page

1.7.2 Checkout

Checkout is a crucial part of the system that influences many other parts of the system. So we tested to make sure the checkouts had these properties:

1. Correct amount and corrects products are checked out
2. The stock is updated correctly
3. Carts can't checkout if stock is updated to be less than amount in cart
4. Empty carts can't checkout
5. Old checked out carts are shown with correct products and amount

Property 1 worked as intended as item IDs are included in the cart even if not visible on the website. A problem with concurrency. If a customer checked out a cart so a product's stock would be less than another customer's cart, the other customer would still be able to checkout their cart leaving the stock at a negative number. Thus property 2 and 3 would not be correct, so to prevent this an additional check was added to not allow a checkout that would result in a negative stock. Property 4 has an if statement to reload the page if the customer does not have a cart so it works as intended. The last property works but not quite as intended. Since all IDs are randomized and our SQL server lists them according to their primary key, their ID, they are not chronically ordered but with the correct amount and products.

1.7.3 Product Page

To test the product page we checked these scenarios:

1. The page show the correct product and information
2. Customers can add products to cart
3. The correct product is added to cart
4. Admins cannot add products

By checking all the product pages we can observe that they show the correct products and information including reviews thus scenario 1 is not an issue. Scenario 2 and 3 work as intended as the correct products with the correct amount are added to the customers carts from the product page. Scenario 4 is not possible cause there's a check to see if the user is logged in as an admin and will then reload the page if they try to add a product to a cart.

1.7.4 Reviews

The review page has many html restrictions so we only tested these scenarios:

1. Rating is in correct range
2. Review text is in the correct length

As mentioned the html restrictions make sure that scenario 1 is correct as and in the range 1-5, it's impossible to enter a number outside of the range or to enter a string. Html restrictions also applies on scenario 2 only allowing texts of the correct length.

1.7.5 Cart

In order to test that the cart works as intended we need to test two functions, adding products to cart from the homepage and change the amount / remove of a product in the cart page.

For the adding in the homepage we tested these scenarios:

1. Adding product as admin
2. Adding product without amount
3. Adding product with negative amount
4. Adding product with amount with wrong type
5. Adding product that already exists in the cart
6. Adding product that doesn't exist in the cart

Since we check for the scenarios 1 and 2, the page would simply reload. Scenario 3 would give us a negative amount in the cart, so a check was added that would reload the page if the amount is ≤ 0 . Scenario 4 is not possible since we specify in the HTML page that only numbers can be entered. Scenario 5 would overwrite the current amount in the cart, so it was changed to add the new amount to the existing amount. Scenario 6 would simply add the new item to the cart as intended.

For changing the amount / remove product function we tested these scenarios:

1. Changing amount with negative amount
2. Changing amount to 0
3. Changing amount to < 0
4. Changing amount to nothing
5. Changing amount to wrong type
6. Changing amount with other items in cart

Scenario 1 resulted in that the product's amount would decrease as intended, scenario 2 would remove the product as intended. Scenario 3 would set the amount to a negative value, so a check was added to remove the product if a product's amount is ≤ 0 . Scenario 4 would cause Internal Server Error, so a check that would reload the page was added. Scenario 5 will as scenario 4 for the homepage adding, not work cause of the HTML restriction. Scenario 6 would simply add or subtract the amount of the correct product as intended.

1.7.6 Login

Since all fields available for the users in all both are chars, it's only necessary to check if the input is of the correct length, if a required field is missing and if the login credentials were incorrect.

All three of these scenarios causes Internal Server Error, so try and catch statements were added to reload the page.

1.7.7 Sign up and Update profile

Since all fields available for the users in all both are chars, it's only necessary to check if the input is of the correct length, if a required field is missing and for sign up that the primary key mail is not already in use.

Both of the first scenarios causes Internal Server Error for both functions, so try and catch statements were added to simply reload the page. If a mail is already in use, it would reload the page with an error message as intended.

1.7.8 Admin page

We tested different kinds of inputs to the removing, publishing and editing functions.

For the publish and edit function we tested these inputs:

1. An input with field(s) with incorrect type(s)
2. An input where required field(s) are missing
3. An input where field(s) are longer than limit
4. An input with all fields and correct types
5. An input with all required fields and correct types

Inputs 1-3 resulted with Internal Server Error, where we should do a try and catch statement and reload the page with an error message.

Inputs 4 and 5 worked as intended and added or edited the correct products and reloaded the current web page.

For the remove function we simply tried inputs with correct id and incorrect id as it has one field that is an int which has a high limit making it hard to test. With a correct id it worked as intended and removed the product with the specified id and reload the page. However if an incorrect id was entered, the site would simply reload the page without removing anything.

1.8 Description of the system's limitations and the possibilities for improvements.

The website has some basic functionality of an e-commerce site, however could still be improved.

One big improvement would be for customers to create a cart without them having to enter their contact information. Currently all carts are related with either a customer ID, *CuID*, or a registered ID, *ReID*.

While the pages has received some CSS styling there are many pages that look plain without color or images and would look significantly better with more CSS styling.

Admin functionality could also be improved by allowing admins to edit or remove products in the product pages.

Currently the only way to see previous orders on the website is in the checkout page. An improvement would be to add a list of previous orders in a registered user's profile and to add a system accessible for admins to see current and completed orders.

The amount of IDs are limited and not structured in a way to easily tell what kind of ID it is by simply looking at the ID. The current limit is 99999999 IDs, in the range 1-99999999.

Another improvement would be to allowing customers to change the amount of several products in their cart at the same time, currently they have to be changed one by one.

The security of the system is very weak as all data is in plain text, so if the data is encrypted in some way the system would be more secure. The system is also vulnerable to SQL injections, to prevent this the user input should more carefully handled.

1.9 Transactions for orders /checkout

The idea behind the checkout transactions is to save a snapshot the current price of the product and the amount of products the customers wishes to buy. This is saved in the table *Item* with the current cart id when it's added to the cart. When customer wants to checkout their cart, then for each product in the cart it's checked if there is enough products in stock, $\text{stock} - \text{amount} \geq 0$. If every product in the cart has enough products in stock then the cart will checkout and remove the amount from the stock. If there is not enough items in the stock for at least one product, then the cart page is reloaded and the cart will not be allowed to enter checkout. If the products that are out of stock are removed from the cart then the cart can be checked out.

1.10 Rating and comments on products

The idea is for regular customers and registered customers to be able to leave ratings and comments on product pages which is then saved on our database in it's table *Rating* and writes out on the product page. The table *Rating* contains several columns, one for the ratings specific and unique id, one for the score of the review (1-5), one for the opinion of the product, then there are columns referencing the product ID of the product being reviewed, customer/registered IDs referencing to the customer that writes the review.

The review needs to contain a score but the opinion section is not mandatory. When the review is displayed on the product page the first name of the customer or registered user is fetched from the database so that other users can see who has left the review.

1.11 References

1. [Digitalocean community](#), used for server and flask set up.
2. [Stackoverflow](#), used for details in Python, flask, HTML, CSS, and SQL.
3. [Flask Documentation](#), used for finding flask commands.
4. [w3schools](#), used for SQL, HTML, and CSS syntax.
5. [Techonthenet](#), used for details in SQL
6. [MySQL Connector/Python Developer Guide](#), used for details in MYSQL

Appendix A – Images

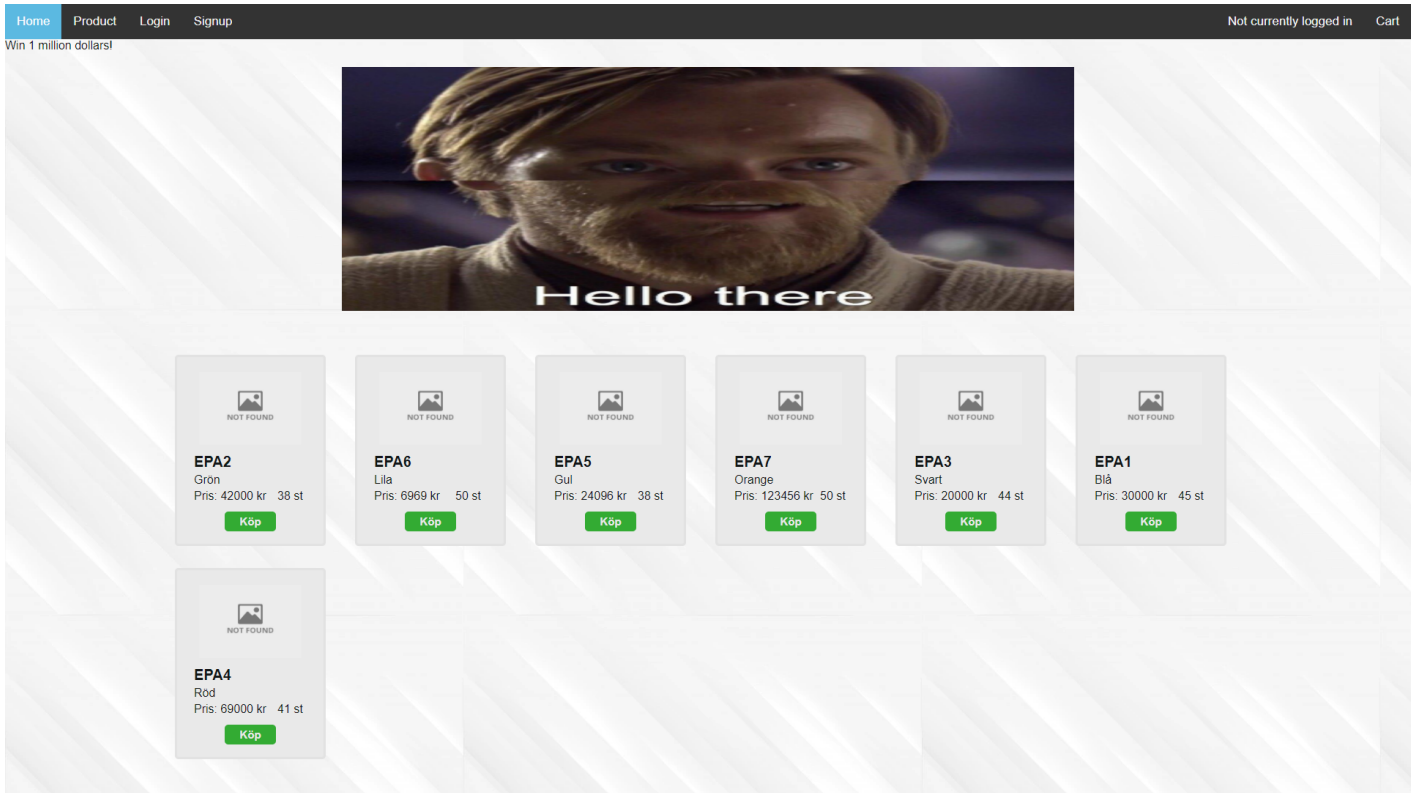


Figure 3: The first page of the website

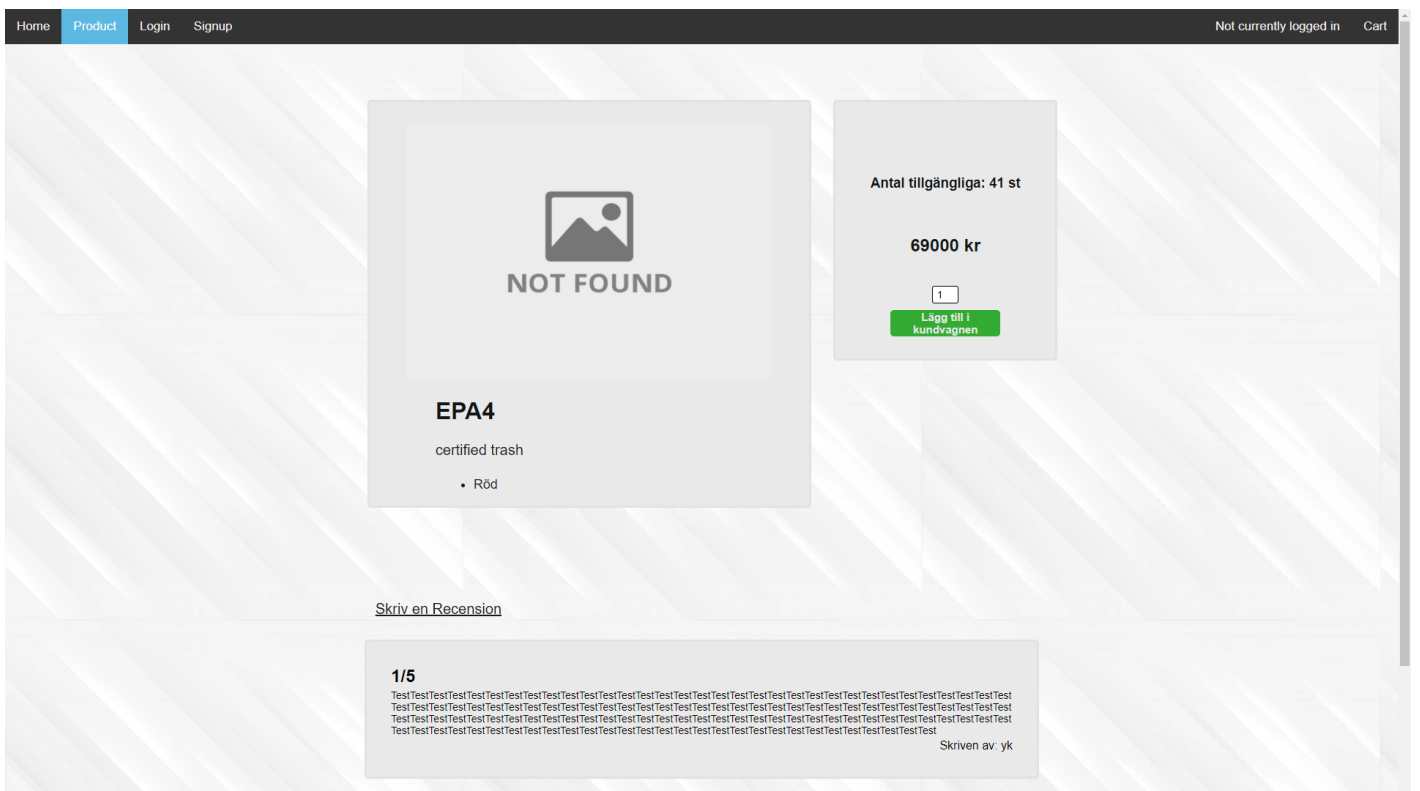


Figure 4: The page for the individual products

[Home](#) [Product](#) [Login](#) [Signup](#) Not currently logged in [Cart](#)

Review

Score*:

Review:

Fields marked with * are mandatory

Submit

Figure 5: The page for submitting reviews

[Home](#) [Product](#) [Login](#) [Signup](#) Not currently logged in [Cart](#)

Submit info

First name*:

Last name*:

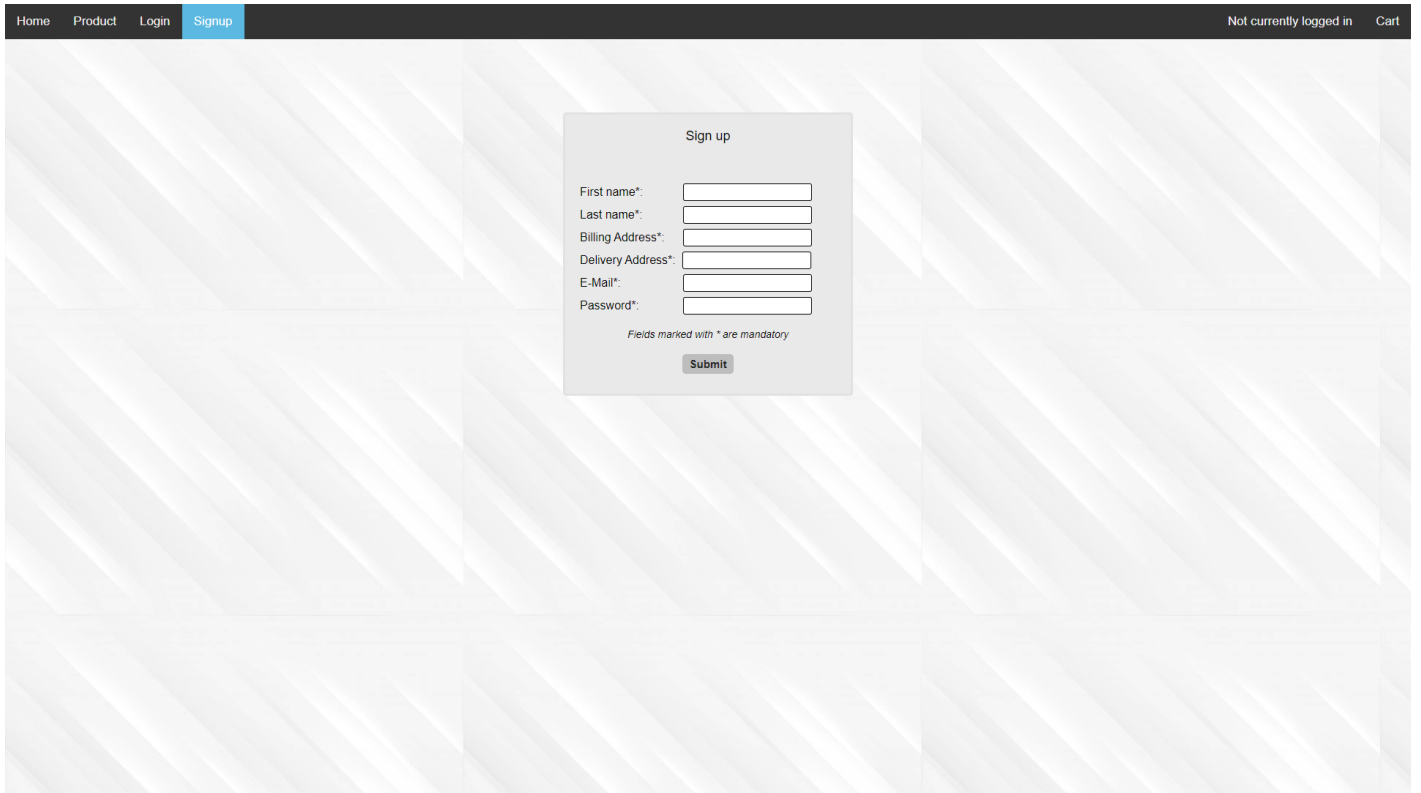
Billing Address*:

Delivery Address*:

Fields marked with * are mandatory

Submit

Figure 6: The page for submitting customer information



The screenshot shows a web application interface with a dark header bar. On the left, there are navigation links: Home, Product, Login, and Signup. The 'Signup' link is highlighted in blue. On the right, the text 'Not currently logged in' and a 'Cart' link are visible. The main content area has a light gray background with a subtle grid pattern. In the center, there is a white 'Sign up' form. The form contains five input fields, each with an asterisk indicating it is mandatory: 'First name*', 'Last name*', 'Billing Address*', 'Delivery Address*', and 'E-Mail*'. Below these is a 'Password*' field. A 'Submit' button is located at the bottom of the form. A small note below the fields states 'Fields marked with * are mandatory'.

Home Product Login Signup Not currently logged in Cart

Sign up

First name*:

Last name*:

Billing Address*:

Delivery Address*:

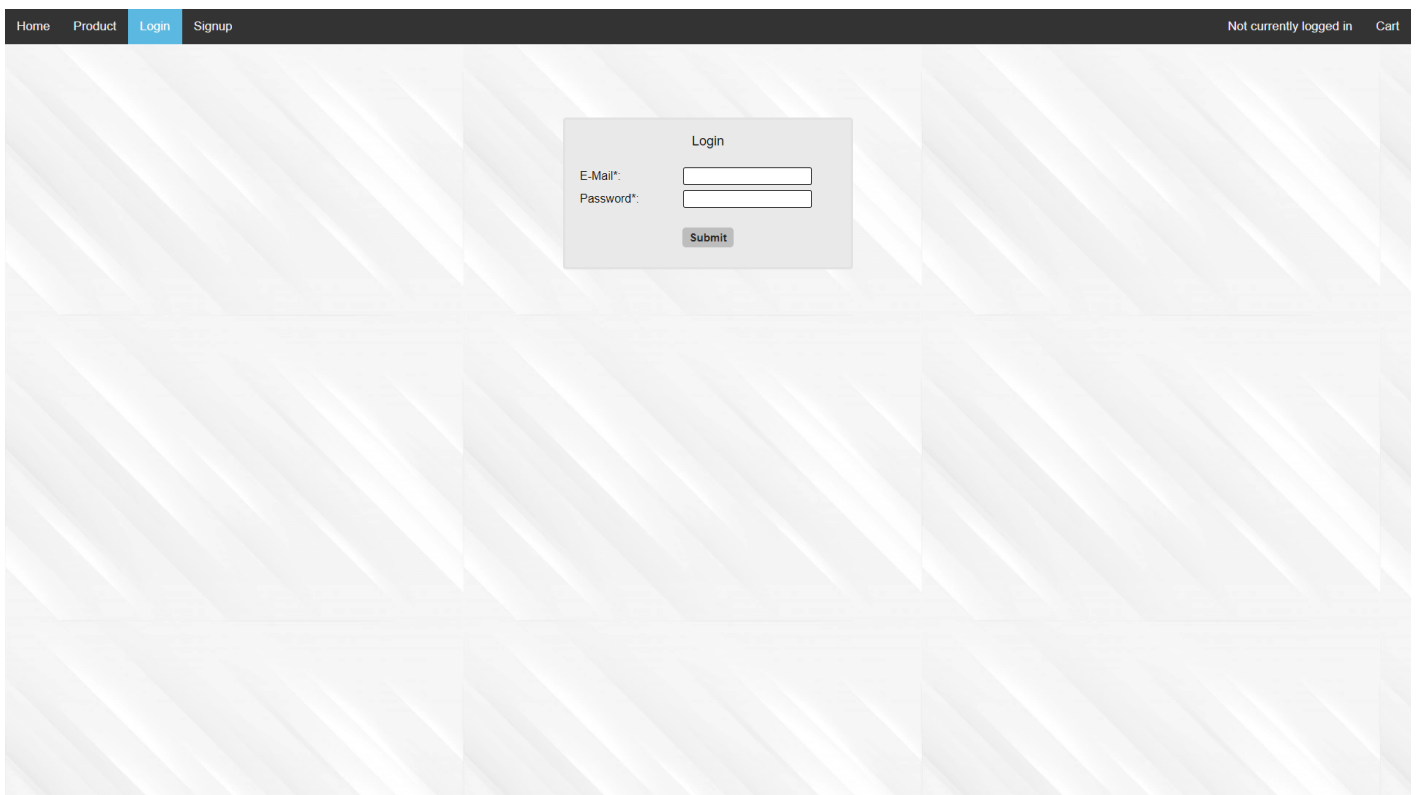
E-Mail*:

Password*:

Fields marked with * are mandatory

Submit

Figure 7: The page for signing up as a registered user



The screenshot shows the same web application interface as Figure 7, but with the 'Login' link highlighted in blue in the header. The 'Sign up' form is no longer visible. In the center, there is a white 'Login' form. It contains two input fields, both with asterisks indicating they are mandatory: 'E-Mail*' and 'Password*'. A 'Submit' button is located below the fields.

Home Product Login Signup Not currently logged in Cart

Login

E-Mail*:

Password*:

Submit

Figure 8: The page for logging in as registered user or administrator

Home
Product
Account
Logout

Jakob is logged in as registered
Cart

Update your information

First name*:

Last name*:

Billing Address*:

Delivery Address*:

E-Mail*:

Password*:

Fields marked with * are mandatory

Submit

First Name:

Jakob

Last Name:

Moregård

Billing Address:

Professorsvagen 17

Delivery Address:

Professorsvagen 17

Mail:

Jakob@mail.com

Password:

Password123

Figure 9: The page for a registered user

Home
Product
Admin
Logout

Jakob is logged in as admin

Add product

Name*:

Color*:

Description*:

Price*:

Stock*:

Fields marked with * are mandatory

Submit

Update product

ID*:

ID of product to edit

Name*:

Color*:

Description*:

Price*:

Stock*:

Fields marked with * are mandatory

Submit

Remove product

ID*:

Fields marked with * are mandatory

Submit

PID	PrID	AvID	Name	Color	Price	Stock
23083754	23083754	23656266	EPA2	Grön	42000	38
10008653	10008653	42790436	EPA6	Lila	6969	50
62690139	62690139	48812328	EPA5	Gul	24096	38
77109580	77109580	54833513	EPA7	Orange	123456	50
60631362	60631362	65043466	EPA3	Svart	20000	44
39579424	39579424	80583035	EPA1	Blå	30000	45
32924536	32924536	87494087	EPA4	Röd	69000	41

Admin ID:

112233

First Name:

Jakob

Last Name:

Moregård

Mail:

Jakmor-8@student.ltu.se

Password:

password

Figure 10: The page for a administrator

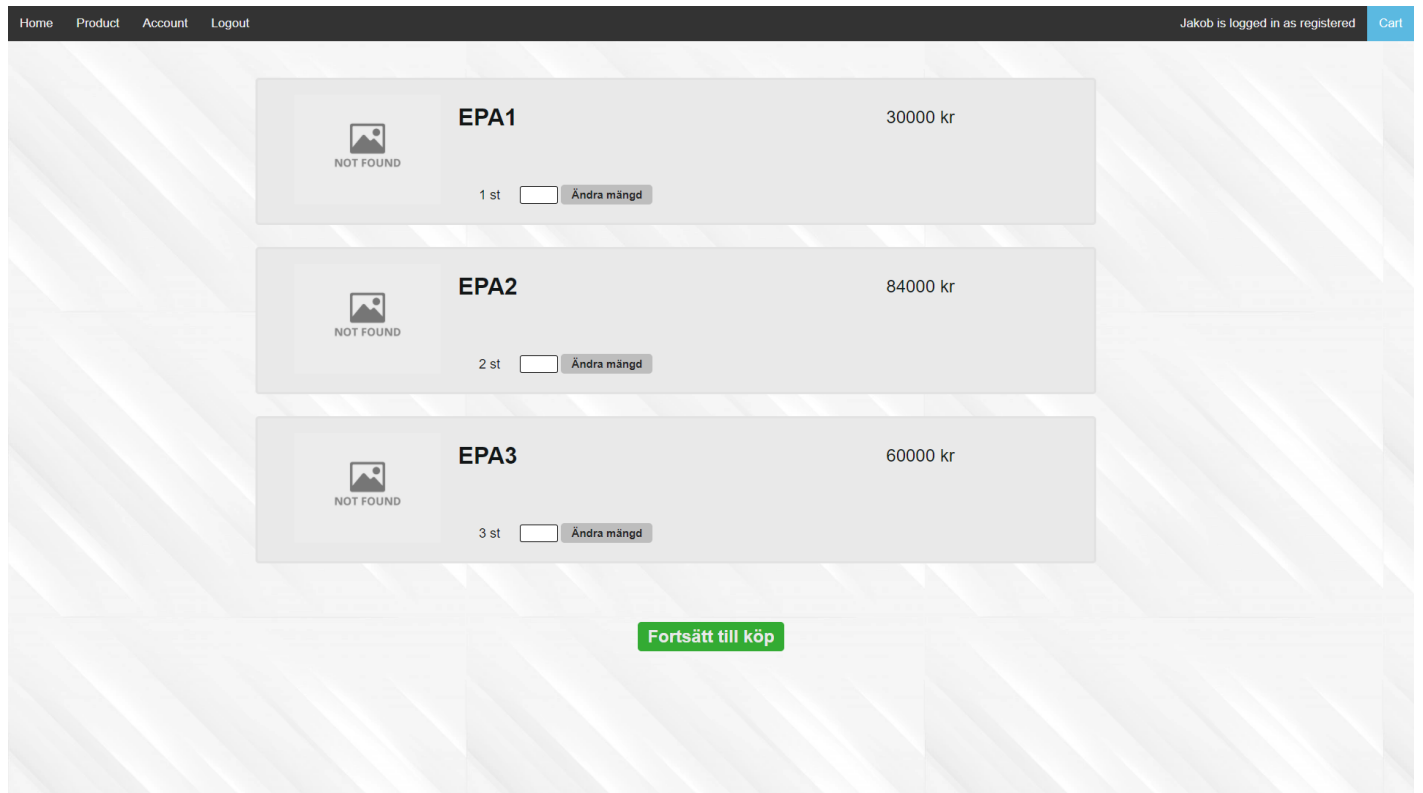


Figure 11: The page for the cart containing items

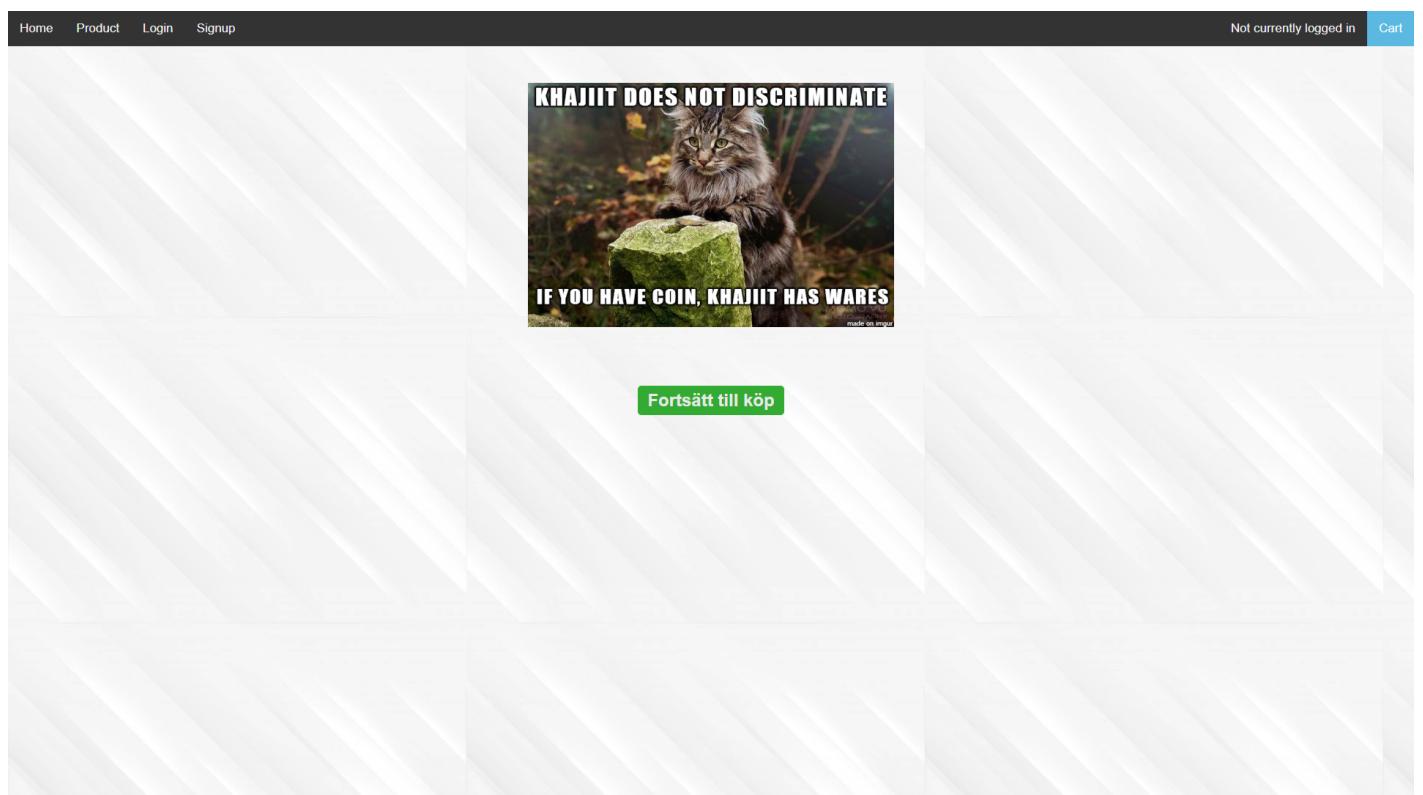


Figure 12: The page for the cart without items



Tack för din order!

EPA4	1 st	69000 kr
EPA5	1 st	24096 kr

Previous Orders

Order 1:

Namn	Pris	Mängd
EPA2	84000 kr	2 st
EPA1	30000 kr	1 st
EPA3	60000 kr	3 st

Figure 13: The page for purchased orders

Appendix B – chronological descriptions

Sprint 3

Executive Summary

This sprint we added several functions, improved the site's appearance and added some exception handling. The functions added are:

- Login
- Logout
- Add product to cart
- Change amount / remove a product from the cart

A banner to easier access pages on the site, a login page, a sign up page, a user page for registered customers and also a cart page were added to the website. Cookies were also added to store session ID and login status of the website user. To avoid the Internal Server Error page some exception handling was added, were most errors would result in a reload of the page.

User Stories

Anyone can connect to the site homepage while the website is hosted on the server.

A *regular customer* can access their shopping cart even if the cart currently is empty. They can add products to the cart from the homepage but they need to enter their contact information before adding anything to their cart. In the cart page they can add or remove products that currently are in the cart or checkout the items currently in the cart. Customers can also rate and leave reviews on products. They can also sign up to be a registered customer.

A *registered customer* is a customer that has been registered with their unique mail and password with their contact information. Registered customers can also login or logout from the site and freely add products to cart from the homepage and change the amount in cart page without having to enter their contact information again. Registered customers can also checkout the items currently in their cart. They also has their own profile page where they can see and update their contact information. Like customers, registered customers can also rate and leave reviews on products.

An *admin* can also login or logout to the site and see available products on the homepage. Admins does not have access to a shopping cart and cannot add or remove products to a cart. Admins however has access to the admin page where they can see all admin users, products and publish, edit or remove products.

System Architecture

The website is hosted with a public LUDD DUST server that has a local SQL server. The website uses python/Flask to create an app, PyMySQL to access the SQL server and HTML templates as the frontend for the web pages.

Current Backlog

ID	Description	Prio	Effort	Sprint	Status
Database setup	Setup relational database for the web page.	3	Medium	1	done
Web page setup	Write simple test web page.	2	Low	1	done
Cooperation	Setup git repository.	2	Low	1	done
Server setup	Setup server for web page.	1	Medium	1	done
W-D communication	Get data from the database to appear on a web page.	3/1	Medium	1/2	done
Expand customer web page	Flesh out the web page interface for the customers	3	Medium	2	done
Publish Products	Publish products on the web page, linked to the database.	1	High	2	done
Edit Products	Edit aspects of current products on web page and database, such as pricing, stock, description, etc.	2	Medium	2	done
Remove Products	Remove products from web page and database.	2	Low	2	done
Exception handling	Handle exceptions to prevent Internal Server Error page.	1	Low	3	done
Cookies	Add cookies that track session data to the web page.	1	High	3	done
Login	Add a login function to the web page.	2	Medium	3	done
Login access	Restrict access to certain web pages based on user login.	2	Medium	3	done
Cart	Add a cart function to the web page.	3	High	3	done
Add to cart	Let customers add products to cart.	1	Medium	3	done
Remove from cart	Let customers remove products from cart	2	Medium	3	done
Buy cart	Let customers proceed to checkout	1	High	4	x
Setup product pages	Implement product pages which shows detailed information about a specific product	2	Medium	4	x
Create reviews	Let customers leave ratings & comments on product pages	3	Medium	4	x

Database schema

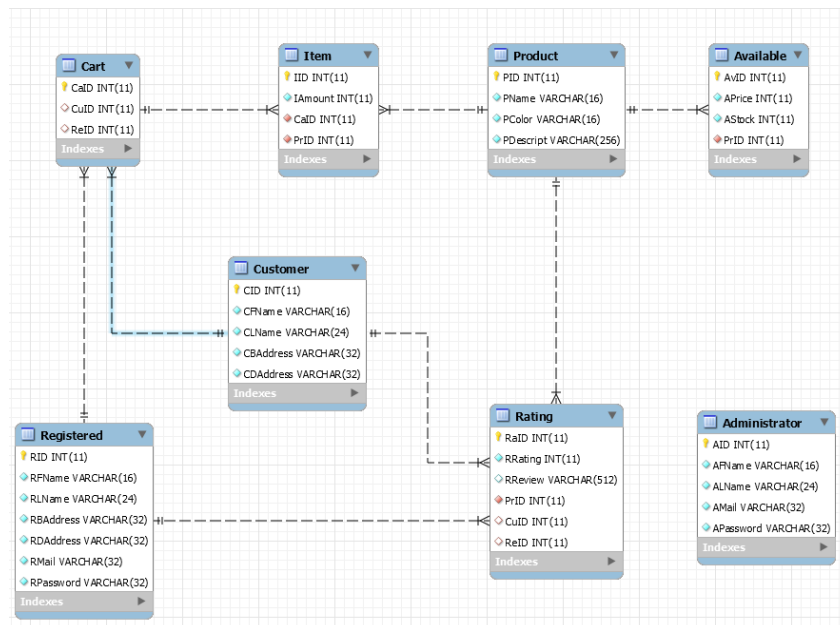


Figure 14: The tables included in the database schema

Link to code and website

[Github repository](#)

[Our website](#) (must be hosted first).

Test case specifications

Demo

[Our website](#) (must be hosted first).

Admin tests:

1. Login (top left corner) with admin account, Mail: Lisa@Master.com, Password: VeryCool
2. Can add, remove or update products
3. Changes show up on homepage
4. Cannot access cart or add anything to cart via homepage

Customer tests:

1. Add product from homepage, if incorrect amount error message or page is reloaded
2. Is sent to enter contact information, all chars
3. If contact information entered incorrectly site will reload
4. If contact information entered correctly will resend to cart page with product and amount specified from homepage
5. Can change amount on product, if ≤ 0 , product is removed
6. Can add new products from homepage without entering contact information again

Registered customer test:

1. Sign up on homepage, if incorrect input page is reloaded
2. Can change profile, if incorrect input page is reloaded
3. Add product from homepage, if incorrect amount error message or page is reloaded
4. Sent to cart page and can change amount on product, if ≤ 0 , product is removed

Cart

In order to test that the cart works as intended we need to test two functions, adding products to cart from the homepage and change the amount / remove of a product in the cart page.

For the adding in the homepage we tested these scenarios:

1. Adding product as admin
2. Adding product without amount
3. Adding product with negative amount
4. Adding product with amount with wrong type
5. Adding product that already exists in the cart
6. Adding product that doesn't exist in the cart

Since we check for the scenarios 1 and 2, the page would simply reload. Scenario 3 would give us a negative amount in the cart, so a check was added that would reload the page if the amount is ≤ 0 . Scenario 4 is not possible since we specify in the HTML page that only numbers can be entered. Scenario 5 would overwrite the current amount in the cart, so it was changed to add the new amount to the existing amount. Scenario 6 would simply add the new item to the cart as intended.

For changing the amount / remove product function we tested these scenarios:

1. Changing amount with negative amount
2. Changing amount to 0

3. Changing amount to < 0
4. Changing amount to nothing
5. Changing amount to wrong type
6. Changing amount with other items in cart

Scenario 1 resulted in that the product's amount would decrease as intended, scenario 2 would remove the product as intended. Scenario 3 would set the amount to a negative value, so a check was added to remove the product if a product's amount is ≤ 0 . Scenario 4 would cause Internal Server Error, so a check that would reload the page was added. Scenario 5 will as scenario 4 for the homepage adding, not work cause of the HTML restriction. Scenario 6 would simply add or subtract the amount of the correct product as intended.

Login

Since all fields available for the users in all both are chars, it's only necessary to check if the input is of the correct length, if a required field is missing and if the login credentials were incorrect.

All three of these scenarios causes Internal Server Error, so try and catch statements were added to reload the page.

Sign up and Update profile

Since all fields available for the users in all both are chars, it's only necessary to check if the input is of the correct length, if a required field is missing and for sign up that the primary key mail is not already in use.

Both of the first scenarios causes Internal Server Error for both functions, so try and catch statements were added to simply reload the page. If a mail is already in use, it would reload the page with an error message as intended.

Admin page

We tested different kinds of inputs to the removing, publishing and editing functions.

For the publish and edit function we tested these inputs:

1. An input with field(s) with incorrect type(s)
2. An input where required field(s) are missing
3. An input where field(s) are longer than limit
4. An input with all fields and correct types
5. An input with all required fields and correct types

Inputs 1-3 resulted with Internal Server Error, where we should do a try and catch statement and reload the page with an error message.

Inputs 4 and 5 worked as intended and added or edited the correct products and reloaded the current web page.

For the remove function we simply tried inputs with correct id and incorrect id as it has one field that is an int which has a high limit making it hard to test. With a correct id it worked as intended and removed the product with the specified id and reload the page. However if an incorrect id was entered, the site would simply reload the page without removing anything.

Description of the system's limitations and the possibilities for improvements.

The website is improving, however it's still a way from having the functionality required of an e-commerce site. The biggest limitation of the site is that in order to create a cart, the SQL server requires either a customer ID, *CuID* or a registered customer ID, *ReID*. To create one of these IDs the customer needs to enter their contact information or create an account on the page, so a cart can't be created without entering their contact information. Most of the pages on the site are still very plain without much color or images even with the added banner.

There is also much room for improvements especially in the functionality of the site the main one being the ability to checkout your cart, the intended purpose of the site. Another improvement would be to add product pages where customers can submit reviews and comments on the product. Setting a range for ids for objects that require an ID would also improve the SQL-structure so that you could recognize an object's type by it's ID. For example setting it so that products have an ID in the range 100000-200000, so if an object with the ID 100001 would easily be recognized as a product ID. Currently it's only possible to add one product to the cart at once, so being able to add multiple products to the cart simultaneously would make the site more user-friendly.

Transactions for orders /checkout

Not currently implemented, however the plan is to set the price of the product when it's added to the cart and when the cart goes to checkout it removes the amount from the stock if the remaining stock is ≥ 0 . If the current stock is less than the amount that wants to be checked out, the site will display an error message with the remaining stock.

Rating and comments on products

Not currently implemented, however the idea is for customers and registered customers to be able to leave ratings and comments on product pages which is then saved on our database in it's table *Rating* and writes out on the product page. The columns of Rating is it's own ID, the product ID of the product receiving the rating or comment, the rating, the comment and the ID of the customer or registered customer that made the review. It would be possible to leave a comment without a rating, a rating without a comment and also to leave both.

References

1. [Digitalocean community](#), used for server and flask set up.
2. [Stackoverflow](#), used for details in Python, flask, HTML and SQL.
3. [Flask Documentation](#), used for finding flask commands.
4. [w3schools](#), used for SQL syntax.
5. [Techonthenet](#), used for details in SQL
6. [MySQL Connector/Python Developer Guide](#), used for details in MYSQL

Feedback Sprint 3

- Make it so items added to cart represent the stock of the products
- Structure orders
- Bugfix Customers adding items to cart

Sprint 2

Executive summary

Added an admin page with functions to add, edit or remove products and made the start page more presentable.

User Stories

A user can connect to the site and see available products.

An admin can also connect to the site and see available products. Admins also has access to the hidden admin page where they can see all admin users, products and publish, edit or remove products.

System Architecture

The website is hosted with a public LUDD DUST server that has a local SQL server. The website uses python/Flask to create an app, PyMySQL to access the SQL server and HTML templates as the frontend for the web pages.

Current Backlog

ID	Description	Prio	Effort	Sprint	Status
Database setup	Setup relational database for the web page.	3	Medium	1	done
Web page setup	Write simple test web page.	2	Low	1	done
Cooperation	Setup git repository.	2	Low	1	done
Server setup	Setup server for web page.	1	Medium	1	done
W-D communication	Get data from the database to appear on a web page.	3/1	Medium	1/2	done
Expand customer web page	Flesh out the web page interface for the customers	3	Medium	2	done
Publish Products	Publish products on the web page, linked to the database.	1	High	2	done
Edit Products	Edit aspects of current products on web page and database, such as pricing, stock, description, etc.	2	Medium	2	done
Remove Products	Remove products from web page and database.	2	Low	2	done
Exception handling	Handle exceptions to prevent Internal Server Error page.	1?	Low	3?	x
Cookies	Add cookies that track session data to the web page.	1	High	3	x
Login	Add a login function to the web page.	2	Medium	3	x
Login access	Restrict access to certain web pages based on user login.	2	Medium	3	x
Cart	Add a cart function to the web page.	3	High	3	x
Add to cart	Let customers add products to cart.	1	Medium	-	x
Remove from cart	Let customers remove products from cart	2	Medium	-	x
Buy cart	Let customers proceed to checkout	2	High	-	x
Setup admin interface	Implement the store admin interface	3	Medium	-	x

Database schema

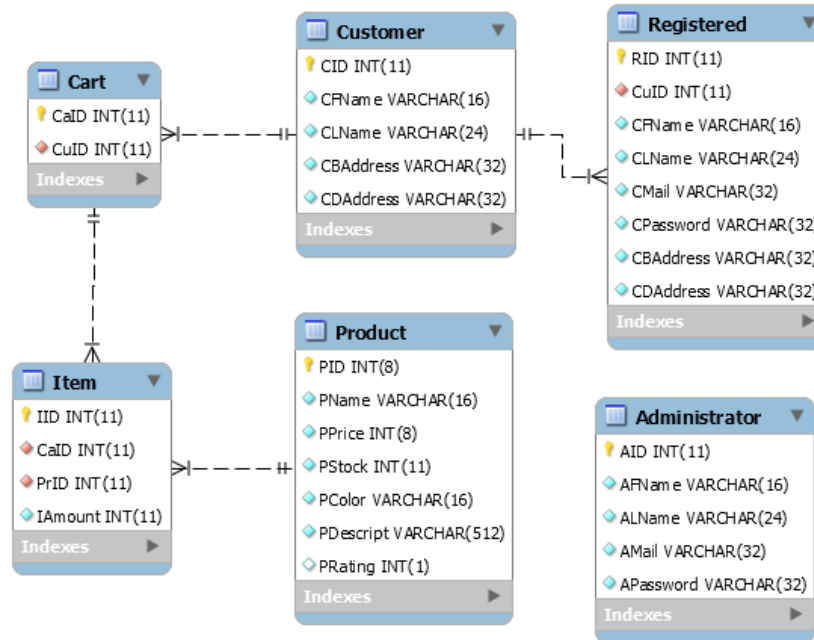


Figure 15: The tables included in the database schema

Link to code and website

[Github repository](#)

[Our website](#) (must be hosted first).

Test case specifications

We tested different kinds of inputs to the removing, publishing and editing functions.

For the publish and edit function we tested these inputs:

1. An input with field(s) with incorrect type(s)
2. An input where required field(s) are missing
3. An input where field(s) are longer than limit
4. An input with all fields and correct types
5. An input with all required fields and correct types

Inputs 1-3 resulted with Internal Server Error, where we should do a try and catch statement and reload the page with an error message.

Inputs 4 and 5 worked as intended and added or edited the correct products and reloaded the current web page.

For the remove function we simply tried inputs with correct id and incorrect id as it has one field that is an int which has a high limit making it hard to test. With a correct id it worked as intended and removed the product with the specified id and reload the page. However if an incorrect id was entered, the site would simply reload the page without removing anything.

Description of the system's limitations and the possibilities for improvements.

The website is still very limited, it's composed of largely static pages with only a few intractable element present throughout the entire site. There are a lot of room for improvements in almost all areas, the most obvious one being styling, all pages of the site are very plain, without images and colors. Many elements on the pages could also be rearranged to produce a more pleasing presentation for users.

More importantly though is the sites functionality, here there is also lot's of room for improvements, for instance the page with the products should perhaps have functionality to click on products and be taken to that specific products page. The page should also (with later sprints) have functionality for purchasing product and adding them to a cart. Other pages will also be needed, such as a customer page where past orders, saved carts, and customer information should be presented to a specific customer. A purchasing page will be need where a customer could actually do what the site is intended for.

A Retrospect of the work done in Sprint 2

Firstly we made it possible to see data from the SQL server on the website. Then we added an admin page with data from the SQL server and forms to publish, edit or remove products. Then we tested the forms to ensure that they worked as intended. After that we cleaned up the the starting page and made it more presentable.

Feedback Sprint 2

- Expand user stories to include the more complete vision of what the website will be able to offer to the user and other actors of the page.

Sprint 1

Executive summary

- Set up a DUST server, created a website using python/flask, set up a git repository and created a relational database on the DUST server.
- Downloaded and installed MySQL on a local machine with a local MySQL server with the purpose of learning SQL scripting in a testing environment.

User Stories

Users can connect to a simple web page and click on a link to the other page of the website.

Current Backlog

ID	Description	Prio	Effort	Sprint	Status
Database setup	Setup relational database for the web page.	3	Medium	1	done
Web page setup	Write simple test web page.	2	Low	1	done
Cooperation	Setup git repository.	2	Low	1	done
Server setup	Setup server for web page.	1	Medium	1	done
W-D communication	Get data from the database to appear on a web page.	3/1	Medium	1/2	started
Expand customer web page	Flesh out the web page interface for the customers	3	Medium	2	x
Publish Products	Publish products on the web page, linked to the database.	1	High	2	x
Edit Products	Edit aspects of current products on web page and database, such as pricing, stock, description, etc.	2	Medium	2	x
Remove Products	Remove products from web page and database.	2	Low	2	x
Cart	Add a cart function to the web page.	1	High	-	x
Add to cart	Let customers add products to cart.	1	Medium	-	x
Remove from cart	Let customers remove products from cart	2	Medium	-	x
Buy cart	Let customers proceed to checkout	2	High	-	x
Setup admin interface	Implement the store admin interface	3	Medium	-	x

Database schema

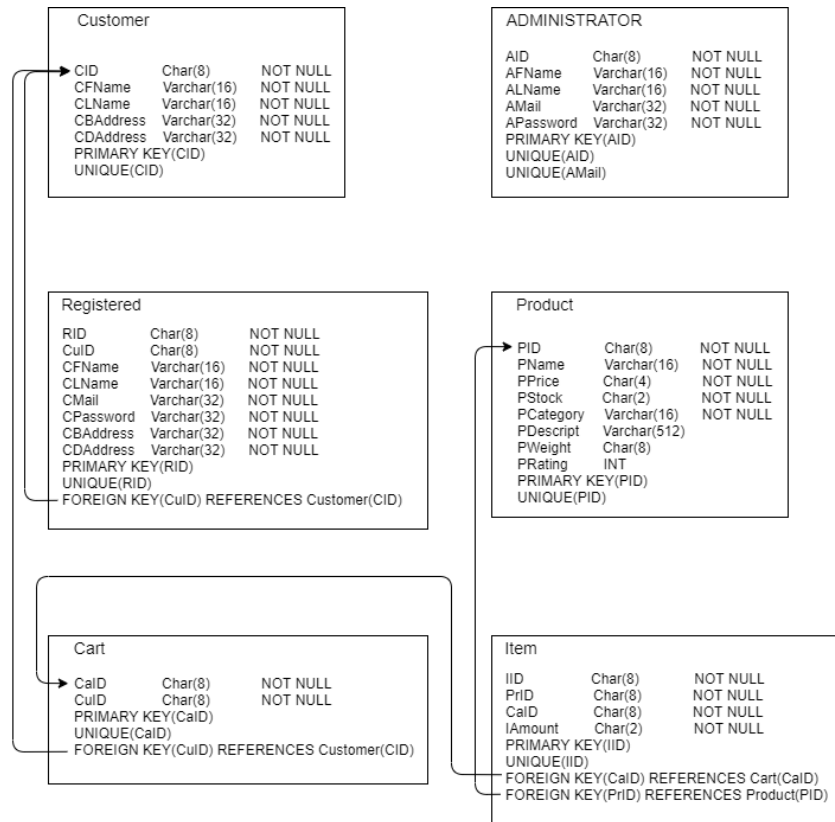


Figure 16: The tables included in the database schema

Link to code

[Github repository](#)

Test case specifications

Can anyone access the server web page.

Description of the system's limitations and the possibilities for improvements.

Currently a simple web page with text and an href, lacks all features of an e-commerce site. It's not currently connected to a database, making it unable to store and upload data. Improvements would be to connect the site to the database and implement features and a proper interface on the web page. These features could for example be to publish and remove products, adding user logins or adding a shopping cart system.

A Retrospect of the work done in Sprint 1

We have set up a DUST server, a local SQL server for script testing, made two simple web pages and created a git repository for the web pages, and later the encompassing project code.

Feedback Sprint 1

- Report structure, headlines instead of list.
- System architecture included in the next one (Frameworks, systems used, graphic representation of the system as a whole).
- Names and group number in report and file name.