D0018E Sprint 3

An Internet e-commerce site using an SQL database

Authors
lelrek-1@student.ltu.se
sunlar-3@student.ltu.se

March 1, 2024

1 Summary

The project involves the development of an e-commerce website with features tailored to both customers and administrators. Key functionalities include account creation and login for customers, product reviews, shopping cart management, purchase completion, and product management for administrators.

During the project discussion happened with group 16.

2 User stories

As a	I want to be able to	so that
Customer	create an account and log into it	I can save my shopping cart
Customer	have an account	I can create reviews for a product
Customer	add and remove items in the shopping cart	so that I can see which product I have selected
Customer	complete a purchase	I receive a receipt
Customer	add rating	a product get a rating
Admin	add new products	customers can buy them
Admin	to edit products	product information and stock is correct
Admin	answer a review	context can be added to a review
Admin	see created orders	I can confirm orders

Table 1: Tabel of user stories

Tabel 1 contains the user stories for the two different roles of customer and admin that use the website.

3 System architecture

3.1 Database

Our choice of database is MySQL which is hosted on AWS. Image 1 shows the database schema in use.

3.2 Web server

For the web server, Python Flask is used, in development it is run on local machines, and migrated to AWS Ubuntu [4] when it leaves development. The connection to the MySQL database is done with the flask_mysqldb library in Python. The creation of queries is made by the use of the library MySQLdb and its cursors.
To make the design of the webpage easier Bootstrap 5 is used.

4 A database schema

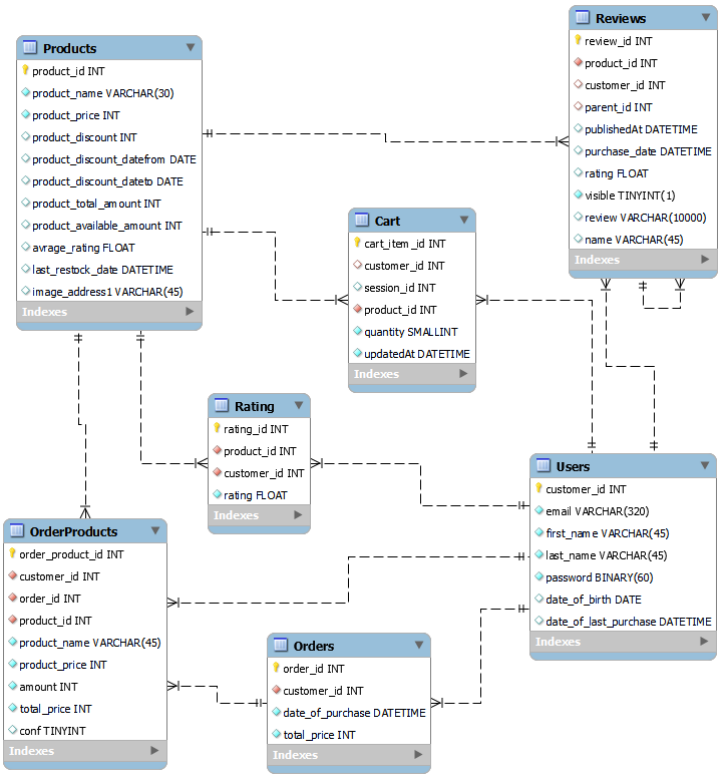

Figure 1: E-R Diagram of the database schema in use

As seen in figure 1 the schema contains seven tables and shows each table's primary key and how foreign keys are connected between the different tables.

5 Test case specification

The test cases used stem from the user stories displayed in table 1, a quick overview of the test cases are displayed in table 2.

Test as	**Action**	**Expected result**
Customer	create a test account and login	possible to login
Customer	login, review a product, previously purchase	review is displayed for the product
Customer	login, review a product, without previously purchase	error needs to have purchase product
Customer	try to review a product without login	error needs to be logged in
Customer	add and remove items in the shopping cart	see changes in the shopping cart
Customer	try to add more of a product than exists in stock	get error that not enough in stock
Customer	complete purchase	get a receipt, double check database for order
Customer	try to complete a purchase without being logged in	get redirected to register page
Admin	add new product	a new product is created in server and on website
Admin	edit product	product information is changed on server
Admin	confirm single order where stock exists	order gets confirmed and disappears from list
Admin	confirm single order where no stock exists	error not enough of product in stock
Admin	confirm all orders	if all products in stock all orders confirmed
Admin	confirm all orders	if some products are not in stock cancel order confirmation + error

Table 2: Table of test cases

5.1 Demo

Step 1: Add one or multiple products to the shopping cart and test what happens if we try to order more products than are in stock.

Step 2: Clear an individual item in the cart or clear all items in the cart.

Step 3: Try to checkout without being logged in and when you are logged in.

Step 4: Try creating an account and then log in to it.

Step 5: Go to the product page as a customer and add reviews (logged in or not) then as an admin.

Step 6: Go to the admin page and individual products of order confirm orders then do the same if the stock gets too low.

Step 7: Check so it displays an updated stock on the product list.

Step 8: Add a product - if we have time

6 Transactions

To purchase a product a customer needs to first add the products to the shopping cart. The shopping cart is contained in the database to give a customer the possibility to return to its cart at a later date, this feature is only possible for logged-in customers.
When adding a product to the cart a check is done to see if a product is in stock, returning an error if not.
To continue the transaction the customer needs to checkout, to perform a checkout the customer needs to be logged in and gets sent to the login page with a prompt if not.
When the checkout has been performed the admin needs to confirm the order, which is done on the admin page, this is when the product stock gets reduced by the amount specified in the order and this is the finished transaction.

7 Grading and commenting

To comment on a product there are two main requirements, first that the customer is logged in, and second that the customer has previously purchased the product it is trying to review. When a customer creates a review it also gets prompted to add a rating to the product this rating is then displayed along the review and used to give the product an overall score. The date of the review and who did the review is also visible.

After a review is created it becomes possible to reply to that review, to reply to a review the previous requirements that exist for a review still need to be fulfilled. However, replying to a review does not prompt you to add a rating to the product.
There is one exception to the requirements for replying and that is for the admin that can always reply.

If a customer only wants to add a rating that is also possible, the same requirements of being logged in and having purchased the product before still stand.

8 Limitations and possibilities for improvement

One limitation of the solution is that to complete a purchase the customer needs to be logged in, this creates the possibility of losing customers who don't want to create accounts. A possible solution to this problem could be that a customer is created in our database without a password so if the customer decides to later create an account they can view their history.

The solution described for this problem would require one improvement to the user creation process, with this being the verification of the email, since with the current implementation there is no verification meaning just knowing the email address is enough to create a user account with that email.

While security has not been a main concern during development and in the course, since MySQLdb.cursors implements protection against injection attacks by providing parameterized queries [1], there exists a basic defense against it.

Another possible improvement area is how the current stock gets handled. If two users add a product with a small amount of stock to their shopping cart and complete the checkout around the same time, then when the first order gets confirmed by the admin the stock might run out so the second order might have to wait for the next shipping before their order gets confirmed.

The possibility of a user seeing its old order history could be created, as first envisioned but it was dropped to limit the scope of the project.

8.1 Bugs during demo

When a customer adds a product to their shopping cart the customer can use a free text field to add the number of products they want to purchase, with the implementation used during the demo this provided a vector of attack. If the customer adds a negative number of a product they can get a "discount" on their order since the price of this negative amount is also negative. This has since been fixed by controlling if the input is an integer and greater than 0.

During the demo confirming a single order instead confirmed all the orders, the reason for this was that the function called when confirming a single order was the function for confirming all orders, this has since been fixed.

Adding a reply as an admin goes under the wrong name displayed this seems to be a problem only when the webserver runs on AWS as when running locally works as intended and the name becomes admin. This might have something to do with how sessions get handled by the webserver. This is probably an area that could be improved by using a better way to control a user's access rights.

9 Backlog

Figure 2 shows the overall backlog

Story ID	Story	Task ID	Task	Satus	Bercenden	Prioritet
1	***Titel:*** Setup MYSQL Server ***Intended use:*** To be used for our e-commerce site ***Desired properties:*** Hosted on AWS ***Testcase:*** Connect to the server and create table.	1.1	***To do:*** Create server on AWS and schema			High
		1.2	***To do:*** Create tabels for User account, orders, products		1.1	
		1.3	***To do:*** Set up access rights for different accounts.		1.1	
2	***Titel:*** Customer login service ***Intended use:*** Give the possiblility to create accounts and login ***Desired properties:*** Hash and salt the password. ***Testcase:*** Test creating and login in with an account	2.1	***To do:*** Allows a user to create an account		1.2	High
		2.2	***To do:*** Allows a user to login			
		2.3	***To do:*** Create a function that encypts the user password		2.1	
3	***Titel:*** Order flow ***Intended use:*** Control the purchase and the available products ***Desired properties:*** Give the possibility of controlling the inventory ***Testcase:*** Add products to inventory and create orders	3.1	***To do:*** Allows a user to purchase a product		1.2	High
		3.2	***To do:*** Make it possible for admin/inventory manager to add/remove product from stock		1.2	
		3.3	***To do:*** Create statistics of purchases		1.2	

Figure 2: continues down below

4	***Titel:*** Frontend webpage ***Intended use:*** An interface to complete purchases, login and product management ***Desired properties:*** Easy and simple webpage ***Testcase:*** Do purchase on the webpage and control in database	4.1	***To do:*** Login page			Medium
		4.2	***To do:*** Product page			
		4.3	***To do:*** Purchase page			
		4.4	***To do:*** Management page			
		4.5	***To do:*** Account page			
5	***Titel:*** Product review feature ***Intended use:*** Give the possibility to review products ***Desired properties:*** Display reviews and average rating ***Testcase:*** Test writing a review and see if it displays	5.1	***To do:*** Review table (DB)		1.1	Low
		5.2	***To do:*** Review page (frontend)		5.1	
		5.3	***To do:*** Function that average the rating for a product and displays it on product page		5.1	
		5.4	***To do:*** Limit reviews to only be possible for products purchased		1.2 5.1	
		5.5	***To do:*** Add possibility to "ban" users from making reviews		1.2 5.1	
6	***Titel:*** Shopping kart ***Intended use:*** Make it possible to add multiple items before purchase ***Desired properties:*** Have the booking of items at purchase ***Testcase:*** Add and remove items from shopping kart then buy	6.1	***To do:*** Shopping kart on frontend			Low
		6.2	***To do:*** Handel multiple orders		1.2 3.1	
		6.3	***To do:*** Handel multiple orders make it possible to remove items from shopping kart		6.1	
7						

Figure 3: Sprint 1: Story 1 finished and started 2 and 4 Sprint 2: Story 2 finished and all other stories started on Sprint 3: Finished all remaining stories and depreciated some tasks

Github repository. All source code for the project is also found there.

10 References

References

[1] Knowledge-base sql_injection

[2] w3school MySQL

[3] w3school HTML

[4] Youtube AWS Ubuntu setup