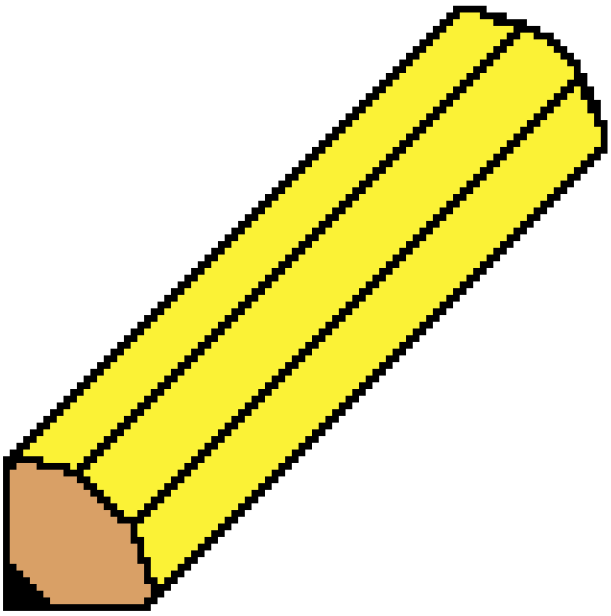
Helixpen, a e-commerce site based on a SQL database project



Group 12
Isak Lundmark, lunisa-9@student.ltu.se
Isak Lundström, isalun-9@student.ltu.se
Ted Selegran, tedsel-9@student.ltu.se
Department of Engineering Sciences and Mathematics

LULEÅ TEKNISKA UNIVERSITET

January 4, 2022

Contents

1 Executive summary

This is a the final version of a continuous report for the lab assignment in the LTU course Database Technology. The goal of the assignment was to create a e-commerce website where the application consist of many requirements. Some of the requirements set where that a admin should have privileges to add and change products, have access and can manage user accounts, see, change and send orders and that a customer can add and remove products to cart, see and change own user information, see, checkout and order whats in cart and lastly see all available products on the website.

The final implementation of the system is based on MySQL, Node.js, Express.js, EJS and CSS and hosted at a DUST server located at LTU maintained by LUDD.

The assingment was divided in four sprints where good progress was made in every sprint and the project was almost complete after the third sprint. Only some bugs and fixing of the report was left to fix before the assignment could be called finished. Retrospects of every sprint can be viewed at the end of this report.

A live backlog of upcoming, in progress and completed tasks was used on a GitHub repository. All file sharing was done using the same repository. A couple simple test cases was defined to test the applications functions.

2 User-Stories

2.1 Administrator

1. As an administrator I can manage products on the page.
2. As an administrator I can get access to customer information.
3. As an administrator I can remove a customer.
4. As an administrator I can see and send orders.
5. As an administrator I can manage reviews.

2.2 Customer

1. As a customer I can create an account, so that I can log in to it.
2. As a customer I can browse products and add/remove them to/from my cart.
3. As a customer I can order what is in my cart.
4. As a customer I can change my own account information.
5. As a customer I can see review for a product and write my own review.
6. As a customer I can remove my account.

3 Requirements

3.1 Administrator

1. Log in with admin account
2. Add/Change products
3. Access to user information
4. Remove an user
5. See/Send/Complete orders
6. Manage user accounts
7. See all reviews
8. Able to delete a review

3.2 Customer

1. Create a account
2. Log in with created account
3. See all available products on the website
4. Add/Remove products to cart
5. See/Change own user information
6. See/Checkout/Buy items in cart
7. See/Make reviews on a product
8. Delete created account

4 System architecture description

The system is built using MySQL and Node.js. The backend is located and hosted at a DUST server maintained by LTUs academic computer society LUDD. Node.js is used to query to and from the database. The Express.js framework is used to send data to the webpage and to the database. The webpage is built using EJS and is styled with CSS. See figure 1.

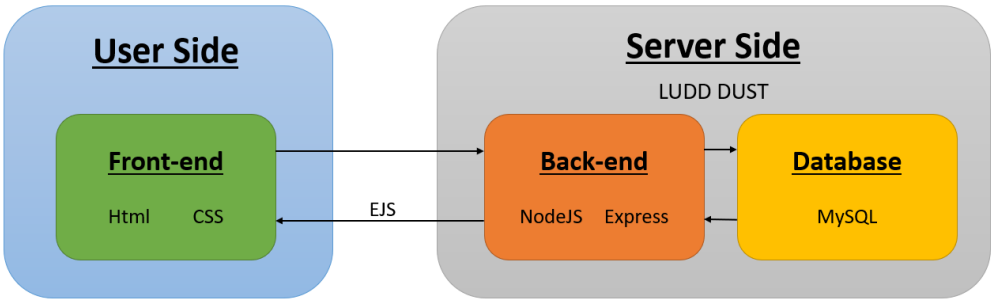


Figure 1: Simplified diagram of the system architecture.

5 Backlog

Snapshot from the backlog can be seen in figure 2.

6 Database schema (E-R Diagram)

The database schema, see figure 3, is defined by five tables, 'Users', 'Products', 'Cart', 'Orders' and 'Reviews'. The Users table contains all users on the site, customers and admins and only the isAdmin attribute. A cart table is used instead of cookies since cookies would not persist between devices and cookies may eventually expire. Orders does not have any references or foreign keys because even if Users are deleted their orders should still remain in the system.

7 Cooperation

All the files used and the backlog is accessed and shared on a GitHub repository[1].

[1] https://github.com/IsakLundstrom/D0018E_LabAssignment

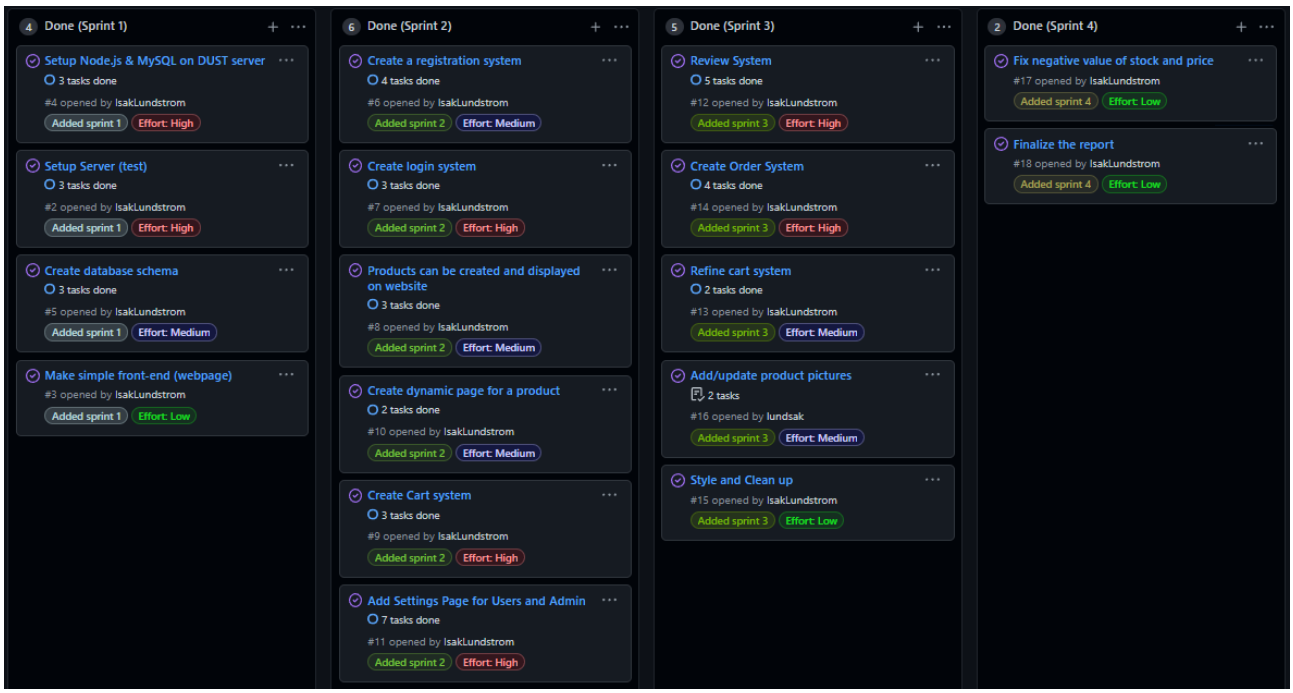


Figure 2: Snapshot of the used backlog.

8 Test cases

8.1 Check connection to website

- Start the application remotely.
- Enter the URL in a web browser.
- See that the website is accessible.

8.2 Registration and login test

- Go to registration page on the website.
- Enter information in the forms.
- Login with the same information to confirm that the registration was successful.
- Go to the settings page and press "Delete Account" to delete the created account.

8.3 Add product to cart test

- Go to the login page and enter an username and password to login to an account.
- Press the "Add to Cart" button to add a product to cart.
- Go to cart page to confirm that the correct product was added to the cart.
- Press the "X" button to remove the product from the cart.
- Press order when the cart is filled with desired items to send an order to the admins.

8.4 User settings test

- Go to login and enter an username and password to login to an account.
- Go to settings page to see the accounts personal information.
- Change some fields in the form and press the "Update" button to update the accounts personal information.
- See Orders table if orders previously have been made.

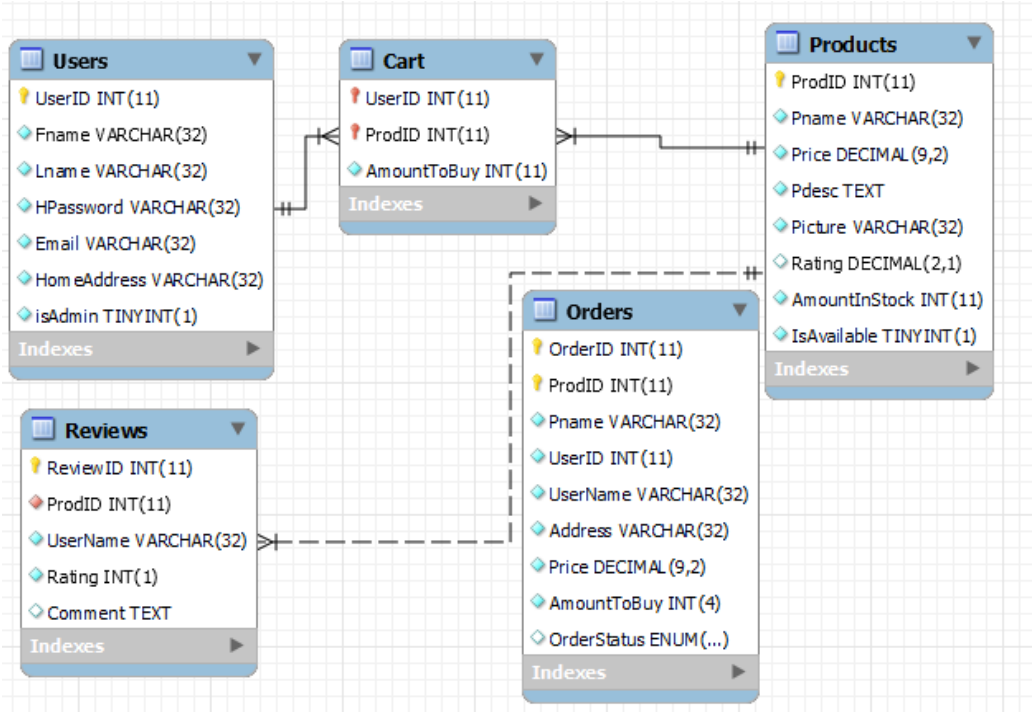


Figure 3: The final database design.

8.5 Admin settings test

- Create an admin from the server side.
- Go to profile page to observe the admin settings.
- See all product and choose which products should be visible on the site using the availability button.
- Add a product to the page using the 'Add a product' section.
- Remove a product using the 'remove a product' section.
- Change the information or image on an existing product.
- Observe the users in the 'users' section.
- Check that it is possible to remove a user from the database using the 'users' table found in /setting.

8.6 Review test

- Login and go to a product to observe the review box.
- Write a review, give a rating and press submit.
- Refresh the website to observe the review.
- Furthermore, observe the average rating on the product has changed.
- Login with an admin, go to a product and then see that you can remove a review.

8.7 Order test

- Login as an admin and go to the admin settings.
- In the order section, click "Go to all orders" to see all orders purchased by the users.
- Click on the truck icon in the "To ship" section to send the order to the "Shipped" section. This means the order is on its way to the user.

- Click on the check mark icon in the "Shipped" section to send the order to the "Completed" section. This means the product has arrived.
- Observe the completed section which contains all completed orders.

9 Limitations and possible improvements

Some limitations in the system is that a product can never really be deleted from the system as the orders rely on there being a ProdID with items in stock. To change this behavior the whole order system, with taking items from the stock when shipping, has to be reworked or a completely different approach have to be taken.

A implementation that has a possible improvement is the order system. Since the OrderID is calculated by counting every distinct value of OrderID any order cannot be removed. This makes it impossible right now to cancel a order. One solution which is not implemented is to allow users to change the OrderStatus to 'Canceled' if the order is not shipped yet.

There still exists some uncaught errors that will crash the server. However under normal circumstances those error will not occur. Those errors should only happen when a malicious user tries to manipulate the website.

One problem with the system is that in some places GET-requests are used where POST-requests should have been used instead. This can cause some weird problems because information is stored in the URL. Here again a malicious user can do bad-natured activities, this time by exploiting the URL.

10 Transactions

First, there is a transaction when an user places an order to make sure that the order gets the correct OrderID. When a user checkout, the items is sent to the Orders table and no items are reserved from the stock. A admin can then decide which items from the order to send. It is not until this last step a transaction is used to reserve products from stock. If the amount to order is greater than amount in stock the admin can not confirm the order.

11 Retrospects

11.1 Sprint 1

In sprint 1 we have the majority of the time studied and looked at different sources for information about how to setup a MySQL database on a server and how to uses Node.js to help us reach our goal of the e-commerce website. After learning the basics of MySQL and setting up Node.js on the server, which is a LUDD DUST server, we experimented with it to make a simple front-end for testing purposes. Then we decided to create the initial database design and it took two iterations to finalize it for this sprint. We learned a way of how to display content on the web using Express.js and EJS.

11.2 Sprint 2

Firstly, we studied how we would retrieve information from the website to the server. This was fairly easy to accomplish and we used the gained knowledge to create a registration system. The system takes user inputs and stores the data in a table in our MySQL database. Then to crate a login system with the created users we needed a way for the application to keep a user logged in the whole session. Session cookies in Node.js was used to solve this problem.

Products was a first added through the command line in the server and displayed on the homepage only. We then added a dynamic product page for any product to use. The page uses the url to determine which product to illustrate.

The cart system was then implemented by using the session-user-id and the product-id to connect the products in the cart to a specific user. A settings page was then created, where a user can update their account information and delete the account if desired. The last task of this sprint was to implement a admin mode where a admin can add and delete products to the database through the website and manage other users.

Through the later parts of the sprint a lot of styling of the front-end was continuously refined and processed. Also the database evolved multiple times this sprint.

11.3 Sprint 3

In this sprint the first task on the backlog was to implement a review system. This was pretty similar of adding a product, which we completed in sprint 2, which made the task fairly trivial. The review takes a grade from 1

to 5 and takes a comment. After the comment is published it is visible for that product on the product page. The average rating is calculated every time the site is loaded to make the rating as consistent as possible.

Secondly, the order system was created by first moving the items in the Cart table to the Orders table. The content had to be copied because the price is set in stone when the order is sent into process by the customer. A customer can view their orders on the settings page and see the status of every product. There are three statuses: 'ToShip', 'Shipped' and 'Completed'. These statuses can be changed by a admin from the orders page when the products are sent and when the orders is finalized.

The cart page was not really finalized since you could only removed all products from the cart and not easily change the amount of items to buy. To solve this a minus and plus button for on the cart page was added for every product in the cart.

We wanted to be able to directly be able to add an image from the browser to the site. So with some gathering of information and thinking of how this was gonna fit out implementation already the function was successfully added. Even .gif is supported.

Over the whole sprint styling with EJS and CSS was being done and going through many interactions. Some reformatting and comments where added to the code to make it more readable and user-friendly.

11.4 Sprint 4

In the this final sprint the only thing that was done was fixing a bug that allowed product stock and product price to be negative.

12 References

Important

- To learn MySQL syntax:
  https://www.w3schools.com/sql/default.asp
- To learn syntax of Node.js and MySQL:
  https://www.w3schools.com/nodejs/nodejs_mysql.asp
- To understand Express.js:
  https://expressjs.com/
- To explain EJS, and example:
  https://codingstatus.com/how-to-display-data-from-mysql-database-table-in-node-js/

Informal

- To learn cookies for login:
  https://www.section.io/engineering-education/session-management-in-nodejs-using-expressjs-and-express-session/
- Minor help with login error:
  https://stackoverflow.com/questions/19035373/how-do-i-redirect-in-expressjs-while-passing-some-context
- Learn include() in EJS:
  https://www.digitalocean.com/community/tutorials/how-to-use-ejs-to-template-your-node-application
- Upload images:
  https://stackoverflow.com/questions/15772394/how-to-upload-display-and-save-images-using-node-js-and-express