D0018E - Database lab report
Lab group 11

Markus Blomqvist —— amuobi-4@student.ltu.se
Ivar Jönsson —— ivajns-9@student.ltu.se
Johnny Lam —— lamjoh-9@student.ltu.se

March 20, 2023

Contents

1 Summary

The goal of the project is to make an e-commerce site using an SQL database as part of the tech stack. The e-commerce site will be selling wine and allow the user to search for a specific wine, filter on some attributes of the wine and review them. Currently, supported functionality includes:

- Register new user.
- Login with an existing user.
- Logout.
- Make a user admin.
- Admin can create products.
- Admin can create categories.
- Admin can create category groups.
- Admin can modify prices.
- Admin can modify stocks.
- Admin can modify and delete reviews.
- Admin can manage orders.
- Admin can manage categories.
- Admin can manage category groups.
- Admin can modify accounts.
- Admin can view and modify users' reviews
- User can open a specific product, displaying more detailed information
- User can look at all products by clicking on different numbered pages
- User can search for a product based on its name
- User can filter products on categories
- User can filter search results by categories.
- User can sort the products based on some filters
- User gets product recommendations based on their basket

- User can view a product ”spotlight” by clicking the product image
- User can view their previous orders.
- Users can view and modify their profile.
- Users can add items to their cart.
- Users can checkout their cart.
- Users can modify the content of their cart.
- User can create and modify a review.

2 User stories

The system currently has 2 roles: admin and user.

- User story 1: Admins can use the website to login/logout, change prices, increase the stock, decrease the stock if there is any problem with the wine, manage orders and categories, delete or add more types of wine to sell and modify and delete reviews.
- User story 2: Customers visiting the website to buy wine can login/logout, look at recommended wines, search for a particular wine to buy, filter and sort results, add wines to the basket, do a checkout and make a 1-5 star review.

3 System architecture

The tech stack is structured like this:

- Database: SQL.
- Backend: Python, Flask.
- Frontend: Jinja 2 templating, HTML layout and JS for dynamic layout.

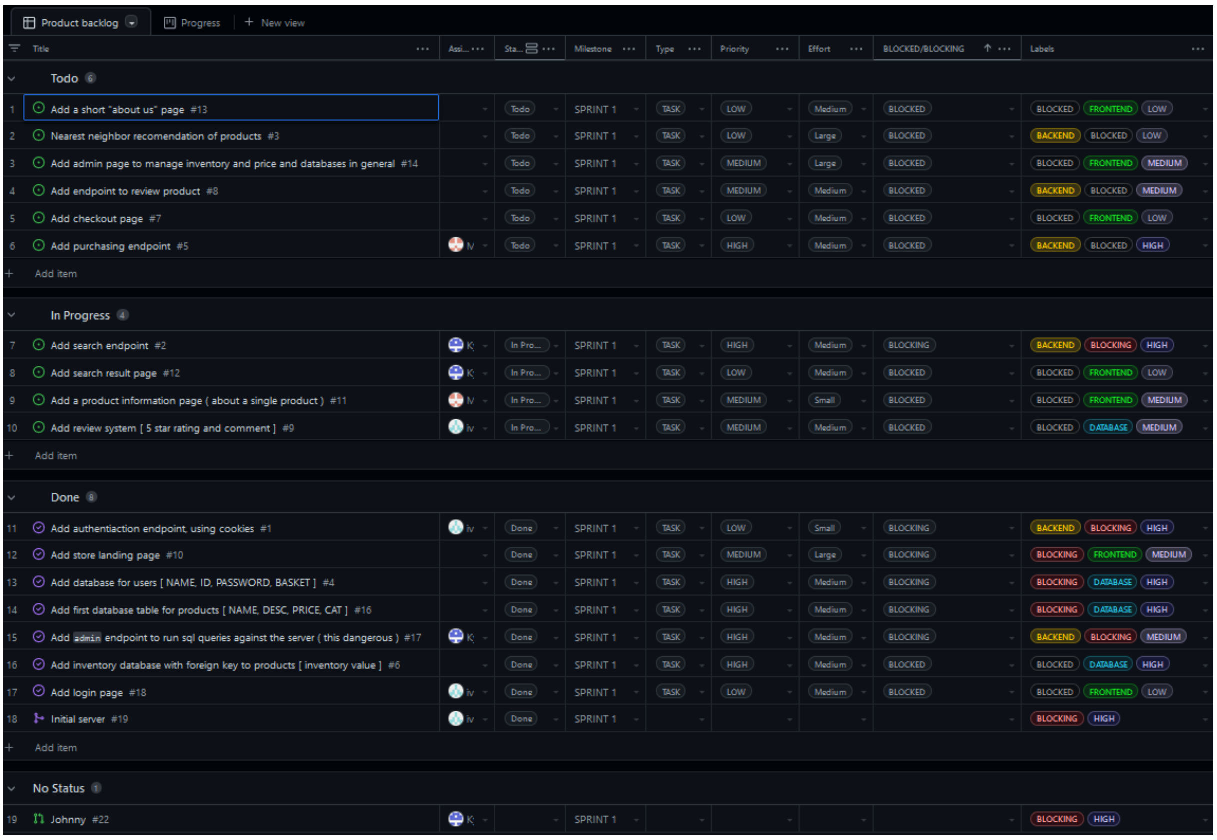
The system is composed of 2 major components, the database and the backend server. The backend server talks to the database using SQL commands passed over an SSH tunnel, ensuring data integrity and data safety when executing commands on protected data. To ensure privacy the users’ passwords are stored as sha256 hashed strings instead of plaintext.

4 Assumptions

The system is not built around a lot of assumptions. We do, however, assume that the back-end server is not on the same physical or virtual machine as the database server. Moreover, we assume that the users of the website all have good intentions since we have not really put a lot of focus on security and that the number of users is limited since the current system does not implement sharding or load balancing.

5 Backlog

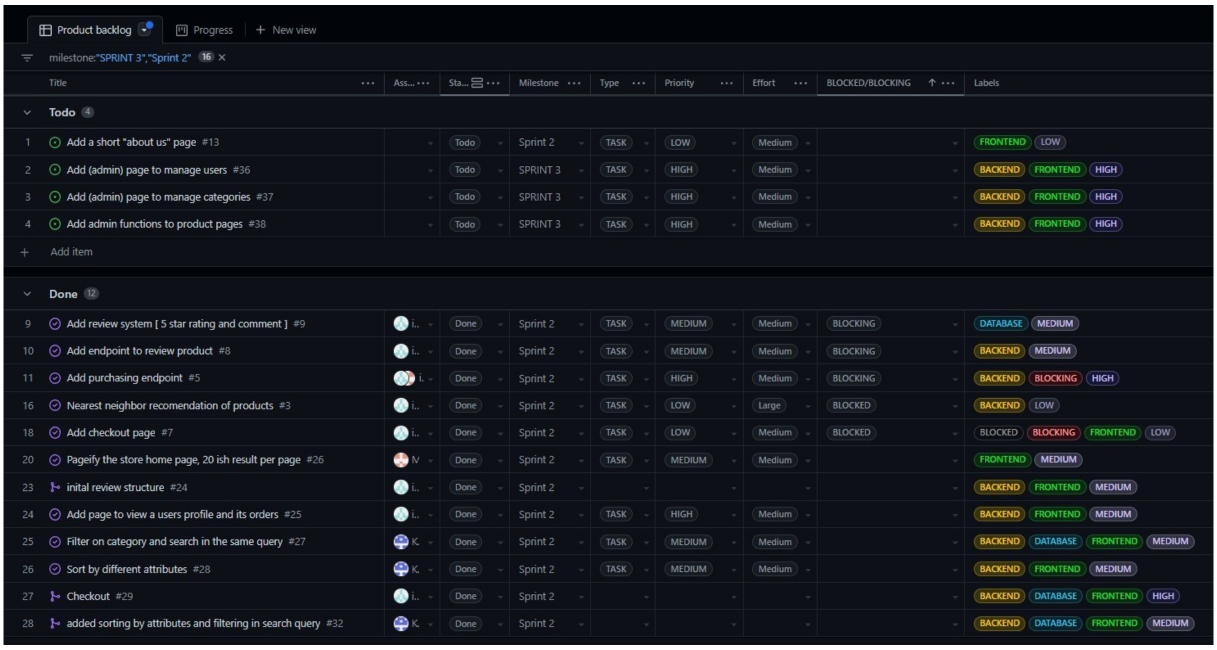
Link to the backlog on GitHub can be found here: GitHub



Product backlog | Progress | + New view

	Title	Assi...	Sta...	Milestone	Type	Priority	Effort	BLOCKED/BLOCKING	Labels
	Todo 6								
1	Add a short "about us" page #13		Todo	SPRINT 1	TASK	LOW	Medium	BLOCKED	BLOCKED FRONTEND LOW
2	Nearest neighbor recomendation of products #3		Todo	SPRINT 1	TASK	LOW	Large	BLOCKED	BACKEND BLOCKED LOW
3	Add admin page to manage inventory and price and databases in general #14		Todo	SPRINT 1	TASK	MEDIUM	Large	BLOCKED	BLOCKED FRONTEND MEDIUM
4	Add endpoint to review product #8		Todo	SPRINT 1	TASK	MEDIUM	Medium	BLOCKED	BACKEND BLOCKED MEDIUM
5	Add checkout page #7		Todo	SPRINT 1	TASK	LOW	Medium	BLOCKED	BLOCKED FRONTEND LOW
6	Add purchasing endpoint #5	M	Todo	SPRINT 1	TASK	HIGH	Medium	BLOCKED	BACKEND BLOCKED HIGH
+	Add item								
	In Progress 4								
7	Add search endpoint #2	K	In Pro...	SPRINT 1	TASK	HIGH	Medium	BLOCKING	BACKEND BLOCKING HIGH
8	Add search result page #12	K	In Pro...	SPRINT 1	TASK	LOW	Medium	BLOCKED	BLOCKED FRONTEND LOW
9	Add a product information page (about a single product) #11	M	In Pro...	SPRINT 1	TASK	MEDIUM	Small	BLOCKED	BLOCKED FRONTEND MEDIUM
10	Add review system [5 star rating and comment] #9	iv	In Pro...	SPRINT 1	TASK	MEDIUM	Medium	BLOCKED	BLOCKED DATABASE MEDIUM
+	Add item								
	Done 8								
11	Add authentiaction endpoint, using cookies #1	iv	Done	SPRINT 1	TASK	LOW	Small	BLOCKING	BACKEND BLOCKING HIGH
12	Add store landing page #10		Done	SPRINT 1	TASK	MEDIUM	Large	BLOCKING	BLOCKING FRONTEND MEDIUM
13	Add database for users [NAME, ID, PASSWORD, BASKET] #4		Done	SPRINT 1	TASK	HIGH	Medium	BLOCKING	BLOCKING DATABASE HIGH
14	Add first database table for products [NAME, DESC, PRICE, CAT] #16		Done	SPRINT 1	TASK	HIGH	Medium	BLOCKING	BLOCKING DATABASE HIGH
15	Add `admin` endpoint to run sql queries against the server (this dangerous) #17	K	Done	SPRINT 1	TASK	HIGH	Medium	BLOCKING	BACKEND BLOCKING MEDIUM
16	Add inventory database with foreign key to products [inventory value] #6		Done	SPRINT 1	TASK	HIGH	Medium	BLOCKED	BLOCKED DATABASE HIGH
17	Add login page #18	iv	Done	SPRINT 1	TASK	LOW	Medium	BLOCKED	BLOCKED FRONTEND LOW
18	Initial server #19	iv	Done	SPRINT 1					BLOCKING HIGH
+	Add item								
	No Status 1								
19	Johnny #22	K		SPRINT 1					BLOCKING HIGH

Figure 1: The backlog at the end of sprint 1.

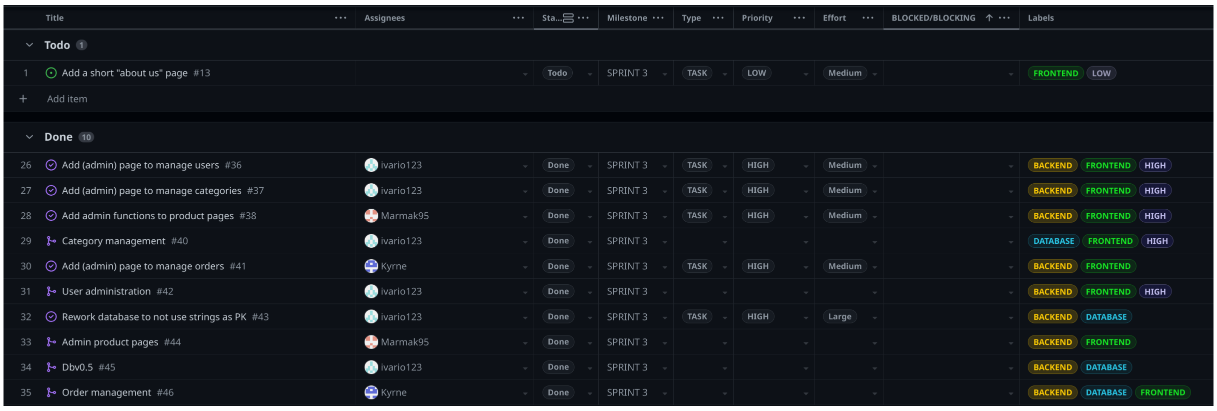


Product backlog | Progress | + New view

milestone:"SPRINT 3","Sprint 2" 16

	Title	Ass...	Sta...	Milestone	Type	Priority	Effort	BLOCKED/BLOCKING	Labels
	Todo 4								
1	Add a short "about us" page #13		Todo	Sprint 2	TASK	LOW	Medium		FRONTEND LOW
2	Add (admin) page to manage users #36		Todo	SPRINT 3	TASK	HIGH	Medium		BACKEND FRONTEND HIGH
3	Add (admin) page to manage categories #37		Todo	SPRINT 3	TASK	HIGH	Medium		BACKEND FRONTEND HIGH
4	Add admin functions to product pages #38		Todo	SPRINT 3	TASK	HIGH	Medium		BACKEND FRONTEND HIGH
+	Add item								
	Done 12								
9	Add review system [5 star rating and comment] #9	i..	Done	Sprint 2	TASK	MEDIUM	Medium	BLOCKING	DATABASE MEDIUM
10	Add endpoint to review product #8	i..	Done	Sprint 2	TASK	MEDIUM	Medium	BLOCKING	BACKEND MEDIUM
11	Add purchasing endpoint #5	i.	Done	Sprint 2	TASK	HIGH	Medium	BLOCKING	BACKEND BLOCKING HIGH
16	Nearest neighbor recomendation of products #3	i..	Done	Sprint 2	TASK	LOW	Large	BLOCKED	BACKEND LOW
18	Add checkout page #7	i..	Done	Sprint 2	TASK	LOW	Medium	BLOCKED	BLOCKED BLOCKING FRONTEND LOW
20	Pageify the store home page, 20 ish result per page #26	IV	Done	Sprint 2	TASK	MEDIUM	Medium		FRONTEND MEDIUM
23	inital review structure #24	i..	Done	Sprint 2					BACKEND FRONTEND MEDIUM
24	Add page to view a users profile and its orders #25	i..	Done	Sprint 2	TASK	HIGH	Medium		BACKEND FRONTEND MEDIUM
25	Filter on category and search in the same query #27	K.	Done	Sprint 2	TASK	MEDIUM	Medium		BACKEND DATABASE FRONTEND MEDIUM
26	Sort by different attributes #28	K.	Done	Sprint 2	TASK	MEDIUM	Medium		BACKEND FRONTEND MEDIUM
27	Checkout #29	i..	Done	Sprint 2					BACKEND DATABASE FRONTEND HIGH
28	added sorting by attributes and filtering in search query #32	K.	Done	Sprint 2					BACKEND DATABASE FRONTEND MEDIUM

Figure 2: The backlog at the end of sprint 2.



	Title	Assignees	Sta...	Milestone	Type	Priority	Effort	BLOCKED/BLOCKING	Labels
	Todo 1								
1	Add a short "about us" page #13		Todo	SPRINT 3	TASK	LOW	Medium		FRONTEND LOW
+	Add item								
	Done 10								
26	Add (admin) page to manage users #36	ivario123	Done	SPRINT 3	TASK	HIGH	Medium		BACKEND FRONTEND HIGH
27	Add (admin) page to manage categories #37	ivario123	Done	SPRINT 3	TASK	HIGH	Medium		BACKEND FRONTEND HIGH
28	Add admin functions to product pages #38	Marmak95	Done	SPRINT 3	TASK	HIGH	Medium		BACKEND FRONTEND HIGH
29	Category management #40	ivario123	Done	SPRINT 3					DATABASE FRONTEND HIGH
30	Add (admin) page to manage orders #41	Kyrne	Done	SPRINT 3	TASK	HIGH	Medium		BACKEND FRONTEND
31	User administration #42	ivario123	Done	SPRINT 3					BACKEND FRONTEND HIGH
32	Rework database to not use strings as PK #43	ivario123	Done	SPRINT 3	TASK	HIGH	Large		BACKEND DATABASE
33	Admin product pages #44	Marmak95	Done	SPRINT 3					BACKEND FRONTEND
34	Dbv0.5 #45	ivario123	Done	SPRINT 3					BACKEND DATABASE
35	Order management #46	Kyrne	Done	SPRINT 3					BACKEND DATABASE FRONTEND

Figure 3: The backlog at the end of sprint 3.

6 Database schema

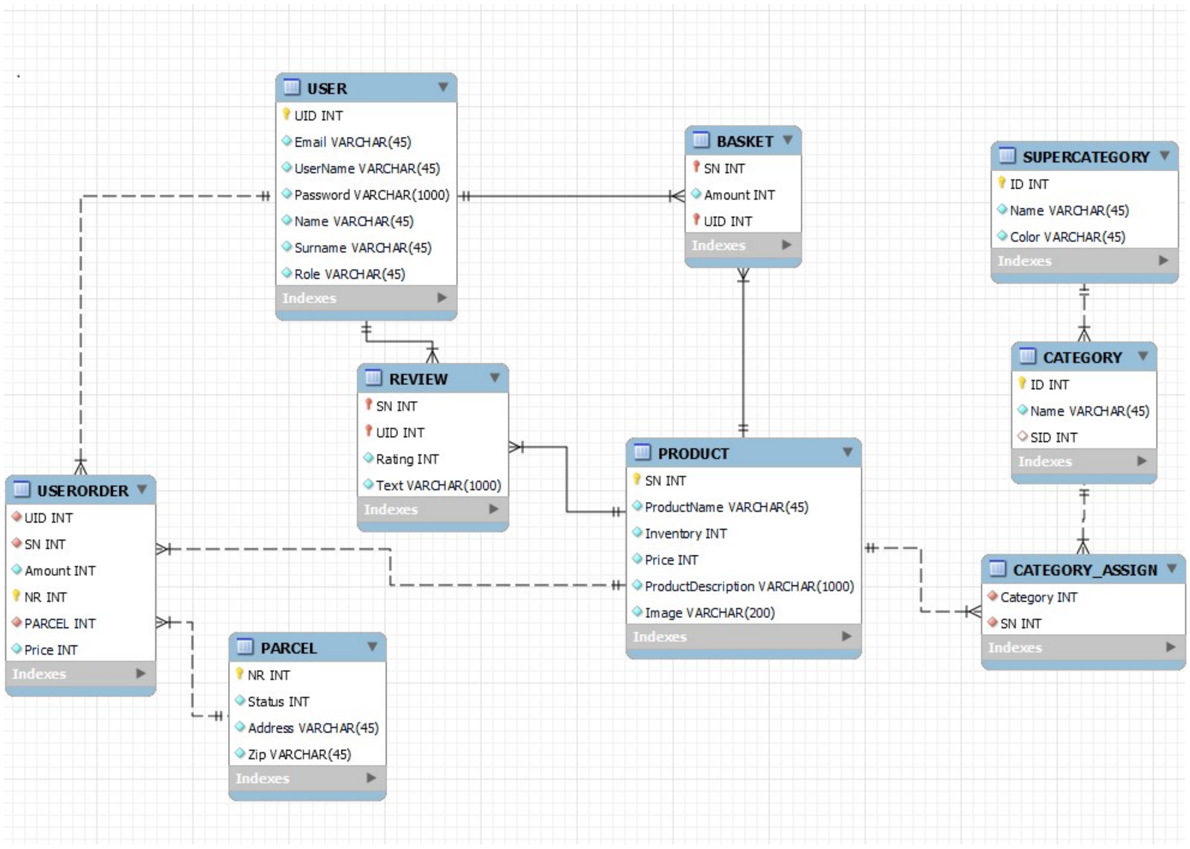


Figure 4: The database schema (EER form).

7 GitHub project code

The project's source code is stored in a GitHub repository to make collaboration simple.

8 Test case specifications

The tests aren't automated, since there is no requirement for that. To test different functionalities the tests will be done with user stories to see if the functionalities work.

8.1 Customer tests

- Register:
  - Register a new user on the $< URL > /register$ endpoint.
  - In MySQL workbench execute the query $SELECT * FROM\ USER$;

  - Ensure that the last entry in returned data contains the email submitted and that the password is not in plain text.

- Login:
  - Register a user.
  - Try to log in at $< URL >$ */login* with the correct email and an incorrect password, expect fail.
  - Try to log in at $< URL >$ */login* with an incorrect email and the correct password, expect fail.
  - Login in with the correct email and the correct password, and expect success.

- Buy a product:
  - Login to $< URL >$ */login*
  - Click on a product's name on the index page and expect a redirect to the product info page.
  - Enter the amount to buy in the "Amount to buy" input box.
  - Click the "buy" button and expect a redirect to the index.
  - Hover over the cart in the top right and expect a drop-down to show up.
  - Click the "Checkout" button and expect a redirect to the Checkout page.
  - Ensure that the product is contained in the list of products displayed.
  - Fill in order details.
  - Click the "Place Order" Button and expect a redirect to the index page.
  - Click the "Orders" button on the left, expect redirect to $< URL >$ */user/orders*.
  - Ensure that the order is present in the list of orders presented.

- Review:
  - Login to $< URL >$ */login*
  - Click on a product's name on the index page and expect a redirect to the product info page.
  - Scroll down to the review section.
  - Click on a star, representing the number of stars that you want to rate the wine, and expect all stars to the left of that star and that star to light up.
  - Enter some text in the text box.
  - Click on the "Submit" button and expect a redirect to the same page.
  - Ensure that the review is now present on that page.

- Recommended products:

  - Login to $< URL > /login$
  - Create a new order as described in the "Buy a product" step, but now include a few different products in that order.
  - After completing that order, add one of the items ordered back into the basket and ensure that the system now recommends those products or some other products depending on wether there have been other orders placed or not.

- Pagination (numbered pages for displaying products):
  - (Admin) Login to $< URL > /login$
  - (Admin) Go to $< URL > /admin$
  - (Admin) Go to $< URL > /admin/create\_product$
  - (Admin) Create a product and repeat so that there are more than 20 products of the same category.
  - (Customer) Login to $< URL > /login$
  - (Customer) Scroll down to the bottom of the index page and click on the numbered pages to see more products with maximum products per page of 20.
  - (Customer) Filter by a category and scroll down to the bottom of the category page. Then, click on the numbered pages to see more products with maximum products per page of 20 of the same category.

- Search for products:
  - Search with product name, try both the exact product name and a substring of the name. Should show products that has the exact name or contains the search string.
  - Search with category, both exact and substring. Should show products that have the exact category or products with categories that contains the search string.
  - Search with super category, both exact and substring. Should show products that have the exact super category or products with super categories that contains the search string.

- Sort products by different attributes, eg. price or name:
  - Filter by price and name, both ascending and descending order, and verify that the order is correct.

- Filter products by category
  - Choose the filters in sidebar and press filter. Products with those category should appear.

- Combination of searching, filtering and sorting
  - This can be tested by searching, filtering by category and sorting, with different combinations, both before making a search and after making a search. Making a new search after filtering or sorting should reset those conditions.

8.2 Admin tests

- Make a user admin:
  - Login to $< URL >$ */login*
  - Navigate to $< URL >$ */admin/make_admin*
  - Enter a Users email address into the textbox, press the "Make Admin" button, and expect an info box with text "User is now an admin"
  - Log out, log in to that user. Ensure that you can now navigate to $< URL >$ */admin*.
- Create a new category group:
  - Log into $< URL >$ */login* with an admin account.
  - Navigate to $< URL >$ */admin/create_super_category*
  - Fill in the super category name.
  - Click "Create" and expect a redirect.
- Create a new category:
  - Log into $< URL >$ */login* with an admin account.
  - Navigate to $< URL >$ */admin/create_category*
  - Fill in the category name.
  - Select a category group from the category group accordion.
  - Click "Create" and expect a notification with the text "Category creation successful".
- Create a new product:
  - Log into $< URL >$ */login* with an admin account.
  - Navigate to $< URL >$ */admin/create_product*
  - Fill in the product information and select one or more categories, expect a redirect to the index page.
- Modify a product:
  - Log into $< URL >$ */login* with an admin account.

  - Click on a product.
  - Enter either a new stock or a new price for that product.

- Modify categories and super categories:
  - Log into $< URL > /login$ with an admin account.
  - Click on the admin button.
  - Click on "Manage categories".
  - Modify a category's name.
  - Delete a category.
  - Delete a category group.
  - Rename a category group.
  - Change the colour of a category group.

- Manage user's orders
  - Make sure there are orders to manage. If there are no orders placed, place some orders.
  - Log into $< URL > /login$ with an admin account.
  - Click on the admin button.
  - Click on "Manage orders"
  - Click "Shipped" to mark an order as shipped.
  - Click "Arrived" to mark an order as arrived.
  - Click "Refund" to mark an order as refunded.

8.3 Presentation test

1. Register a user.
2. Login to that account.
3. Look at the product listings.
4. Filter on wine colour.
5. Choose some wine to order.
6. Add that wine to the basket.
7. Search for a wine.
8. Sort result by price ascending.

9. Add wine from the result to the basket.
10. Checkout basket.
11. Look at the user's orders.
12. Write a review for some wine.
13. Modify that review.
14. Make the user into an admin user.
15. Logout, login in again.
16. Increase the stock of some wine.
17. Change the price for some wine.
18. Modify some user comments.
19. Remove some user comments.
20. Update the status of some user orders.
21. Modify some categories.

9 Limitations and possible improvements

The system is currently limited by a lack of real-world connection, there is no payment processing and no real inventory system to connect to. Aside from these limitations, the system is fully featured for a simple e-commerce site. Future improvements could include:

- Admins could have an easier way to search for accounts to modify, by adding an admin page to search for accounts.
- Add payment processing.
- Add integration to a real inventory tracking system.
- Make order fetch from courier API to update status.
- Improve frontend to be more dynamic and interactive.
- Implement some form of 2FA.
- Improve backend to be more DRY.
- Implement CI using Selenium and Github actions for full-stack CI.
- Implement CD using Github actions.
- Move to HTTPS server instead of HTTP.

10 Usage of transactions for placing orders

When placing an order, each and every element that is in the user's basket has to be moved into the list of user orders. Moreover, the stock of those items also has to be updated. So 3 tables are affected, which means that if any command fails it could cause inconsistencies in the database. To mitigate the risk that is inherent in moving data between tables we can utilize transactions, this means a few things. Resource locks are introduced on the resources that we access when we try to write to them, if an error occurs we can simply roll back the entire transaction nullifying the data transfer altogether. While this assurance that the data is not transferred is great for the data owner the user might have to place their order again since the data transfer was cancelled.

11 Review system

Our website implements a review system, which allows the end user to submit a review containing a string of text describing their thoughts on the product and a 1-5 star review of the product. Every such review is then represented on the product page. The newest reviews are displayed at the top of the list of reviews on a product information page. These reviews are then used to recommend products to users that do not have anything in their basket. To allow for content monitoring the user can at a later date modify the text in their review, they can not, however, modify their star rating. The same goes for admin content monitoring.

12 Project setup information

To test the solution, clone the GitHub repository and modify the ssql.cfg file adding this configuration

```
[ssh]
user_name = ivajns-9
password = TbhNoxzKJbujfwZU
host = 130.240.200.92
port = 22

[mysql]
user_name = handledare
password = Handledare!123
host = 127.0.0.1
port = 3306
database = V0.5
```

or run the setup tooling supplying the same information. The setup tool configures the ssql.cfg file and installs the dependencies.

```
# On linux
chmod +x setup.bash && ./setup.bash
# On windows, in powershell
.\setup.ps1
```

Then you need to start the app:

```
# On any system that has python installed except Debian-based Linux distros
cd src
python app.py
# or on Debian-based distros
cd src
python3 app.py
```

After going to your local ip address you need to make your account an administrator. You do this by logging in to your account, going to $< URL >$ */admin/make_admin*, entering your email, and then logging in again.

12.1 Setup

The local Integrated Development Environment (IDE) that is used is VScode. We're using the virtual machines from LUDD Dust. The virtual machines are used to host the database server. During development, the web servers are hosted on our local machines but for deployment, the server could easily be started on any virtual machine that has an internet connection and supports python 3.

To interact with the database we use MySQL connector through an ssh tunnel, this is done using the ssql library as a wrapper. This allows secure data transfer over a non-secure network.

Since we are running MYSQL on the DUST virtual machines we have not used the utbweb. Though we did use the MYSQL workbench to create all of the needed database schemas and such.

12.2 Technology stack

The technology stack used is as follows

- Frontend:
  - CSS framework Bulma
  - Structure, normal HTML
  - Dynamic elements, normal JS
  - Templating engine, Jinja
- Backend:

    - Server: Flask
    - Authentication: Passlib

- Database MYSQL-server

We chose Flask/Jinja/Bulma for two reasons, firstly since some people in our group had worked with the tooling previously but also because the documentation for Bulma is excellent and the community support for Flask and Jinja is outstanding.

12.3 References

The external sources we have used are documentation for Bulma, Jinja and Flask. Links to these documentations can be found below:

- https://bulma.io/documentation/
- https://jinja.palletsprojects.com/en/3.1.x/
- https://flask.palletsprojects.com/en/2.2.x/

Stack Overflow has also been used for some cases.