D0018E Lab assignment

An Internet e-commerce site using an SQL database

Overview

The objective for this assignment is to approach an SQL database by building a small e-commerce site, which is accessed by standard web browser clients. Your task is to set up and manage your database on a server and build a basic web frontend. An SQL database schema shall be established in this server to store the data handled by your web site.

Please note that the focus of this course is on databases, so the core effort should be on database design and on database application programming, while your web page just need to have a basic layout.

Your software shall be designed so that it would run on a web hotel or other cloud platform. You may choose such a target platform, read the requirements, and design accordingly so that you later could upload your e-commerce site there. However, for the lab course your site can be hosted locally at LTU, or at your own server.

The result of this lab will become part of your portfolio of accomplished demo-friendly works, which may serve a purpose when eventually applying for a job.

Working methods

There are a few guidelines for the assignments:

- You must build your own solution in a team of 2-3 students. You must make sure that everyone in the team is doing hands-on work. Assign yourselves to the empty groups that are created in Canvas.
- Each team must build its own solution. However, we strongly encourage open cooperation across teams. You may discuss and even design parts of the solutions together with other teams. Please also use the Canvas chat for asking questions, helping each other and posting interesting links or findings. We really want everyone to be an active party of the course community, consisting of all students in this year's course.
- Another key ability is to search, find and re-use existing knowledge, software and other information that can bring you forwards effectively. Please use the web extensively, and don't forget to make references.

In your reports you must name the people and teams you have cooperated closely with and which persons you have discussed briefly with (if it has had impact on your solution). Obviously you must reference all external information (web-pages, papers etc.) that you have used. The list should have two levels: Important (heavily used) and Informal. There is a lot of useful information on building e-commerce sites.

The assignment is generally defined, so we expect all solutions to be different to some extent.

We do enforce agile and demo driven development, i.e., development and adaptation in small steps accompanied with a demo at each step. In this course we gently introduce a lightweight version of SCRUM in real small projects of typically 2-3 members (the agile and scrum concepts in team-based development is covered in a document under Files in Canvas, and will be covered in more depth in later project courses). So, for now just focus on:

- User stories/roles (e.g., administrator, customer), use-cases, and requirements: Define scenario, actions, key requirements and key functions for each role that you would like in your system.

- Keep a prioritized backlog: e.g. a list of identified functions that will go into current and future sprints. As soon as you identify the need for a new function or item of development, add it to the backlog. Each item should have ID, description, priority (importance), effort (small, medium, large), sprint no (when assigned), status.
- Small Sprints: Make small development cycles, producing demos (e.g., every other week), each one adding a small set of functions as identified above. If time is short, drop functions and finish in time.
- Test-driven: define test cases early. They should be saved and reused over and over.
- Get ready for the sprint review with your supervisor (lab assistants). You should be able to present demo and documentation (see below) and have a plan for the next sprint.

Already in the first week, you must make your first sprint plan, including initial backlog, etc. See Canvas, under Assignments, for more instructions. You shall show and discuss this with your lab instructors at the first lab lectures, and upload in Canvas the same day.

Documentation shall be established early (already as part of sprint 1) and evolve through the course, always describing the current state on the above-mentioned bullets. Documentation contains:

- Executive summary
- User stories (different roles), use-cases, requirements, assumptions.
- System architecture description and overview of the implementation.
- Current backlog.
- A database schema (E-R diagram) with descriptions, identification of keys etc.
- Links to code (i.e., your teacher should be able to check the code).
- Test case specifications
- A description of the system's limitations and the possibilities for improvements.

Below are some actions that should be covered. This is a suggestion and you may modify the order. This is NOT a sprint plan. You are personally responsible for the planning of current and next sprint. You will not be told by you supervisor what to include in each sprint (but he/she will review and comment on it). As said, it is in any case important that you develop/implement your solution in small steps.

1: Establish a minimalistic web-site

1. Decide on local Integrated Development Environment (IDE), e.g., VScode, Jetbrains, Eclipse, NetBeans, plain editor or other favourite. On LTU lab computers it should be already installed, but it is a good idea to install the development environment on your own laptop or PC too. Although you can use the lab database server at LTU, I recommend that you setup your own server environment on your personal computer too. This is not difficult, and you will then learn how to be a system administrator of your site in all respects. For this, you can use a virtual machine at LUDD Dust https://dust.ludd.ltu.se , or you can install a hypervisor (e.g., virtualbox) and set up a virtual machine on your own computer. Thereafter install your database and web server.

   Alternatively, to use the LAMP lab server provided at utbweb, your site can be hosted under public_html in your home directory. You may create a specific subdirectory for this course/exercise. You can place files in your publc_html in at least two ways:
   a) By FTP (FileZilla)
   https://www.ltu.se/ltu/it-support/IT-support-personal/Din-dator-och-dina-dokument/Anslut-till-hemkatalog-via-SFTP-filezilla)

b) By VPN and file explorer https://www.ltu.se/student/Tjanster-och-service/IT-support-student/Dina-dokument-och-programvaror/Ansluta-mot-public-html-via-VPN-1.202458)

The web server named “utbweb” supports the LAMP tools (e.g., PHP, Java/AJAX, MySQL) that you can use. You can access the page at: http://utbweb.its.ltu.se/~user, where user is your LTU user name. The utbweb url is only reachable inside ltu, so if you are outside ltu you must use VPN to access the site. Se instructions: https://www.ltu.se/student/Tjanster-och-service/IT-support-student/Undervisning/Distansundervisning. (note that you can also access your site through http://students.www.ltu.se/~user, but there the LAMP tools such as PHP and MySQL are not supported).

Alternatively, as said, you may host your site at your own server (e.g., a virtual machine in the cloud), where you can install your preferred server environment. A key requirement in this case you must provide access to your instructor to test and check your solution.

2. Setup the simplest possible web page for this assignment and verify that it works. This page will be the showroom of the rest of the assignment. Consider which tools you like to use. You may go for a traditional solution (e.g., HTML, PHP, AJAX, JSON, XML/SOAP, etc). Some useful sites for instructions are: http://www.w3schools.com/ , http://php.net/ . You may alternatively use for example Python/Flask, Ruby on Rails, Python/Django or Node.js. Talk to your instructor about which environment you choose.
3. Get acquainted with the database. On utbweb, MySQL is already configured with a user and a database for each student. Your MySQL user name and your password are the same as your 8 digit birth date (e.g., 19960908) and there you have your own database named the same but preceeded by ¨db¨ (e.g., db19960908). If your login does not exist, it is because collision on your birth date, so just add *the last* digit from your other four digits of your personal number. Your password is initially the same as your login, so change it when you have logged in. To explore the database we have installed a tool for remote management called “phpmyadmin” so that you can login to MySQL on utbweb without having any ordinary user account on utbweb. You can login to MySQL on https://utbweb.its.ltu.se/phpMyAdmin1/, by specifying localhost and your MySQL user and password. In these tools you can study the content of your database, change the content and most importantly run command-line SQL statements. Login an change your password by running the SQL command “SET PASSWORD FOR 'user'@'%' = PASSWORD('your-new-password') . If you like to use your own personal computer you will as said to install web-servers support (e.g. LAMP) including MySQL and establish a user and a database.

   You must also use the tool MysqlWorkbench. It should be already installed on the computers in the lab, and you should install it on your laptop too. With MysqlWorkbench you can connect to the utbweb.its.ltu.se (on the standard port for mysql, 3306) by using your MySQL username and password). MysqlWorkbench is a powerful tool that you will use a lot in this exercise.
4. Setup some database table(s) the simplest possible web to access/modify your data. There are lots of instructions on the web on how to access MySQL from the tools you have selected.
5. Write a short report on the selections you have made under 1-5, and report shortly on 6. Submit to your instructor and give a very short demo.

2: Establish a relational database for e-commerce

1. Define use-cases, requirements, data, etc. Design a database schema. The database should (at least) contain tables for Assets, Customers and Orders. The data should be defined so that a simple relational schema can handle it. However, we require that there is some

variable data, such as a counter for how many items there are in stock of a certain asset. Remember that you can find a lot of examples on the web that you can use as starters.

2. Establish documentation according to the structure described above at the end of "Work Structure". Synchronise with your instructor.
3. It is your choice what each spring will include, the first one could include writing a simple front-end for listing assets, for placing an order for one item, at the time etc. You may (later?) want to define customer accounts so that customers can login before shopping. Note that you may use the command-line interface to the SQL database during development and test.
4. Extend your report, essentially showing the documentation at this point. It should now clearly have the format that is explained at the end of the section "working methods" above. Sent it to your instructor, together with a link to your site and prepare to make a short demo for your instructor.

3: Add shopping basket

1. Add the concept of a shopping basket, so that you can add and subtract assets to the basket, and check-out to place orders for multiple assets.
2. Explain your considerations: When do you book an asset? Is it when put in basket or when checking out? Trade offs?
3. (Optional Parallelism/loads test: Emulate a large number of auto shopoholic clients and try performance (actually if you are using your own server do it locally, I am however a bit worried about the load on our common server if everyone does this simultaneously...))
4. Extend your report explaining how the shopping basket was defined and your considerations. Send it to your instructor, together with a link to your site and prepare to make a short demo for your instructor.

4: Add grading and comments

1. Add functionality for customers to provide grades and comments on particular assets. Can comments be related in some way (e.g, list or tree)?
2. Explain your solution and the challenges with adding such structures.
3. Add to your report on how grading and comments were defined. Sent it to your instructor, together with a link to your site and prepare to make a short demo for your instructor.

5: Other functions that you have identified yourself

1. Add as you like. Make your personal e-commerce site as you like.

Summary, final report and portfolio

This assignment is your creative piece of work and it will produce a solution that you may use in the future when you want to setup an e-commerce site (at least as a start or as a lesson learned…).

The final report, code and demo define the contribution to your portfolio. The final report should be intended for a 3rd party reader (not for your supervisor who already knows what this is all about), so please gently introduce a reader to your outcome and your contribution. Get to the point early and push details forwards so that a reader, wherever he/she stops reading, gets the most important points. Any chronological descriptions on your development history (if any) would be pushed to the end or in appendix.

With this result being part of your portfolio of accomplished demo-friendly works, it may serve a purpose when going to job interviews.