D0018E Lab assignment 1: 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. The site shall be accessed by standard web browser clients. Your task is to setup and program a centralized server. An SQL database schema shall be established in this server to store the data handled by the web-site.

Please note that the focus of this course is on databases, so the website should be simple with a basic layout. The core effort should be on the database design and database application programming.

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, note that for the lab course your site can be hosted locally at LTU.

Working methods

There are a few guidelines for the assignments:

- Every student must build their own solution. This is to make sure that everyone is doing the hands-on work by themselves.
- However, we strongly encourage open cooperation. You may form groups to discuss and even design the solutions together.
- Another key ability is to search, find and re-use existing knowledge, software and other information that can bring you forwards effectively. Please do that and don't forget to make references.

In your reports you must name the people you have cooperated closely with and which persons you have discussed 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 strongly encourage agile and demo driven development, i.e., development and adaptation in small steps accompanied with a demo at each step. The agile development concepts will be covered in later courses and are typically applied in team-based development. So, for now just focus on:

- User stories/roles (e.g., administrator, customer), use-cases, requirements: Define scenario, actions, key requirements and key functions that you would like in your system.
- Keep a prioritized backlog: e.g. a list of identified functions that will go into future sprints.
- Small Sprints: Make small development cycles, producing demos (e.g., every 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.

Documentation shall be established early and evolve through the course, always describing the current state on the above mentioned bullets. Documentation contains:

- 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. As said it is 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., Elipse, NetBeans, plain editor or other favourite. On LTU lab computers it should be already installed, but you may like to install the development enviroment and even the server environment (e.g., Linux LAMP) on your personal computer too.
2. Your site will be hosted under public_html in your home directory. You may create a specific subdirectory for this exercise. There is a web server named "utbweb" that supports the tools (e.g., PHP, Java/AJAX, MySQL) that you need for this exercise. Please contact your teacher if you find that some additional software should be installed. You can access the page at: http://utbweb.its.ltu.se/~user, where user is your LTU user name. (note that you can also access your site through http://students.www.ltu.se/~user, but there tools such as PHP and MySQL are not supported)
3. 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 (e.g., HTML, PHP, AJAX, JSON, XML/SOAP, etc). Keep it simple. Some useful sites for instructions are: http://www.w3schools.com/ , http://php.net/ ,
4. Get acquaintet with the database. On utbweb MySQL is already configured with a user and a database for each student. Your MySQL user is the same as your LTU user where the '-' is removed (e.g., if your LTU user is batman-0, your MySQL user is batman0). Your password is initially the same as your login. To explore the database we have installed a tool for remote management called "chive" so that you can login to MySQL on utbweb without having any ordinary user account on utbweb. You can login to MySQL on http://utbweb.its.ltu.se/chive , by specifying localhost and your MySQL user and password. In chive 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'@'localhost' = 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.
5. 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.
6. Sidetrack (reading exercise). Read briefly what a CMS (Content Management System) is. Explore what a CMS would brings to you (e.g., Vosao, Wordpress, Joomla, Drupal). In this

course we will not use any CMS.

7. 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”. Define a (first) sprint. Synchronise with your instructor.
3. Start your sprint: It is your choice what it will include, it 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. Write a short report, essentially showing the documentation at this point. 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. Parallelism: Emulate a large number of auto shopoholic clients and try performance (actually if you are using your own server do it locally, I am a bit worried about the load on our common server if everyone does this simultaneously...)
4. Write a short report on how the shopping basket was defined and your considerations. Show results on performance measurements. Sent 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. Expain your solution and the challenges with adding such structures.
3. Write a short 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.
2. Write a final report and send it to your instructor, provide code and final demo.