Cupcake mini project

This assignment must be delivered on moodle (Go to moodle for details: what, when etc.)

This assignment is to be carried out in groups. (If you work together in pairs on one computer, you have to take turns at the keyboard distributing time evenly). You must use github for your shared code repository and deploy the application on all of your linux servers (So that each member has a working application on their own server.)

Web shop

You are to develop a simple web-shop using MySql database, java servlets and jsp pages on the backend and html, css and javascript on the frontend.

The web shop sells cupcakes, but only as pick-up. Customers can use the web shop to place an order and then show up in person to pick up the cupcakes. Sending cupcakes in the mail has shown to be a bad idea as they turn up with bite marks if at all.

The bakery has a very fast cup cake-machine, so the instant the order is placed the cup cakes are ready for pickup.

The cupcakes have a bottom and a topping which can be combined in many ways, but a cup cake must always have both a bottom and a topping. Bottoms and toppings can be found in appendix A.

Costumers each have an account with the shop and orders can only be placed if the account hold enough money to cover the price. Payment is handled by another system and as of now deposits will have to be handled manually in the database, but withdrawals happen when cupcakes are ordered.

In order to pay with their account the customers will have to use a username and a password to login before placing an order.

See working example here: https://www.bugelhartmann.dk/cupcakeshop/

Let's break the project down into steps:

1 Connect to database

Create a web application with a class which is responsible for the database connection. This class holds all the information needed to make a connection. You can use MySql server on your droplet on Digital Ocean to make the connection to. This will make it easier when you need to deploy the project and it means that all members of the group can connect to the same database.

2 Login

1. Create a database and add a user table. A user has a username, a password and a balance.
2. You will need a User class to hold the user information when it has been fetched from the database.
3. Create a class (a data mapper) that contains all methods that reads from and write to the database. In this class create a method, which takes a username sends a query to the database and then returns the corresponding user object if it exists or null if it does not.
4. Create another method that can take email, username and password and create a new user.

3 Registration and login

(*Read all 3 assignments before you write any code!)*

1. First make a servlet that accepts information about a new user and saves him in the system.
2. Create a login sevlet where the user can input a username and a password. No half measures or shortcuts this time. Make a proper solution where the password is compared to that in the database. How will you remember (on all pages) who is logged in right now?
3. Since the two features a very much alike they can be written as a single servlet, where an extra parameter determines what should be done with the input. This extra parameter will later be made as a "hidden" parameter.

4 The shop

The shop servlet shows who is logged in and the account balance of this user.
It also displays a list of all the cupcakes (bottoms and toppings) and their prices.
You will need to expand the database to hold cupcakes (see appendix A) – *hint you might want to consider using more than one table.*
A cupcake has a bottom, a top and a total price. (Unless you want to keep it very simple and just work with a list of Cupcakes. No toppings and buttons then just a list of cupcakes e.g. List<Cupcake>)

1. You will need a Cupcake class to hold information about the cake, price etc.
2. A data mapper class with methods to create top and buttom objects from the database. E.g.

```
List<Buttom> buttoms getAllButtoms);
```

If the shop servlet is provided with parameters that indicates an amount of a given cupcake. This information can (in assignment 5) be forwarded to another servlet that will store it in the shopping cart.

5 The shopping cart

The shopping cart holds Line Items which has information of which cupcake (bottom and topping) and the quantity of cupcakes. The Line Item also has an invoice_id to prepare it for assignment 6.

1. Create a ShoppingCart class that has a list of LineItems (create this class too)
2. Create a servlet: ProductControl.java that can take information from the shop servlet (when a user selects a cupcake from the list) and create a LineItem with the cupcake and the desired number and add it to the ShoppingCart object (in the session)
3. In the shop servlet also show all the lineitems in the shopping cart and the total price.
4. The shopping cart should be stored in the session (Why do you think?).
5. When a cupcake is added, a new Line Item is created and added to the cart. If the user order the same cake twice we can add yet another LineItem (or the quantity of an already existing Line Item can be incremented).

6 The invoice

When cupcakes have been added to the cart the order can be finalized. This creates an invoice with an id and a customer(user). This invoice and all the Line Items in the shopping cart are stored in the database.

1. Create the necessary tables in the database to persist the shopping cart
2. Create a data mapper that can take the shoppingCart data and persist it to a final order in the DB with order date.
3. Wrap the database inserts in a transaction to ensure that either the full order or no part of the order is persisted if something goes wrong.

4. Create the necessary functionality in the ProductControl servlet so that we can receive message from the user that she wants to finalize her order.(when they click the checkout button on the products.jsp page.)
5. You will need to calculate the total sum of the order and withdraw it from the users balance.
6. If the user can't afford it, the order is not saved and a message is returned to the user.

7 Customer page

1. Create a page(servlet) for the user who is logged in (has a session running). The page should show all the invoices for this particular user.
2. On the top of the page show the name of the user that is logged in.
3. If an invoice is selected, it shows the invoice details for the chosen invoice

8 Admin page

The cake shop administrator needs to know which orders have been placed

1. Extend the user table with 2 roles: customer and admin
2. Make an admin page with a list of all the orders.
3. When admin clicks on an order a new page is requested on the server containing the order details of that order.

9 Styling

1. Style the user registration page (not using bootstrap) to have
    a. The form centered on the page
    b. The form inside a dark brown frame (hint: use div)
    c. Add a nice cupcake picture to the page
    d. Create a top menu with a logo and menu links to the pages
    e. Make the background color chocolate brown
    f. Make the font color: cream
2. implement the top menu in a way that it can be used on all the pages (hint jsp:include).
3. Style the products page using bootstrap. Make it as attractive as you can. (hint: you can use a theme like on of these: https://startbootstrap.com/)
4. Put the products inside an html table and style it with bootstrap so that it becomes zebra striped.
5. Style the admin page with 2 bootstrap columns
    a. column 1 contains the list of orders
    b. column 2 contains the details of the order that was last clicked on

10 Interactive design

We will use javascript to make the html pages more interactive. All javascript is run in the browser, so if we can use javascript to prevent wrong or unnecessary requests to the server, then we can optimize the application by minimizing the work load on the server.

1. Make form validation on the registration form so that a nice discreet message is shown to the user when either a field is not filled out or has the wrong data format. (Hint: by descreet means not a popup but rather a small text in red next to the input field or something)
2. On the customer (or admin) page use javascript to get an array of all the orders.
3. Make 2 buttons: ‘Sort by date’ and ‘Sort by customer’. Create event handlers to the buttons to sort the orders and change the DOM node containing the orders with the new sorted content.
4. In the products.jsp, where the LineItems are you can put a css class on a span element surrounding the price of each LineItem. E.g: `<span class=”price”>10.00</span>`.
5. In admin page use javascript to get the collection of elements with that class (Hint: document.getElementsByClassName(“price”). Get the values (innerText) from each element. Calculate the total price of the order every time the users adds a new LineItem.
6. Get the users account and if the account is less than the total order price, then disable the button that finalize the order.
7. Provide the user with a message saying that there is not enough money on the account and they need to remove items from the list.
8. Make a remove button on each line items in the shopping cart (make sure they have unique id’s).
9. Make a javascript event handler to the ‘remove button’, that removes the item from the list.

---------- next semester stuff: ----------

10. Make an AJAX (fetch()) call to the the server to remove line item from the ShoppingCart object on the session.
11. Make the necessary servlet functionality to remove the line item from the server session.

11 Extra tasks for when you just need something to do

1. In you database tables apply indices columns that are likely to be searched more often
2. Make an automatic backup plan for your database that backup data every night

Appendix A – The cupcakes

The bottoms

Bottom	Price
Chocolate	5.00
Vanilla	5.00
Nutmeg	5.00
Pistacio	6.00
Almond	7.00

The toppings

Topping	Price
Chocolate	5.00
Blueberry	5.00
Rasberry	5.00
Crispy	6.00
Strawberry	6.00
Rum/Raisin	7.00
Orange	8.00
Lemon	8.00
Blue cheese	9.00