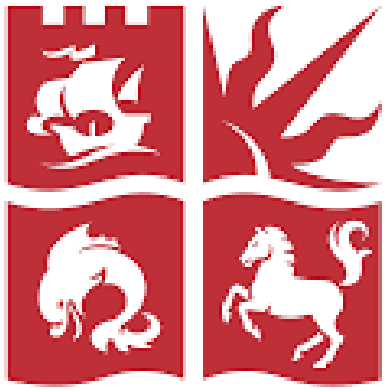

University of BRISTOL

Web Technologies
COMS32500

Aiyu Liu & Tram Nguyen,
(cr18164, fg18837)

Final Report

June 18, 2020

Contents

Preamble

This final report for the Web Technologies Project serves to be a continuation of the preliminary solution. To ease the reader from re-reading content, changes to the previous report will be noted with MODIFIED , marked with red color or explicitly stated.

Worth noting is the fact that when we commenced this unit, we were complete beginners in terms of developing web sites with an embedded database. However, throughout this project we have widely expanded our skill set within this area; as evident from the resulting website.

1 Introduction

For this project in Web Technologies (COMS32500), we have created a webstore for a physical store named Jinghua (English spelling), which primarily focuses on selling Asian goods [1]. The website is made such that site visitors are able to (1) get information about the physical store, (2) view all available products, (3) sign up and place orders. As these were the primary functionalities of the website, more features have been implemented along the way. Below is a list of the website's feature-set:

Pages	Index, products, contact (sub-page of index), admin page, login and registration-form, shopping cart
Navigation-bar	Navigation between the pages.
Display products	Display products by category, filter by price, sorting by price, the quantity of products left, or alphabetical.
Coupon Code	Pop-up image for coupon. Coupon code can be applied to the total shopping cost on the the shopping cart page.
Shopping Cart	Users have who are logged in can access their individual shopping cart page. If not authenticated, an error message is displayed using the module *flash-connect*, and we can verify whether the user is authenticated with the inbuilt functionality of the *passport* module. Products can be added to the shopping cart, and there are options for *Delete all of this item*, *add one (instance of this item) to cart* and *remove one (instance of this item) from cart*. This page is accompanied by the order summary for which there is a coupon code field, which is not working as of now. Naturally, an user can go through the products page and add items to the cart. The database module has been extended with the respective functionalities required (e.g *getAllProductsInShoppingCart*, *getNumProductsInCart* etc.)
Login	User is able to login and is now able to add items to his own shopping cart . The button for *"or create an account here"* does not work, as the functionality for closing the current modal and then switching to another is difficult to implement.
Registration	A client is now able to register oneself as an user to go into shoppingCart page.
Search bar	Display the products which match the product name.
Admin	Admin user is able to view all the products in the database and add/edit/delete any products, or upload images for the products. In order to view all the products you must log in with: **Username**: *adm* **Password**: *pass*
password encryption	Using the `bcrypt` module, we have now been able to hash and compare passwords for extra security measures, utilised for Admin, Login, Registration.

Please follow the instructions on our `https://github.com/tramnguyenJC/webTechProject` in order to run the web app.

[1] This is a kind gesture to my father who is the storeowner. NOTE: We are not receiving commission by creating this website.

2 Self-assessment

Based on our work, we have assessed each heading of the assessment criteria:

Heading	Mark
HTML	**A**
CSS	**A**
JS	**B**
PNG	**A**
SVG	**B**
Server	**C/B** →B
Database	**B/A** →A
Dynamic Pages	**A**
Depth	Shopping cart

In the following sections, we will describe our work for each heading.

3 HTML

We have used the *Node.js* web framework application, *Express JS*, for our website creation. We believe this heading deserves an A as we have worked extensively with creating and routing different pages. For instance, we have created an admin page which is able to route to editing/deleting/adding products; being able to handle POST requests when adding new products.

However, we have not looked into XHTML delivery, or a validator for ensuring our pages are correct.

4 CSS

We have applied extensive CSS styling to our website in order to fit our own liking. I.e. different stylesheets have been made to meet different purposes; with the stylesheets contained in a separate *stylesheets* folder for modularity. Also we have made sure that no styling tags are used in the HTML pages. Overall, our confidence with CSS is quite good, thus deserving a mark of A.

5 JS

This heading deserves a mark of B as developing a high-level understanding in Javascript has not been our primary focus; scripting only came about on a per-need basis. For instance we needed a button functionality for creating a pop-up login form. Additionally, we have separated our small scripts in a different '*javascripts*' folder for which each file has a designated scripting purpose for a specific task.

Our most significant script is filtering the products-page based on the prices. This was adapted form the work of this source `https://bootsnipp.com/snippets/5MzD5`(also been inspiration for our products page). We do not claim of having gained a high level of expertise in using client-side frameworks.

6 PNG & SVG

For this heading we decided to create an image containing a promotional coupon for the event of Chinese New Years. Firstly, we created the vector-based outline in *Inkscape*, mostly based on free-hand drawing.

Secondly, we exported the outline to the pixel-based software, *Medibang Paint Pro*, to which colors were applied. For this process, many tools and layers were practiced.

Figure 1: Promotional coupon in which the code can be applied at store checkout

The option to apply this coupon-code at a checkout cart will be implemented if time allows it. Progress pictures of this image-making can be found in the appendix.

7 Server

We have not only setup express but also used it's module creation API extensively to improve our application's modularity. Additionaly, we have used middleware modules to make certain tasks easier for ourselves. For instance, using *Passport* for user authentication, *Nodemailer* for handling emails and *Multer* for handling image uploads etc. Thus, we believe that we have satisfied the C requirement and crossed into the B category, marking ourselves with C/B.

We have not dealt with the speficic requirements for the B and A categories.

The section below has been modified.
The main feedback for the previous work was the fact that the implementation was not good at accounting for either database or or web server security.

We have chosen to restrict access to our local machine only, which is why we have not dealt with security to a greater extent:

```
http.createServer(app).listen(app.get('port'), 'localhost',
  function(){
    console.log("Express server listening on port " + app.get('port'));
});
```

Furthermore, password hashing has been employed for extra security measures.

Although, once the project is deployed, the developers need to add heavy security measures. One of these is URL validation, for which only known valid URLs are furhter processed for instance *www.jinghua.com/products.html*, *www.jinghua.com/shoppingcart.html* etc.. This can be implemented using direct string matching, and when the website becomes more advanced in the number of pages, reg ex can be utilised.

Also, webpage needs to ensured to have the proper format. I.e. if an URL ends with /, *index.html* needs to be added to the back, and if it is not .html, discontinue further processing and reply accordingly to the client.

In terms of content negotiation, the developer can configure `static` in express to call a content negotiation function before client delivers to server and server delivering to client - ensuring that the correct extension types are delivered.

8 Database

For database integration, we have used *sqlite3* to create a relational database, which handles our *products*, *users* and *shopping cart* . Our separate server-side module is able to extract, insert and update data. Although, we have not ensured that things happen in the right order.

Our database creation commands:

```
CREATE TABLE IF NOT EXISTS Products
  (id INTEGER PRIMARY KEY AUTOINCREMENT,
   name VARCHAR(255) NOT NULL,
   category VARCHAR(255),
   price DOUBLE(10, 2),
   quantity INT,
   imgUrl TEXT)
```

```
CREATE TABLE IF NOT EXISTS Users
  (id INTEGER PRIMARY KEY AUTOINCREMENT,
   username VARCHAR(255) NOT NULL UNIQUE,
   password VARCHAR(255) NOT NULL)
```

```
CREATE TABLE IF NOT EXISTS Users
  (username VARCHAR(255) PRIMARY KEY NOT NULL,
   password VARCHAR(255) NOT NULL,
   isAdmin  BOOLEAN)

CREATE TABLE IF NOT EXISTS ShoppingCart
  (username VARCHAR(255) NOT NULL,
   productid INTEGER NOT NULL,
   quantity  INTEGER NOT NULL,
   FOREIGN KEY(productid) REFERENCES Product(id) ON DELETE CASCADE,
   FOREIGN KEY(username) REFERENCES Users(username) ON DELETE CASCADE,
   PRIMARY KEY (username, productid))
```

The first red frame is the old Users table, while the green frame are the modifications/additions. It can be seen that we have decided to set a primary key on the username, as this would get rid of the unnecessary `id` field. An additional `isAdmin` field is added to check for admin access, required for modifying items in the database. Admin users are predefined and can only be created within the website application, see 1 for the credentials.

To support our new functionality for having individual shopping carts for each user, we create the `ShoppingCart` table, which is identified by the composite key of `(username, productit)`. We have `ON DELETE CASCADE` keywords, since if a user is deleted from the database every entry, said user's entries should not be stored within the table. The same goes for a product. These modifications are the reason why we think this heading deserves an A from A/B.

~~Additionally, We don't have any relationships between our database entities yet, but this is something we aim to do with adding a purchase history entity, thus linking the users to the products.~~

~~Currently, we have made the database variable exposable to our application when using user authentication (`exports.db = db`). This contradicts the fact that the database has a 'seperate server-side' module, but as soon as everything is setup with user authentication, we will create specific database commands for user authentication purposes.~~

9 Dynamic Pages

We believe that we have become very fluent in using *Express JS* as evident from our website and as described above.

10 Depth

We think the Shopping cart is significant work that can be classified under this section.

11 Further work

In our codebase we are checking for errors extensively, although most error messages are either printed to the console or sent to the client by `res.send()`, which is quite plain and boring.

Also, it would be ideal to add the `logout` functionality. This should not be difficult, although due to time constraints has not been implemented.

12 Appendix

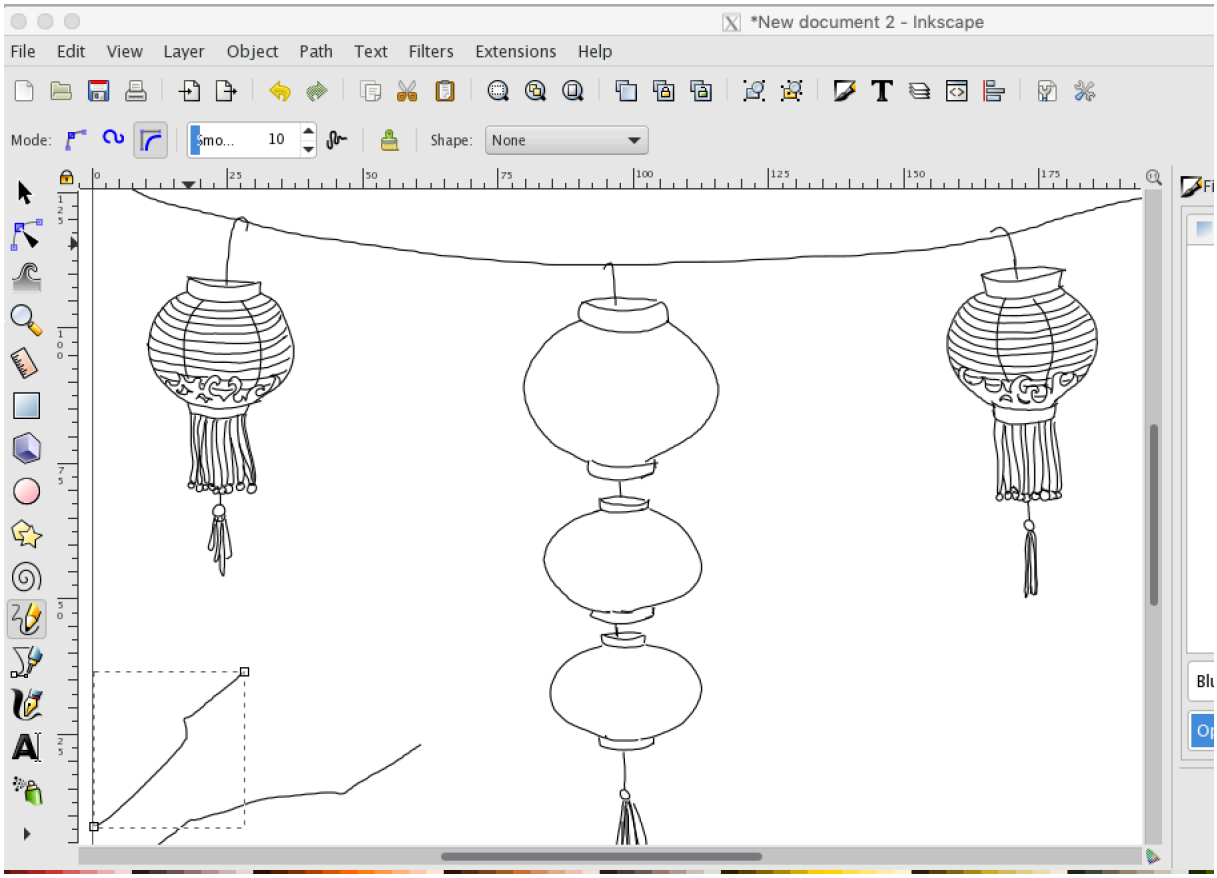

Figure 2: First progress picture (Inkscape)

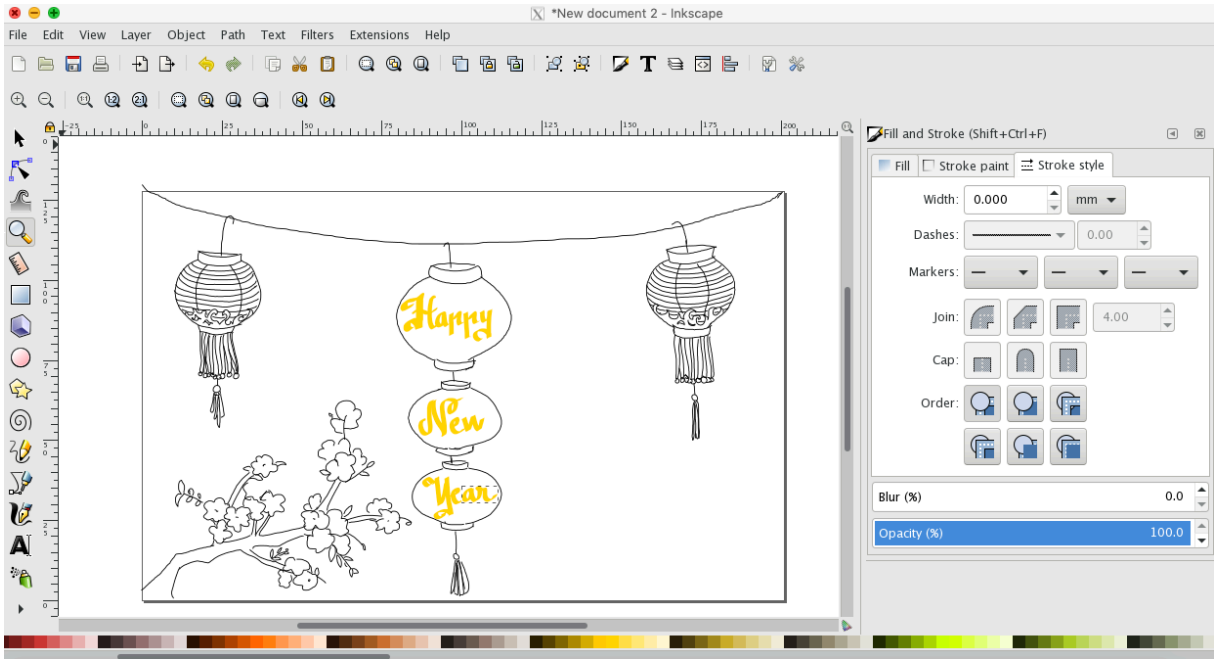

Figure 3: Second progress picture (Inkscape)

Figure 4: Third progress picture (Medibang Paint Pro)

Figure 5: Last progress picture (Medibang Paint Pro)