Team Name: Byteme
Team Member Names: Jay Gupta (Student ID: 921804023), Kishan Patel (922243969), Gabriel Gidey (922049034)
GitHub UserID (for Submitted Assignment): jaygupt
Class: CSC 0317-04

Group Project #5

1) What was Done: We developed our website further to include Node & MySQL. Node allowed us to provide a backend to our app, where we could serve routes & data, perform queries, and provide a more secure web application. Furthermore, using the routing features, we were able to minimize the number of files needed, and using npm packages such as ejs, our ability to transmit data to the individual pages was further simplified. We also used MySQL to store data about our users, their purchases, and the products that are available for purchase. Overall, the development process was made much more streamlined in this iteration, using the knowledge we gained throughout the semester.
2) Who worked on What:
    a) Gabriel worked on the account information page, creating the HTML structure, the CSS, and the JavaScript to make it interactive.
    b) Kishan worked on the login/registration page, allowing users to create an account, and sign in to their accounts. This data was stored in the database, and he ensured that their passwords were kept encrypted. He also created a personalized navbar that would display once someone would sign in, with items such as the user's username.
    c) Jay created the MySQL database & tables, and converted the project from a standard HTML/CSS/JS webapp to a Node app that uses MySQL and EJS. He connected the "Add to Cart" feature to the database, such that a product's quantity is updated throughout the website. He created the EJS template files, such that instead of having 30 individual product pages, there is now 1 template page. He created the search bar, which displays the available products that contain the user's query. Jay also updated the cart page, which displays the user's cart based on the corresponding table in the database. He also worked on the charge summary (subtotal, tax, total, etc.), which is displayed to the user on the cart and review pages.
3) How Work was Divided: Gabriel was new to most of the tools, and as such, we gave him the account information page, which was a good introduction to retrieving/posting data from/to a database. Kishan was working on login/registration for the past few iterations, and as such, we wanted him to continue working on that part of the project. Jay, like Kishan, was working on Add to Cart, Cart, and the product pages, and as such, continued iterating on these parts of the website. He had some prior experience in MySQL and EJS, and as such, was given parts of the website which had a strong use case for templating, such as the product pages and search results page.
4) Problems we Encountered & How we Solved Them:

Team Name: Byteme
Team Member Names: Jay Gupta (Student ID: 921804023), Kishan Patel (922243969), Gabriel Gidey (922049034)
GitHub UserID (for Submitted Assignment): jaygupt
Class: CSC 0317-04

Group Project #5

a) The first major problem we encountered was how to select multiple products from the products table, without writing multiple WHERE clauses, or having an unsecure SQL query.
    i) To solve this issue, we read the MySQL documentation, as it relates to Node, and found the “IN” operator: using IN, we could iterate through the name column of the products table, and if the name was in the array we provided, it would be displayed to us in the results. We used the “?” syntax, such that when we are querying, we give the sql statement followed by a comma, and then the data, which would replace the question mark. This is much more secure than most other options, and reduces the chances of a SQL Injection attack.
b) Our second issue was when we had multiple promises to fulfill; for instance, when we were making multiple queries to the database, we had to wait until all of them were fulfilled before sending the data. With .then statements, this would not make the code very readable.
    i) We learned about Promise.all(), which in its body, returns individual Promises, to create an array of Promises that will need to be fulfilled. Using Promise.all(), only one .then is needed, which is very clean to read.
c) Our third issue was with routes; we needed a way to separate them based on their general category (for instance, some routes are related to auth, some with products, etc.). If we didn’t separate them, it would make the app.js file very long, and difficult to maintain.
    i) What worked for us was to create specific modules for each general category, and use “express.Router()” in each of them. Using this, we were able to specify an entry point into the module (such as “/products” to go to the products route), and furthermore, we could access sub-routes by going one level deeper within this initial route (for instance, to get to the trending products, one would go to “/products/trendingData”). This made the overall process of adding and configuring routes much more straightforward.
d) One issue that was consistent throughout the application was on asynchronous operations; a process, such as querying the database, would take time; however, since JavaScript doesn’t wait for commands to be over before moving on to the next one, it jumps over the query, which produced a lot of errors since statements following the query relied on its result to

Group Project #5

produce output. Hence, we had to find a way to first finish the query before moving on.

i) The solution to this was Promises; one place this was especially needed was in the add_to_cart.js file, which was only to be run when all of the Add to Cart buttons were displayed on the DOM. Initially, there were errors, as we didn't realize that the DOM displayed after the add_to_cart.js statements were executed; as such, we put all of the statements in the file into an asynchronous function, which ran only after the DOM was displayed. This solved all of our issues, and we similarly used Promises and .then statements throughout the rest of the application.