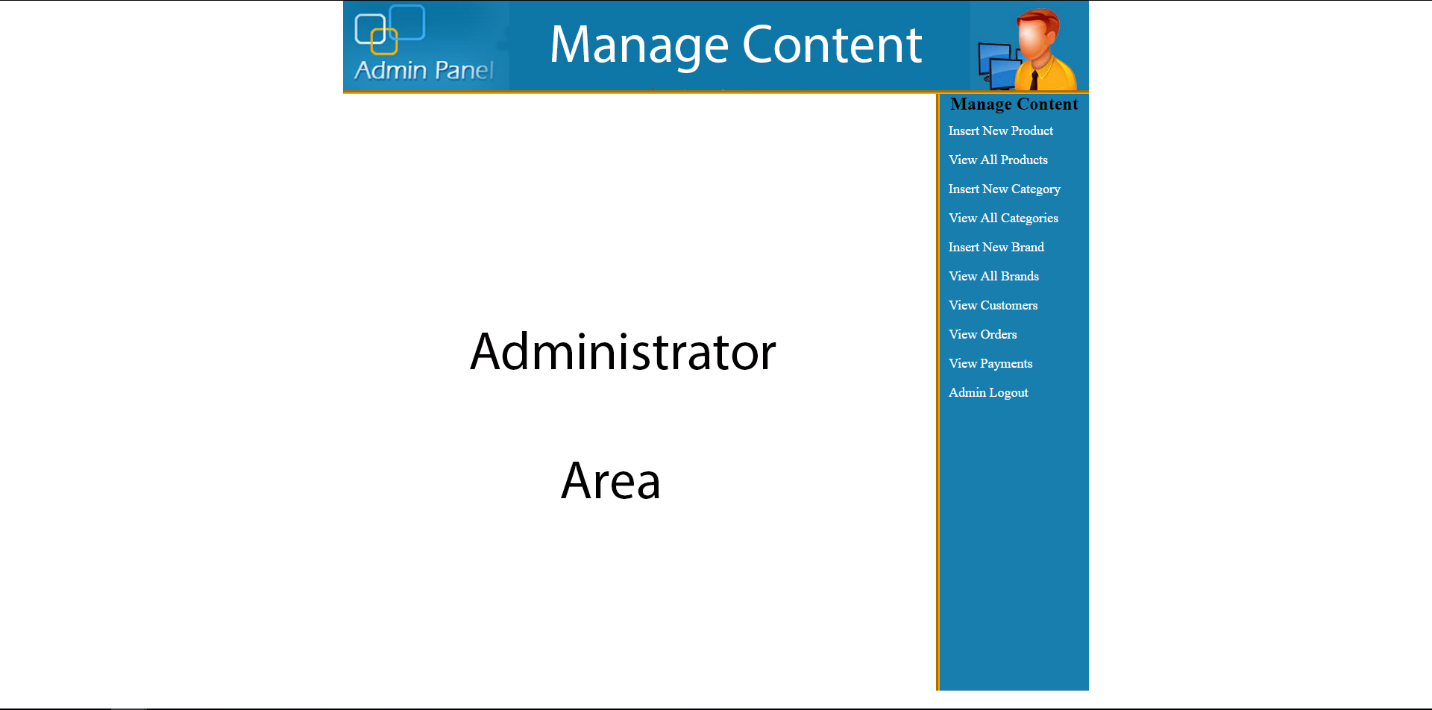
A. User Manual

The end user of this software should instinctively know how to use it. On its own, this is a website that allows a user to shop as would any normal website (Amazon, eBay, Walmart, etc.). However, the user has control as to what goes on the website. At the end of the day we want our user to be able to supplement their brick-and-mortar place of business with an online web presence. To this end, we have given them the freedom to put whatever they want on the website that they would like to sell. The user can do this by simply going to their domain we have them set up and adding a /admin\_area to the end of the URL. This takes them to their personal administrator page. They will also have access to the SQL database and subsequently the ability to add as many, or as few, other administrators as they would like.



After that, the user can insert more products, view all the products they have on their site, remove products, and anything else relating to their website. We want it to be as seamless as possible for the user to talk to someone about a product, and be able to sell them that product in their store as well as online. After the administrator has added the products that they desire, they can check the progress of their site on the homepage. Every item they manipulate is managed in real-time and immediately viewable, and purchasable, to everyone who visits their site. Once the user is satisfied with their products on their page, they can continue on about their day. The goal of Kwik-E-Kart is to make the end users experience just that, quick.

B. Instructions on how to compile/run program and test suite

The end program is a website, so when the user goes to a specific website the program automatically runs. However, since I was not able to get a domain for the website, this runs locally on my machine using a program called “XAMPP” which is a free webserver and database suite. To test the program, I ran it extensively and did my very best to try to break it. To make sure the HTML, CSS, and PHP code was styled well and productively implemented I used Webgrind and Xdebug profiling. Making sure that every div tag and PHP function call was implemented correctly gave the me the peace of mind knowing that the user will have a seamless experience. That being said, the test suite was very difficult to implement. I needed to install Webgrind, then install Xdebug, and then configure Xdebug to communicate on the same port and server as Webgrind. Ultimately, getting the suite up and running took a few hours, but it was well worth it to make sure that everything got tested extensively.

C. A set of automated test cases using a testing framework. Also, discuss experiences testing product

My experience testing the service was an exercise in frustration. It seemed as though that every time I went to test a feature, something else broke. At one point, the service was at a complete stand still because everything went blank. All of my CSS tags, PHP functions, everything was gone. It wasn’t until I realized that I accidentally changed the port of where the webserver was talking to the debugger that I was able to get everything to show up again. Ultimately I learned that taking the time to test as you code will save you potentially several hours at the end. I tried just testing it after my service was finished and it continuously set me further back while I had to fix a number of errors that we present from day one. I am sure that I have not found every bug in my service, nor do I think I could. However, I believe that it is a very solid, production-level service that I wouldn’t mind at all presenting to end-users.

D. Features implemented successfully. Limitations (known problems). Plans for next phase (improvements for next semester)

I was very proud to get most of the features we initially set out to accomplish working. In the end, it was the payment service and order tracking that were the things I simply did not finish. Given a lengthy extension, I feel as though I’m well equipped to finish both of those features and use this site to sell anything that I might need to, or anything a user would want to. One of my favorite features to get working was the admin area. Being able to dynamically add and remove products, customers, etc. from one page, have it talk to the database, and effect another page was really neat. It was the first time that during the project that it felt as though something was actually accomplished. Granted, almost immediately after I accidently dropped the whole customer table and had to start all the way over, but it was nice knowing that I could successfully fix it. As a whole, the features that got fully implemented are: ability for the admin to add new categories, ability for the admin to add new brands, ability for the admin to add new products, ability for the admin to add new customers, ability for the admin to delete categories, ability for the admin to delete brands, ability for the admin to delete products, ability for the admin to delete customers, ability for user to visit site, ability for user to add items to shopping cart, ability for users to register as a new customer, ability to go to checkout page and pay via paypal (however, this is not fully functioning

E. Meeting Minutes

10/26 – 10/30 – Brainstormed ways to keep MeteorJS as the framework and still achieve desired result. Mainly tried to learn MeteorJS quickly enough to have it be a viable option for a project

10/31 – took the day off for Halloween

11/01 – took the day after Halloween off

11/02 – Ended up coming to the realization that I couldn’t use MeteorJS for what I want to have the service do

11/03 – 11/05 – Took the weekend and brainstormed ways to make PHP/HTML/CSS do what I needed it to do

11/06 – Began implementing PHP/HTML/CSS into an XAMPP framework and coding the web service

11/07 – Implemented the base for the database in SQL. Using “ecommerce” as the table, I implemented a structure of databases called [admins, brands, cart, categories, customers, and products] each of which holding their subsequent data. Admins, hold admins; customers hold all of the customers; etc.

11/08 – Finished coding the initial Index.php site and linked it to the database through phpmyadmin

11/09 – Added the logo, banner, and decided on a style to implement

11/10 – Added the .css styles sheet to the web service

11/11 – Wrote the functions.php page to make sure that all system calls were accounted for and localized

11/12 – Linked it to Github but never uploaded it. Not by design, I just kept thinking it was backing up but it wasn’t

11/13 – Added the customer section to the web service

11/14 – Added the ability to add customers via register customer

11/15 – Implemented the cart feature

11/16 – Began working on admin area

11/17 – Implemented the login/logout feature along with my account page

11/18 – took Saturday off

11/19 – 11/21 – Added the ability for admin to dynamically add/delete/view products, categories, customers, brands, and payments

11/22 – Added the paypal payment feature (implemented but not fervently tested/working)

11/23 – Added the order tracking feature (implemented but not fervently tested/working)

11/24 – took the day off for Thanksgiving

11/25 – took the day after Thanksgiving off

11/26 – Finally realized Github problem and began to fix

11/27 – Took a slack day to finalize the web service and make sure bugs were sufficiently squashed.

11/28 – Finished up very last touches and started transferring notes to Deliverable III.

F. Member contribution table:

|  |  |  |  |
| --- | --- | --- | --- |
| Member name | Contribution description | Overall Contribution (%) | Note  (if applicable) |
| Axel Yates | User Interface  Backend Database | 100% |  |
| Paul Gerard | None | 0% | Group member left and contributed nothing before. |