1. **Requirements specifications:**

Software Requirements Specification

for

Kwik-E-Kart

Version 1.0 approved

Prepared by Axel Yates and Paul Gerard

CSCE4444

09/01/17

Table of Contents

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Project Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Features 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

3. System Features 3

3.1 System Feature 1 3

3.2 System Feature 2 (and so on) 4

4. External Interface Requirements 4

4.1 User Interfaces 4

4.2 Hardware Interfaces 4

4.3 Software Interfaces 4

4.4 Communications Interfaces 4

5. Other Nonfunctional Requirements 5

5.1 Performance Requirements 5

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

6. Other Requirements 5

1. Introduction

1.1 Purpose

The purpose of our product is to facilitate the smooth transition of small business owners to the open market on the internet. Throughout constant revisions of our product we will be focusing on security for clients and customers, smoother transition to the online marketplace, and the overall success of our clients. This SRS will cover our business product as a whole and every detail therein.

1.2 Document Conventions

This document will contain a table of contents, normal Arial type 12 point font and consist of roughly 20-25 pages. Each topic will be centered around our project as a whole and subsequently the entire document can and will be used as a reference point as to where our project currently is and where we see it going in the coming months.

1.3 Intended Audience and Reading Suggestions

Overall this is a document that is intended to be used as a reference for us and an overall quality assurance to our future clientele. Users, testers, and developers will also be able to gain insight into our project and philosophy by reading this, making their jobs easier by understanding exactly what it is we are trying to do and the process we plan to take. As such, we suggest reading this document in the order that applies to you from the table of contents contained herein. You need not read this cover to cover or all the way through as each section will be able to stand on its own. Developers and designers may benefit from the UML Design Documents in section two, and the project managers may benefit more from the test plan we have created in section three.

1.4 Project Scope

To begin with we will be leveraging the Meteor JS platform to create this web application. Upon further development we will transition the application into something more proprietary, however within the timeframe and scope of this current project, we decided that Meteor was the most relevant and useful situation for our team. However, we have already begun to see the limitations that a pure Javascript platform like Meteor can have, which is why when we decide to take this project further we will be developing more of a proprietary, in house application.

2. Overall Description

2.1 Product Perspective

The idea of this project hits very close to home to a member of our team. The idea was formed when that members mother had an idea to open a shop to sell the jam and jelly she was making from home to prospective buyers. Her product was of a high enough caliber that she would regularly sell to her close friends and neighbors. However, she wanted to widen her margin and increase her customer base. Our team came up with the idea herein and have since pitched it to her with great aplomb and received positive feedback. She has already signed on board and is one of our main motivating factors for making this the best product we can.

2.2 Product Features

One of the features that sets our product apart from others like it, is the ease of use. What we really wanted to focus on was someone who had an amazing product or idea, transitioning to the web as easily as possible. There are a significant number of factors that play a role in starting and maintaining an online web presence. Our group wanted to make sure that our clients hit the ground running and only had to worry about their product, manufacturing, and other things more in their immediate control. Along the same vein, we made sure that our final product would help our clients maintain that ease of use in each and every aspect. We give them complete freedom to design their website however they would like, but make sure that it flows as seamlessly as possible.

2.3 User Classes and Characteristics

We completely intend to have users from all walks of life. We believe that no matter how vast your knowledge of computers or the internet is, you would like to have the process of setting up a web presence be a little bit easier. Therefore, we cater to the completely new, and the most advanced user. The distinguishing factor of our product between these two is that of guidance. We will more readily guide a user who may not be as familiar with the internet over a user who can program websites in their sleep. Both are of equal value to us and we want to make sure they know that, but we also understand that being treated below your level can seem demeaning to some users and as such, our process will a lot for that.

2.4 Operating Environment

Initially we will be running the whole process on the Meteor platform and the Javascript programming language within the Ubuntu Linux environment. However, when we continue this process further, we fully expect to build a proprietary process that can run on Windows 7 and above, all flavors of Linux, and MacOS.

2.5 Design and Implementation Constraints

Understanding the development that we have used so far would take a new developer time. Using the Meteor platform is sensational when you understand its minute intricacies. However, if you’ve never used Meteor before then there is an initial learning curve that might sway some developers away from using it. Therefore, if a developer has any experience with Javascript, particularly with Meteor, then they’ll be at more of an advantageous position than someone who did not. That being said, we also deal with a lot of databases and fixing security vulnerabilities which can be a lot to grasp. For our users however, we made sure that all of the design and implementation constraints are merely for us to handle so they have more of a streamlined experience.

2.6 User Documentation

We are very aware that in the beginning of any new venture, communication is paramount. That is why we will be delivering a comprehensive “starter guide” along with 24/7 access to someone who knows the ins and outs of our service to provide the best customer service possible if need be. As previously mentioned, we pride our work as being something that anyone can follow, but no one can account for every use case. To that end, we will be providing that aforementioned service along with a comprehensive walkthrough as the process is being implemented. What this means is that when we are getting the customer's website up and running for your business, we will have an open line of communication with the customer making sure that everything is going well and they have someone to talk to if they run into any initial problems.

2.7 Assumptions and Dependencies  
Even though there are more and more people in the world with a wide breadth of knowledge when it comes to computing, we find it easier to initially assume that our customers will need as much help as we can possibly provide. To that end, we will scale back the help if a customer knows what they are doing, or we will provide even more help if need be. Using the Meteor platform to develop this web application has provided us with some hindrances and oftentimes we need to make sure that a plugin or package is working correctly. The development of our web application will be under constant scrutiny by our quality assurance team and they will make sure every bug is squashed. While we do not want any problems with our web application, assuming we will helps us make sure that we are equipped to take them on when, and if they do.

3.System Features

3.1 Ease of use

3.1.1 Description and Priority

Ease of use is the bread and butter of what we do. Transitioning our users to the internet in an easy way is of the highest priority. The benefit of putting such an emphasis on the quality of our transitioning of users is that if it is easy for them, that will make them happy, and when they are happy they inherently tell their friends about our service. Word of mouth advertising is vastly underrated in today’s market, and we intend to make it work for us. This brings up a glaring cost however. Namely, that we will have to work a lot more on the back end of the application to make sure that the user experience is as great as we expect.

3.1.2 Stimulus/Response Sequences

The user gets this benefit throughout the process with no necessary action on their part.

3.1.3 Functional Requirements

Back end user account management:

One of the biggest hurdles that we are going to have to deal with is the management of our users accounts. As a premium service, we hold ourselves to a higher standard of security and as such, we need to have the infrastructure to back up the amount of users and website traffic that we will inevitably see.

Front end user experience:

Something that is just as important as the security of the back end, is the user experience of the front end. If we have the most secure service in the world but no one knows how to navigate it, it will not do us any good. We need to make sure that the user does not get confused or nervous about anything we choose to implement. An example of something we would not do is “opt-out advertisements” this has been shown to make users more hesitant and less trustworthy of the service they are receiving.

Quality Assurance:

To maintain the utmost quality, we need to hold ourselves to a high quality assurance standard. This means going through several levels of error checking, multiple test cases, and preventative security measures.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**3.2 System Feature 2 (and so on)**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4.External Interface Requirements

4.1 User Interfaces

While we have not nailed down a final User Interface, some of our ideas include having a consistent search bar at the top of the screen to help users find whatever they need and each screen will be made for a specific purpose. For example, when the user first goes to the web site they will be greeted with our welcome screen, then they will be prompted to login in or sign up. If they are a returning user and would simply like to check how their website is doing, they will be greeted with such a screen. The account management page will be set up where there is a column of rows on the side wherein they can choose where they’d like to go. From there it is as simple as what the user would like to check on or change. However, for a new user it is slightly more complicated. After they sign up for an account, they will be sent to a screen where they can build their site. What this means is that we will take them step by step, screen by screen, and layout by layout for what they would like their site to look like. They will have the option to make a banner across the top of the page, their own login (with users they would manage but the information would be securely stored on our own servers), however they would like to have the pictures of their products laid out, etc.

4.2 Hardware Interfaces

Since we live in a mobile world and this is a web application, we are assuming it will inevitably be accessed on multiple screen sizes, screen ratios, and operating systems. Fortunately our web application will be able to service any screen size, ratio, or operating system. There will however, be a minor change in what information is available for the user to see. For example, if the user logs in on a phone with a screen size of four inches, they will see a partial amount compared to someone who logs in on a desktop with a twenty-one inch screen. That being said, we will have a mobile version and a desktop version of the web application, however, at this time there will be the same information displayed on both.

4.3 Software Interfaces

As stated previously, our web application was built on the Meteor platform and uses Javascript. What this entails is also the use of NodeJS and the Node Package Manager (NPM). That being said, when we develop version 2.0 of our web application we are looking to transition further away from Meteor and being able to provide a greater service with a wider scope of benefits.

4.4 Communications Interfaces

As a web application, we will rely heavily on things such as HTTP, FTP, e-mail, web browsers, etc. Our service will initially not have an e-mail client of it’s own, but in a future release we would like to have our users be able to make their own e-mail addresses using our service to promote their webstore. That being said they will be able to communicate with us using our e-mail. Every form that the user fills out and subsequent users that shop at their store fill out will be based on Javascript and rely on HTTP. Our security encryption will all come from our back end and Quality Assurance teams.

5.Other Nonfunctional Requirements

5.1 Performance Requirements

Our service will be available 24 hours a day and 7 days a week. The only requirements we have is that the user has a working device with internet access.

5.2 Safety Requirements

So long as the user doesn’t drop their computer on their foot, our service is fairly safe.

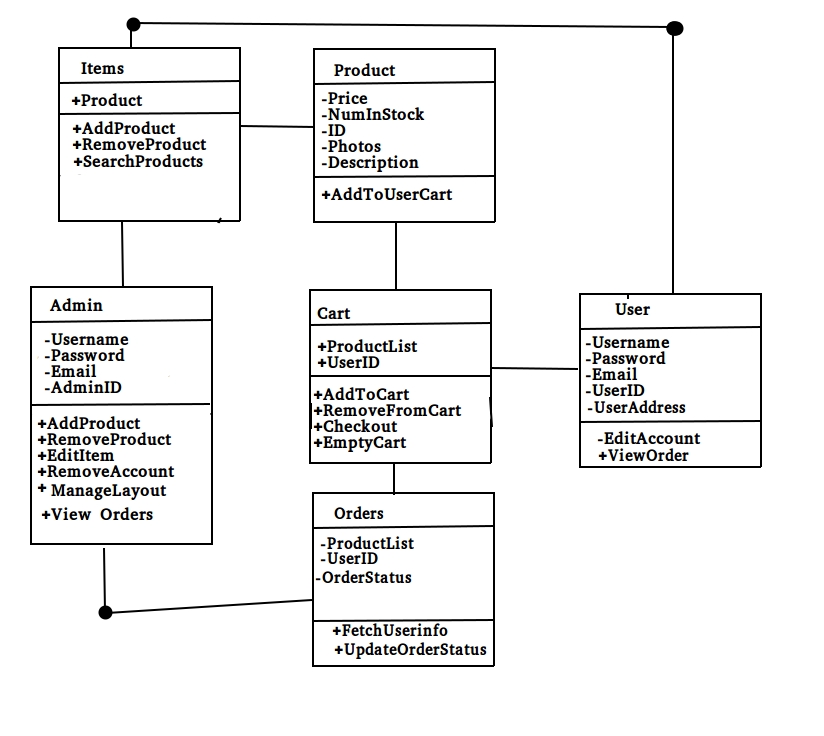
5.3Security Requirements

Inasmuch as we will try to protect a user's data from being compromised, we can not fix, or account for every situation. We will try and sway users to using strong passwords and keeping those passwords in a safe location, but if they don’t then they will run the risk of their website being taken over by someone else. Along that same vein, we will make sure that we do everything in our power as a company to protect them if that does happen and not get hacked ourselves. Considering we will be the only ones with administrator privileges on the website as a whole, we can do our best to make sure that being hacked does not happen to us.

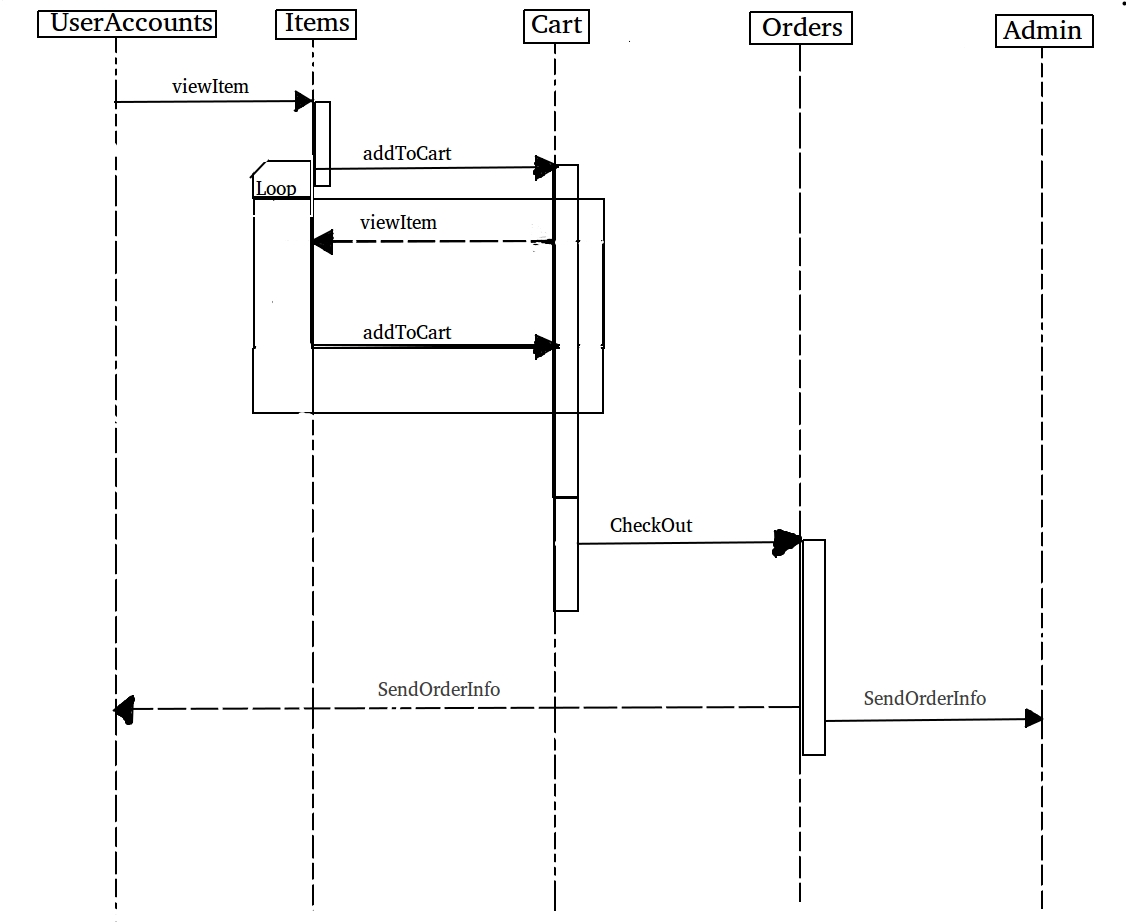
Appendices

Appendix A: Analysis Models

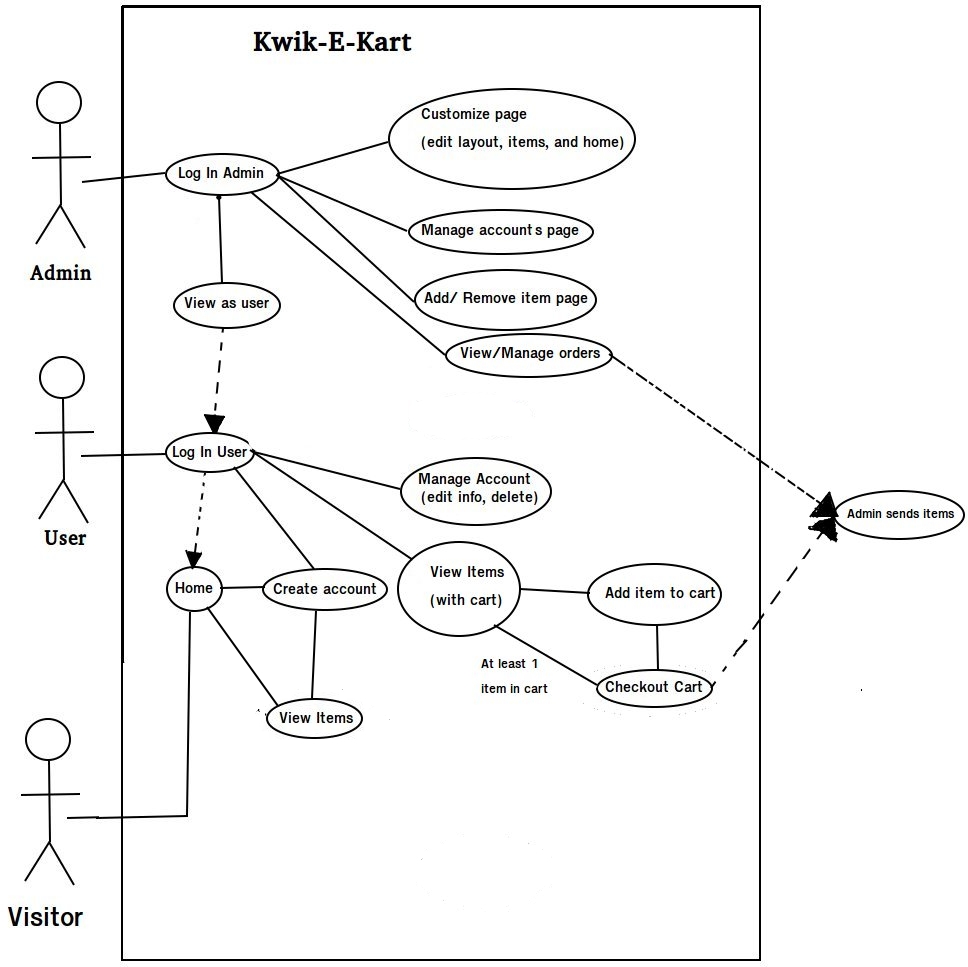
a. Class diagram



b. Sequence diagram

Customer\_Order\_Diagram\

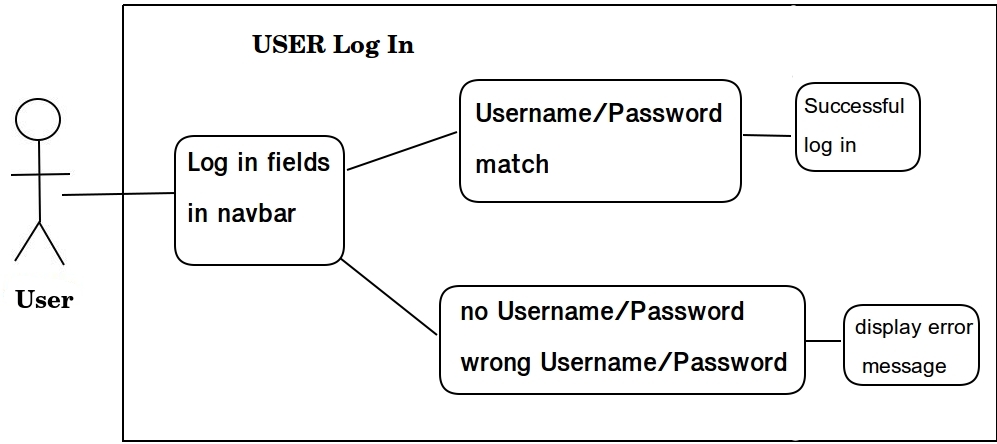
**General Use Case Diagram**



**Test plan**: The purpose of this section is to outline a plan for test cases that will be used to determine if the website is operational.

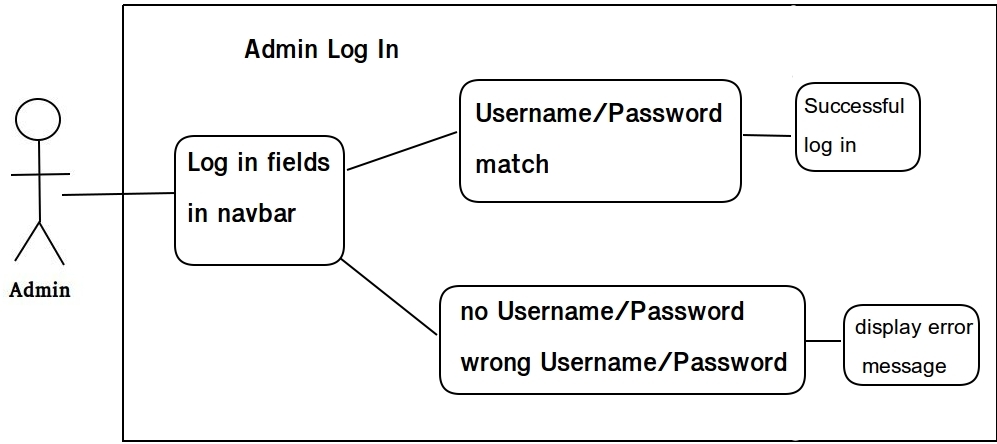
1. **Test Cases**  
     
   **Test Case 1: Customer Login**

* Incorrect input: Wrong username and password and empty requirement fields.
  + Pass criteria: If username and password entered into their respective fields does not match any created user account in the customer accounts database then error message is displayed and user is not logged in.
* Correct input: Correct username and password entered into the correct field.
  + Pass criteria: A username and password given match a username and password in customer accounts database.



**Test Case 2: Admin Login**

* Incorrect input: Wrong username and password and empty requirement fields.
  + Pass criteria: If username and password entered into their respective fields does not match any created Admin account in the Admin accounts database then error message is displayed and user is not logged in.
* Correct input: Correct username and password entered into the correct field.
  + Pass criteria: A username and password given match a username and password in Admin accounts database.



**Test Case 3: Customer Account Creation**

* Incorrect input: no input given in any one of the required fields (email, username, and password).
  + Pass criteria: On the account creation page, if not all required fields are filled then display an error message.
* Correct input: All required fields are filled with information of the correct format.
  + Pass criteria: Take the information in the required fields and add them to a new user account object. Log the user into their newly created account.

**Test Case 4: Admin Account creation.**

* Incorrect input: no input given in any one of the required fields (email, username, and password) or incorrect admin password used (admin password needed to create admin accounts).
  + Pass criteria: On the account creation page, if not all required fields are filled then display an error message. If an incorrect admin password display relevant error message.
* Correct input: All required fields are filled with information of the correct format
  + Pass criteria: Take the information in the required fields and add them to a new admin account object. Log the admin into their newly created account.

**Test Case 5: Remove Account**

* Prerequisite: Successful log in to any account.
* Incorrect input: Wrong username and password given at confirmation screen
  + Pass criteria: On confirmation screen for deletion display an error message if wrong username and password are given
* Correct input: Correct username and password given and then submit deletion button is pressed.
  + Pass criteria: Delete the user account associated with the given username and password, If the account is the same as the one currently logged in then log out the current user before deleting the account. If account being deleted is the only admin account then display an error message and do not delete the account.

**Test Case 6: Edit account details**

* Prerequisite: Successful log in to any account and correct fields filled on account editing page
* Incorrect input: Not logged into an account or the all fields are empty
  + Pass criteria: Account edit page not visible if not logged in and error message displayed if none of the fields are filled.
* Correct input: Replace account information with information in related fields and hit confirmation button.
  + Pass criteria: Account Database updated with information in any field filled on the edit account page if the confirmation button is pressed.

**Test Case 7: Add item to cart**

* Prerequisite: Successful log in to any account.
* Incorrect input: Not logged into an account.
  + Pass criteria: Cart not visible until logged into an Admin or User account.
* Correct input: Add to cart button pressed on any item’s page when an account is logged in.
  + Pass criteria: Take item information and add it to the cart. Cart contains all items added in current session and automatically empties when account logs out or completes purchase.

**Test Case 8: Remove individual item to cart**

* Prerequisite: Successful log in to any account and the cart is not empty.
* Incorrect input: Not logged into an account or the cart is empty.
  + Pass criteria: Cart not visible until logged into an Admin or User account and remove item button not visible till items are in cart.
* Correct input: Remove item button pressed on cart page when an account is logged in.
  + Pass criteria: Remove the item attached to the button pressed from the cart. Cart contains all items added in current session and automatically empties when account logs out or completes purchase.

**Test Case 9: Remove all items from cart**

* Prerequisite: Successful log in to any account and the cart is not empty
* Incorrect input: Not logged into an account or the cart is empty.
  + Pass criteria: Cart not visible until logged into an Admin or User account and remove all items button is greyed out if cart is empty.
* Correct input: Remove all item button pressed on cart page when an account is logged in, account logs out, or purchase is completed.
  + Pass criteria: Remove all items from the cart. Cart contains all items added in current session and automatically empties when account logs out or completes purchase.

**Test Case 10: Purchase items in cart**

* Prerequisite: Successful log in to any account and the cart is not empty
* Incorrect input: Not logged into an account or the cart is empty.
  + Pass criteria: Cart not visible until logged into an Admin or User account and remove all items button is greyed out if cart is empty.
* Correct input: Complete purchase button pressed
  + Pass criteria: go to page requesting payment information, process order information, provide order confirmation to user, and remove all items from the cart.

1. **Item Pass/Fail Criteria**

Testing will be considered complete if all of the test cases are completed with proper results for both passing and failing cases. If a incorrect input does not lead to the proper error message or the correct input does not give the proper result then the project will be considered unfinished and testing will be repeated after the errors are assumed to be fixed. Once all test cases complete with the correct results the website will be considered operational.

4. **Updated risk management**

1. Illness / Emergency - Due to the fact that there are only two of us. If one of us is incapacitated for an extended period of time due to any number of complications, it’ll put a very large workload on the other to make our set deadlines. This is our number one risk mainly due to the fact that as hard as you try, there may just not be any way around this.

a. Contingency: Honestly, there is no real fix for this issue. In the unfortunate event that one of us falls ill or needs to take care of an emergency, the other will have to do their best. Specifically with the winter months coming up, sickness is bound to happen. Mitigating the illness when it props up is our primary focus

2. Lack of Effort - If one of us simply doesn’t put forth very much effort then it will fall on the shoulders of the other team member to pick up the slack and make sure we have a finished product by the deadline.

a. Contingency: To fix this issue we will both get in agreement on what the other is supposed to finish by the specified times. Both of us are committed to the other to make sure that we put forth our best effort possible.

3. Out of Scope - If our project explodes out of the initial scope we set for it there could be cause for concern. While we are taking preventative measures ahead of time to make sure that this does not happen. We are planning for the unfortunate event that something pops up that neither of us saw coming.

a. Contingency: To fix this issue we would scale back the project or change the end goals to better suit a more viable timeline. Along the same vein, we could also make sure that we do not try to make the project fancier or flashier than intended. What we mean by this is that if a few extra graphics would push us too close to the deadline, we would remove that from our workload.

4. Technical Difficulties - If, heaven forbid, our files get accidentally deleted or irreparably damaged, or we have our computers crash in the middle of work without it backing up we could be set back countless hours. This is mainly a risk if we let it slip, both Paul and I are good about consistently saving work.

a. Contingency: To fix this issue, we will constantly back up our progress whenever we can and make sure that we aren’t putting ourselves at risk by having older / deprecated software or hardware. Also, having multiple backups running concurrently as well as having everything backed up to GitHub at certain times throughout the project will help us avoid this pitfall.

5. Only Two of Us - We briefly touched on this in issue 1 but it begs it’s own number. Since there are only two of us we have to make sure the scope is within bounds and no matter how we slice it, we’ll both end up having to do significantly more work than if there were twice as many more people.

a. Contingency: There’s not really a fix for this other than trying to keep the scope within reason. Also, working as hard as we can to finish the best project possible.

6. Meteor Package General Issues: Meteor uses many open source packages to accomplish various tasks that would otherwise be cumbersome to code from scratch. The problems with this approach are that some projects may be incompatible with each other or even versions of Meteor that were released after development of the package ended. This problem is more threatening the further along the project goes and the further we widen the scope of what our site can do.

a. Contingency: Doing alot of research on what combination of packages can accomplish various modular functions allows us to design a blueprint of what tools we can use together before we actually assemble the site. While this research takes more time in the beginning of the project it should save much more should we run into compatibility issues with the packages we selected.

5. **Updated project plan**

The project so far has gone mostly to schedule but there have been a few setbacks. We’ve had to use a slack day on the second deliverable report. Researching the meteor platform has also taken a bit longer than initially so our initial date for having a working template of the site has not been reached. The estimates for these goals were given in the PERT chart from the last deliverable. At the any milestones that are dependant on the uncompleted milestones are more than a week away meaning that we still have slack time to take care of them. Furthermore the Milestones that were passed were not official class given milestones and were more of goals set by the development team with an optimistic attitude. The project otherwise is on the same schedule given in the previous report.

**6. Meeting minutes**

Paul Gerard and Axel Yates present at any mentioned meeting  
  
9/26/2017 3:20 P.M. - 3:50 P.M.   
Topics discussed: Meteor troubleshooting and package research.   
  
9/27/2017 3:20 P.M. - 3:50 P.M.  
Topics discussed: Division of documentation duties.

10/2/2017 3:20-3:50 P.M.

Topics discussed: Division of documentation duties.

10/9/2017 3:20-3:50 P.M.

Topics discussed: Editing Deliverable II.

**7. Progress report**

We are still mostly still researching how to use the technology stack we have chosen for our project as well as experimenting with the functionality of the tools provided. Coding has started on the site but at the current state there isn’t really anything much to show that works. Even still we’ve gotten through a decent amount of troubleshooting with the Meteor platform as well as the packages we chose to use for this projecct. It turns out development for Meteor runs into some unexplained errors on the Windows platform so Linux Ubuntu is being used for development at this point. When the project is complete we will test to see if it runs on multiple platforms (Ubuntu, Mac, and Windows).

**8. Member contribution table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Member name** | **Contribution description** | **Overall Contribution (%)** | **Note**  **(if applicable)** |
| Axel Yates | Backend  Infrastructure, Documentation | 50% | N/A |
| Paul Gerrard | Frontend  GUI, Documentation | 50% | N/A |