Creating a User-Friendly Personal Wine Collection Inventory Management System

By: Peter Loddengaard
Technical Advisor: Dr. Reza Pouraghabagher

Table of Contents

Abstract

This project set out to create a user-friendly wine collection management software. Existing systems were reviewed and discussions with collectors took place to establish the user requirements for this type of inventory management system. This led to the scope of the project being narrowed to making a wine bottle entry form that was quick and easy to use. The programming language, Ruby on Rails, was used to design this system, and learning the software was a large part of the project itself. It was established that four forms would be designed using combinations of two ergonomic factors: an “F” shaped layout and the use of autocomplete text boxes. Two “F” shaped forms were developed: one with autocomplete and one without autocomplete. The other two forms had a simple vertical text box format: one had autocomplete, and the other did not. A one-way ANOVA followed up by a Fisher comparison test revealed that the form with no “F” shape and no autocomplete had the lowest mean entry time. These results were surprising because the altered layout and autofilling text boxes were expected to improve the users experience. An analysis of the unexpected results was conducted, but the data still allowed for a conclusion on which form was the most user-friendly. Also, the results could apply to other systems besides just a wine bottle entry form because the experiment focused on form-filling ergonomics.

Introduction

Many wine collectors have hundreds, sometimes thousands of bottles of wine in their collections. Considering that rare bottles of wines can cost hundreds, sometimes thousands of dollars, and that wine can go bad if it is not drank in the correct time frame, keeping track of wine collection inventories is very important. Often with these large collections, the task of managing them is very tedious and difficult. There are many software packages that handle the task of managing these types of inventories, but many of them are not very user-friendly. To design an ergonomic personal wine inventory management software tool, many industrial engineering principles can be factored in.

Background and Literature Review

There were many directions from which to approach creating this inventory system. First, it was important to look into existing wine inventory database software to investigate various important aspects of wine collecting as well as to find any shortcomings of said systems. Second, deciding the medium to use to create such a system was also relevant. This entailed looking into the benefits of different types of software. Third, it was also important to do some research that shows how to make an ergonomically sound web form. And finally, using statistics to analyze different form layouts was very useful for determining the most ergonomic design. All of these industrial engineering concepts were used to aid in the design of a user-friendly personal wine inventory management system.

There are several existing wine inventory management databases available online. For the scope of this project, three existing systems were reviewed. These

include the Personal Wine Curator Wine Cellar Software (PWC) ("The Personal Wine Curator Wine Cellar Software"), Vinote ("Vinote Cellar Software And Tags"), and Cellar Tracker (Levine). The Personal Wine Curators have an extensive web application that allows users to manage wine inventory, to have access to wine/food pairings, to identify bottle labels, and a few other features. This was useful to see an existing system, and to observe its shortcomings such as possibly being overly complicated. As seen in Figure 1, the form is very cluttered which could lead to users being confused or not being able to find features they are looking for. ("The Personal Wine Curator Wine Cellar Software"). This could ultimately lead to the users becoming frustrated with the system. Cellar Tracker seems to have found a good balance between being a comprehensive tool without being overly complicated, but parts of the layout can still be cluttered as seen in Figure 2 (Levine). In addition, looking into existing wine tracking systems was useful to get a sense of how to scale down a wine inventory management system for the scope of a senior project. With limited resources and time, only certain features, such as logging wine bottles and being able to view and sort a collection, were realistically accomplished.

The next aspect of the project to be considered was the medium for creating the inventory system. In industrial engineering at Cal Poly, Microsoft Access is the database software that is taught. Access has its limitations though because it can only be used locally and not through the web. Ruby is a programming language that is easy to learn and, when used in combination with a web application framework called Ruby on Rails, it can be used for rapid web development. This allows a designer to do anything that Access can do but within the framework of the web.

Figure 1: Home Screen (PWC)

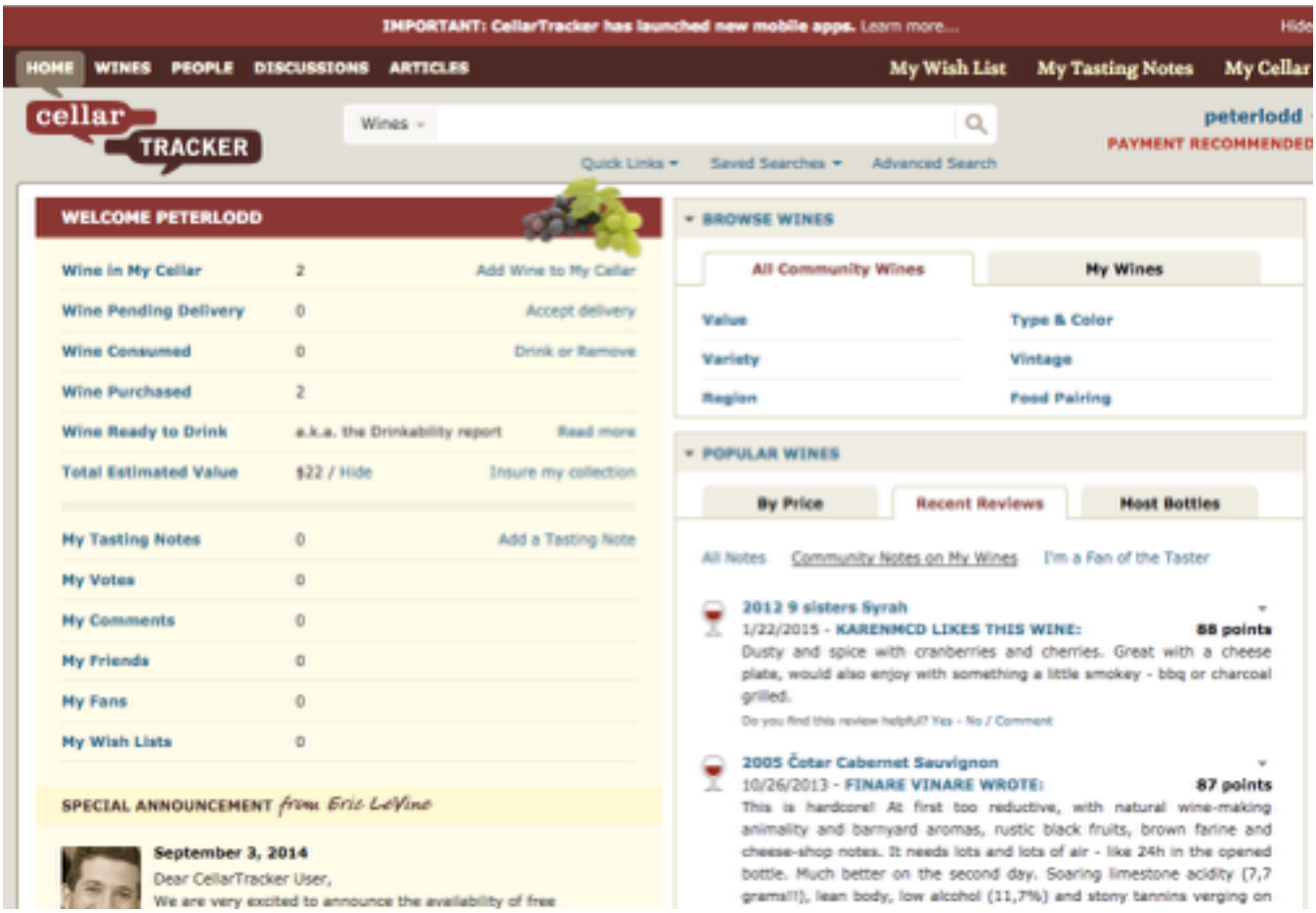


Figure 2: Cellar Screen (Cellar Tracker)

The article entitled, “Rapid Web Application Development: A Ruby on Rails Tutorial,” details a short tutorial on how Ruby on Rails can be used to make forms on the web. The article suggests that Ruby is a very dynamic and flexible, object-oriented computer language (Viswanathan). In addition, there are existing inventory systems that have been developed with Ruby already. The article, “Warehouse Management System in Ruby on Rails Framework on Cloud Computing Architecture,” describes how Ruby on Rails was used to accomplish a web-based warehouse inventory database (Durski, Murlewski, Makowski, and Sakowicz.) For this senior project, Codecademy, an online programming tutorial website, was used to learn Ruby (Sims).

The next component of this project was a review of form ergonomics. The first article reviewed on this subject was entitled “Applying Ergonomics to Improve Usability Design of the Interface of the San Nong (Agriculture, Rural Areas and Farmers) E-government Website” (Sun, Yu). This article described how farmers in rural China were having trouble interacting with the government websites because of poor layouts. The article goes into how ergonomics were applied to these websites in order to make the users have to think as little as possible. Elements analyzed were the importance of the navigation bar, how the links on the page work, and even the importance of color choices. Although there were many things analyzed, the navigation of the page was the foundation for these changes. These principles can be applied to a personal wine inventory system as well because, although the users wouldn’t be rural farmers, there would still be varying levels of computer savviness.

Continuing the review of literature concerning form ergonomics, an article entitled, “Gathering Information About Web 2.0 Applications for Usability Engineering,”

went into how information can be gathered about usability of web applications (Martin). Also it discusses the tie between usability and ergonomics. Ergonomics is all about human compatibility with product design and using statistics to analyze how to best accommodate all users. Usability involves effectiveness and efficiency of task completion. The issues with navigation as seen in the previous article is heavily tied into usability. Since many wine collectors end up losing track of their inventory because they get tired of inputting and withdrawing entries, the ease of use of the proposed system is crucial to a successful design.

The next article reviewed was “Research on Consumer to Consumer (C2C) E-Commerce Website Usability Evaluation System” (Huang). This article discussed how usability of C2C e-commerce is evaluated. This could be useful in evaluating any kind of interactive web page. The use of empirical research is very important in this evaluation. This empirical research used questionnaires to evaluate usability. For this senior project, questionnaires were used for gauging whether or not the proposed system is easy to use.

One other article on software usability was researched, and it was entitled, “Using Three Layer Model (TLM) in Web Form Design: WeFDeC Checklist Development” (Idrus, Razak, Talib, and Tajuddin). This article took a deep look into design checklists that are used to make sure that a website has an acceptable level of usability. Part of the study was to determine if these checklists are valid tools of web form analysis. Government web pages were used because in the past they have been very hard to navigate. Also the three layer model was discussed in great detail. This model involves “three points of view which [are] relationship, conversational, and

perceptual or appearance" (Idrus). What this means is that there are several aspects to making forms easy to use, and one way is to make the form conversational: the form asks the right questions at the right times. Also the attractiveness of the form is very important. The checklists mentioned earlier make sure that these various aspects are covered in a form.

These articles were very useful in seeing different thought processes of how to analyze web page ergonomics and usability. These are very useful tools that can tie industrial engineering concepts such as statistical analysis and ergonomics with the ever-growing fields of web page and computer inventory management system design.

Design

The main components to this project were creating the wine inventory web application, conducting an ergonomic experiment designed to analyze the user friendliness of different bottle entry forms (see Methods section), and performing an economic justification for the system. These tasks required combining skills learned in industrial engineering, such as ergonomics, statistics, and economic analysis with the ever-growing field of software development.

Economic Justification

Since wine bottles can range from tens of dollars to thousands of dollars and collections can range from tens of bottles to thousands of bottles, it is difficult to come up with averages for collectors. Because of this, consider two possible scenarios for collectors. In a conservative example, if every collector worldwide (estimated at 875,000 collectors by Newcomb) only lost five bottles per year due to poor inventory management, and the average cost of bottles in those collections was $150, each

collector would lose $750 per year meaning over $650 million lost per year worldwide. In a more liberal example, if every collector worldwide only lost 10 bottles per year due to poor inventory management, and the average cost of bottles in those collections was $250, each collector would lose $2500 per year meaning over $2 billion lost per year worldwide. Ultimately a collector is not going to continually use inventory software that is tedious to use considering the volume of bottles that some collectors have. This means that it is very important to create software that is user-friendly because, as seen in this economic justification, a lot of money and wine could be at stake if your wine inventory is poorly managed.

Wine Inventory Web Application

The wine inventory system that was created was made with the computer programming language Ruby on Rails. This language was chosen because it is accommodating to beginners in web development. Rails allows the developer to pick fields such as “Maker,” “Grape,” and “Region,” and automatically generate a series of web forms in HTML. These forms include pages where a user can view the database, make new entries, delete entries, and edit entries. Figure 3 displays the final design for the inventory viewing page that was created. The initial Ruby on Rails output had no color, no collection values, and all of the table headers were immediately next to each other, making the form very cluttered and hard to read. Once the Rails framework was in place, HTML, CSS, Bootstrap, and a little Javascript were used to touch up the forms and make them look interesting. Ruby and HTML were also used to add in the “Collection Value at Purchase” and “Current Collection Value” table that can be seen in Figure 3. Also, the headers of the wine table can be clicked for sorting.

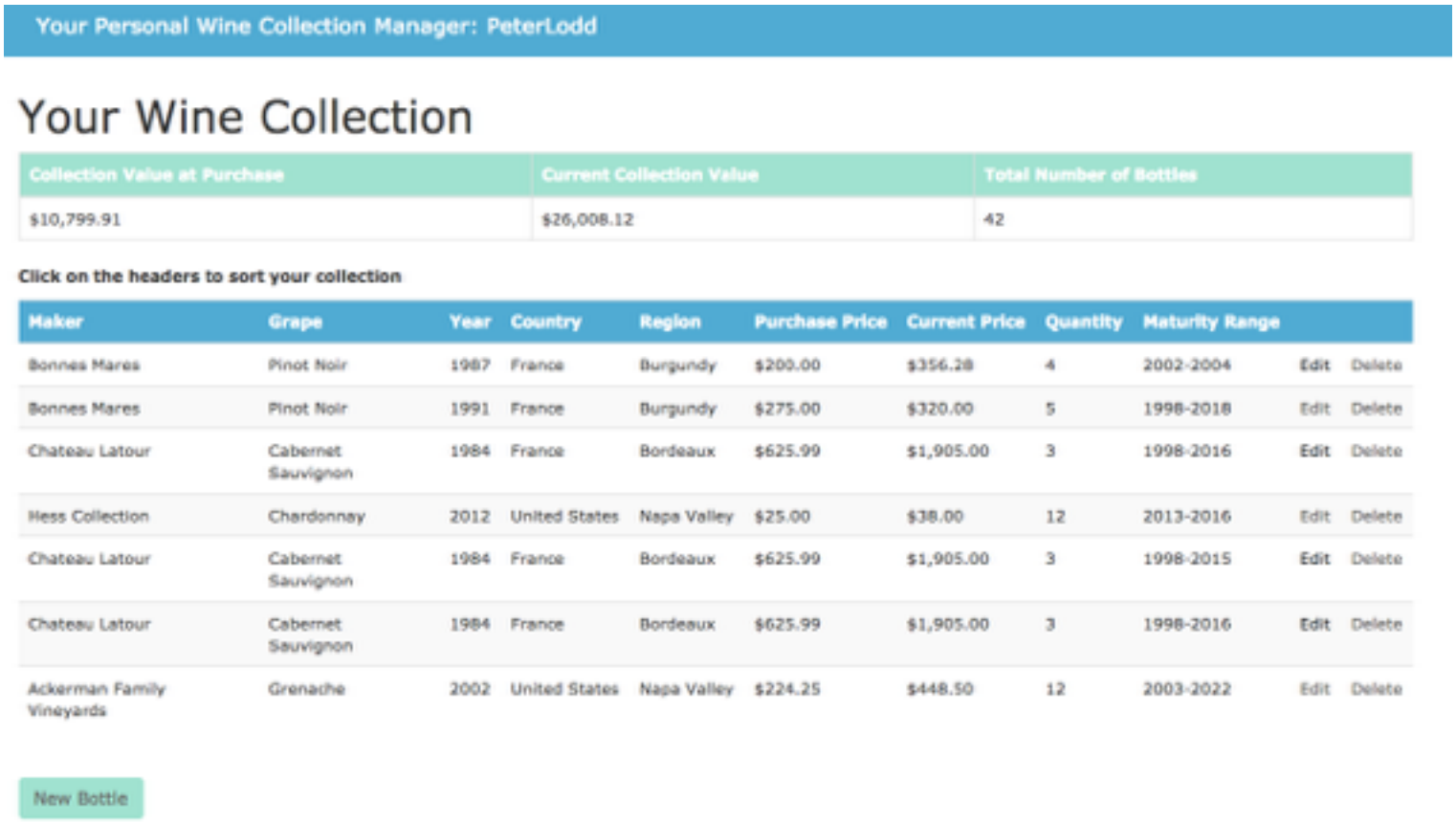


Figure 3: Collection Viewing Screen

Since users stop using wine collection inventory systems because of the tediousness of entering bottles into the system, the main focus of the project was the bottle entry form. Four forms were created in order to test a few different designs. Two of the forms were the shape of an “F” which is considered a good ergonomic design for online forms (see Figure 4): One had an autocomplete function built-in to the text boxes and the other one did not have autocomplete. The other two forms were in a regular vertical format (see Figure 5). Again, one of the vertical forms had autocomplete built-in to the text boxes, and the other vertical form did not. These four forms were chosen to test different combinations of the autocomplete and “F” shape ergonomic factors to determine the most user-friendly form.

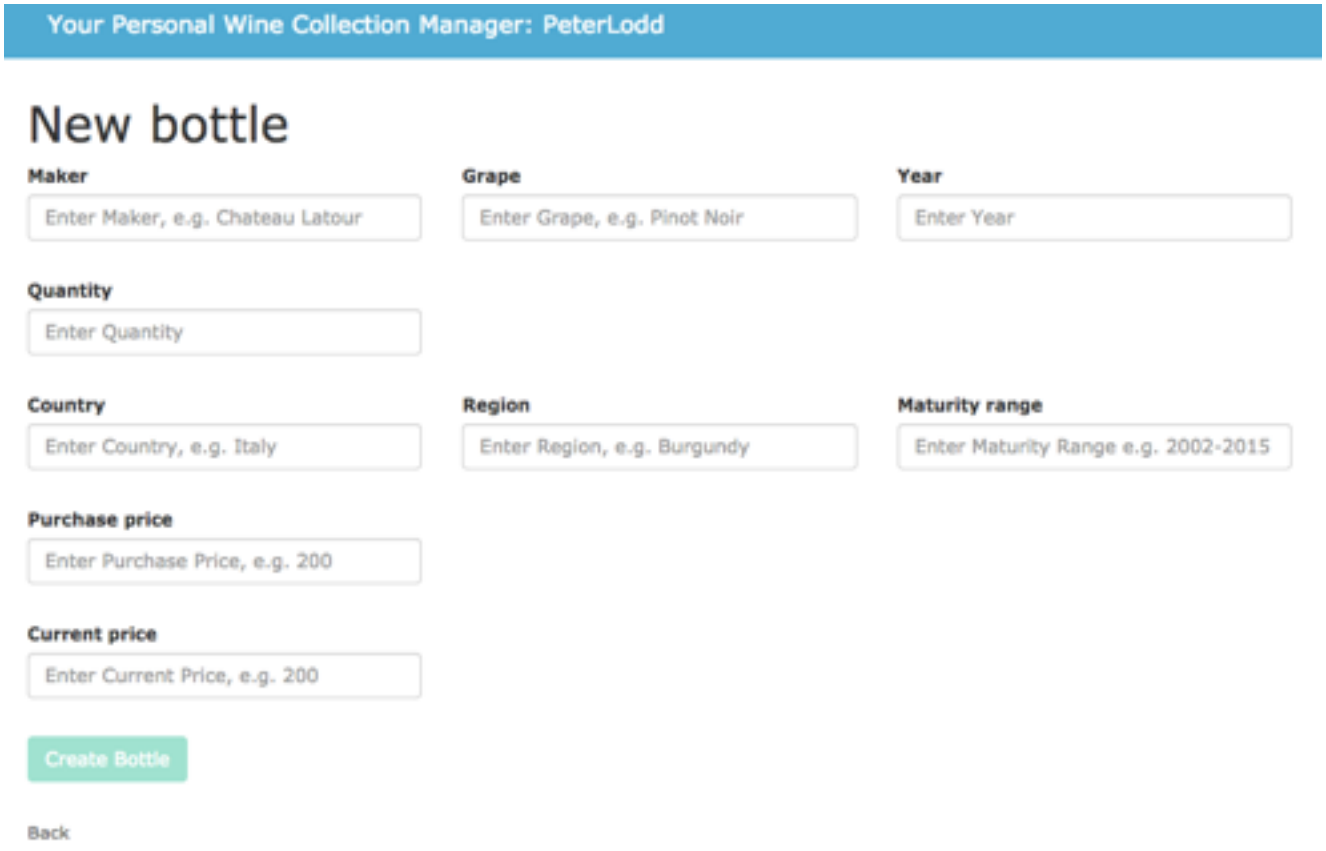


Figure 4: “F” Shape Entry Form

Changing the layouts, colors, and functionality of the forms required using HTML, CSS, Bootstrap, and some javascript which were learned through a couple different mediums. The Codecademy tutorial for Ruby was the first step in learning the new software (Sims). Also, a book called “Ruby on Rails Tutorial” was used to learn Rails (Hartl), and “HTML & CSS: Design and Build Websites” was used to learn HTML (Duckett). Anything else that needed to be learned, such as syntax problems and interpreting error messages was taught by a consulting software engineer or found on Google.

The next step of the project involving the inventory system was the ergonomics experiment to analyze the user-friendliness of the forms

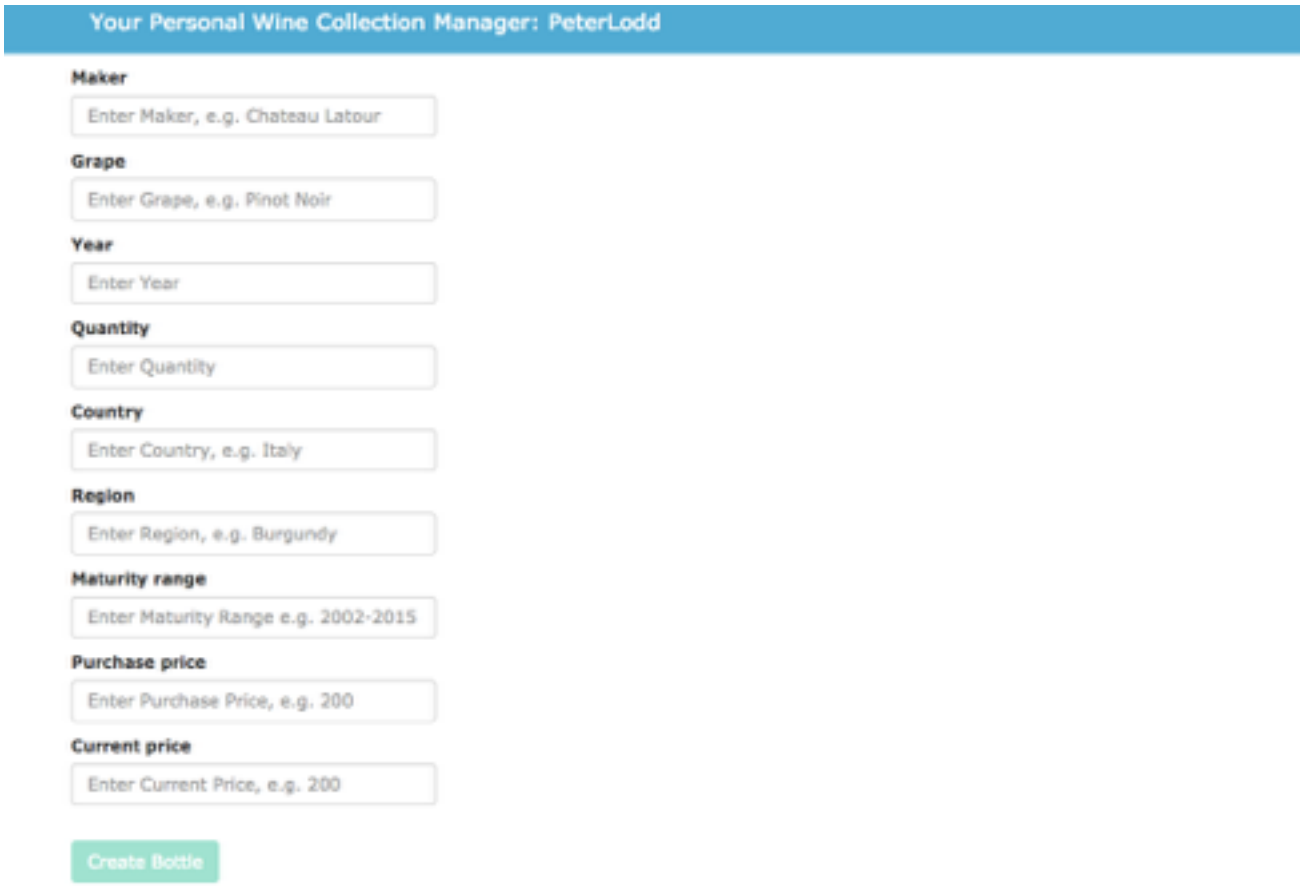


Figure 5: Vertical Layout Entry Form

Methods (Ergonomics Experiment)

For the ergonomic analysis of this project, an experiment was be designed to collect data on the usability of the web application. As stated previously, multiple layouts of the inventory management system were be developed to test different ergonomic factors.

Controls

- Experiment was performed in Building 192 Room 237
- Experiment was performed February 18th and 19th from 10am-2pm
- All subjects were the same distance from the computer
- None of the participants had any wine software experience

Independent Variable

- Form Type (With 4 levels)

Dependent Variables
- Form Filling Time (in seconds)

Target Subjects
- 40 Cal Poly Males

A one factor, completely randomized ANOVA was performed. The ANOVA had four levels: four different wine bottle entry forms. Forty subjects participated and each subject only filled out one of the forms (ten per form). The order in which the participants were exposed to the forms was completely randomized. This allowed the completely randomized, one factor ANOVA to be performed.

Cal Poly males were targeted because they were the largest demographic that was available for testing. Ideally, the subjects would have been middle-aged wealthier males. Test subjects were asked to fill out the experimental form to enter a new wine bottle into the system. The timer started when the user clicked the “New Bottle” button (Figure 3), and ended when they clicked the “Create Bottle” button (Figures 4 and 5). Also, when the experiment was complete, each user filled out a questionnaire, that asked them to rate the “Overall Experience,” “Overall Appearance,” “Layout,” and “User-friendliness” on a scale of one to five.

Results and Statistical Analysis

To analyze the data that was collected in the ergonomics experiment, A one-way ANOVA was performed. First, an Anderson-Darling normality test was conducted on the data. With a P-Value of 0.299 it was concluded that the data could be considered normal (Appendix A). Once the normality assumption was validated, the ANOVA was performed. The ANOVA, which can be seen in Appendix B, revealed with a P-value of 0.081 that it was plausible that there was a statistically significant difference between

forms. To follow up the ANOVA, a Fisher Comparison Test was run with a 95% confidence level. This test revealed that Form 3, the form with no autocomplete and no “F” shape, had the lowest mean time and that this was a statistically significant difference from the other forms (Appendix C). These results were surprising considering that the “F” shape and autocomplete functions, in theory, should have benefited the user.

Also, a correlation test was run between the “Overall Experience” and the mean times. The theory was that high completion times should correlate with low “Overall Experience” ratings. This test showed that there was a weak correlation of -0.200, but a P-Value of 0.215 revealed that it was not statistically significant.

The final statistical test that was run was a Kruskal-Wallis test. This test was run in order to analyze the user questionnaire data. The Kruskal-Wallis test was chosen because it is a non-parametric comparison test, and the ratings did not follow a normal distribution. The results showed that only Form 1 (“F” Shape, No autocomplete) was ranked significantly worse than the other forms in only the “Layout” rating category.

Although the statistics revealed that the simplest form, the one with no autocomplete and no “F” shape, was the most user friendly in terms of entry time, there might be some reasons as to why the Autocomplete and F shape weren’t quite as effective as expected. For instance, many of the users ignored the autocomplete complete function altogether. Many of them stated after the experiment that they thought they weren’t supposed to use it, and that it was the last users entry, not a built-in autocomplete function. Also, some users used “Tab” to move the next text box, while some users used the mouse to click to the other boxes. Finally, some users stated that

the F shape was confusing because, while the form was in the shape of an F, the attributes that they were supposed to fill in for the experiment were listed vertically. These factors could've contributed to the high standard deviation that was seen in the "F" layout forms. Also, this could reflect the lower "Layout" rating that the F shape received. It's possible that if the users were told that they could use tab and autocomplete, the results would've been different.

Conclusion

This project aimed to create a user-friendly wine inventory system. This involved looking into existing systems, considering different types of software to use, learning details about the wine collection industry to perform an economic justification for this system, researching web form ergonomics, designing forms to incorporate these ergonomic advancements, and testing the designs. These tasks were accomplished:

Goals Accomplished

- Researched tools to improve user-friendliness of forms
- Performed a form-filling ergonomics experiment
- Learned Ruby on Rails, HTML, CSS, and Javascript
- Combined Industrial Engineering skills with Software Engineering skills
- Interesting, surprising results were found through the ergonomics experiment

The statistical results of the project showed that a simple form with vertical text boxes and no autocomplete would be the most ergonomically sound design. This was a surprising, interesting result considering that the "F" shape and autocomplete were designed to aid a users experience. Much of the general feedback from test subjects was that they liked the "F" shape and autocomplete, but because of certain conditions of the experiment (see Results section), these factors were not always properly utilized. This was a very valuable lesson in experiment design, showing that statistical analysis

needs to be interpreted with qualitative factors in mind. Overall, the goals of this project were achieved and many lessons were learned about coding, how to perform a web form ergonomics experiment, justifying a project economically, and designing forms to suit a user.

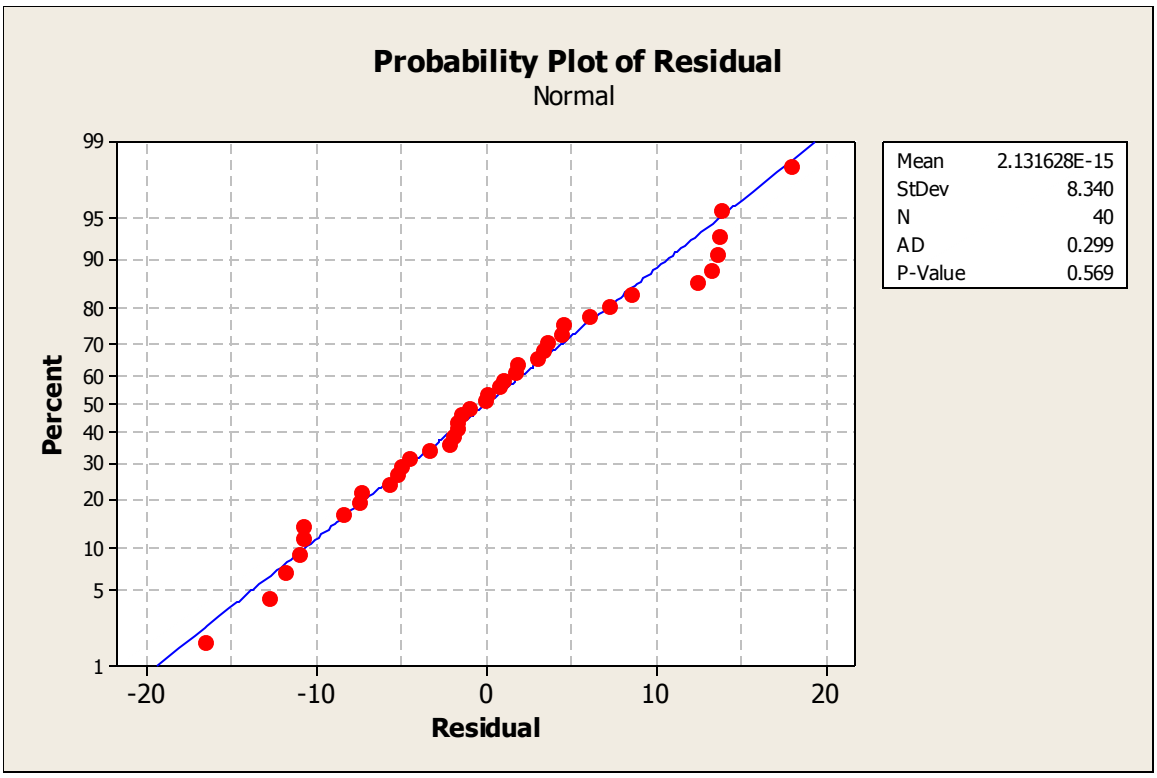
Although the project was mostly successful, there are a few things that would have been done differently in this project if it were to be completed again. For instance:

- Wine collectors would have been officially surveyed about their experience with existing wine inventory software
- The scope of the project would have been narrowed earlier on in the project
- The experiment would have been controlled better
- Users would have been informed of using the autocomplete

Despite these shortcomings of the project, the overall goals were still achieved.

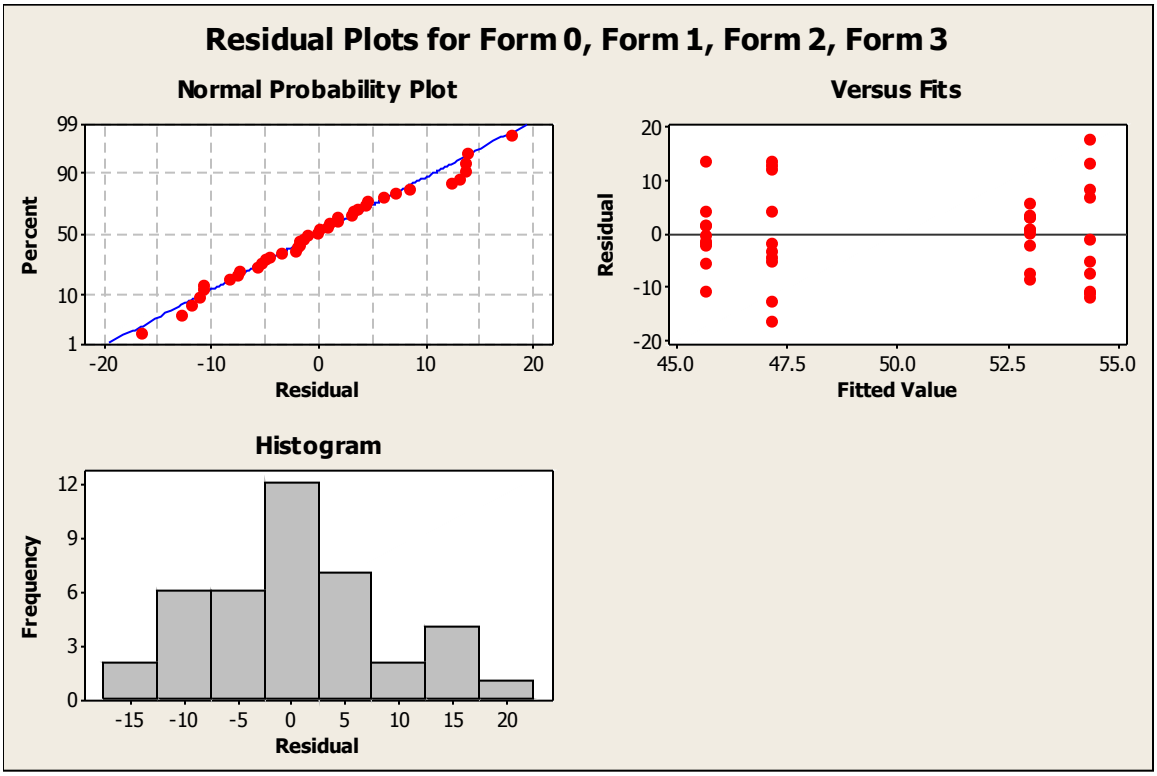
Appendix A

Test for Normality



Appendix B

One Way ANOVA Test with Fisher Comparison



One-way ANOVA: Form 0, Form 1, Form 2, Form 3

```
Source  DF      SS     MS     F      P
Factor   3   548.5  182.8  2.43  0.081
Error   36  2712.5   75.3
Total   39  3260.9

S = 8.680   R-Sq = 16.82%   R-Sq(adj) = 9.89%


                          Individual 95% CIs For Mean Based on Pooled StDev
Level    N    Mean   StDev     +---------+---------+---------+---------
Form 0  10  47.113  10.751        (----------*----------)
Form 1  10  54.317  11.040                   (----------*----------)
Form 2  10  52.950   4.745                 (----------*----------)
Form 3  10  45.621   6.435     (----------*----------)
                               +---------+---------+---------+---------
                              40.0      45.0      50.0      55.0

Pooled StDev = 8.680
```

Grouping Information Using Fisher Method

	N	Mean	Grouping
Form 1	10	54.317	A
Form 2	10	52.950	A B
Form 0	10	47.113	A B
Form 3	10	45.621	B

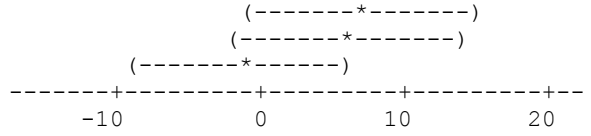
Means that do not share a letter are significantly different.

Fisher 95% Individual Confidence Intervals
All Pairwise Comparisons

Simultaneous confidence level = 80.32%

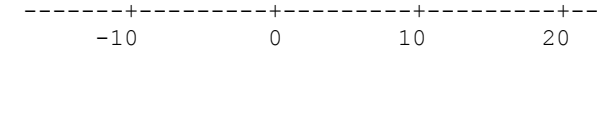
Form 0 subtracted from:

	Lower	Center	Upper
Form 1	-0.669	7.204	15.077
Form 2	-2.036	5.837	13.710
Form 3	-9.365	-1.492	6.381



Form 1 subtracted from:

	Lower	Center	Upper
Form 2	-9.240	-1.367	6.506
Form 3	-16.569	-8.696	-0.823



Form 2 subtracted from:

	Lower	Center	Upper
Form 3	-15.202	-7.329	0.544

Appendix C

Kruskal-Wallis Tests for Questionnaire

```
Welcome to Minitab, press F1 for help.
```

Kruskal-Wallis Test: Layout versus Form

```
Kruskal-Wallis Test on Layout

Form      N  Median  Ave Rank      Z
0        10   5.000      19.7  -0.25
1        10   4.000      12.5  -2.50
2        10   5.000      24.9   1.37
3        10   5.000      24.9   1.37
Overall  40              20.5

H = 7.56  DF = 3  P = 0.056
H = 10.00  DF = 3  P = 0.019  (adjusted for ties)
```

Kruskal-Wallis Test: Overall Appearance versus Form

```
Kruskal-Wallis Test on Overall Appearance

Form      N  Median  Ave Rank      Z
0        10   5.000      23.1   0.81
1        10   4.500      16.9  -1.14
2        10   5.000      21.9   0.45
3        10   5.000      20.1  -0.12
Overall  40              20.5

H = 1.63  DF = 3  P = 0.651
H = 2.32  DF = 3  P = 0.509  (adjusted for ties)
```

Kruskal-Wallis Test: User-Friendliness versus Form

```
Kruskal-Wallis Test on User-Friendliness

Form      N  Median  Ave Rank      Z
0        10   4.500      17.0  -1.09
1        10   5.000      17.7  -0.87
2        10   5.000      26.5   1.87
3        10   5.000      20.8   0.09
Overall  40              20.5

H = 4.11  DF = 3  P = 0.250
H = 6.41  DF = 3  P = 0.093  (adjusted for ties)
```

Kruskal-Wallis Test: Overall Experience versus Form

```
Kruskal-Wallis Test on Overall Experience

Form      N  Median  Ave Rank      Z
0        10   5.000      19.8  -0.22
1        10   4.500      15.0  -1.72
2        10   5.000      25.5   1.56
3        10   5.000      21.7   0.37
Overall  40              20.5

H = 4.18  DF = 3  P = 0.242
H = 7.33  DF = 3  P = 0.062  (adjusted for ties)
```

Appendix D

Correlations: Time, Overall Experience Rating

```
Pearson correlation of Time and Overall Experience Rating = -0.200
P-Value = 0.215
```

Works Cited

Durski, Kamil, Jan Murlewski, Dariusz Makowski, and Bartosz Sakowicz. "Warehouse Management System in Ruby on Rails Framework on Cloud Computing Architecture." Dept. of Microelectron. & Comput. Sci., Tech. Univ. of Lodz, Lodz, Poland, 23 Feb. 2011. Web. 30 Oct. 2014.

Duckett, Jon. HTML & CSS: Design and Build Websites. Indianapolis: John Wiley & Sons, Inc., 2011. Print.

Hartl, Michael. Ruby on Rails Tutorial. 2nd ed. Upper Saddle River, NJ: Addison-Wesley, 2013. Print.

Huang, Liqing. "Research on C2C E-Commerce Website Usability Evaluation System." IEEE Xplore. Jiangsu University, n.d. Web. 13 Oct. 2014.

Huihui, Sun, and Zhang Yu. "Applying Ergonomics to Improve Usability Design of the Interface of the “San Nong” (Agriculture, Rural Areas and Farmers) E-government Website." IEEE Xplore. 2010 International Conference on E-Business and E-Government, 2010. Web. 8 Oct. 2014.

Idrus, Zanariah, Nor Hafizah Abdul Razak, Noor Hasnita Abdul Talib, and Taniza Tajuddin. "Using Three Layer Model (TLM) in Web Form Design: WeFDeC Checklist Development." 2010 Second International Conference on Computer Engineering and Applications, 19 Mar. 2010. Web. 2 Nov. 2014.

LeVine, Eric. CellarTracker. 2003. Web. 2 Nov. 2014. <https://www.cellartracker.com/Default.asp>.

Martin, Ludger. "Gathering Information About Web 2.0 Applications for Usability Engineering." IEEE Xplore. 2010 Seventh International Conference on the Quality of Information and Communications Technology, 2010. Web. 11 Oct. 2014.

Newcomb, Jim. "How Many Wine Collectors Are in the World?" Quora. N.p., 09 Aug. 2013. Web. 25 Feb. 2015.

"The Personal Wine Curator Wine Cellar Software." Features of The Personal Wine Curator Wine Cellar Software. The Wine Curators, Web. 8 Oct. 2014.

Sims, Zach, and Ryan Bubinski. Codecademy. Aug. 2011. Web. 29 Sept. 2014. <http://www.codecademy.com/learn>.

“Vinote Cellar Software And Tags.” Vinote, Web. 21 Oct. 2014. <http://www.vinote.com/WineCellarSoftware.aspx>.

Viswanathan, Viswa. "Rapid Web Application Development: A Ruby on Rails Tutorial." Seton Hall University. 11 Nov. 2008. Web. 30 Oct. 2014.