A Web Application for the La Crosse Symphony Orchestra

A Manuscript

Submitted to

the Department of Computer Science

and the Faculty of the

University of Wisconsin-La Crosse

La Crosse, Wisconsin

by

Joshua Newman

in Partial Fulfillment of the

Requirements for the Degree of

Master of Software Engineering

December, 2010
A Web Application for the La Crosse Symphony Orchestra

By Joshua Newman

We recommend acceptance of this manuscript in partial fulfillment of this candidate’s requirements for the degree of Master of Software Engineering in Computer Science. The candidate has completed the oral examination requirement of the capstone project for the degree.

____________________________________ _______________________
Dr. Kasi Periyasamy Date
Examination Committee Chairperson

____________________________________ _______________________
Dr. Kenny Hunt Date
Examination Committee Member

____________________________________ _______________________
Dr. Mao Zheng Date
Examination Committee Member
ABSTRACT


The La Crosse Symphony Orchestra has been suffering with mediocre tools for quite some time. The Orchestra’s website has little functionality and its DOS auction application was developed over twenty years ago. The Orchestra has been unable to implement new technology to improve its every-day work. A web application that combines the functionality of the old DOS program with the ability to improve everyday tasks for the Orchestra is the aim of this project. This manuscript describes the development of a web-based application that will allow the La Crosse Symphony Orchestra to manage all of its auctions, events and concerts while keeping its customer data organized and easily searchable.
ACKNOWLEDGEMENTS

I would like to sincerely thank my project advisor Dr. Kasi Periyasamy for his help keeping me on the path towards graduation as well as his assistance with the development of this project and the writing of this manuscript. I would also like to thank Tracy Fell, executive director of the La Crosse Symphony Orchestra, as well as the entire Symphony Orchestra for bringing this project to Dr. Periyasamy and for answering any questions I had while developing the application. Lastly, I would like to thank my family and friends, who have put up with me talking about this project for the last nine months and are most likely glad that it is finally finished.
TABLE OF CONTENTS

ABSTRACT................................................................................................................... iii
ACKNOWLEDGEMENTS............................................................................................. iv
TABLE OF CONTENTS ............................................................................................... v
LIST OF FIGURES...................................................................................................... vi
GLOSSARY.................................................................................................................... vii
1. Introduction ............................................................................................................. 1
2. Life Cycle Model .................................................................................................... 4
   2.1 Possible Life Cycle Models ............................................................................... 4
   2.2 The Waterfall Model ...................................................................................... 5
3. Requirements Gathering ....................................................................................... 7
   3.1 Customer Discussions ................................................................................... 7
   3.2 Reengineering the DOS Application ............................................................. 8
   3.3 Finalizing the Requirements ......................................................................... 9
4. Design ..................................................................................................................... 10
   4.1 Use Cases ....................................................................................................... 10
   4.2 Class Diagram ............................................................................................... 19
   4.3 EER Model Diagram .................................................................................... 21
5. Coding .................................................................................................................... 23
6. Verification ............................................................................................................. 25
7. Limitations ............................................................................................................ 27
8. Continuing Work ................................................................................................... 28
9. Bibliography ......................................................................................................... 29
Appendix ..................................................................................................................... 30
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1: Administrator Account Management Use Case Diagram</td>
<td>11</td>
</tr>
<tr>
<td>Figure 2: Administrator Event Management Use Case Diagram</td>
<td>12</td>
</tr>
<tr>
<td>Figure 3: Administrator Concert Management Use Case Diagram</td>
<td>14</td>
</tr>
<tr>
<td>Figure 4: Administrator Auction Management Use Case Diagram</td>
<td>15</td>
</tr>
<tr>
<td>Figure 5: Administrator Auction Report Management Use Case Diagram</td>
<td>16</td>
</tr>
<tr>
<td>Figure 6: All Users Use Case Diagram</td>
<td>17</td>
</tr>
<tr>
<td>Figure 7: Customer Management Use Case Diagram</td>
<td>18</td>
</tr>
<tr>
<td>Figure 8: Class Diagram Relationships</td>
<td>20</td>
</tr>
<tr>
<td>Figure 9: EER Diagram</td>
<td>21</td>
</tr>
<tr>
<td>Figure 10: Home Page of the Web Application</td>
<td>30</td>
</tr>
<tr>
<td>Figure 11: Customer Home Screen for the Web Application</td>
<td>30</td>
</tr>
<tr>
<td>Figure 12: Administrator Home Screen of the Web Application</td>
<td>31</td>
</tr>
<tr>
<td>Figure 13: The Main, DatabaseBasic, and Database Connection Classes</td>
<td>32</td>
</tr>
<tr>
<td>Figure 14: Account, Auction, Event, Concert, and Item Classes</td>
<td>33</td>
</tr>
<tr>
<td>Figure 15: All Report classes, donation, singleTicketTransaction, seasonTicketTransaction, auctionRegistration, and eventRegistration classes</td>
<td>34</td>
</tr>
</tbody>
</table>
GLOSSARY

LSO
Short for “La Crosse Symphony Orchestra,” a non-profit organization that is commissioning and sponsoring this project.

DOS
Short for “Disk Operating System,” an acronym for several closely related operating systems that dominated the IBM PC compatible market between 1981 and 1995.

HTML
Short for “HyperText Markup Language,” the predominant markup language for web pages.

Javascript
An implementation of the ECMAScript language standard and is typically used to enable programmatic access to computational objects within a host environment.

Jquery
A cross-browser JavaScript library designed to simplify the client-side scripting of HTML.

CSS
Short for “Cascading Style Sheets,” a style sheet language used to describe the presentation semantics (the look and formatting) of a document written in a markup language.
**PHP**

Stands for “PHP: Hypertext Preprocessor,” a widely used, general purpose scripting language that was originally designed for web development to produce dynamic web pages.

**MySQL Workbench**

MySQL Workbench is a visual database design tool that integrates SQL development, administration, database design, creation and maintenance into a single, seamless environment for the MySQL database system.

**IE**

Windows Internet Explorer is a graphical web browser developed by Microsoft and included as part of the Microsoft Windows operating systems.

**Mozilla Firefox**

Mozilla Firefox is a free and open source web browser descended for the Mozilla Application Suite and managed by Mozilla Corporation.

**Google Chrome**

Google Chrome is a web browser developed by Google that uses the WebKit layout engine and application framework.
1 Introduction

The La Crosse Symphony Orchestra is a non-profit organization that tries to bring music to the community of La Crosse. They also try to help educate young people to the wonders of music. Like any other non-profit organization the LSO does not always have the funds to keep up with constantly changing technology. The LSO currently has a web site that is designed and maintained by Meridian Corporation at no cost to the LSO. Because the website is designed and maintained for free, the LSO is usually at the mercy of Meridian for implementing requested improvements and typically has to wait longer for changes to be made. In addition, the web site does not have many of the functionalities desired by the LSO. While there is plenty of helpful information on the La Crosse Symphony Orchestra’s current website, many parts of the web site are not working properly. For example, if someone wants to buy season tickets or tickets to a concert, they have to select the tickets they would like to buy, submit their contact information as well as their phone number(s), and wait for a response from the LSO. A LSO representative will read this message, call the customer, and record the credit card information from the customer(s). It almost makes going to the website to order tickets redundant, because you could have simply called the LSO directly. Other problems with the current website include:

- The seating chart is supposed to show an approximate view from a section if you hover the mouse over it, and it only shows a black screen.
- Currently, no checking of the fields takes place when buying tickets. You can submit a ticket order and as long as something is included in each field, the e-mail is submitted.
- The roster of orchestra members includes a number of people who have names with hyperlinks, but the links do not work.
- The link for directions does not take the user to directions or even give the address, it merely sends them to Mapquest.

Needless to say, the Symphony Orchestra would like to improve upon the system that is currently in place by adding significantly more functionality.
Because the website does not currently record any information of its own, the staff at the Symphony Orchestra is forced to keep track of customer purchases and donations manually in different formats such as paper copies, EXCEL files, and WORD documents. This has resulted in a large number of Excel spreadsheets with a variety of customer data which makes finding a specific group of users extremely difficult. For example, if the Orchestra would like to recognize all patrons who have donated over $1000 and are also season ticket holders, they must compare two different spreadsheets to find the group for which they are searching. This can be painstakingly difficult. If all of the Orchestra’s customer data was centrally stored and could be accessed by a central search function, it would make the LSO’s staff’s job much easier.

Another large event that the Symphony Orchestra holds every year is the Valentine’s Ball, a money-raising auction in which people or local businesses donate items or services to be sold to the highest bidder with the proceeds going to the Orchestra. The administrators of the Symphony Orchestra currently use a DOS-based application that is connected to a simplified database to keep track of all auction attendees and items that have been donated. Every year when this auction is conducted, the original creator of this application is asked to come to the LSO and handle an application that almost no one else understands. He completely erases the previous year’s data from the locked database. Then, the Orchestra staff must input all new data about the attendees and the items that have been donated. Once this tedious task is completed, the staff must bring a laptop with the DOS-based program on it to the auction. From this laptop, the staff associates the auction items with the patrons who have bid the highest for the items and use this information to print out receipts for everyone at the end of the evening. In order to print out these receipts, though, the staff must employ a special printer because the program only recognizes this solitary device. After completing the recording and printing processes, the database locks itself, and must be unlocked once again the following year. To make matters even worse, this DOS-based application will not run on any 64-bit operating
system; consequently, when the Symphony Orchestra tries to upgrade its staff computers, this program will become useless to the organization.

Taking into account the status of the website, as well as the unsustainable and intolerable nature of the DOS auction program, the Symphony Orchestra staff came suggested plenty of functionality they would like to see in the web application. The Orchestra’s major requests included:

- The ability for any customer to buy any type of tickets online through the LSO’s web site. These tickets may be either season tickets, or tickets to a specific concert, event, or auction.
- The amalgamation of the organization’s entire customer data into one system so that the staff will not only keep track of purchases and donations, but also will be able to query any specific set of users.
- Re-engineer the DOS application so that its limitations would be reduced or eliminated. In particular, its portability as well as the tedious data entry that accompanied the Valentine’s Ball every year must be improved.

While the ability to purchase tickets online will be included in the web application, the committee agreed that storing and processing credit card information was a significant legal liability for the university, the developer, and the Symphony Orchestra. It was determined that the system will simply send the customer’s credit card information directly to the Orchestra via e-mail instead of processing it online. Further enhancements to the system could include secure online purchases, perhaps through a third-party, such as PayPal.

The web application will be accessible directly from the La Crosse Symphony Orchestra’s current website, so the need for security is high because anyone with Internet access could potentially access the system. However, because credit card information is not being stored in the database, the risks are limited to personal information, most of which can be found in a phone book or by searching the internet.
2 Life Cycle Model

Software engineering is a good process for developing applications because of the planning that goes into the model that is selected. Adherence to a software engineering model can take a project from acceptable to almost perfect, but that will only occur if the correct model is chosen.

2.1 Possible Life Cycle Models

After being initially introduced to the project and its basic requirements, the developer immediately considered three possible life cycle models: the waterfall model, the spiral model, and incremental prototyping. The waterfall method was the prime candidate for selection from the beginning because the required system seemed to be a straightforward data entry system. The spiral model was the next method considered because it is very thorough and, after the requirements have been gathered, the system can be perfected through repeated iterations of prototypes. Any risks that were unaccounted for or unknown at the beginning of the project would be fixed in the repeated iterations. The spiral method would allow the system to be as close to perfect as possible. Incremental prototyping would allow the customer to visualize the product in gradual increments. It would also allow the project to be easily adapted to changing requirements.

During the first meeting with the La Crosse Symphony Orchestra the developer observed that the staff had very specific ideas as to what functionality should be included in the web application. This enabled the developer to create a well-defined subset of requirements for the project. Coupling these specific requirements with the general design requirements of a web-based application, the project risks seemed minimal, and so the spiral model was not chosen. Because the requirements for this project were straightforward and well-understood, the developer determined that the waterfall model was the correct approach for this application. Even though the developer anticipated that some changes in the requirements and/or design could occur, these changes could still be accommodated through the feedback path of the
waterfall model. In other words, there is no need for a new cycle as is typically required in the prototyping model.

2.2 The waterfall model

The waterfall method proceeds from one phase to the next in a sequential manner, and only after the activities in each phase are completed. Generally, it is not necessary to visit a previous phase again, except for minor changes. For example, the design phase is begun only after the requirement phase is completed. That is, all requirements pertaining to the project are gathered, analyzed and well documented into a well-organized and structured format. The waterfall method does not adapt well to changes in the requirements. To successfully employ the waterfall method, the developer should have a very good initial understanding of the project and should also know where potential pitfalls may lie. The Symphony Orchestra’s requests were not very complex, and upon further review of the DOS auction program, the plan for this data management system became very clear.

Another potential problem with the waterfall method is its ability to adapt to unforeseen problems with projects. Once a particular phase of the project has been completed, it is possible that some essential part of the system was overlooked or missed. If this happens, the earlier requirements phase has to be repeated from the beginning and all of the time previously spent on this phase has been wasted. Because the LSO data management project was a straightforward data management system, and because the requirements were well defined and unchanging, the developer concluded that any potential problems could be identified and easily handled. Further, if these problems were adequately anticipated, the issues would not develop into serious concerns. While unforeseen problems can always arise, the developer had sufficient experience with data management and web application systems and believed that unforeseen events could be overcome without much difficulty.

Having confronted all of the possible problems with the waterfall method, and given that the project seemingly fits perfectly with this model, the waterfall method
was selected by the developer and work was begun on the first phase of requirements gathering.
3 Requirements Gathering

Requirements for this project came from two main sources: the staff at the La Crosse Symphony Orchestra, mainly Executive Director Ms. Tracy Fell, and Development Director Ms. Diana Miller, and from the DOS auction application that was to be reengineered.

3.1 Customer Discussions

The main source of the requirements for this project was the employees of the Symphony Orchestra themselves. In the first meeting with the LSO, the project was discussed in broad terms, with general requirements being the starting point for the conversation. The LSO employees described the current operations as well as the problems they have maintaining the regular activities. From there, they discussed what would make the daily processes of their jobs easier and more maintainable. In the second meeting with the LSO, more specific functionalities were discussed. While the LSO was sometimes unsure what was possible to be implemented in the program, the developer was able to guide their thoughts and ideas into executable requirements. From these broader requirements, specific features and capabilities were discussed in the next meeting.

The LSO was very emphatic about certain features being included, such as the ability for customers to buy tickets to concerts, events, and auctions through the website. The employees at the LSO were also focused on improving the procedure for their annual auction. The LSO was sure that their current process of handling the auction could be improved by a large amount with this program and the developer was sure to fully encompass all parts of the auction in the application. The rest of the functionalities were extracted from the descriptions of daily operations from the LSO. The other source for requirements was the DOS auction application that the La Crosse Symphony Orchestra had asked to be reengineered and included in their web application.
3.2 Reengineering the DOS Application

The La Crosse Symphony Orchestra has been using the DOS application for many years in order to keep track of attendees and donated items in their yearly auction, the Valentine’s Ball. One of requirements for the current project was to re-engineer this DOS application and to include it in the web application that was being developed. To understand and run the auction application, the Orchestra was asked to provide insight into how exactly they used the application. Based on the discussion with the LSO staff, the developer was able to identify the following functionalities from the DOS application:

- Add a new auction attendee to the application
- Add a new auction item to the application
- Generate reports on attendees and items separately
- Generate receipts for those attendees who won auction items and also receipts to be kept by the LSO. Receipts must include auction price, applicable taxes and total.

These four functionalities are given by the LSO staff. The exact details of these functionalities were required to be gathered by running the old DOS program and observing its inputs and outputs. This posed a lot of challenges. For example, the developer gathered the details of the first functionality by adding different users with different names and related information one at a time and then generating reports. This was done a number of times to figure out what each input field was for, as well as to figure out what the automatically generated numbers meant. Adding an item to the system was performed in the same manner, by adding an item and then generating a report. This was more difficult because of the abbreviations that were used for some of the fields, as well as the unknown nature of the inputs. When adding an attendee, fields such as first name, last name, and address were pretty straight forward. But when adding an item, it took some time to figure out what bidder number, table number, minimum bid, and the abbreviation RO# all have in common.
The generation of reports was a little more complex because of the large number of possibilities included in the reports. The reports could be generated from any field that was included in either the auction attendee record, or the donated item record, thus leaving many possible reports to decipher. This was observed throughout the requirements extraction process. At the end, receipts were generated correctly and the print function was understood.

Once all of the parts of the DOS auction application had been observed, the generated requirements were presented to the Symphony Orchestra. They reviewed and confirmed that the findings were correct. Combining these requirements with the requested requirements from the staff of the Symphony Orchestra, a proof of concept implementation was developed.

3.3 Finalizing the Requirements

Once the proof of concept implementation was finished, it was shown to the Symphony Orchestra staff. The proof of concept was just a front end written in HTML, with javascript, and jquery included. The proof of concept was created so that the Orchestra staff could visually see what their requirements would look like once the project becomes fully operational. The developer was also hopeful that this front end design would be acceptable to the Symphony Orchestra for the actual project. All requirements that were agreed upon with the Orchestra staff, as well as the requirements from the DOS auction application were included in the proof of concept. Seeing their requirements in action sparked a few additional functionality requests from the Symphony Orchestra staff. Some of these additions included the ability to print customer searches in a specific way, as well as allowing the customers to register for events and buy tickets for auctions through the system. These additions, while not complex, had to be worked into the overall requirements before the requirements phase was complete. Once these requirements were completed, the design phase was begun.
4 Design

After the requirements were gathered, completed and revised after meeting with the Symphony Orchestra, the design phase started. The first step to this process was discovering the use cases for this project and then creating the use case diagram. The use case diagram is used as a high level view of each action that the system performs. The use case diagram not only enables the designer to view the functional requirements but also serves as the basis to extract information for the class diagram. The second step was to take the completed use case diagram and translate it into a more concrete class diagram that would then be used for coding. Each use case can become one or many functions that are represented in the class diagram. Usually, each use case is the starting call from the main class to make the represented action occur. Lastly, the Enhanced Entity-Relationship (EER) Model diagram was created to help model the database design and plan out the back end of the project before the actual coding would take place.

4.1 Use Cases

The requirements which serve as the basis for use case diagrams were broken down into two main sections, requirements for the administrators, and requirements for the customers. The requirements for the administrators were further broken down into four groups, account management, auction management, event management, and concert management. The requirements for the customers were further broken down into two groups, account management and ticket management. Event management and concert management were separated out from the whole and each broken down separately, with each group having its own use case diagram. Because auction management had so many requirements, this was broken down into two different use case diagrams, one including auction reports, and the other including all other auction management requirements. The customer ticket management requirements were also broken down this way, and also have their own use case diagram. Because some of the account requirements overlapped for both the customers and the administrators,
one use case diagram was created with the overlapping use cases, and a separate diagram containing just the additional administrator account use cases was also created. Overall, there were a total of six use case diagrams.

The first use case diagram that was compiled from the requirements was the administrator account use case diagram seen in figure 1.

![Use Case Diagram](image)

**Figure 1. Administrator Account Management Use Case Diagram**

This diagram was completed first because of its overall importance to the system and the necessity for it to be correct. Without these fully correct use cases, the system would be effectively useless because no administrator would be able to use or manage the system and its users. The five use cases included in this use case diagram are add administrator, add new user, record transaction, delete account, and search account.

The ‘add administrator’ use case is used to add additional administrators to the system so that more people can help manage the system in addition to the one built-in administrator. The use case ‘add new user’ is used only when a customer is not in the system and makes a purchase directly from the Orchestra. In this case, in order to record the transaction, the user must first be added to the system. The ‘record transaction’ use case is used to record a transaction that is not made through the
The possible transactions that can be recorded are: purchasing season tickets, purchasing single concert tickets, donating money to the orchestra, registering to attend an event, or buying tickets to an auction. ‘Delete account’ use case is used to delete any accounts that have been inactive for at least one year. Because the Orchestra stated that they would like to continue to keep this data after the account has been “deleted,” this delete will be a soft delete, meaning that logging in will no longer be possible for this user, and they will not show up on any account searches unless inactive accounts are requested. Lastly, the ‘search account’ use case is used to find customer account in the system based upon certain criteria. These criteria include first name, last name, whether the user is a season ticket holder, whether they have purchased concert tickets, whether they have attended an auction, whether the user has volunteered with the Orchestra before, and whether they have donated money or items to the Symphony Orchestra.

The next two use case diagrams that were compiled were the event diagram and the concert diagram. In the system, events were defined as one time happenings that may or may not have a cost involved. Events may overlap with each other. The event management use case diagram can be seen in figure 2.
Each event has a name, date, start time, end time, location, price, and description associated with it. Events are all scheduled occurrences that are not concerts or auctions, but that the Orchestra would still like people to attend or notice on the calendar.

The five use cases in this diagram are: ‘create event’, ‘modify event’, ‘cancel event’, ‘delete event’, and ‘search event’. The use case ‘create event’ is used to generate a new event in the system associated with the given input data. As long as the date is in the future, and the start time is before the end time, as many events as needed may be required. According to the Orchestra, they may only have three or four such events a season. The use case ‘modify event’ is used to change any of the data associated with the event that is to be modified. The same constraints from ‘create event’ apply. ‘Cancel event’ use case is used in the rare case that an event will not be taking place anymore. Once an event is canceled, a list of all customers who have registered to attend the event is returned so that the Orchestra may contact them and alert them or offer them a refund. The use case ‘delete event’ is used to make events older than a year as inactive so that they do not show up when searching through past events. Lastly, ‘search event’ use case is used to find any event in the system by event name, time period, or both. Deleted events will not be included in the search unless otherwise specified before the search is started.

Concerts were defined to be one time occurrences to which tickets are sold and cannot overlap. The concert management use case diagram can be seen in figure 3. A concert consists of a name, director, guest artist, date, start time, end time, upper balcony ticket price, lower balcony ticket price, main floor ticket price, location and a description.

The five use cases included in the concert management use case diagram include ‘create concert’, ‘modify concert’, ‘cancel concert’, ‘delete concert’, and ‘search concert’. The use case ‘create concert’ is used to generate a new concert in the system. The required parameters include a concert name, date, start time, end time,
and the three ticket prices. A concert cannot be created on a date in which a concert already exists or is in the past, and the start time must come before then end time.

![Administrator Concert Management Use Case Diagram](image)

`All Dependencies are of type <<include>>`

**Figure 3. Administrator Concert Management Use Case Diagram**

The rest of the concert use cases, ‘modify concert’, ‘cancel concert’, ‘delete concert’ and ‘search concert’ all follow the same pattern as their event counterparts but use the information associated with concerts instead of the information associated with events.

The final administrator requirements group, auction, was broken down into two use case diagrams because there were so many use cases. These two use case diagrams are the main auction use cases seen in figure 4 and the auction report generation use cases seen in figure 5.
An auction is defined as a onetime event that cannot overlap with any other concert or auction to which tickets are required to attend. People who buy tickets are considered auction attendees, and are associated with the auction, along with any items that are donated to the specific auction. A specific auction consist of a name, date, start time, end time, price, location, and a description as well as a list of auction attendees and a list of auction items.

The use cases ‘create auction’, ‘modify auction’, ‘cancel auction’, ‘delete auction’, and ‘search auction’ each function the same way as their equivalent in the event and concert use cases. New use cases include ‘add attendee from system’, ‘add attendee
not in system’, ‘add item from system’, ‘add item not in system’, ‘remove attendee’, ‘remove item’, and ‘record item winner’. Both the ‘add attendee from system’ and ‘add attendee not in system’ are used to add auction attendees to an auction. If the customer is already registered with the system, the ‘add attendee from system’ is used. If the customer is not yet in the system, then they will be added to the system as well as added to the auction with the ‘add attendee not in system’ use case. The idea is the same behind the use cases ‘add item from system’ and ‘add item not in system’.

If the item donor is not in the system, the ‘add item not in system’ should be used, which will then add the donor to the system as well as adding the item to the auction. The ‘remove attendee’ and ‘remove item’ use cases are fundamentally the same, each of them removing for an auction what their name implies. Lastly, ‘record item winner’ is used after an auction is complete to associate the auction attendee with the items that he or she won.

The second auction use case diagram, the administrator auction report management use case diagram contains all five reports that are used by the Orchestra to generate data about their auctions. These use cases are ‘generate full auction report’, ‘generate item report’, ‘generate donor report’, ‘generate attendee report’, and ‘generate receipts report’.

![Figure 5. Administrator Auction Report Management Use Case Diagram](image-url)
‘Generate full auction report’ returns all data associated with the auction, including name, date, start time, end time, price, location, and description, as well as all of the auction attendees with their information, and all donated items and the information associated with the items. The use case ‘generate item report’ returns all information about a specified item, or about all items in the system if desired. ‘Generate donor report’ returns all information about the item donors for the specified auction. ‘Generate attendee report’ returns all information about a specified attendee, including all purchases made in the auction. Lastly, ‘generate receipts report’ returns a receipt for either a specific auction attendee or all auction attendees so that their items may be collected and the attendee can be charged.

The overlapping use case diagram, which includes use cases that both the administrator and the customer use, can be seen in figure 6 below.

Figure 6. All Users Use Case Diagram

The four use cases included in this diagram are ‘login’, ‘logout’, ‘change password’, and ‘modify information’. The ‘login’ and ‘logout’ use cases are self-
explanatory. ‘Change password’ is used to update a user’s password, and ‘modify information’ is used to update and personal information that may have changed for the users.

Lastly is the customer management use case diagram as seen in figure 7. This diagram includes all of the functionalities that are available to just the customers in this system.

![Figure 7. Customer Management Use Case Diagram](image)

The six use cases included in this diagram are ‘register’, ‘buy single tickets’, ‘buy season tickets’, ‘donate money’, ‘event sign-up’, and ‘auction sign-up’. The use case ‘register’ is used the very first time a customer is using the system. They provide their first name and last name, and if a record of their data is in the system they are matched to it. They are then asked to fill out all missing information and provide a
password. After this is complete, any time after they would like to access the system, they use the login use case. The ‘buy single tickets’ use case is used to buy tickets to a specific concert. The concert is selected and credit card information is provided and all of this is e-mailed to the La Crosse Symphony Orchestra for processing. ‘Buy season tickets’ is very similar, except a concert is not specified. ‘Event sign-up’ is used when a customer would like to register to attend an event. Just like ‘buy single tickets’, an event is specified and the credit card information is supplied and all the given information is e-mailed to the Orchestra for processing. Finally, ‘auction sign-up’ is used to buy tickets for an auction and then specify who is attending the auction.

After all use case diagrams were completed and the use case narratives were written, the class diagram was started.

4.2 Class Diagram

The class diagram for this project is shown in Figure 8. Each main grouping of use cases needed its own class. For example, the initial set of classes became account, auction, event, and concert. It was also prudent to include classes for each report, because they included lots of information and having a specific class for each one allowed for better organization. Event registration and auction registration were created to help keep track of all customers who were signing up for events and auctions. They allowed the system to make a connection between account, and either auction or event. The same can be said for the singleTicketTransaction class. This allowed the system to make a connection between account and concert. Donation and seasonTicketTransaction are separate from everything because they have nothing to do with any specific event, concert, or auction, but only associate to the customer account. Lastly, the database connection class connects to the basic database class that includes all database transactions that are used throughout the system as accessed by the main class.
The attributes and methods for each class can be found in the appendix. The class diagram was built using the use case diagrams, as well as the developer’s past experience developing class diagrams in previous projects. After the class diagram the EER Model was designed using the built-in tool in MySQL Workbench.
4.3 EER Model Diagram

The built-in tool in MySQL Workbench allows a user to design the tables in a database along with foreign keys associations and it then generates the EER model diagram. The EER diagram for the web application is seen in figure 9.

Figure 9. EER Diagram
The database tables were created from a combination of knowledge taken from the class diagram, and remembered from past experience of the developer. The relationships between the tables are very similar to those in the class diagram, with the user table taking the place of the account class, and every other table taking the place of the associated class.
5 Coding

Once the design phase was completed, the coding phase was ready to begin. The first step to coding is deciding what platforms will be used in each of the three parts of the application. Because this was a web application, HTML was the main language used in the user interface portion of the application. In addition to HTML, CSS was used to keep the look and feel of the application the same no matter what page the user was viewing. Javascript and jQuery were used to add some moving parts and color themes throughout the user interface. They are responsible for components such as the calendar feature that shows up when choosing dates, the time picker that shows up when choosing times, and the accordion style in which the event calendar and search results are displayed.

Because the proof of concept implementation was written with these languages, not much needed to be changed when the actual project was being developed. The only parts that needed to be added into the front end was the server code.

The server code was forced to be in PHP. Because Meridian Co. is responsible for the La Crosse Symphony Orchestra’s web site, they deal with the hosting company, which is GoDaddy.com. Meridian Co. has purchased from them the lowest level package which only allows PHP, ASP, or Silverlight on the Linux based plan. PHP is what Meridian Co. uses PHP for all of the sites they design, so the developer thought that sticking with that language would be a good idea.

The difficulty here was that PHP was a new language to the developer. After looking at many examples and code samples, the most basic parts of the project were implemented first. The functionalities to add events and concerts to the system were the first parts added. This was a slow going process learning how to include the PHP code with the HTML and javascript as well as figuring out how to make a database connection without having to write a new line of code every time a database query was made. Luckily, things began running smoother and the finer points of PHP were discovered, thus allowing efficient coding throughout most of the coding phase.
Lastly, the database was chosen to be MySQL database because of its ease of use, facility to accommodate large storing area, as well as the fact that this is the database that is included with the GoDaddy.com plan.
6 Validation

Since the application is voluminous, it is almost impossible to test every possible action that a customer or administrator could complete. Therefore, the developer was forced to pick and choose the best way to test the most important pieces of the system. Wherever possible, manual input was eliminated with the thought being the more point and clicking, the less chance of mistakes. However, testing was still required in many places. For this reason, testing was broken down into two parts, testing a portion of the project individually, and testing it as a part of the whole.

When testing a portion of the project individually, one specific functional requirement was tested at a time. For each functional requirement, the list of actions was performed in the order they were listed. Also, all exceptions for each functional requirement were tested to make sure that no errors or unexpected results would occur. This same process was done with the use case narratives, using the precondition list to test against for each use case. As soon as one functionality was completed, it was tested on all possible boundary conditions, if there were any. If manual input was inevitable, an educated guess was made to determine what the most likely erroneous inputs would be. These erroneous inputs were then tested and protected against. If a given input was incorrect and not in the erroneous set it may still cause a problem. To limit these errors, the idea was to limit the number of manual inputs to as few as possible. For example, while there is no good way to test if a customer’s credit card number is correct, it is possible to limit some of the most obvious mistakes, such as making sure that the input is numerical, and has the correct number of digits. This helps cut down on mistakes.

Once the individual functionalities had been tested, it was now time to make sure that they worked together as a whole. This involved navigating through the application with a mindset like a customer or an administrator would have. The Symphony Orchestra staff helped with some part of this process. This testing was done by using the application as it would be used once it had become active. Enrolling in events and auctions, buying season and concert tickets, and donating
money all functioned the same no matter what order they were accessed. This was easier to do for the customer side of the system because there is not as much functionality as there is on the administrator side. However, this was the general idea behind the testing for the system on the administrator side. The developer input a large amount of made up data that was created to closely resemble what the real data would look like. Then the developer spent time manipulating that data in using all functionalities of the program while checking for errors.

While the hope is that all errors are found and corrected, the chance of a system this size developed by one person having no errors is small. Hopefully if errors are found once the system is put into operation, they will be small cosmetic errors that do not affect the operation of the system.

In addition to testing the code, the system was tested for workability on different browsers. The application was developed using IE8 as the test browser, and it works great in this browser. It also works well in the most recent versions of Firefox and Google Chrome. It only has trouble with earlier versions of Internet Explorer, and there is a fix built into the system to fix the issues with IE7. Anyone using an earlier version of Internet Explorer will have to upgrade if they would like to use the system.
7 Limitations

There were two limitations to the development of this application, the developer’s lack of experience with PHP, and the developer’s lack of experience with databases in general.

The lack of PHP experience has led to what is most likely inefficient code. This is important in a web based application because all of the server code has to be processed and the results sent back to the user. If the server code is inefficient, then the time between page loads may become too high and users may complain that the application is too slow. This could cut down on the number of users that use the application thus making it useless.

The lack of database experience has most likely led to database tables that could be organized and interconnected better. The lack of organization and interconnectedness leads to the search functions becoming long and complicated procedures that could probably be simplified down with a better database design.
8 Continuing Work

The next possible step in this project would be directed towards automatic checkout when purchasing tickets or registering for an event. Currently, while better than the previous system, the orchestra still has to process the credit card information that is sent to them via e-mail notification. However, a future addition to the application could be adding something like PayPal, where an account can be set up for the Orchestra. Another addition is to include a secure credit card processing function that would automatically charge credit cards when the information was submitted. While some people balk at using PayPal, having secure processing capabilities would add to the application and improve on its usability.

Another possible addition to the system would be the ability to print out tickets from home after purchasing them. This would save on mailing costs for the Symphony Orchestra, and it would be more convenient for customers as well.
9 Bibliography


Appendix

Figure 10. Home Page of the Web Application

Figure 11. Customer Home Screen for the Web Application
Figure 12. Administrator Home Screen of the Web Application
Figure 13. The Main, DatabaseBasic, and Database Connection Classes
Figure 14. Account, Auction, Event, Concert, and Item Classes
Figure 15. All Report classes, donation, singleTicketTransaction, seasonTicketTransaction, auctionRegistration, and eventRegistration classes