# Inf 43 – Fall 2014 – Homework 1

**Student Name:** **Tommy Anteater**
**Student Number:** **12345678**

| Awarded Points | Maximum Points | Document Aspect |
|---|---|---|
| | 15 | Clarity of writing (spelling, grammar, sentence construction) and Clarity of expression (flow, structure, making logical arguments). Roughly 7.5 each. |
| | 15 | Introduction / Executive Summary (can be different sections or combined into one) |
| | 7.5 | Application Context / Environmental Constraints (can be different sections or combined into one) |
| | 35 | Functional Requirements, including use-case diagram and each use case (following a use-case template). |
| | 7.5 | Software Qualities and Non-functional Requirements |
| | 5 (+5) | Other Requirements and Other Items. At least a Glossary of Terms. You can earn up to 5 points Extra Credit if you go beyond Glossary |
| | 7.5 | Assumptions / Risks (can be different sections or combined into one) |
| | 7.5 | Priorities / Implementation Phases;<br>Future Directions and Expected Changes |
| | **100** | **TOTAL** |

# BeachBurn Manager System Requirements

October 28, 2014

Tommy Anteater

# Introduction

The purpose of the software described in this document is to provide a cohesive environment for the ABC event management firm. The event managers have been struggling to organize all of the information surrounding the festival, so there needs to be software that will allow all of the different managers to store the information and work with it all in one rather than many separate systems.

The current problems that lead to the development of the software is the inability for the managers to be updated on the current state of the festival and the individual users. It does not account for obsolete seating, tickets sold from different vendors at the same time, band cancellation, refunds, the various sections of the festival, and more.

Each user of the system will need the ability to have different kinds of access and abilities so that they can work with the system properly. This document will specify the requirements of the software and all that it should contain and do.

This BeachBurn Manager (BBM) system will allow an admin to give the managers access to certain aspects of the software so that they can fulfill their jobs successfully and without complications. The process of ticket handling for customers will be simpler, and the seating arrangement handling will be more flexible yet logical. The system allows the managers to work together and have their responsibilities mix together while still allowing for separation to avoid confusion.

# Overview / Executive Summary

The goal of this system is to combine and organize all of the needs of the managers of the company into one software system. With the complications that surround the festival, the ticket selling, the seat assigning, and the band handling, there needs to be a system that can work with all of these components smoothly. The major managers that will be working with this software system are: band/schedule manager, stadium manager, ticket pricing manager, user administrator.

The band/schedule manager will be managing the bands and organizing who will go on when. They will handle scheduling the bands and working with them. The main problem they need resolved is what to do when bands cancel. If the band cancels, he will need to find a replacement. The system will need to recognize the cancelation and be able to handle a change. The band manager will need to be able to input the bands, the times they are performing, and make sure that information is available.

The stadium manager handles the seating. They must be able to eliminate obsolete seats, such as ones that are behind pillars so the customer cannot see the stage. There is also assigned seating, so the stadium manager must be able to adjust any seating arrangement issues. The program should recognize that there are a total of 500 seats, and so once all 500 have been assigned it cannot assign anymore. In addition, the software needs to have the ability to adjust the different sectors of the arena and the size of them for every day. Each day, there could be a different number of sectors (such as general admission, VIP, etc.). There is also a variable amount of people for each sector depending on the day. The program must be able to adjust to this according to how the stadium manager decides it should be.

The ticket pricing manager handles the sale of the tickets and the pricing. The system must be able to handle multiple vendors selling tickets without having them sell the same seat to different people due to them being sold at the same time. The ticket price should be able to be manipulated by the ticket pricing manager. Different sectors should cost a certain price, and the prices should vary by the day. It should be possible to still manually change the ticket price so the manager could provide discounted tickets if needed. This also leads to refunds. If a band is cancelled, a customer may want to refund their ticket (or for any other reason). If they would like to do this, the ticket manager should have the capability to locate their ticket and have their information on file in the system. They should then be able to issue them a proper refund.

The user administrator has control of what screens and views the other users will have. They grant access to users based on their job functions and what they need to know. The user admin should have the ability to add new users on to the system and allow them to access certain aspects of the program. The user administrator must assign a user to the components of the system in order for them to use it. This is the only way it can be accessed. Users should be able to get in through a password.

The ticket information should be saved in an excel spreadsheet. This information includes the customer's name, the sector of the stadium they purchased, the price of their ticket, and what day they are going. This must be accessible to the ticket manager and the stadium manager.

In addition, it is extremely important that the software have the stadium and ticket managers work together so that they don't sell more seats than they have. They need to work together so the program knows how many seats have been sold and which seats they were.

## Application Context / Environmental Constraints

This software should be a web-based application only, so it does not need to work on multiple different platforms. The software does need access to the network drive. Since the software will not be used by a large amount of people, and not by customers, the attractiveness of the interface is not a priority. However, since the users are not necessarily technical, it still needs to be simple and usable. The program needs to be cohesive and cross-functional for the various users.

The program will generally only be used on the local network in the office and/or ticketing station of the stadium. It should hypothetically not be limited based on the operating system. The program will assist the stadium manager, band/schedule manager, and ticketing manager in performing vital tasks for their jobs with the festival. This software is necessary in allowing customers to purchase the tickets properly, so it will have a major impact on both the users of the system and the people the users will sell to. It will be the primary method of working with many aspects of ticket handling for the festival. The program will need to be able to create a spreadsheet of the information of the customers who purchase the tickets, but aside from this it should not affect any other software. It does not need to change anything on a website that customers could potentially see. Any programming language can be used as long as it can properly perform the necessary functions.

# Functional Requirements

Figure 1 shows the use case diagram for the system.

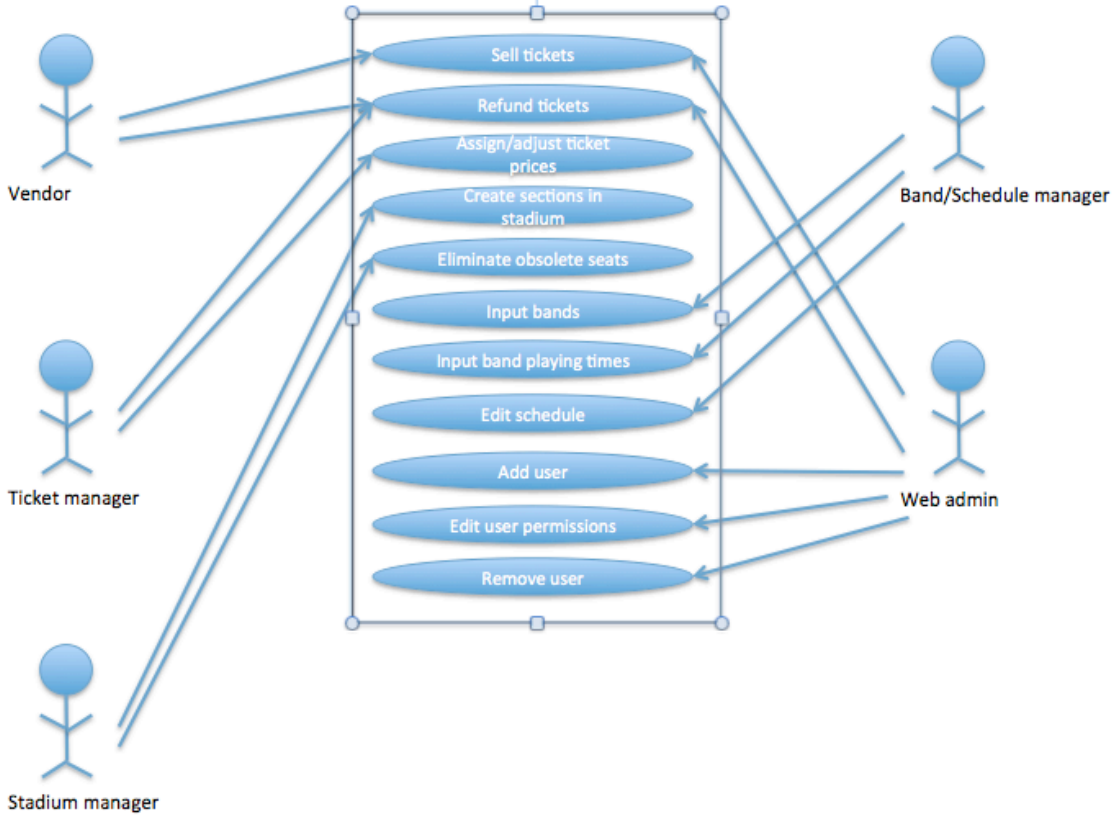


**Figure 1: BeachBurn Manager Use Case Diagram.**

The main functions of the system should be handled depending on how they occur. The following is each requirement and the potential flows that could occur:

**Selling Tickets:** Basic flow – the vendor inputs the customer's name, ticket price, the day of the festival they are attending, which section they are sitting in, and payment information. The system takes this information and then prints it on to a ticket. The information is also then stored by the system into a spreadsheet document that records the information.
Alternatives – the vendor may need to change the price of the ticket before the purchase occurs if the band lineup changes, if the customer needs a discount for any reason (such as a friends and family discount), etc. The vendor will need to be able to change the price in the system and then sell the ticket to the customer at the updated price. It is also possible that the customer will want to attend multiple days, so the vendor will have to input multiple days associated with the customer's purchase. The system should reflect this addition. The customer should also have the ability to upgrade their ticket to a more expensive section. This could occur if they at first bought general admission seats but then decided later that they wanted to be in the VIP section. The system should allow the

customer's order to be edited and then modify the spreadsheet. The new, more expensive seat should then be taken and the old seat should be made available.
Exceptions – if there is an issue with payment, such as the credit card is declined or the customer does not have enough cash, then the system will not be able to complete the transaction. If this occurs, the system should be able to have the customer attempt a different payment method, and if there is no other method available then the order can be cancelled without assigning a seat or adding to the spreadsheet. Another scenario the system needs to be prepared for is if the customer attempts to purchase a ticket for a day of the festival in which all of the seats are already taken. The software should recognize if a day is sold out and not allow any customers to buy tickets for sold out days. The vendor should be given the option to input a different day for them to purchase or cancel the order.

**Refund tickets:** Basic flow: A vendor should be able to obtain a customer's name, find their information in the system, and then refund them for their ticket if the customer desires. The refund should be noted and their seat in the system should go back to being available and ready to be sold.
Alternatives: If the customer just needs a partial refund, the system should allow that to occur. This could be a potential necessity if the customer bought the ticket full price but later discovered that they could have taken advantage of a discount (such as a friends and family one). This should also be doable if they want to downgrade their seat from a more expensive section to a less expensive one. This should be reflected in the system. The customer's information should be modified so that it is noted that they changed their seat, and then the new seat should be taken and the old one made available to purchase again. In addition, a partial refund could be necessary if the price of the ticket decreases after a customer already purchased it at a higher price. If this is the case, their seat should not be released but they should still get some of their money back, and the price of their ticket should be adjusted in the system.
Exceptions: Before allowing the vendor to refund the customer, the system needs to be able to search for the name of the person and retrieve their information. If the name the customer gave is incorrect and not in the system, the vendor should be not be able to refund them regardless. There needs to be proof of their previous purchase in order for the refund to occur. In addition, if someone "bought" the seat at $0, if they try to get a refund then they should not get any money back but should still have their seat released back into availability.

**Assign and adjust ticket prices to different sections:** Basic flow: after the sectors for the stadium for each day has been decided, the ticket vendor accesses the system and assigns a price to each section in the stadium. The system should sync the price of the ticket with the section automatically when a ticket is purchased.
Alternatives: if the band lineup changes for any of the days, the price will need to be changed depending on who the replacement is (if there is one). The ticket manager will then need to be able to change the prices for that day in the system and then have all future purchases acknowledge the change. The change should be made note of, however, so that if customers who purchased tickets before the change choose to ask for the partial refund, it can be accounted for.

Exceptions: If the sectors have not been assigned yet, the ticket manager will not be able to price them. The stadium manager must assign sectors before the ticket manager is allowed to assign the prices.

**Create different sections in the stadium:** Basic flow: the stadium manager will go into the program and pull up a map of the stadium. The manager will divide up the stadium into different sections. These should be able to vary for each day. They should be color coded, and they should be able to be named.
Alternatives: it is possible that the sections will need to change. For example, perhaps there is not enough demand for a particular section, so the stadium manager decides it should be eliminated. The software should allow this to be possible.
Exceptions: all seats need to be in a sector, and all of the sector needs to contain seats. The program needs to not allow the stadium manager to assign this improperly.

**Eliminate any obsolete seats:** Basic flow: the stadium manager wants to eliminate seats that people shouldn't be sitting in, such as ones that are behind pillars. The stadium manager will be able to go into the program and choose which seats should not be sold or counted in the seat count. It should be as though the seat does not exist.
Alternatives: Some seats that were previously obsolete may be made available for some of the other showings. If the seats need to be made available, they should be opened up and allowed to be sold based on the commands of the stadium manager.
Exceptions: if the seat is already eliminated then it should not be eliminated again. In addition, if the seat has already been sold to a customer then the stadium manager should not be able to eliminate the seat.

**Input which bands will play:** Basic flow: the band manager should be able to input the bands that are going to play in the system. This should be available to the other users of the system, and it should be subject to change. This will need to work with the timing schedule so that it can be recognized when exactly they will play.
Alternatives: if the bands cancel or change, the band manager should have the capability to change which bands are going to play.
Exceptions: If there is no available time slot for the bands to play, then they shouldn't be mentioned as a band that will play if it is not possible.

**Input when the bands will play:** Basic flow: the band manager will be able to input the time that each band is playing on each day. This will be available to everyone that can access the system.
Alternatives: If the band times/days need to get rearranged, the band manager should have the capability to edit this in the system. The system should be able to recognize the updated schedule and provide the accurate information real-time.
Exceptions: if there is already a band assigned to a certain time slot, the band manager should not be able to add another band in there anyway. The original band needs to be removed before a different one can replace it.

**Add new user to system:** Basic flow: the user admin should be able to have the main control of the system. They should be able to access a section of the program that allows

them to add a new user to the system. They can input the information about the user and give them a password.
Alternatives: the user may just need to have their password changed, which the user admin should allow them to do. If the user has been in the system before but now is not, it should still allow them to make a new account.
Exceptions: if the user already exists in the system (username and regular name already exist in the system), then the user admin should not be able to make a second account for them.

**Assign specific permissions for which screens a user can view:** The user admin should have the capability to choose the specific screens each user can view, edit, and work with. They should do this by allowing certain permissions to each person. The user admin should have a screen that allows access to certain aspects of the program.
Alternatives: the user admin may need to remove access for people if they were at once working with certain aspects of the system but no longer need to. In addition, some screens should be accessible to multiple people but only allow certain things to be done to the program depending on the permission of the person.
Exceptions: some users should not be allowed to have certain permissions. The user admin should not be able to assign other users some of the screens the user admin has. They should not be given the capability to assign users, including themselves and others.

**Remove user from system:** Basic flow: the user admin should be able to remove a user from a system if they no longer need permission to access the program. The user admin can delete them and no longer allow them to log in.
Alternatives: the user that needs to be deleted may need to get back in the system for whatever reason. If this occurs, the user admin would need to create an account for them and allow them specific permissions.
Exceptions: the user admin cannot delete a user from the system that does not exist. It also should not be able to delete itself from the system.

# Software Qualities and Non-functional Requirements

The software has many significant qualities that it should focus on in its implementation. These are some of the most significant non-functional requirements:

- **Security:** The security of the information the software gets is important when it comes to the customer information. Since the software is working with transactional details, it needs to keep it secure so that there is no payment leakage. If customers paid with credit card and the security of the system is weak, then that could be very dangerous for the customers.
- **Flexibility:** The system needs to be extremely modifiable and flexible. Since there is a lot of potential for change, the software must be flexible to unexpected changes and ready to implement them immediately. It is likely that bands will cancel, customers will want refunds, and seating sectors will grow and shrink. This means that if the system is too rigid, it will not be able to account for these changes properly. The system must be ready to accept change and work with it properly.

- **Integrity:** The system must have data integrity. Since it is handling such precise information, it needs to be accurate and honest. The system cannot allow for slip ups such as allowing two customers to purchase the same seat or charging a customer twice. Obsolete seats need to be ignored in the system and not sold. It is one of the most important aspects of the system that it assigns tickets and seats to customers accurately and precisely.
- **Maintainability:** The software should have the capability to be updated and changed in the future. The festival may not always be the same number of days, stadiums, seats, etc. This means that the software may need to be modified to account for these changes in the future, and it should be reasonable to update it rather than having to implement a whole new system.
- **Reusability:** The ability to be reused year after year is important in particular for this festival. Since it will likely be annual, the software should be usable as the years go by. It should still be relevant even if there are some minor changes.
- **Usability:** While the users of the program should be generally capable of handling it without a strong focus on attractive graphics, they are still not assumed to be technologically advanced. The system should be simple enough to use that it should not require extensive training to understand how to use it. It does not need to be "high tech."

# Other Requirements

Some of the non-functional requirements mentioned above mean something different than the implied definition without context.

Security is more than just guarding the information. In the system, there needs to be password protection and data protection. It is not physical security; it is security of data and transactional information.

Flexibility refers to the needs for the program not to fail even when things change. While it does include the definition of needing the ability to be ambiguous and adjustable, it also needs to work properly even when the unexpected occurs.

Integrity is generally thought of as doing the right thing regardless of circumstances. While this is certainly needed in the software, it also needs to be considered from the perspective of being accurate and not just sometimes working. The ability to avoid slip-ups is significant with this term.

Maintainability is not just about keeping the program the same and working. It is also about upgrading and updating easily as time passes. The software will certainly need to be modified, and maintaining the software will mean constant improvements. This needs to be allowed and easily doable with the program.

Usability seems obvious: it is the ability for the program to be used. However, in terms of this program, it needs to be thought of as being able to be used *easily* without extensive training. It cannot be so technical that only highly skilled people that understand the program very deeply can work with it. There should be some level of intuition in working with it.

## Assumptions / Risks

There are a few risks associated with this software. One is the potential for counterfeit. There is no barcode scanning system, so counterfeiting could happen. If it does, it could be difficult to prove who has the real ticket, though they will need ID. This leads to another risk of multiple people having the same name. If two people have the same name, this could cause a problem and lead to more of a counterfeit risk. If someone has a counterfeit ticket, then unfortunately someone else may lose their seat.

Another risk associated with the program is two vendors selling the same seat at the same time. The software should prevent this from happening, but it is still a possibility. There needs to be a sync among the vendors so that this problem can be avoided. It can be a major issue if multiple people purchased the same seat.

The assumptions for this software are that there will be a stadium manager, a ticket manager, and a band manager that control the various aspects of it. If there is no stadium manager, then pricing the tickets would not be doable because there would be no sections to assign prices to. Without a band manager, there would be no festival, so the band manager is necessary. The ticketing manager is essential because without them there would be no ticket prices, so none could be sold.

It can generally be assumed that the users of the system will be adept enough to the festival and their jobs around it that they do not face a large amount of complications trying to understand the system. While they don't have to be technologically advanced, they should still understand what the system should do and how they are supposed to work with it.

## Priorities / Implementation Phases

While most of the features are must haves, some are more high priority than others. High priority must haves would include:

- Assigning ticket prices to certain sections
- Creating sections in the stadium
- Implementing the band schedule/information
- Assigning screen permissions for each user
- Storing ticket information in an excel document

The features that are more "should haves" but not as important are:

- Color coding/naming of sections
- Performing partial refunds
- Allowing all users of the system to see the band line-up

The "nice to have" features that are of lowest priority are:

- setting ticket price to $0/giving deep discounts
- system being compatible in all browsers

## Future Directions and Expected Changes

There are some changes that will be expected to be necessary for the software at some point. One such change is the potential for tickets to be purchased online. At this time, tickets can only be purchased in person from a vendor. However, at some point in the future it would not be unreasonable to consider that the tickets may need the capability to be purchased online. In fact, the system may need the ability to work more

with the website in general as far as automatically updating the band schedule, ticket prices, and more (which currently does not occur).

In addition, the number of stadiums may change at some point. While there is currently just one stadium for the five days, there could be more stadiums, which could lead to some seating complications. It is important that the software can be modified to handle this potential change.