BU Theatre:Costume Database

Butler University EPICS: Spring 2016

Team Members & Roles:

Heath Barkdull:
Team Lead, Database Coding

Amy Hendricks:
Web App Design

Client: Teka England/BU Costume Shop

Professor/EPICS Administrator: Dr. Panos Linos

Table of Contents:

Abstract:

This report details the work conducted during the fall semester of 2015 for the EPICS project working with the Butler Theater Costume department on the development of an application that enables them to track and maintain inventory. This project was continued in Spring of 2016 Included are the various things we researched and information we collected that enabled us to complete the client's requirements. The team structure and project management are outlined as well as recommendations for future work that could be implemented by the next EPICS team to take on this ambitious project.

Project Summary:

Project Objective/Overview:

Client Description:

Butler University Theater department is an incubator for the next generation of theatre artists. Renowned as one of the best programs in the country, Butler Theatre strives to push the boundaries of art as we create innovative productions and instill in our graduates an entrepreneurial spirit that will help them to succeed in an ever-changing market. With a strong liberal arts foundation, we foster the development of well-rounded, critical-thinking artists who can plot their own successful path in theatre and in life. A critical portion of the Theatre department is the costume shop. Under the umbrella of the costume shop; props, clothes, shoes, and other items are created, modified, maintained and provided to the overall department for use in productions. As a result of the support role that the costume shop gives to the theatre department a lot of items are collected and stored for future use. These items are numerous and at times difficult to keep track off.

As consequence to this tireless pursuit, the department has accrued a tremendous amount of costumes and props over the years which they are finding more difficult to manage as the years go on. The university has just recently granted the department a large space beneath the soon-to-be-constructed parking garage to store their costumes and props, and the department would like to acquire an efficient cataloging system to coincide with the massive transfer.

The Butler University Theater Costume Shop department is looking to develop an organizational app for costume storage. The hope is to attach barcodes to our costumes and large accessories and associate that barcode with a page that allows an item history to be tracked via pictures and search phrases/categories in terms of available rentals, current location, and last time of use.

That is where we come in. We at EPICS have been tasked with creating an application that will facilitate the qualitative cataloguing of the Theatre Department's stock alongside a few added bells and whistles such as rental and administrative systems. With the guidance of facilitator, Dr. Panos Linos, we aim to provide an elegant solution.

Contact Person: Teka England (tengland@butler.edu)

Requested Features & Implementation Plan:

Overview of Requested Features:

Database Features:

- Catalogues items based on physical attributes, storage location, and other identifying features. Items are uniquely identified through a barcode ID and sequential database ID.
- Related attributes are divided into tables in order to cut the fat and increase the ease of navigation.
- Database has a built in auto-backup feature that will periodically create a snapshot of the database that can be reverted to in case of an emergency.
- Modification and searching of the database is done through the PHP web application.

Web Application:

- Filter/Search costume pieces by one or more of the following characteristics:
    - **Characteristics** (color, measurements, material, ect...)
    - **Era** (20's, 30's, medieval, fantasy ect...)
    - **Clothing Type** (hat, dress, shoe, ect...)
    - **Location in Storage** (tentatively denoted by section and row)
    - **Previous Productions** (Nutcracker: 1980, A Midsummer's Night's Dream: 2000, ect...)
    - **Check-in Status** (to filter what may currently be in stock)
- Search results may be displayed as a text list or in an image grid displaying the primary image of each item. Results may be printed or sent to PDF
- Item information screen contains all relevant information about the item in question as well as pictures to identify the item. Linked items and rental history will populate as they are filled.
- As an admin, one has additional actions available to take:
    - One can add, remove, and edit items in the database through the front end. This includes the manufacturing of "outfits" or individual items that link together.
    - One can manage rentals to both Butler students and other organizations using the barcode system.
    - After reading in the item barcode and inputting the renter information, one can print out a rental sheet for records.
    - One can revert to a previous database snapshot should it be necessary.

In-Depth Rundown of Implementation Plan:

Barcode Reading

- *Explanation*: Program needs to be able to read in barcode values via a camera to associate the barcode value with the item it is paired with. This item's value will be put into a database and stored with its associated data. (ie. Picture, Category, Tags, and In/Out Status)
- *Implementation*: Barcodes are essentially long strings of characters and numbers in a graphical format. Reading in a barcode would be used to create a unique item ID that could be used as half of the primary key to uniquely identify every item in the database. The other half of the key would be a local Database ID.

User Interface

- *Explanation*: This undertaking requires some sort of GUI through which the user can perform all of the necessary actions of the application through the front end. The UI needs to be able to run on Mac OS devices.
- *Implementation*: Instead of creating a GUI in the form of a stand-alone program or application, we will be using PHP to link our SQL database to a web application with a GUI that can be used from both Mac and Windows device. This would serve as the workaround to our inability to code Mac applications while still having the program be accessible from the department's Mac devices.

Tree Organization by Item Type

- *Explanation*: We need a system in order to logically sort through the items in the department based on qualities intrinsic to their being.
- *Implementation*: This is an easily remedied though the features intrinsic to a database. The SQL language allows for the calling of certain items based on a combination of their qualities. With the plls from the database that PHP can make, this can be implemented via coding of the front end.

Tagging System within Organization

- *Explanation*: We would like to be add multiple items under the same tag for an item. Things such as previous shows, related item ID's, and things of that nature. Within the web app, we would like to view these multiple items
- *Implementation*: Either through seeding tables to host these multiple items or just listing multiple values within a single database cell, the tagging system is implementable within the rules of SQL. If we want related items to populate on the other end that would have to be written into the backend of code, but that could very well be a later implementation.

Support for Multiple Pictures [ie. Different Views of a Pictures]

- *Explanation*: In both GUI and backend, we would like to have an option for view multiple pictures of the same item, in order to accommodate items that, say, have multiple layers or details on the front and back. Due to Butler's manly server space and the non-astronomical size of the database, I don't foresee there being too much issue in the way of space, but obviously, some system of compression will need to be present.

- *Implementation*: The database will have calls to compressed image files corresponding with its item and PHP, being the SQL-friendly language that it is will make a call to the images in order to display them as needed.

Adding and Removing Items with Admin Permissions

- *Explanation*: As the BU Theatre Department's stock will inevitably grow over time, we need to allow for the staff to add to their database as needed. Unique item ID values, pictures, and attributes need to be both creatable and editable once created, and items must be able to be deleted at an admin's discretion.
- *Implementation*: On backend, we can create login credentials for a higher tiered user. We simply write a script that would populate the database with a new item and then sequentially ask for the user to input the attribute values. This higher user would also be able to drop an item identified by its Unique ID value through PHP hooks to SQL.

Checking Items In and Out

- *Explanation*: Allow clients of the customer to check items in and out of the store. The customer's client information will be recorded in the program by the customer who will use that to track who has what and when it should be returned.
- *Implementation*: This is probably the last step in base level implementation for the project. There will be item attributes for who checked out the item and when that check out occurred. If it is a Butler student who checked out an item, both the name and ID number will be recorded. This information will be run through a program that creates a check-out sheet with all relevant information about the item being checked out and the person who is checking it out. This sheet is then saved as a PDF ready to print.

Automatic Periodic Server Backups

- *Explanation*: In order to prepare for the worst, we would like the database to snapshot on occasion in order to facilitate backups. This is just a precautionary measure.
- *Implementation*: MySQL has a built in feature that backs up the database on specified intervals. We need only specify the interval.

Project Approach:

Preamble:

As no member of our team had prior experience in database design prior to this class, much of our time was spent acquiring the skills necessary to undergo this project and figuring out the best medium in which to complete this task. While this did reduce the amount of time we could spend developing the final product, as students, the journey towards finding the correct path to take was beyond valuable to our individual growth, having learned multiple computer languages and figured out our way around many applications. Furthermore, the pitfalls that we encountered and took note of in our efforts allowed us to more swiftly and knowledgeably move forward in our final path. We include this section to help illustrate the less visible efforts made during the course of this project.

Assumptions and Constraints:

We started this program with 2 members on the team. While this is a normal amount for any other project, it seemed small for the task we had at hand. One member had knowledge of a few programming languages, while the other needed to be brought up to speed on the programming languages that would be used during the course of the project. In the previous semester, there were 4 members who needed to be brought up to speed on 3 different languages, this time one member needed to be brought up to speed on one language and it streamlined the entire process.

Building off of the previous groups work, we cut a lot of development time by mapping out our program and executing our plan of action. We also trimmed time by settling on one programming language and avoiding flipping back and forth.

Timeline:

- **2/1-3/1:** Project planned to be completed using Visual Basic for the front-end development and database calls while having our database be managed through Microsoft Access. Since Amy is unfamiliar with Visual Basic, she will be brought up to speed during this time.
- **3/1-4/1:** Progress was slow during this month. Both members are in the pep band and spent the first 2 weeks of March Travelling to support the Basketball team. However, the team did regroup at the end of the month and decided to narrow their focus on the app and completed several important milestones, including the coding of the search, delete, and edit functions. A lot of progress on learning databases was made.
- **4/1-5/1:** At this point, our entire team has acquired the skills necessary to implement the plans developed throughout the semester**.** While we have been working on individual components of the project prior to this period, it is at this point that we all have the required knowledge to begin linking these components together as well as continue development of the application components.

Design

Database Design Preamble:

As it is difficult to accurately display the ins and out of an entire database system in a single report, we have opted to include the hierarchy used to design our database. In order to properly understand the layout of our database system from the hierarchy, please keep the following in mind:

1. Our database is broken up into many smaller tables containing information about qualities that may or may not be applicable to a certain item. This was done to reduce the null space created by a larger tabling system. As such, an item, uniquely identified by a "Database_ID" (a sequential value that locally identifies the item) and a "Barcode_ID" (the string created when reading in an item's barcode), may have information stored in multiple tables depending on whether the qualities identified in each table are applicable to the item in question.
2. The hierarchy uses a line to separate the individual tables, and the table title, which denotes what item types these qualities are applicable to, are **emboldened** and <u>underlined</u>.
3. Each table contains columns to denote both "Database_ID" and "Barcode_ID" in addition to columns for the qualities listed within. This is, again, done to link the qualities of an item by means of its primary key.
4. The location identification is only a placeholder. We will devise a applicable system when the plans for the storage space are finalized.

ID	Barcode	Gender	Era	Color	Waist	Hips	Pattern	Style	Length	Notes	Click to Add
(New)	0										

Above: Costume Inventory Database in Entirety before making changes through the app

Upload Pants

Barcode: 123456
Gender : Male
Era : Contemporary
Color : Blue
Pattern: Distressed

Image Upload:
Choose File blue-jeans.jpg

Style: Jeans

Waist 36 Hips 36 Length 30

Notes:

Save Please be sure to have entered a barcode prior to clicking save!

Pant entry form

Pants

ID	Barcode	Gender	Era	Color	Waist	Hips	Pattern	Style	Length	Notes	Click to Add
23	123456	Male	Contemporary	Blue	36	36	Distressed	Jeans	30		
(New)	0										

Above: Entry Submitted to Database

Database Hierarchy:

General:

Gender

- Male
- Female
- Unisex

Era

- Fantasy
- Ancient (greeks/romans/celts)
- Medieval/Renaissance (1200-1500)
- Tudor/Elizabethan (1500-1600)
- Cavalier (1600-1700)
- Colonial (1700-1800)
- Regency/Empire (1800-1820)
- Civil War/ Early Victorian (1830-1860)
- High Victorian(1860-1890)
- 1900s
- 1910s
- 1920s
- 1930s
- 1940s
- 1950s
- 1960s
- 1970s
- 1980s
- Contemporary
- Non-Western (Asian, Indian, Balinese, African, South American, European)
- N/A

Color

- Red
- Pink
- Orange
- Yellow
- Green
- Blue
- Purple
- Black
- White
- Brown/Tan
- Warm

- Cool
- Neutral
- Multi

Check-In Status

- In
- Out
  - Rentee Identification

Location

- Row
  - R1
  - R2
  - ect...
- Column
  - C1
  - C2
  - ect...

Clothing:

Size

- Chest
  - 24-26”
  - 27-29”
  - 30-32”
  - 32-34”
  - 35-37”
  - 38-40”
  - 40-42”
  - 43-45”
  - 46-48”
  - 49-51”
  - 52-54”
  - 55-57”
  - 58-60”
  - 61-63”
  - 64”+
  - N/A
- Waist
  - 24-26”
  - 27-29”
  - 30-32”
  - 32-34”
  - 35-37”
  - 38-40”

- 40-42”
- 43-45”
- 46-48”
- 49-51”
- 52-54”
- 55-57”
- 58-60”
- 61-63”
- 64”+
- N/A

- Hips
  - 24-26”
  - 27-29”
  - 30-32”
  - 32-34”
  - 35-37”
  - 38-40”
  - 40-42”
  - 43-45”
  - 46-48”
  - 49-51”
  - 52-54”
  - 55-57”
  - 58-60”
  - 61-63”
  - 64”+
  - N/A

Pattern:

- Floral
- Geometric
- Plaid
- Stripes
- Dots
- Paisley
- Asymmetric
- Distressed
- N/A

Pants

- Formal
- Informal
- Athletic
- Jeans

- Occupational

Skirts

- Length:
  - Maxi
  - Knee Length
  - Mini
- Shape:
  - Mermaid
  - Ball Gown
  - A-Line
  - Wraparound
  - Bodycon
  - Bubble
  - Pleated
  - Pencil
  - Tiered
  - Yoked
  - Circle
  - Asymmetric
  - Paneled
  - Godet
  - Tulip

- Formality
  - Formal
  - Casual
  - Occupational

Shorts

- Athletic
- Everyday
- Formal

Shirts

- Woven
- Collared
- Buttondown
- Wing Tip
- Mandarin
  - Not Collared
  - Long Sleeve
  - Short Sleeve

- Occupational
- Knit
  - Short Sleeve
  - Long Sleeve
  - Polo
- Occupational
  - Military
  - Other

Jackets and Blazers

- Fashion
- Occupational

Blouses

- Sleeveless
- Short Sleeve
  - Cap sleeve
  - Normal Sleeve
  - Flutter Sleeve
- Long Sleeve
- Occupational

Sweaters

- Cardigan
  - Short sleeve
  - Long sleeve
- Pullover
  - Short sleeve
  - Long sleeve
  - Turtleneck
- Vests
  - Button front
  - Pullover

Sweatshirts

- Hooded
  - Zip front
  - Pullover
- Crewneck
  - Open front
  - Pullover

Dresses

- Length:
  - Full length
  - Tea Length
  - Knee Length
  - Mini
- Shape
  - Sheath
  - Hourglass
  - Mermaid
  - Princess
  - Shirtdress
  - A-Line
  - Smock
  - Empire
  - Wraparound
  - Sundress
  - Bodycon
  - Asymmetric
- Sleeves/Straps:
  - Halter
  - Strapless
  - One Shoulder
  - Long Sleeve
  - Short Sleeve

Suit

- Two piece
- Three piece

Jumpsuits and Tracksuits

- Athletic
- Occupational

Outerwear

- Waist length
  - Lightweight
  - Heavyweight
- Full length
  - Lightweight
  - Heavyweight
- Occupational
- Cloaks/Capes

Accessories

Men/Women/All
Color (see clothing colors)
Era: (see clothing eras)
Materials:

- Straw
- Leather
- Felt
- Fabric
- Fur
- Vinyl
- Flowers/Foliage
- Feathers
- Plastic
- Metal
- Gemstones (?)
- Webbing

Hats:

Brim Size

- Brimless
- 1-2”
- 3-4”
- 4+”

Type

- Religious Hats
- Balaclavas
- Baseball and Trucker Hats
- Berets
- Academic Hats
- Mortarboard
- Biretta
- Boaters
- Bowlers
- Bucket Hats
- Knit Hats
- Cadet Caps
- Coifs
- Derby
- Driving Caps/Newsboys
- Clippy Hats
- Fantasy Hats
- Fedoras

- Greek Fisherman Hats
- Ethnic Hats
- Homburg
- Hood
- Occupational
- Drover
- Pork Pie
- Ring Hats
- Top hat
- Tricorn/Bicorn
- Turban
- Trilby
- Toques/Beanies
- Walking Hat
- Wedding Veil
- Fez

Shoes:

Size:

- Men's 6/6.5
- Men's 7/7.5
- Men's 8/8.5
- Men's 9/9.5
- Men's 10/10.5
- Men's 11/11.5
- Men's 12/12.5
- Men's 13/13.5
- Men's 14/14.5
- Men's 15/15.5
- Women's 5/5.5
- Women's 6/6.5
- Women's 7/7.5
- Women's 8/8.5
- Women's 9/9.5
- Women's 10/10.5
- Women's 11/11.5

Heel height

- 0-1"
- 2-3"
- 4-5"
- 5"+

Style:

- ● Platform
- ● Pump
- ● Peep Toe
- ● Mule
- ● Slingback
- ● Mary Jane
- ● T-Strap
- ● Character Shoe
- ● Oxfords
- ● Monkstrap
- ● Wingtip
- ● Loafer
- ● Espadrille
- ● Bedroom Slipper
- ● Sneakers/Athletic
- ○ Boat Shoes
- ○ Slip-ons
- ○ Velcro
- ● Sandal
- ○ Athletic
- ○ Saltwater
- ○ Gladiator
- ○ Flip Flop
- ● Dance Shoe
- ○ jazz shoe
- ○ pointe shoe
- ○ ballet slipper
- ● Boot
- ○ Wellington
- ○ Cowboy
- ○ Combat
    - ○ Chukkas
    - ○ Heeled Boot

_____

Scarves and Shawls

Fiber Type:

- ● Wooly
- ● Silky
- ● Cottony
- ● Polarfleece

Size

- ● Small (head sized)

- Medium (around neck sized)
- Large (around shoulder sized)
- XL (bigger than large)

Shape

- Triangular
- Rectangular
- Square
- Oddball

Gloves

Use:

- Winter
- Utility
- Formal

Length

- Opera
- Elbow
- Wrist

Belts:

Style:

- Formal
- Casual
- Occupational
- Skinny
- Embellished
- Woven

Waist Measurement:

- <24”
- 24-26”
- 27-29”
- 30-32”
- 33-35”
- 35-37”
- 38-40”
- 41-43”
- 44-46”
- 47-49”
- 50-52”
- 53-55”
- 56-58”
- 59-62”
- 63+”

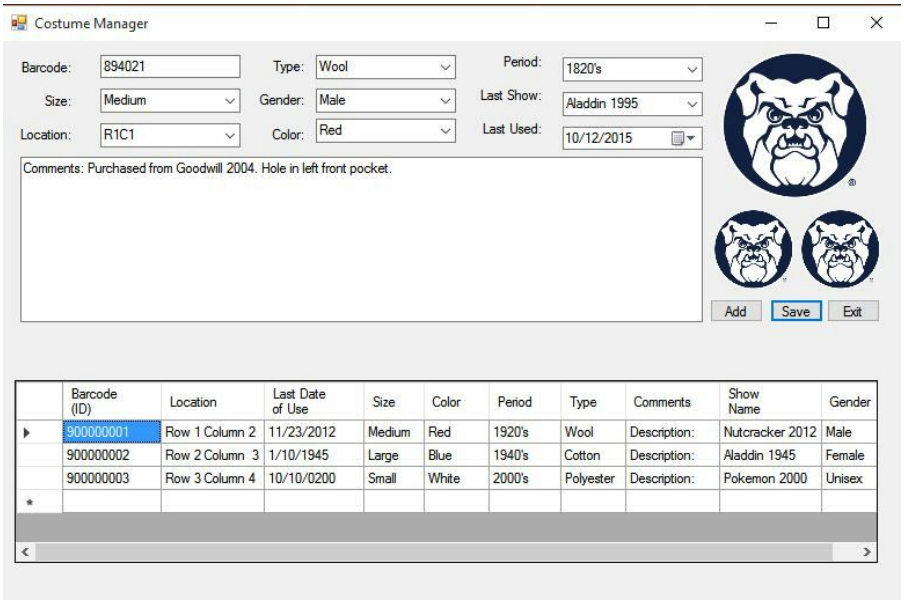
Handbags:

- Style:
- Backpack
- Tote
- Clutch
- Cross Body
- Messenger

END HIERARCHY

Web Application Design:

The web application was developed in ASP.net and Visual Basic with use of Visual Studio 2015. We have finished the basic, barebones app and began adding features that were considered stretch features.



Above: Mock up of the original “Item Information” screen.

Since the first semester focused on the development of a backend, this semester focused on the actual creation of a web app to handle the database backend.

Relevant Code Snippets:

The following code snippet is a sample of the program that commits changes to the database.

```
Protected Sub buttonSave_Click(sender As Object, e As EventArgs) Handles buttonSave.Click
    Dim i As String
    Dim j As String
    Dim k As String
    j = TextBox1.Text
    j = LTrim(j) 'Trim Functions take off spaces in the filename to avoid issues
    j = RTrim(j)
    i = j + ".jpg" 'This merges everything together to make the next command cleaner
    uploadPhoto.SaveAs(Server.MapPath("~/photos/") & i) 'This handles the photo saving.
    'server.MapPath tells the program you want to access the information within the programs
    'directory, and the following ~/photos/ tells it where to find it within the directory
    'Doing it this way only allows .jpg to be saved.

    Dim conn1 As New OleDbConnection("Provider = Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\Heath\Documents\Visual Studio
2015\WebSites\WebSite4\Databases\theater.accdb")
    conn1.Open()
    If (Label2.Text = "Pants") Then
      k = "INSERT INTO Pants(Barcode, Gender, Era, Color, Waist, Hips, Pattern, Style, Length, Notes)
VALUES ('" & Me.TextBox1.Text & "','" & Me.DropDownList1.Text & "','" & Me.DropDownList2.Text &
"','" & Me.DropDownList3.Text & "','" & Me.TextBox2.Text & "','" & Me.TextBox3.Text & "','" &
Me.DropDownList4.Text & "','" & Me.DropDownList5.Text & "','" & Me.TextBox4.Text & "','" &
Me.TextBox5.Text & "')"

    End If
    If (Label2.Text = "Shirts") Then
      k = "INSERT INTO Shirts(Barcode, Gender, Era, Color, Pattern, Style, Substyle, Other, Notes)
VALUES ('" & Me.TextBox1.Text & "','" & Me.DropDownList1.Text & "','" & Me.DropDownList2.Text &
"','" & Me.DropDownList3.Text & "','" & Me.DropDownList4.Text & "','" & Me.DropDownList6.Text &
"','" & Me.DropDownList7.Text & "','" & Me.DropDownList8.Text & "','" & Me.TextBox5.Text & "')"
    End If

    Dim cmd1 As New OleDbCommand(k, conn1)
    Dim da1 As New OleDbDataAdapter(cmd1)
      Dim dt As New DataTable
    da1.Fill(dt)
    conn1.Close()
    MsgBox("RECORD ADDED")
```

This code snippet handles the loading of items from the database.

```
Protected Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
    If DropDownList1.Text = "Pants" Then
       Me.GridView1.DataSourceID = "SqlDataSource2"
    End If
    If DropDownList1.Text = "Shirts" Then
       Me.GridView1.DataSourceID = "SqlDataSource3"
    End If
    i = TextBox1.Text
    Image1.ImageUrl = "~/photos/" + i + ".jpg"


  End Sub
```

Future Work:

Things Left Undone:

While we did spend a good chunk of time settling on the framework of the project, doing so has made the project path exceedingly clear going forward. What follows is a list of features that yet need to be completed.

- Development of Stretch Functions
- Implementation of the Filter search
- Implementation of multi-image view
- Creation of tiered administrative privileges
- Continue to reach out to Nathan Partenheimer to find the database a permanent home
- Connect the front-end to the restore database backup function
- Streamline the addition portion of the database editing functions

While more tasks may pop up as the projects nears completion. These are the only issues that remain in the scope of the current objective.

The Hands of Tomorrow:

Things we have learned this semester include database programming in MYSQL and Microsoft Access , familiarity with Visual Basic, ASP.Net. We got exposed to a lot of different languages and methods of thinking. In the end were able to implement the basic requirements for the client and include a few stretch goals.
In the future Amy and Heath would both like to continue working on this project whether it is in EPICS or on our own time.

References

Contact Information:

Development Team
Heath Barkdull: hbarkdul@butler.edu
Amy Hendricks: achendri@butler.edu

EPICS Program Administrator
Dr. Panos Linos: linos@butler.edu

Theatre Department Client
Teka England: tengland@butler.edu

Relevant Websites:

https://www.codecademy.com
- Primary website for learning necessary programming languages

https://www.mysql.com
- Product site for MySQL

https://www.jetbrains.com/phpstorm
- Product site for PHPStorm

https://www.apachefriends.org/index.html
- Product site for Xampp

https://www.butler.edu/theatre
- Webpage for Butler Theatre

http://epics.butler.edu
- Webpage for Butler EPICS program

Appendices

Relevant Code:

The application prototype can be found on the artifacts page.

Powerpoint Presentation:

The powerpoint can be found on the artifacts page.