Tracking System for Theatre Student Production Experience

Project undertaken for:
Patsy Lavender - Virginia Tech School of Performing Arts

Keegan Weiler, Zachary Sullivan, Austin Sedita

5/2/2018
CS 4624 Multimedia, Hypertext, and Information Access
Instructor: Edward A. Fox
Virginia Tech, Blacksburg VA 24061

TABLE OF CONTENTS

Table of Figures

Table of Tables

I. Executive Summary

Our client, the Virginia Tech Theatre Department, is required by their accrediting body (National Association of Schools of Theatre), to maintain records of their students' experience in various productions. They asked us to modernize their previously existing system, which was impractical and had fallen into disuse. We decided that the optimal solution would be a web-based interface to collect and store records entered by the students. Our main priority was that the system require minimal upkeep, while still having the flexibility and scalability to meet the growing needs of the School of Performing Arts.

To this end, we decided upon designing a front-end webpage and a back-end database, both with an accessible and intuitive interface. Our design should enable the department's students to easily self-report their data, while the built-in functionality incentivizes them to do so.

The project was completed in two main phases: designing the front facing webpage, and building the database on the back end. Designing the web form consisted primarily of communicating with the client to understand and analyze their needs. We collected information on the data to be recorded, what the client thought an optimal design would be, and any requirements they had for implementation of the system's front-end. We analyzed example records that former students had submitted, and assessed what aspects had caused the previous system to fail. From this data, we created several wireframes, and sample question sets which were sent to the client for feedback.

Once we had a sufficient understanding of the system parameters, we created a MySQL database, and built a backend in PHP. Our design priority for the backend was ensuring that modification and maintenance does not require any technical knowledge, while not compromising on security and stability.

Our project enables the Virginia Tech Theatre Department to easily meet their national accreditation requirements, while maintaining sole management and ownership of their data. We are proud to provide a modern, scalable system that will continue to meet any new challenges the department might face as it grows.

II. Introduction

A. Objective

Our objective is to create an efficient and effective system for tracking student participation in productions for the School of Performing Arts. We needed to ensure that modification and maintenance would not require any technical knowledge, and that the department maintained sole management and ownership of their data. This is comprised of two elements: the front-end webpage and the back-end database. The webpage will be login-based. Students will be able to submit new records of their activities, as well as view and edit their own previously submitted records. Administrators will be able to view all of the records, with various tools to sort and search them. They will also have the ability to modify or extend the functionality of the tracking system. As the project will be maintained by staff with little, if any, coding experience, we will make our system as easy to understand as possible, and include extensive documentation.

B. Client Background

Our client is Patsy Lavender, who requested this project for Virginia Tech's School of Performing Arts. She is the Director of Undergraduate Advising for all Theatre and Cinema majors. In addition, she specializes in Community Engagement, Marketing, and Arts Management.

C. Project Background

As part of the Virginia Tech Theatre Department's national accreditation by the National Association of Schools of Theatre (NAST), they are required to keep track of the productions each student is involved in as well as their roles. This accreditation process occurs every 10 years, so the data needs to have a long lifespan. The Theatre Department has been using a paper system since it was first accredited, but wanted a better way to keep track of these records. They believe that a web-based system will enable students to more consistently record their production data, as well as provide more permanent storage.

D. Statement of Functionality/Requirements

At a minimum, our client requires essentially the same functionality as the paper system. See Figure 1 in the Design Document for a copy of the original tracking form. The client stated before starting the project that they “are open to any configuration that would be accessible to individual students and faculty/staff.” In addition, we wanted to take the opportunity to expand the functionality of the system to make it more useful to both students and staff. To this end we are implementing the following functionalities:

- A webpage with a login
- Separate functions for student and administrators
- Students: submit a form about their production record
- Students: view and edit their own records
- Students: download a copy of their records
- Administrators: can view all student production records
- Administrators: sort and search utilities
- Administrators: edit the form/types of records
- Backend: website and database will be available 100% of the time
- Backend: backups of all data
- Backend: input sanitation and other security measures

E. Statement of Scope

The scope of this project is simple: we need to create a system that provides the functionality above. The School of Performing Arts already has a running website, so we simply need our service to be available from their website. The Theatre Department will be responsible for maintaining the system and making changes to it as necessary. We will provide utilities and documentation supporting making these changes. However, actually making these changes is out of scope. In addition, we will train staff members on the system’s use, and walk them through any process if they desire.

F. Extensibility

One major aspect of our project is making it extensible. Our client wanted us to make a system that the Theatre Department could modify without altering the underlying code. One of their future goals is to add more majors (such as music) and roles to the tracking system. This document provides detailed documentation on how to make these, and other changes. The only changes that staff members will need to make involve adding new information to the database, which will then be displayed dynamically on the webpage. To learn more about extensibility, see the User Manual and Future Work sections of this report.

G. Report Structure

This report has seven main sections: Requirements, Design, Testing, User Manual, Developer Manuel, Lessons Learned, and Future Work. The User Manual is the most important to the client and the Lessons Learned is the most important in this project to us - we learned a lot. Future Work describes what the Theatre Department can and will do with the system in terms of changes and everyday use. The Appendices at the end of this report deal with our progress as we created the system. They include Team Logistics, Milestones, and Responsibilities. The Requirements and Design Document were created at the beginning of the project. The Manuals and Future Work were created at the conclusion of the project, and the rest was created and updated throughout the duration of the project.

III. Requirements

There are a variety of requirements for this project, both those asked for by the client, and those we have self-imposed as good industry practices. The following is a list of those requirements, as well as a brief description of why the requirement exists.

A. NAST Accreditation

The Theatre Department at Virginia Tech, in order to meet the National Association of Schools of Theatre Accreditation requirements, must keep track of student participation in plays, shows, and other performances. In the past, the Theatre Department did this manually, on sheets of paper like the one shown in Figure 1 in the appendix. This system has become obsolete and infrequently used by the Theatre Department, prompting them to seek a new system. Our task is to create a digital replacement of this old system to allow the Theatre Department to maintain its NAST accreditation.

B. Simplicity

In our meetings with the client, both parties agreed that the product we delivered had to be simple and easy to use, for both the students and the faculty of the Theatre Department. This means that we could not just give our client a product that looks good to the user; it has to be polished inside and out so that members of the Theatre Department faculty with limited-to-no knowledge of programming or web-development would still be able to use this system.

C. Extensibility

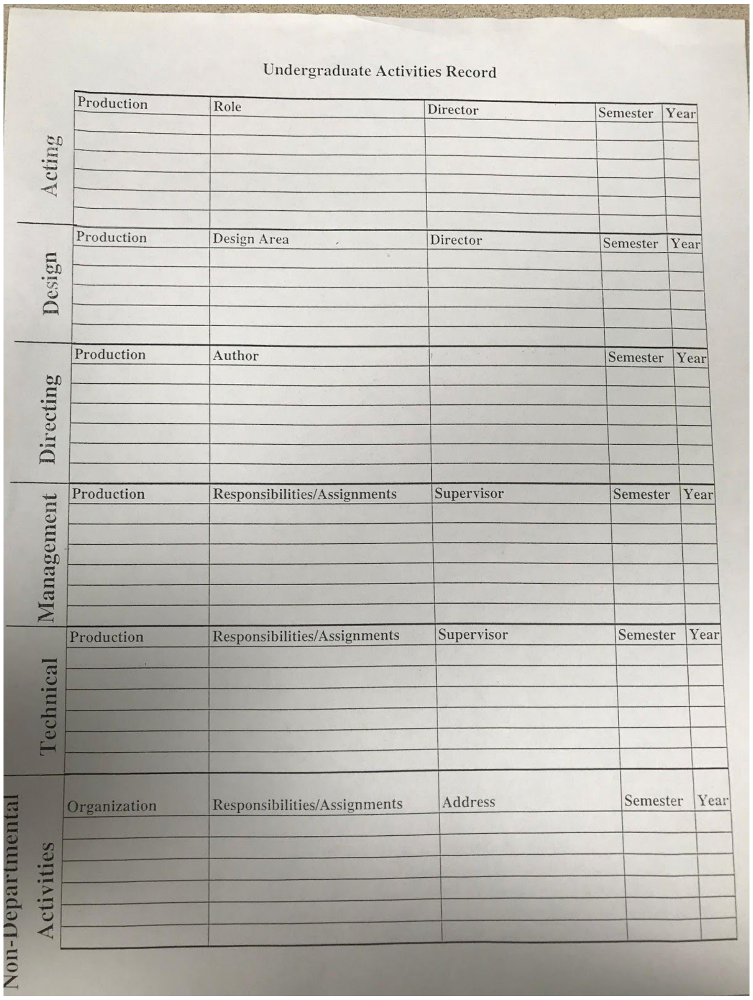
The Theatre Department isn’t the only department at Virginia Tech, or even in the School of Performing Arts, that has similar accreditation requirements. In particular, the Cinema and Music departments have similar requirements. As such, one of the goals of this project is for it to be easily extended to encompass Cinema and Music department requirements.

D. Detailed Documentation

When we have finished our work on this project, we will be turning it over to the Theatre Department here at Virginia Tech. We have taken it upon ourselves to ensure that we provide as complete documentation as possible, of both our code and how to use our product. This will help to ensure that our system remains usable for our client, even as their needs change and grow, as well as allow anyone who picks up this project in the future to understand and continue our work.

IV. Design Document

Figure 1 illustrates the old system that the Theatre Department used to track the different production experiences that students were involved in. This showcases the basic data that the database needs to store and also in what way the database needs to store it. One aspect of this system we decided to keep was the relatively open ended input. This was a large motivation for the design of the system.



Undergraduate Activities Record

Acting

Production	Role	Director	Semester	Year

Design

Production	Design Area	Director	Semester	Year

Directing

Production	Author		Semester	Year

Management

Production	Responsibilities/Assignments	Supervisor	Semester	Year

Technical

Production	Responsibilities/Assignments	Supervisor	Semester	Year

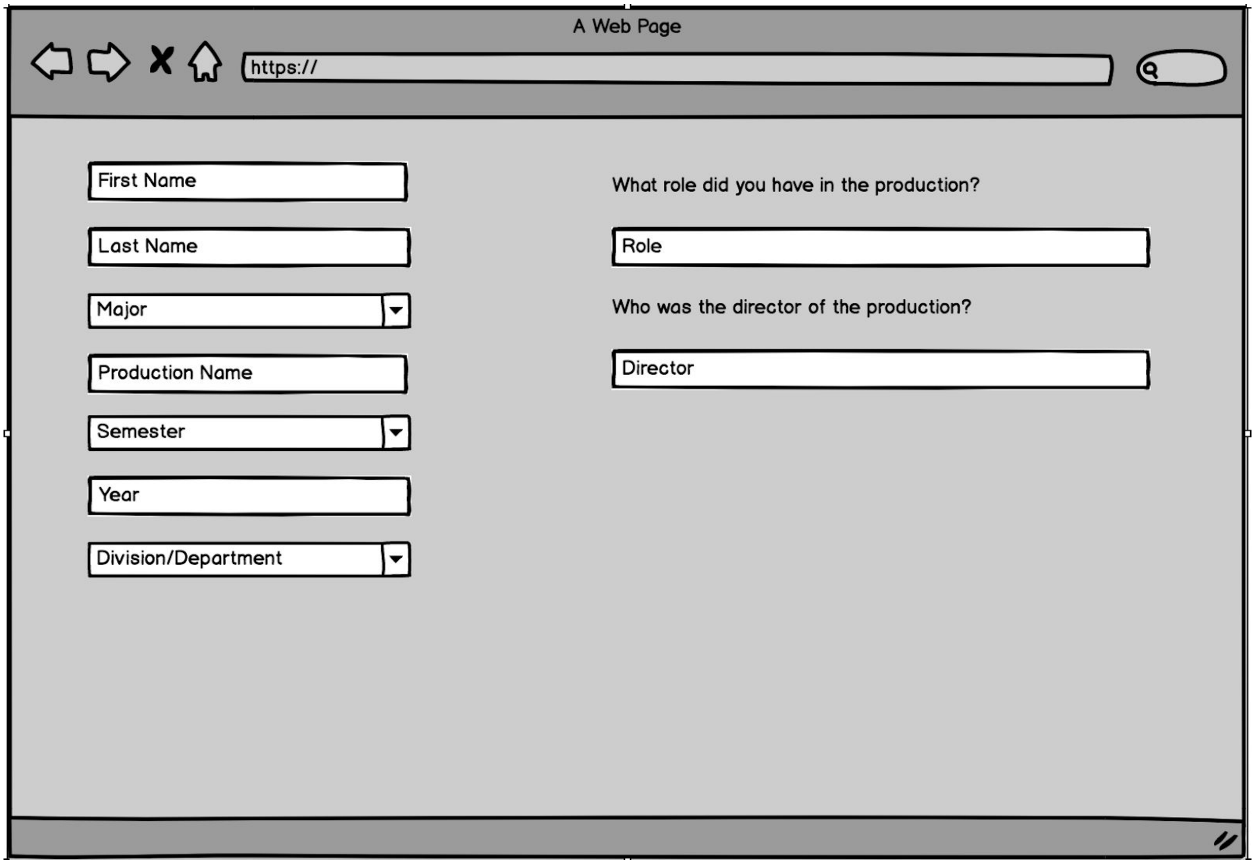
Non-Departmental Activities

Organization	Responsibilities/Assignments	Address	Semester	Year

Figure 1: Old Theatre Department Activity Tracking Form

The following wireframes and design documents are examples of what we sent to the client for feedback in the early stages of the design life cycle. To make it as simple as possible, we decided that the drop-down menus, and the corresponding questions displayed, should be the most easily extensible part of the system.

A. Sample Web Form



What role did you have in the production?

Who was the director of the production?

Figure 2: Mock-up web form design

Figure 2 is an example of what the web form could look like. There are two different types of input for the user. The first one is just a general text input. The user sees what they need to enter in a small text box and if they click on it, they can enter whatever input they want. For example, the user can input their first name in the first box on the left. The following information is received via this input style: First Name, Last Name, Production Name, Year, and any of the questions on the right of the webpage.

B. Drop Down Menu Options

The other type of input is a drop-down selection. The user can only enter information that is found in the drop-down menu. The following information is received via this drop-down style in this wireframe: Major, Semester, Division/Department (anything with an arrow in the box).

- **Major:** theatre, Cinema, Music (Eventually)
- **Semester:** Fall, Spring, Other
- **Division/Department:** Acting, Design, Directing, Management, Technical, Non-Departmental Activity

When the user chooses a Division/Department, the right side of the web form appears with the appropriate questions. For instance, when the user selects Acting in the drop-down menu, two questions appear on the right side of the web form: What role did you have in the production? Who was the director of the production? (These are shown in the picture above.) Below are all of the possible Divisions/Departments with the appropriate questions from this initial draft.

- **Acting**
  - What role did you have in the production?
  - Who was the director of the production?
- **Design**
  - In what design area did you work in the production?
  - Who was the director of the production?
- **Directing**
  - Who was the author of the production?
- **Management**
  - What were your responsibilities/assignments for the production?
  - Who was your supervisor?
- **Technical**
  - What were your responsibilities/assignments for the production?
  - Who was your supervisor?
- **Non-Departmental Activities**
  - What was the name of the organization?
  - What was the address?
  - What were your responsibilities/assignments for the production?

The advantage of using these drop-down menus is that you can add, edit, and delete options as needed. All of the options and questions are pulled dynamically from the database, and not hard coded into the HTML.

C. Questions for Client

1. Are Acting, Design, Directing, Management, and Technical divisions or departments? Right now, I have it as Division/Department for the drop-down menu. What should be the name of this drop-down menu?
2. Are there any questions you want us to add to a certain division/department?
3. If there is anything else you want us to add we can do it!

D. Early Design of Database

The following diagram is an early design for the database. It was not sent to the client in the design document, but it was an important structure that motivated our design of the system. We made significant changes to the database in the later phases of the project.

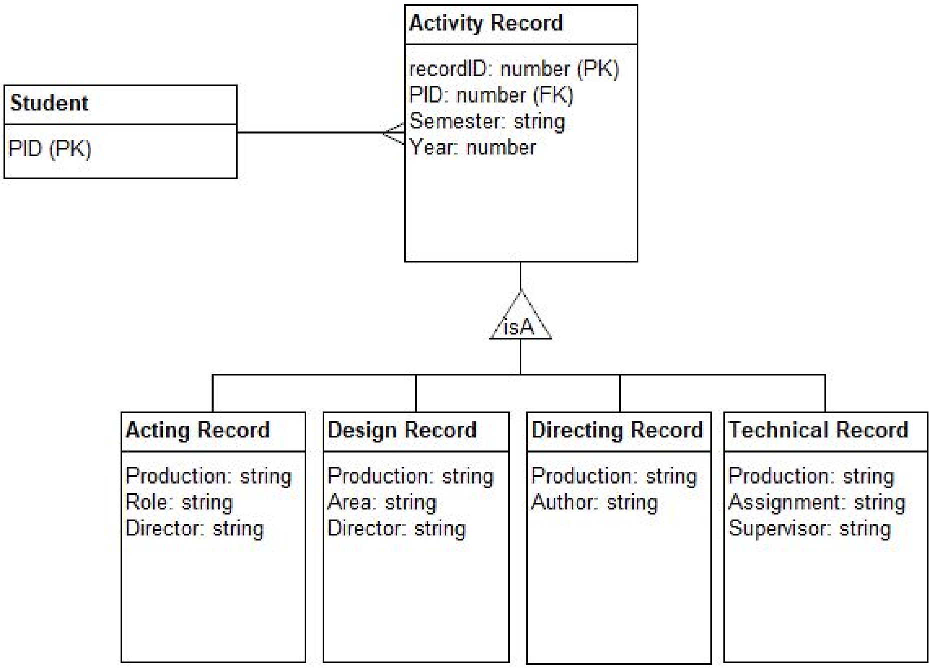


Figure 3: Early Design of Database

V. Implementation

A. Database

After our initial meeting with our client it was clear to all of us that we would need some sort of database to store all the records for our client. No one in our group had ever done any significant database work, but some group members had used MySQL in the past. This made our decision pretty simple. In order to better manage and work with the MySQL database, we used HeidiSQL. This was a program found by one of our group members after we had started work on the database. We did not consider alternatives to HeidiSQL, as it met all our needs the moment we discovered it.

The way we implemented the database ended up being slightly more complicated than we originally planned for. Because our client expressed an interest in widening the user base for this project in the future, we decided it would be helpful for her, as well as future developers, for us to separate the questions into their own table. This would make it easier to create new questions, edit existing ones, or delete unnecessary questions in the future.

B. Web-Forms

Our client needed a system that would allow users to create and store entries. When we clarified that this system would be a website that users would visit, we knew that this would require some sort of web form. We considered using Java EE, Javascript, and Python for this task, but in the end we settled on PHP. we chose PHP because it made the architecture of the project the simplest. However, this was an issue, because none of us knew or had ever used PHP before. After a fair amount of self teaching, and a few key libraries found online, we finish this project with a moderate understanding of PHP.

In order to develop with PHP, we needed an IDE. We considered an Eclipse PHP plugin and a PHP addon for NetBeans, but in the end we settled on PhpStorm (https://www.jetbrains.com/phpstorm/). We choose PhpStorm because we wanted something that was made specifically for PHP.

In addition to PHP, we used CSS to format and style our website. This is a widely used standard practice, and we had used CSS before.

The design of the PHP pages ended up being less complicated than we initially imagined. Because PHP allows us to easily change what the user sees on a page without having to navigate to a new page, we ended up only having 3 pages: a login page, that users will land on when they navigate to the site, and 1 page each for users and for our client as a site admin. These pages provide all the information and functionality needed.

C. Hosting

In order to test and display intermediate versions of our project, primarily amongst ourselves, but also for progress reports, we decided to host the project online. We considered a variety of options for hosting, and our final choice came down to the following: 1) WordPress, a popular free hosting service; 2) Amazon Web Services (AWS), Amazon's web hosting service; and 3) VTHosting, Virginia Tech's private hosting service. In the end we chose AWS, because there was a free tier that would more than suit our needs.

In order to use and test our database online, we decided to use XAMPP (https://www.apachefriends.org/index.html). One of the reasons we choose XAMPP is because it comes with a built in integration for PHP, which was very convenient.

V. Testing

A. Internal Testing

Rigorous testing was essential to ensure proper functionality and stability for both the website and the database. There was two main stages for testing each. First was simple unit tests of the individual functionalities of each system. We needed to ensure that there was not a way, either accidental or intentional, that some function could fail or be misused. To accomplish this, we essentially tried our hardest to break every aspect of the system. This includes simple things such as leaving fields blank or null, to make sure we are validating everything at every stage of its life cycle where it could fail. It also included testing for SQL injection vulnerabilities, as well as things like trying to access password protected pages without logging in.

The second part was using third party tools to analyze our website and database. We use Google Developer tools (https://developers.google.com/web/tools/chrome-devtools/), available in Google Chrome, to analyze the source code for things like performance, memory and network use, security, and adherence to recommended practices.

B. User Trial (Empirical Evaluation)

For testing our completed system, we decided to do an Empirical Evaluation. Thus, we had non-professional participants (the users) try out our application in order to receive feedback and also test for errors. The students involved in this testing were given real PIDs that were given to this project by the Theatre Department. The users were asked to do the following tasks:

- View your current production records
- Add a new production
- Delete a production

Each of the three users did not have any difficulty in performing these tasks. In fact, the feedback was overwhelmingly positive. No errors were found in this phase of testing - which is good news because it means our internal testing worked! The following is the feedback we received from our Empirical Evaluation:

- The design looks very modern and feels like a Virginia Tech application
- All of the instructions were easy to follow even if you are not a theatre major
- The table for viewing the records is easy to read - the export to PDF function is a good idea!
- If I had to make a complaint, it would be that the panels should have an animation associated with them instead of just appearing

The following is a list of observations that we made during the user testing:

- All of the users were able to view the records without any trouble. They also understand how to navigate and customize the table that showed the records.
- One user was surprised to see the different questions appear when they chose a type of activity from the drop-down menu. He tried changing the option just to see if the questions would change too.
- A user had trouble going back from the view records page. He looked for a back arrow, but then quickly realized he could press the “Submit a record” button.

It is important to note that all of the participants had little to no experience in Computer Science and none of them were actual Theatre majors (but they used a Theatre major PID). We tried user testing only after doing the internal testing that is described in detail above. If we had more time, we would have liked to have other Computer Science majors test our system in order to find errors that the users in the Empirical Evaluation would not be able to highlight due to their lack of knowledge in the field. Overall, the testing was a success as all of the errors that were found were fixed and the users who tested our system enjoyed their experience.

VI. User Manual

A. Login

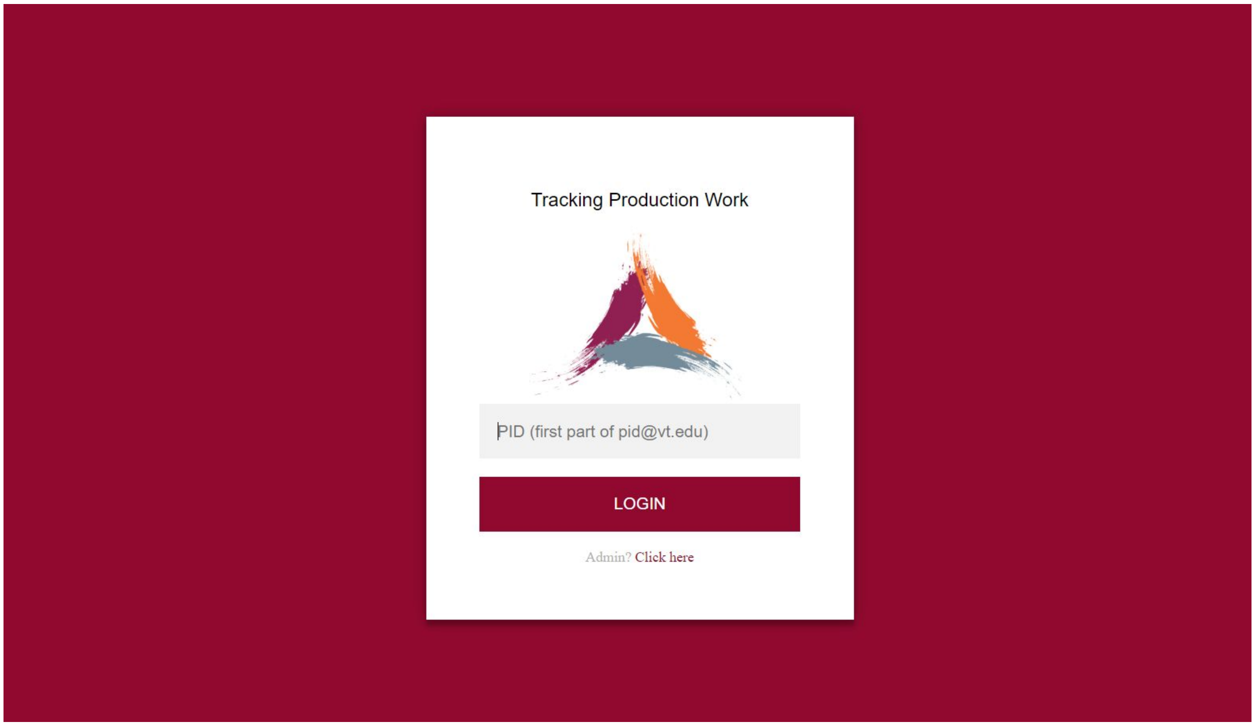
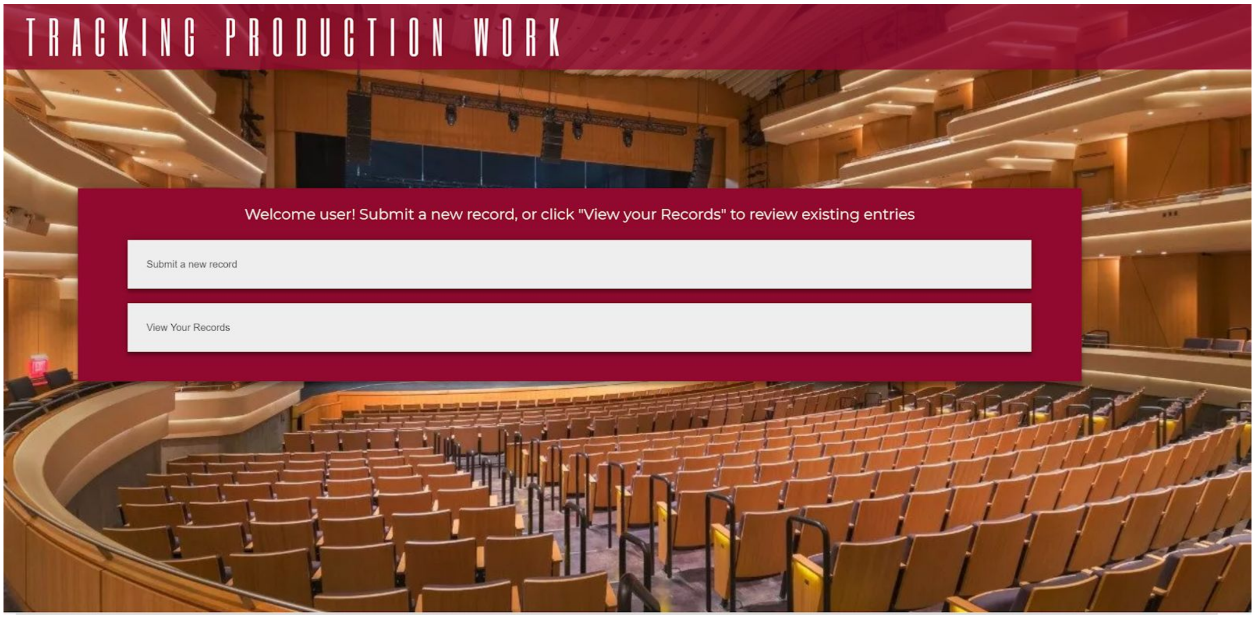


Figure 4: Theatre Production Database Login Page

The screen shown in Figure 4 is the first screen students will see. Students will enter their PID where prompted to gain access to their personal Theatre Production Database. After typing their PID the user should press the “LOGIN” button. This will take the student to their homepage (see Section B).

B. Homepage

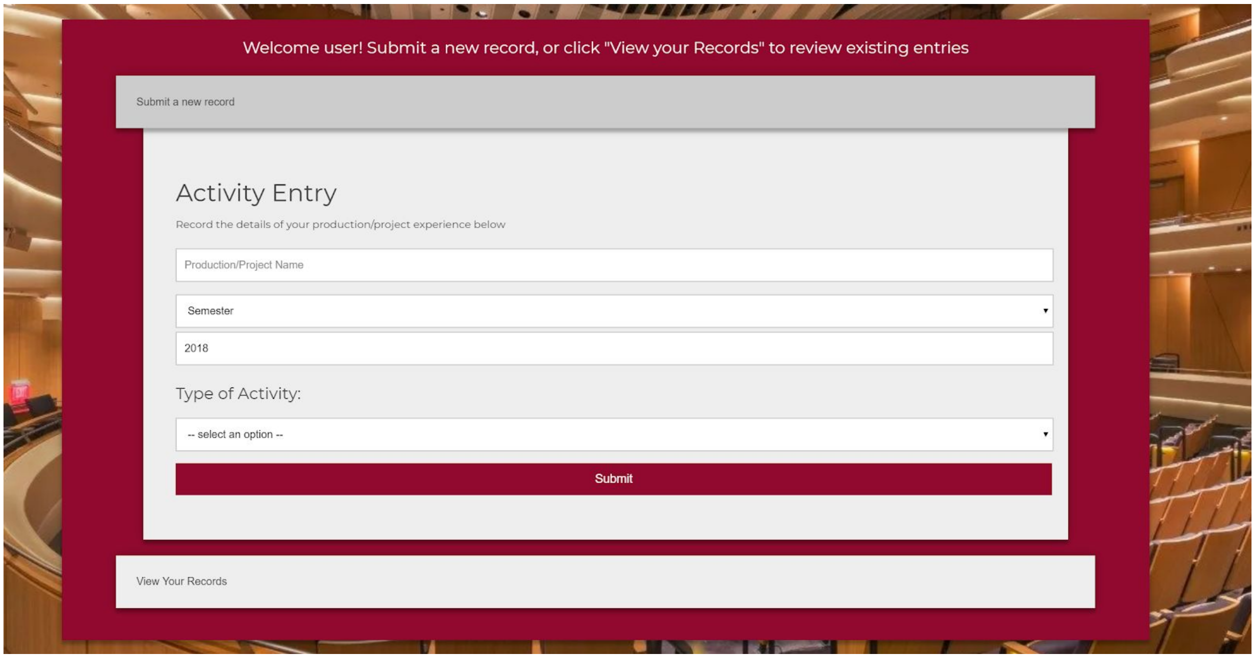


Welcome user! Submit a new record, or click "View your Records" to review existing entries

Figure 5: Theatre Production Database Main Menu

Figure 5 shows the homepage of the application. After a user logs in, this the page that they will see. It has two buttons: “Submit a new record” and “View your records”. If the user would like to add a new production experience to the database, the user should press “Submit a new record” (see Section B). If the user wants to view all current experiences in the database, the user should press “View your records” (see Section C).

C. Submit a New Record



Welcome user! Submit a new record, or click "View your Records" to review existing entries

Submit a new record

Activity Entry

Record the details of your production/project experience below

Production/Project Name

Semester ▾

2018

Type of Activity:

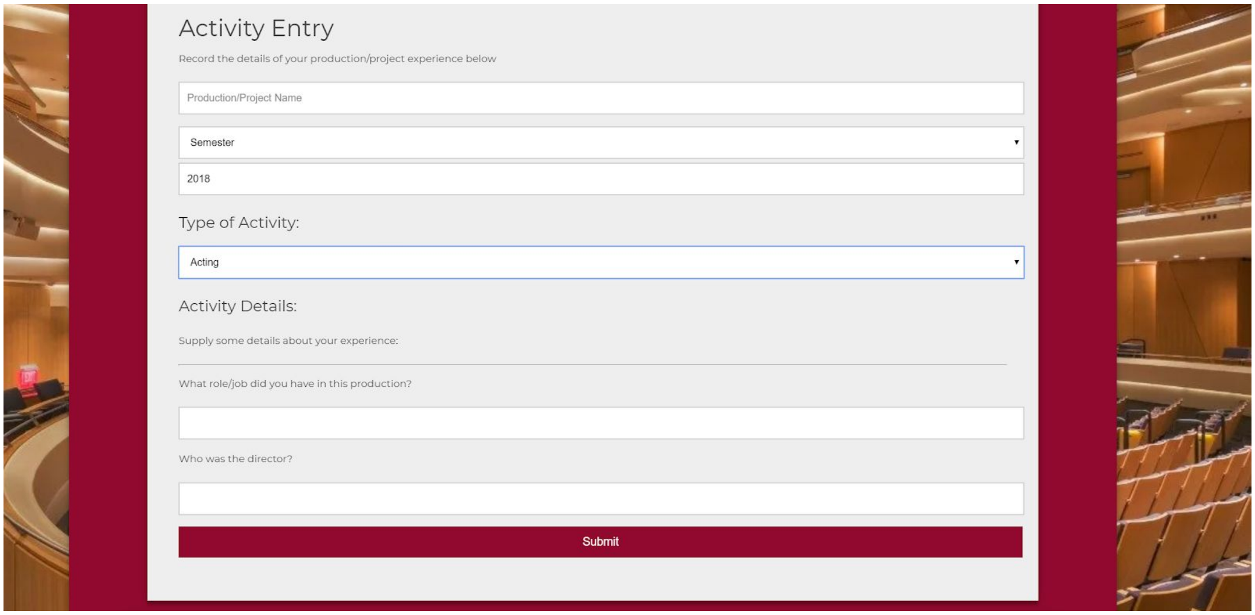
-- select an option -- ▾

Submit

View Your Records

Figure 6: Record Creation Page

If the user selects “Submit a new record” on the homepage, a new “Activity Entry” panel will appear. The “Activity Panel” asks the user for different information about their production experience which will be saved in the database once submitted. The user should put the name of the production in the first text box labeled “Production/Project Name”. Next the user should select the semester that the production was shown to the public from the dropdown menu “Semester”. If the user was involved in a production that was not during the normal school year, the user can select “Other” from the dropdown menu. The user should then put the year of the production in the text box labeled “Year”. The user must then select the type of activity from the dropdown menu under the title “Type of Activity”. If none of the current options completely describes the type of activity a user was involved in, the user can select “Other”. When the user selects an option, a new section titled “Type Specific Questions” will appear under the “Type of Activity” section. These questions are dependent on the option that the user selected in the “Type of Activity” dropdown menu. The user can select a different option from the dropdown menu at any time, and the correct questions will still appear. The user must answer these questions in order to submit the information to the database. The questions all have text boxes so the user can input the answer in their own words. An example of these questions are shown in Figure 7 (where the user has selected Actor as their Type of Activity).



Activity Entry

Record the details of your production/project experience below

Production/Project Name

Semester

2018

Type of Activity:

Acting

Activity Details:

Supply some details about your experience:

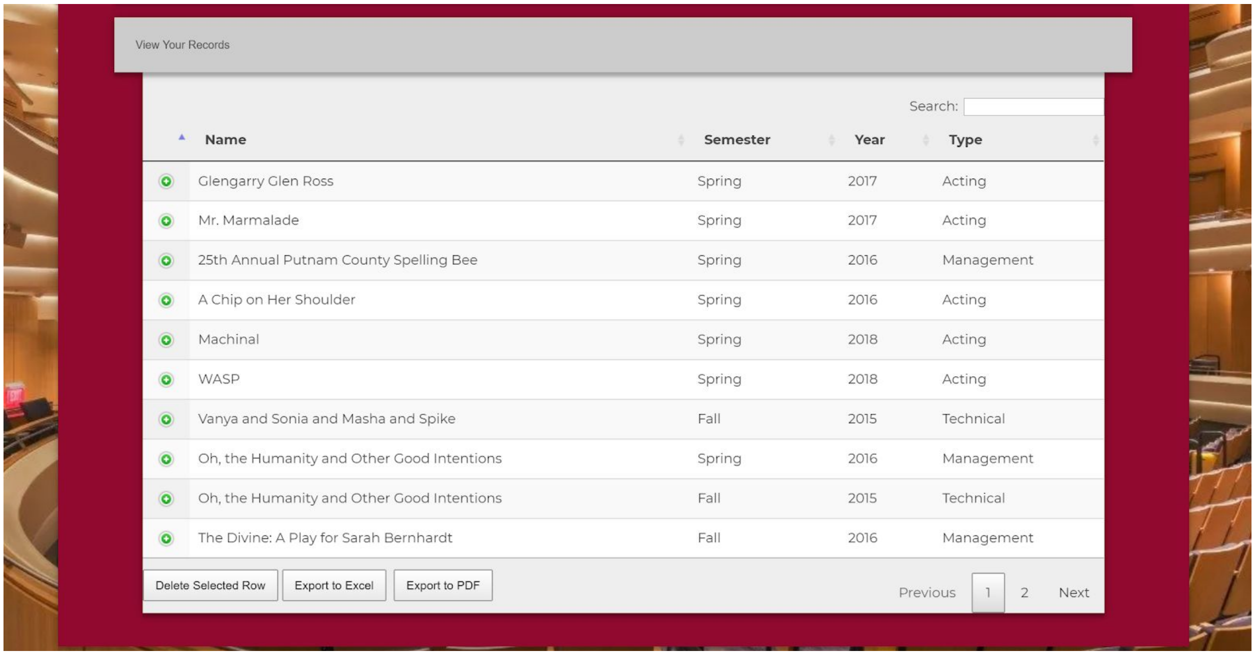
What role/job did you have in this production?

Who was the director?

Figure 7: Record Creation Page showing additional fields.

After the user has filled out the required information, they can press the maroon “Submit” button to submit their information to the database. After submitting, the user is sent back to the homepage. A confirmation message will appear on the homepage saying the information was successfully stored (the message appears in green above the “Submit a new record” button).

D. View Your Records



	Name	Semester	Year	Type
⊕	Glengarry Glen Ross	Spring	2017	Acting
⊕	Mr. Marmalade	Spring	2017	Acting
⊕	25th Annual Putnam County Spelling Bee	Spring	2016	Management
⊕	A Chip on Her Shoulder	Spring	2016	Acting
⊕	Machinal	Spring	2018	Acting
⊕	WASP	Spring	2018	Acting
⊕	Vanya and Sonia and Masha and Spike	Fall	2015	Technical
⊕	Oh, the Humanity and Other Good Intentions	Spring	2016	Management
⊕	Oh, the Humanity and Other Good Intentions	Fall	2015	Technical
⊕	The Divine: A Play for Sarah Bernhardt	Fall	2016	Management

Figure 8: Record View Page with detail highlight

If the user selects “View your records” on the homepage, a new panel will appear that contains a table showing information about each entry in the database. The user can sort the tables by clicking on the four options in the first row of the table (Name, Semester, Year, and Type). Each option has an icon of an up/down arrows in its cell. The user can change the number of entries shown in the table by selected a number from the dropdown above and to the left of the table. If there are more entries that are not being shown in the table, the user can navigate to the bottom of the table and select which page they would like to view (either by moving to the next/previous page or by choosing a page number). If the user would like to search for a specific entry, there is a search bar above and to the right of the table. The user can use this search bar to search a specific Name, Semester, Year, or Type of entry. If the user has found an entry and wants more information about it, they can select the green button to the left of the entry (in the same table row). More information will appear under the entry. This information includes the answers to the type dependent questions that appear under the “Type of Activity” section on the “Submit a new record” page (see section B of this manual for more details). If the user does not want to see this information, they can press the red button to the left of the entry (in the same table row).

VII. Developer Manual

The project consists of two main parts, the website and the database. The website is written in HTML and PHP, styled with CSS, and with some jQuery adding functionality. It is hosted on an Apache server. The database is a MySQL database.

Documentation for the underlying technologies can be found here:
PHP: http://php.net/manual/en/
JQuery: https://api.jquery.com/
MySQL: https://dev.mysql.com/doc/

Third Party Libraries Used:
Datatable: https://datatables.net/ - jQuery library for making tables
MeekroDB: https://meekro.com/ - php library for database access

A. Technical Structure and Flow

The website consists of 3 real pages: the login page, the student page, and the admin page.   All the pages are PHP files, which generate HTML files that are displayed to the user. See Figure 9 for an illustration of how they are related. Javascript/jQuery is used to add dynamic functions to the webpages, and they are styled with CSS.

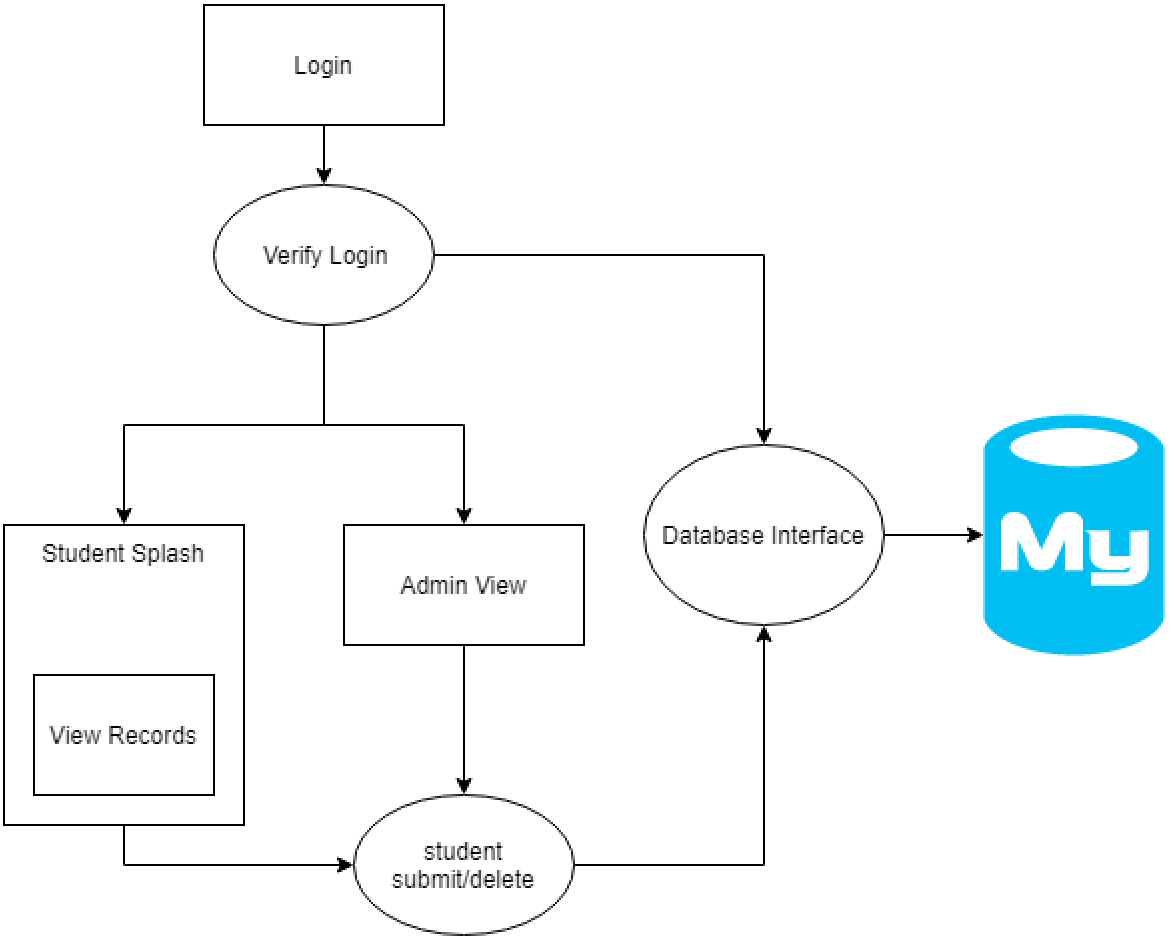


Figure 9: Site Map Diagram

The database is a MySQL relational database. It contains the submitted records as well as information about the form. It consists of 6 tables (see database diagrams in the appendix). The adminlogin table contains username and password for admin login accounts. The student table contains limited information about the students. Only their PID is stored, to avoid complications of storing personal/identifying information. The Activity_Types table includes a data row for each type of activity (Acting, Directing, etc). The Questions table includes the type specific questions, referenced to the types table. Adding either a type or a question will add it to the student form. The activity_base table includes the basic information that each activity shares. The answers table records students' answers to the type specific questions, referencing both question and base activity. See Figure 10 for a graphic illustrating these tables and their relationships.

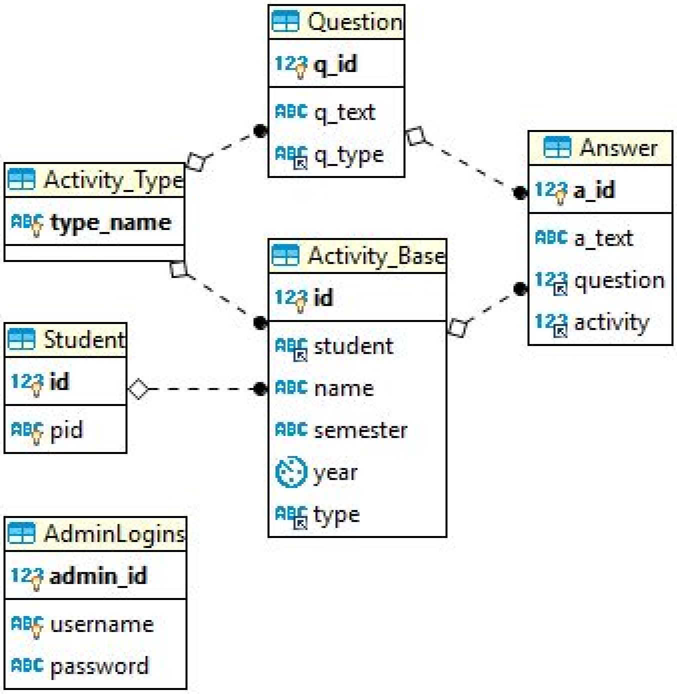


Figure 10: Database Relationship Diagram

This design was made to increase the ease of editing the form. The service owner does not need to edit the HTML in any way to modify the types/questions in the form. They can use any database GUI instead, which greatly reduces the required technical knowledge.

All the database interaction occur through the DatabaseInterface.php class's methods. This is to keep all the database operations in one location, for ease of changing database information, protecting against injection attacks, etc. DatabaseInterface.php uses meekroDB to interact with the MySQL database.

Initially, a user can only access login.php. All other pages check a PHP session variable to see if the user is logged in, and redirects to login.php if not. The student login does not have a password, only a username. The admin has a username and password, which are stored in the database. On login form submission, verify_login.php checks the information against the database, and on failure returns to login. On success, either student_splash.php or admin_view.php is loaded.

Student splash contains two panels, both initially collapsed. Clicking on them will expand them, using jQuery. The Submit Record panel contains a form for submitting a new activity record. There is a select item for the record type. The options are populated from the Activity_Type table of the database. Choosing a type will display addition questions for the record type, populated from the table Questions. Clicking submit will call student_submit.php to add the record to the database.

The second panel contains a table displaying this student's records. This is made in view_records.php and included in student_splash.php. The table is constructed from the database, using rows from the table Activity_Base that match the student ID. The Base_Activity information is added as table cells. The questions and matching answers for this activity are parsed and added to the table row as data. It will be displayed as an expandable child row in the table.

Once the HTML table is made, we use the TableData jQuery library to format the table and add functionality. This adds an expandable child row that has the supplemental questions and answers. It also add search, sort, and export functionality.

The admin_view.php had the same functionality as the student record view table, except it contains data for all students. It functions in the same way; see above for details.

B. Required Technologies & Programs

Further development can be done on any web development environments that support the LAMP stack, either hosted locally or on the web. LAMP stands for Linux, Apache, MySQL and PHP, so you will need access to all of these technologies or an equivalent.

The L in LAMP stands for Linux. This is the OS that hosts the website. Although we used Linux for hosting, Windows is also perfectly acceptable, and some of us even used Windows for development. There should be no issues with using any recent version of Linux or Windows for development or hosting. We have not done any testing or work with any MacOS, and make no promises regarding that.

The second part of LAMP is Apache, and for our development we used XAMPP. Apache is a program that allows you to host and configure web servers. To install XAMPP, go to www.apachefriends.org/download.html and select the version that corresponds to your operating system. XAMPP is extremely simple to install, and you should have no trouble with the default options.

We used MySQL as our database management system, though you may use another if you prefer. MySQL can be found here www.mysql.com/downloads/ and there are a myriad of options available, however, you only need the basic functionality.

The last part of LAMP is PHP. PHP is a scripting language that runs on the server to build the front facing webpages. Alternatives to PHP include Python or Perl, but using them would require a significant amount of revision to the project. You find a download for PHP at php.net/downloads.php where version 7 is the latest and recommended version, but if you have version 5 or later it should work.

In addition to these technologies, you need a web development IDE. This IDE is where you actually write and edit the code for the webpages. We used PhpStorm, which is part of the IntelliJ suite. PhpStorm can be downloaded from www.jetbrains.com/phpstorm/download/ and is relatively simple to install. It is perfectly acceptable to use another IDE, as it is just a frame around the code you are writing.

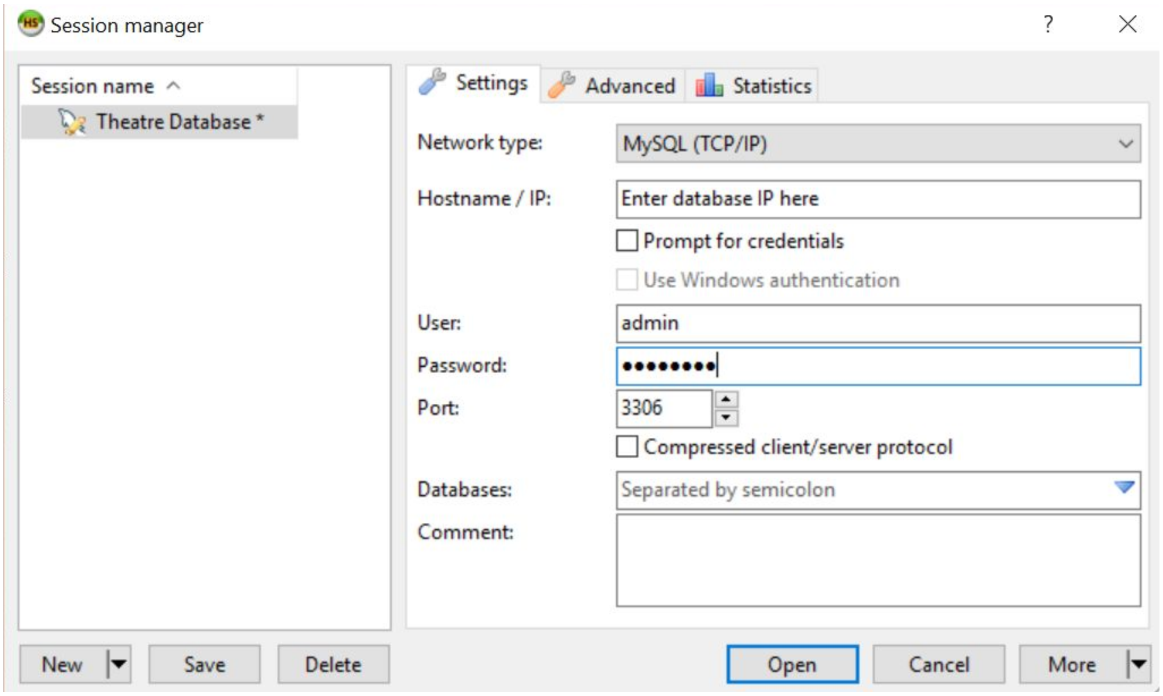
The final piece of the development suite we used was a database GUI. This is used to view and work with the MySQL database. We used HiediSQL, but if you have another preference that is fine. HiediSQL can be downloaded from www.heidisql.com/download.php and installed with all the default options.

This is everything we used in our development, and while alternatives exist for almost everything we used, we recommend these pieces of software on the basis that there should be no compatibility issues.

C. Maintenance & Further Development

After you have set up all the required technologies, you can begin development. If you want to edit a specific page, simply open the corresponding file in whatever IDE you are using. If you want to make changes to the questions or the database, use the database GUI.

When using the database GUI, you have to connect it to the database that you are running. In HeidiSQL, this can be done by clicking new and entering the IP or Hostname you are using to host the database, as seen in Figure 11. Note that this is not the IP or address of the website.



Network type: MySQL (TCP/IP)

Hostname / IP: Enter database IP here

☐ Prompt for credentials

☐ Use Windows authentication

User: admin

Password: ••••••••

Port: 3306

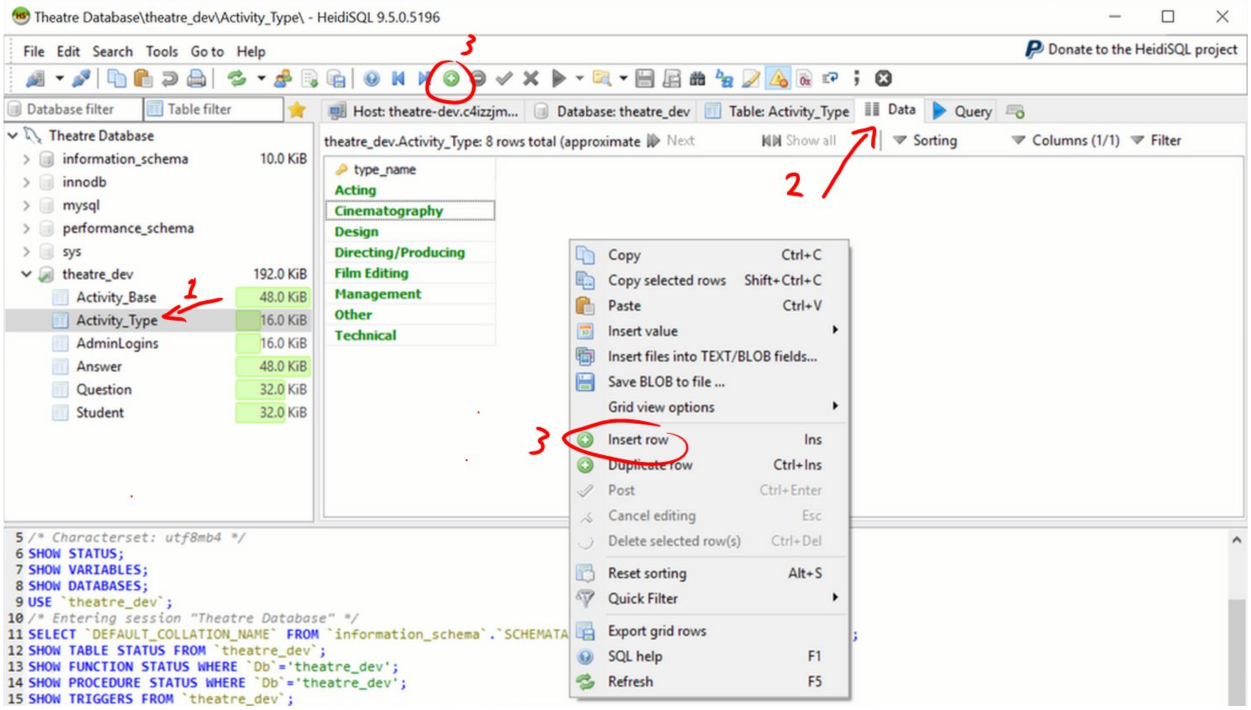
☐ Compressed client/server protocol

Databases: Separated by semicolon

Comment:

Figure 11: HeidiSQL database connection screen

Once you have connected to the database, you can begin to modify it as you see fit. To add a new type of activity, select Activity_Type on the left, and then select the Data tab at the top of the screen. Then either right click and select Insert Row, or simply click the green circle with a plus at the top of the page. Double click on the new row to edit it. See Figure 12 for a graphic of these instructions.



Theatre Database	
information_schema	10.0 KiB
innodb	
mysql	
performance_schema	
sys	
theatre_dev	192.0 KiB
Activity_Base	48.0 KiB
Activity_Type	16.0 KiB
AdminLogins	16.0 KiB
Answer	48.0 KiB
Question	32.0 KiB
Student	32.0 KiB

theatre_dev.Activity_Type: 8 rows total (approximate

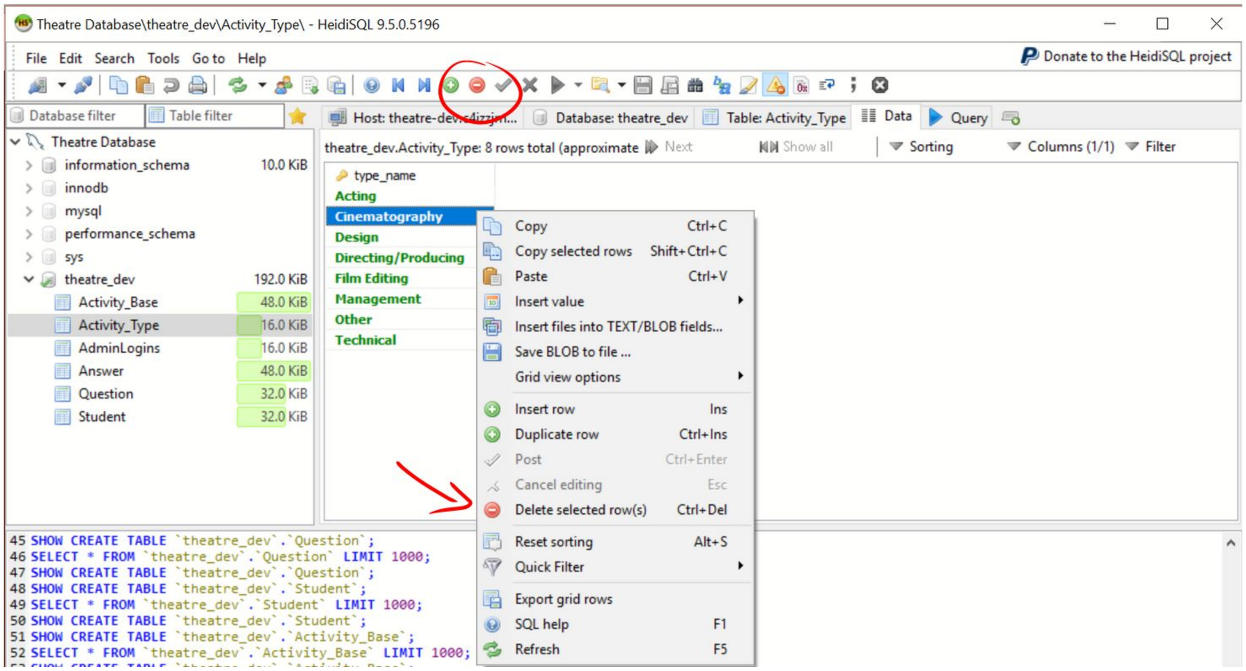
type_name
Acting
Cinematography
Design
Directing/Producing
Film Editing
Management
Other
Technical

```
/* Characterset: utf8mb4 */
SHOW STATUS;
SHOW VARIABLES;
SHOW DATABASES;
USE `theatre_dev`;
/* Entering session "Theatre Database" */
SELECT `DEFAULT_COLLATION_NAME` FROM `information_schema`.`SCHEMATA
SHOW TABLE STATUS FROM `theatre_dev`;
SHOW FUNCTION STATUS WHERE `Db`='theatre_dev';
SHOW PROCEDURE STATUS WHERE `Db`='theatre_dev';
SHOW TRIGGERS FROM `theatre_dev`;
```

Figure 12: Insert new record instructions

The process for adding a new type specific question is very similar. Navigate to the data tab of the Questions table, and add a new row. The Questions table has 3 fields, q_id, q_text, and q_type. When you make a new row, q_id will be (NULL); leave that as is. Type the question text into the q_text column, and set the q_type column to the value in the Activity_Type table.

Activity Types and Questions can be edited as well. To edit an existing record, simply click on it twice (this is not the same as double clicking); this will make the field editable. If it is a text field, like Activity type_name, you will be able to type and delete text. If it is a type field, like Question q_type, you will be given a drop down menu. However, keep in mind that editing a type of question will not change the answers. If you are changing more than just the phrasing, it is better to delete the type/question and add a new one.



type_name
Acting
Cinematography
Design
Directing/Producing
Film Editing
Management
Other
Technical

```
SHOW CREATE TABLE `theatre_dev`.`Question`;
SELECT * FROM `theatre_dev`.`Question` LIMIT 1000;
SHOW CREATE TABLE `theatre_dev`.`Question`;
SHOW CREATE TABLE `theatre_dev`.`Student`;
SELECT * FROM `theatre_dev`.`Student` LIMIT 1000;
SHOW CREATE TABLE `theatre_dev`.`Student`;
SHOW CREATE TABLE `theatre_dev`.`Activity_Base`;
SELECT * FROM `theatre_dev`.`Activity_Base` LIMIT 1000;
```

Figure 13: Deleting rows from the database

To delete a record, simply left click the record to select it. Next, either right click and select the ‘Delete selected row(s)’ option, or click the red circle with a minus at the top of the page. See Figure 13 for a graphic of this. A confirmation window will appear; click ‘OK’ to delete the selected record.

D. Inventory of files

The following is a list of major files and folders contained in this project, each including a brief description of their purpose/function.

Directory: resources - Contains the image files that are displayed on the webpage

File: calender_logo.png - login logo

File: image.jpg - background image

Directory: stylesheets - Contains the CSS style sheets used to style the webpage

File: login.css - styles the login page

File: main_page.css - styles all other pages

Directory: support - Contains the back end .php files, that are not pages a user can visit

File: DatabaseInterface.php - This PHP class contains all the functions needed to interact with the database. It also contains the database login credentials (left blank). It uses meekroDB library to simplify operations. See https://meekro.com/ for documentation.

File: meekrodb.php - See above

File: student_delete.php - This PHP file uses DatabaseInterface to delete a record from the database. It is called from both student_splash.php and admin_view.php

File: student_submit.php - This PHP file uses DatabaseInterface to insert a record into the database. It is called from student_splash.php

File: verify_login.php - This PHP file is called from login.php and used DatabaseInterface.php to either verify an admin login, or a student login. It then displays admin_view.php or student_splash.php, respectively. Failed logins will return to login.php

File: view_records.php- This file gets a student’s records and makes a table from them. It is included/displayed in student_splash.php

Base Directory:

File: login.php- Displays a login form, switching between student and admin login displays using jQuery. It calls verify_login.php to verify the login information, and is styled with login.css.

File: admin_view.php - Contains the admin view, which is a table displaying the records of all students. The data for the table is fetched from the database and build into an HTML chart. DataTables is used to style the chart and give additional functionality.

File: student_splash.php - The base page for students; it contains a panel for submitting records, and a panel for viewing records. The submit activity calls student_submit.php to a new record. The record view is included from view_records.php.

VIII. Lessons Learned

The most important lesson that we learned doing this project was the importance of sticking to milestones that are created at the beginning of the project. Several milestones were not finished on time. The initial database design and research was due on February 10 and we had barely even contacted our client at the time. We let other classes dictate our milestones instead of seeing the importance of scheduling. Because we had not started the project on time, the due dates for presentations and client meetings started to creep up on us. Our late start meant that we had to meet with the client, create a presentation, and do important research all in three days! We held a group meeting after everything was finished and planned on never missing a future deadline because we wanted to create the best possible deliverable on time. We decided to make intermediary milestones so that we could check our progress. One intermediary milestone that we created was finishing the web design on Balsamiq and sending it to the client for feedback. This allowed us to move forward with the creation of the web forms because we knew exactly what the client wanted.

Another lesson that we learned was that of presentation skills. On our first presentation we thought we could just wing it and get a good grade - but, that was not the case. According to our feedback, we appeared disorganized and uninformed on our topic. For our second presentation, we made sure to meet in-person the weekend before we had to present. We practiced what we were going to say and gave each other feedback. The main problem that we were having was that we had too many words on the PowerPoint slides - so we ended up just reading them word for word. To fix this, we tried to put more pictures and key words on the slides so that it forced us to look at the audience and engage them.

A technical skill that we all learned was that of database creation. None of us had ever worked with a database before - which is why we all took on this project. We all wanted to learn how to create an effective database that anyone can manage - even without technical experience. We successfully learned how to use MySQL to achieve this. We also had to figure out the connection between a simple web form and the database we were creating. A lot of our time was spent on researching simply because we were all excited to learn how to create a database!

The biggest problem we faced in this project was figuring out how to get the web forms and the database to talk to each other. Initially we thought we were going to use just HTML and JavaScript for the webpages, but we ran into a wall pretty fast with getting JavaScript to work. Rather than spending a large amount of time trying to climb this wall, we realized that it would be easier to just go around. It was much easier for us to use PHP to communicate with the database, even though we had to learn how to write PHP. This solution helped us learn to be flexible with the technologies we use. You aren't always told what technologies to use, so you have to make a choice. And if that choice isn't correct, or another option is better, you have to make the switch.

IX. Future Work

A. Extensibility

Our client, Dr. Lavender, mentioned that the cinema and music schools also have similar requirements for their accreditations. We decided that one of our final objectives would be to design our product in such a way that it could easily be modified for use by these other departments. All of the extensibility guidelines can be found in the developer manual. We anticipated all of the additions that the Theatre Department would make such as adding more questions, departments, and roles. At any time in the future, the Theatre Department can make changes without the fear of messing up the database.

B. Future Upkeep

After we are done with this project, the only upkeep should be the continued hosting of the webpage and database. These will be transferred to the current host of the Theatre Departments official website as part of our deliverable. In the event some changes are requested by the Theatre Department, they would be able to provide that service easily.

In addition to thorough documentation of all parts of this project, we have the added stretch goal of creating a simple interface for our clients in the Theatre Department to give them some ability to customize the site, as they wish.

X. Acknowledgements

We would like to acknowledge our client Patsy Lavender of the Virginia Tech Theatre Department, as well as Paul Steger and Jane Harrison, for being the source of this project. We would like to acknowledge our professor Dr. Edward A. Fox for connecting us to our client and providing help and feedback along the way.

XI. References

"Add Advanced Interaction Controls to Your HTML Tables the Free & Easy Way." *DataTables | Table Plug-in for JQuery*, www.datatables.net/. Accessed April 20th, 2018

"Apache Friends." Apache Friends RSS, www.apachefriends.org/index.html. Accessed February 20th, 2018

Becker, Ansgar. "What's This?" HeidiSQL - MySQL, MSSQL and PostgreSQL Made Easy, www.heidisql.com/. Accessed February 1st, 2018

"Chrome DevTools | Tools for Web Developers | Google Developers." Google, Google, developers.google.com/web/tools/chrome-devtools/. Accessed April 20th, 2018

"MeekroDB." *MeekroDB -- The Simple PHP MySQL Library*, www.meekro.com/. Accessed April 24th, 2018

"PHP MySQL Database." *PHP: MySQL Database*, w3schools.Com, www.w3schools.com/php/php_mysql_intro.asp. Accessed March 4th, 2018

"PHP 5 Form Handling." *PHP 5 Form Handling*, w3schools.com, www.w3schools.com/php/php_forms.asp. Accessed February 20th, 2018

"PHP 5 Tutorial." *PHP 5 Tutorial*, w3schools.com,

www.w3schools.com/php/default.asp.
Accessed February 15th, 2018

“PhpStorm: Lightning-Smart IDE for PHP Programming by JetBrains.”
JetBrains, www.jetbrains.com/phpstorm/.
Accessed February 1st, 2018

“Testing for SQL Injection (OTG-INPVAL-005).” OWASP,
www.owasp.org/index.php/Testing_for_SQL_Injection_
(OTG-INPVAL-005).
Accessed April 24th, 2018

Young, Denise. The Moss Arts Center's Anne and Ellen Fife Theatre in the Street and Davis Performance Hall. Digital image. Virginia Tech Magazine. Virginia Tech, n.d. Web.
https://foursquare.com/v/moss-arts-center/4cc2facaf49676b0b9af71d5?openPhotoId=528a5cd411d2e03c089315b9
Accessed February 15th, 2018

Appendix A: Team Member Roles and Responsibilities

Table 1: Member Roles

Responsibility	Primary	Secondary
Front-end Design and Implementation	Keegan Weiler	Zachary Sullivan
Report Creation	Zachary Sullivan	Austin Sedita
Presentations	Zachary Sullivan	Austin Sedita
Database Design and Implementation	Keegan Weiler	
Technical Documentation	Keegan Weiler	Austin Sedita
User Manual	Zachary Sullivan	

This table outlines the responsibilities of each team member throughout the project. These assignments were initially determined based on prior experience and interests; however as we progressed, many of the lines were blurred. The progression of the project required each team member to take on multiple roles in order to meet each deadline. The team member in the 'Primary' column either did the most work or took a leading role for that particular responsibility. The team member in the 'Secondary' column contributed some work for that responsibility or had another responsibility that overlapped with it.

Appendix B: Milestone Progress

Table 2: Milestones and Deadlines

Milestone	Date	Status	Notes
Database Design	2/10	Late, Met	Once we met with our client and discussed what was needed, this milestone was accomplished easily.
Front-End Design	2/20	Met	This milestone was accomplished easily after meeting with our client to discuss their needs.
Functioning Prototype	4/1	Late, Met	We were unable to meet this milestone on time due to a combination of unexpected difficulties and our busy schedules.
Style and Look	4/12	Late, Met	This milestone was late because we were waiting on client feedback about our design document
Deployment	4/20	Met	Once there was a working version of the project, hosting it on one of the team members machines was simple.
Documentation	5/2	Met	In addition to comprehensive user and developer manuals, this milestone includes this report paper
Delivery	5/1	In Progress	This deadline involves a final presentation to the class, submission of the final report to the instructor and VTWorks, and transfer of the project to the client

1. February 10: Database Design

The requirements for our database were that it needed to store individual events that were to be recorded by students in the Theatre Department. To this end, we structured our database as shown in Figure 2 in the Design Document. The key points of this structure are that each entry is linked to a student's account, which is just their PID. Each entry is one of several types of Activity Record, depending on which type of entry the student selects. Each Activity record is made up of several fields that contain the information the student enters about the event. The number of fields and their lengths is variable, dependent on the type of event and how much information the student decides to ender.

2. **February 20: Basic Front-End Design**

The front end for this project is a series of web forms that would allow students to Create, View, Edit, and Delete entries into the database. These CRUD forms are the standard operations for database management. An example of one of these forms is included in Figure 1 in the Design Document. These web forms interact with the database through PHP and MySQL commands.

3. **April 1: CAS Integration**

CAS is the Virginia Tech Login Service that allows all students to log into various Virginia Tech related services all with their single official VT student login. Integrating the CAS would replace the existing, basic login system that we made. The CAS login would be much more secure and simpler for the users of our product.

4. **April 10: Style and Look**

When we first designed the front-end web forms, we made them to be functional, not pretty. After we have the website working functionally we will revisit the look of the webpages. Webpages' appearances can be easily edited with CSS files, so this isn't a hard task, but it is important to making our product look finished and professional.

5. **April 20: Documentation and Deployment**

One of our major goals for this project was to be thorough in our documentation, so that our client and future people working on this project would be able to understand what we did and how to work with it. As such, we plan to make extensive User and Developer manuals to deliver to our client. Additionally, we must deliver the project itself, in a way they can use, as all our testing has been done locally. This will be done by hosting the website on a site of their choosing. It has not yet been deployed as of the time of this submission.