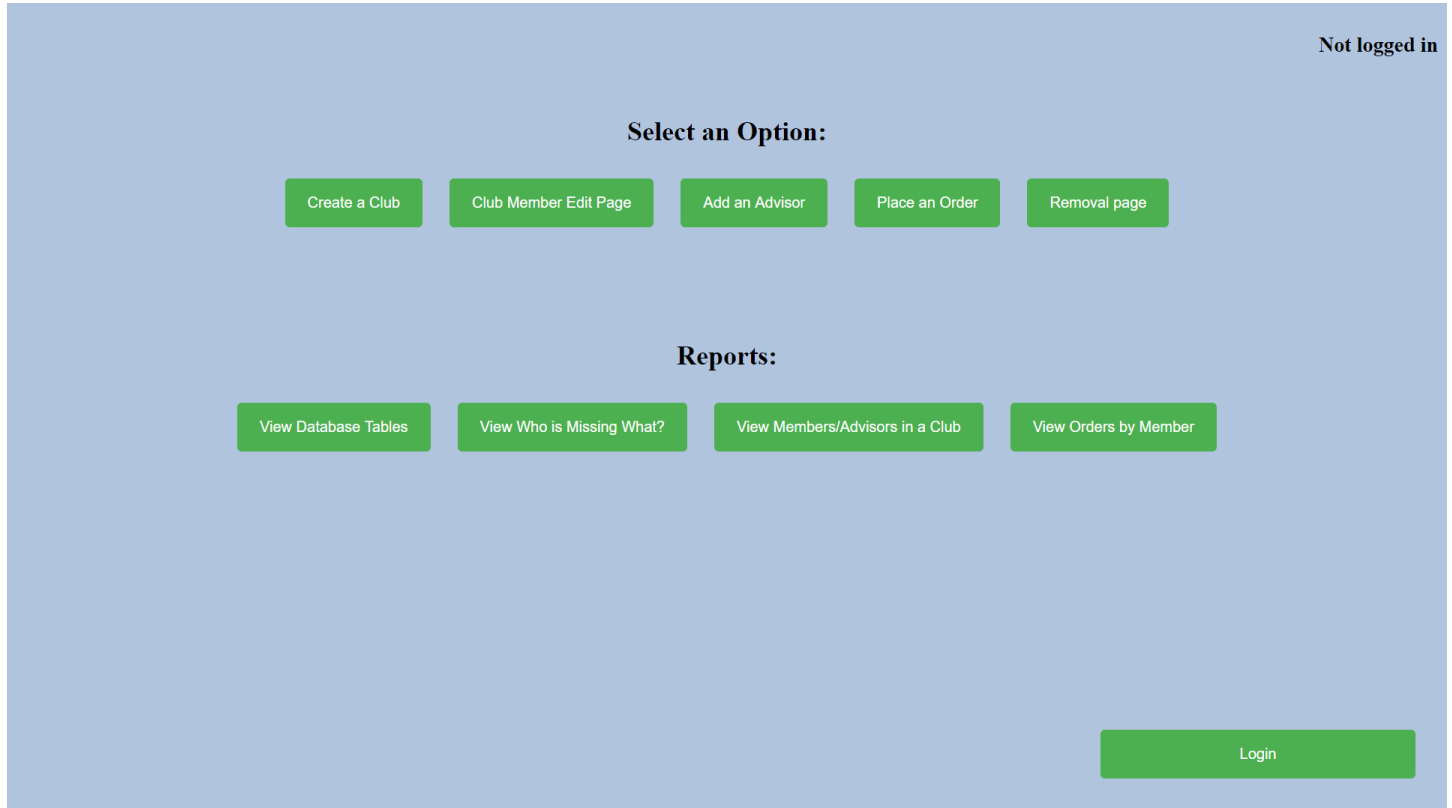


## General Layout

Using an Apache HTTP server, the functionality of the web pages which use the club database are demonstrated here. All the web page files are located in the directory labeled “htdocs,” which is the default configuration for XAMMP Apache HTTP servers. The home page appears as follows:



The home page essentially links all the other pages together and contains an option to login. The top right corner denotes the login status of the user. The section labeled “Select an Option:” links the user to form pages where they can add elements to the database, and a removal page where they can remove elements from the database. The section directly below that, labeled “reports,” uses the information stored in the database to dynamically generate reports on the web pages and provides functionality for them to be downloaded for offline use.

## Logging in

There is no registration page for the club database because, conceptually speaking, if this database were to be employed, viewing should be authorized only to certain individuals. Thus, a method must be used to add users to the database user relation. For simplicity, I used a simple PHP script (which would obviously not be viewable if it were implemented) to add users and their hashed passwords to the database. Here is an example of some users being stored in the database with hashed passwords:

id	username	password
19	fakeuser1	\$2y\$10\$hMHUPmnGuD6tzxntQDsNoOqwav.a5omPEDOw7h.RDV5DvnqzADjNy
21	red@80	\$2y\$10\$8Zz4nuCXWF9WVNbEh1Y72.HsOqNozs6tgJKbmwOHZf2jhjelYD72.

To login, click the login button. A form will pop up where the user can enter their credentials. After an attempted login, the login status will change. Here is what the login status should display on a successful login:

**Logged in as fakeuser1**

Log Out

Since no data should be viewable to unauthorized individuals, if the user is not logged in, and they proceed to another page, a message stating their restricted access will appear:

Results will appear here:

**You must be logged in to add a club!**

**Login**

Username

fakeuser1

Password


.....

Login

Close

### Forms (adding data)

The forms are relatively self-explanatory. There are several descriptive elements to guide the user in filling out the information (e.g. red asterisk for required fields, popups for inputted values that do not satisfy domain constraints etc.). Here is an example of one of the forms:



**Add a Club Member**

Club Member ID (1-5 length numeric [1-99999]): \*

Club Number (1-5 length numeric [1-99999]):

First Name: \*

Last Name: \*

Does this member need a bus pass?: \*

Yes ▼

Is this member a student?: \*

Yes ▼

Add

The results of the insertion attempt will appear at the bottom of the page. A simple JavaScript function is used to automatically scroll the user to the bottom of the page on an insertion attempt. Here is an example of the results that appear from adding a club member with the id of 10:

### Results will appear here:

Successfully added club member [10] - Landon Shenberger.


If an error occurs, a user-friendly message will be displayed. For instance, if a user attempts to add an advisor with a unique id that already exists (duplicate primary keys), this error message will appear:

### Results will appear here:

Error: an id of the value you submitted already exists; you cannot have duplicate ids.

## Removal Page

The removal page is used to remove Advisors, Members, and clubs (the rest is handled by the database via “on delete” referential integrity constraints). The user is unlikely to know every id, and it would be tedious for them to have to refer to another sheet. Thus, a HTML selection form is dynamically generated according to the retrieved data, displaying the id along with the name for each selection. For instance, when the “remove a club” button is clicked, a list of clubs is displayed on another selection form for the user to delete from:



### Select an Option:

Remove an Advisor

Remove a Member

Remove a Club

[1] - MVNU w/ Resort: Mad River  
[2] - Whitefish High School w/ Resort: Whitefish Mountain Resort  
[3] - Mount Vernon Ski Club w/ Resort: Snow Trails  
[4] - The Racers w/ Resort: Breckenridge Ski Resort  
[5] - Princeton University w/ Resort: Mountain Creek  
[6] - High valley Ski Club w/ Resort: No Resort

Remove

To remove a club/member/advisor, select an option (in this case, “[6] – High valley Ski Club” is selected), then click “Remove.” A simple string will appear, confirming the removal:

Successfully removed ski club: [6] - High valley Ski Club

## Reports:

There are several pages for generating reports. The “View Database Tables” essentially views the database in its raw format (i.e. no filtering, joins, sorting etc.). Here is an example from my local instance:

### Choose a Table:

Advisors  
Bus Passes  
**Club Members**  
Forms  
Insurance Information  
Orders  
Passes  
Clubs

Display

### Results will appear here:

Download CSV

Member ID	Club	First Name	Last Name	Requires a Bus Pass	Is a Student?
1	[1] - MVNU	Jim	Smith	No	No
2	[1] - MVNU	Carol	West	No	Yes
3	[2] - Whitefish High School	Nick	Goepper	Yes	No
4	[3] - Mount Vernon Ski Club	Terra	West	Yes	Yes
6	[1] - MVNU	Landon	Shenberger	Yes	Yes
7	[1] - MVNU	Alex	Pager	No	Yes
8	[5] - Princeton University	Tom	Lee	No	No

The “Download CSV” button uses a JavaScript to download the HTML table as a Comma Separated Values file. It automatically writes a timestamp to the file name as well as a few other descriptive attributes. A file named “club\_member-Sun Dec 22 2019 19\_42\_57 GMT-0500 (Eastern Standard Time).csv” (which is generated from the report above) is included to demonstrate the output this function produces.

The “View Who is Missing What?” page provides useful information in the case wherein a club advisor may need to determine who does not have all their stuff turned in (e.g. waiver form, emergency form, medical information etc.). The page has these options:

Who needs a bus pass but does not have one?  
Who has not submitted medical information?  
Who has not submitted an emergency form?  
Who has not submitted a waiver form?  
Who has not bought a pass?  
Display all missing elements by member.

Display

They all display a simple HTML table with simple data values just like the reports aforementioned, except for the last option, that is, “Display all missing elements by member.” This option generated a selection form using the same methods (specifically, functions) as the removal page. It displays an HTML table with images to denote whether the selected member has the component in question. For instance, the results of selecting a random member on my local instance are as follows:

**Results will appear here:**

Download PDF

**Member: 2 Carol West:**

Component	Yes/No
Bus Pass	
Medical Form	
Emergency Form	
Waiver Form	
Resort Pass	

It is probably not a good idea to export such data into a CSV file as it contains images; instead, the functionality to export the table into a PDF file is provided for this option. The function uses a couple external libraries to carry out a PDF conversion. Meta data is included in the file name just as it is with the download CSV option. A PDF file named “missing\_report\_for\_2 Carol West-Sun Dec 22 2019 20\_13\_40 GMT-0500 (Eastern Standard Time).pdf” is included to demonstrate the output of this function.

The “View Members/Advisors in a club” page is used to display the members/advisors in a specific club. It initially displays a selection box with two options: advisors or members. Selecting either option will display a dynamically generated form with all the clubs listed as options. The page should look like this:

### Choose an Option:

Display members by club  
Display advisors by club

Select

### Results will appear here:

**Note: only clubs with club members are displayed (i.e. the club is not empty).**

[1] - MVNU w/ Resort: Mad River  
[2] - Whitefish High School w/ Resort: Whitefish Mountain Resort  
[3] - Mount Vernon Ski Club w/ Resort: Snow Trails  
[5] - Princeton University w/ Resort: Mountain Creek

Select

After selecting a club, the table result will appear below. For example, with my local instance, if I wanted to “show all members in the MVNU club,” I would get this result:

[1] - MVNU:

Download CSV

Member ID	First Name	Last Name
1	Jim	Smith
2	Carol	West
6	Landon	Shenberger
7	Alex	Pager

Lastly, the “View Orders by Member” page is used to display the orders a particular member has placed. On the page, it will display a selection box of only members who have at least one order. On my local instance, the page appears with only one member (as that member is the only one with any orders):

## View Orders by Member:

Only club members with orders are included

[1] - Jim Smith

Select

After selecting the member, it displays an HTML table:

[1] - Jim Smith's order(s):

Download CSV

Member ID	Item Type	Custom Name Label	Size	Color	Quantity	Price Per Unit	Order ID
[1] - Jim Smith	T-Shirt Long Sleeve	I love databases	Large	Light Grey	2	\$35.99	10
[1] - Jim Smith	Hoodie	:D	Large	Light Grey	1	\$70.00	11
[1] - Jim Smith	T-Shirt Short Sleeve	N/A	Large	Red	1	\$15.99	12

**Total Cost of the Order(s): \$157.97**

Notice the total cost of all the orders for that member is displayed at the bottom. This is achieved by using a PostgreSQL function to calculate the total cost of a member’s orders given their id.