Abstract:

The purpose of this project was to organize data (questionnaire screens) in a manner that allowed it to be easily accessed and displayed to users. In order to demonstrate a complete understanding of this project, the proposed system has been implemented in PHP, SQL, HTML, JavaScript, and AJAX on a testing server. The source code for this report is available on Git and explained in the 'Implementation' section below.

Data Structure Overview:

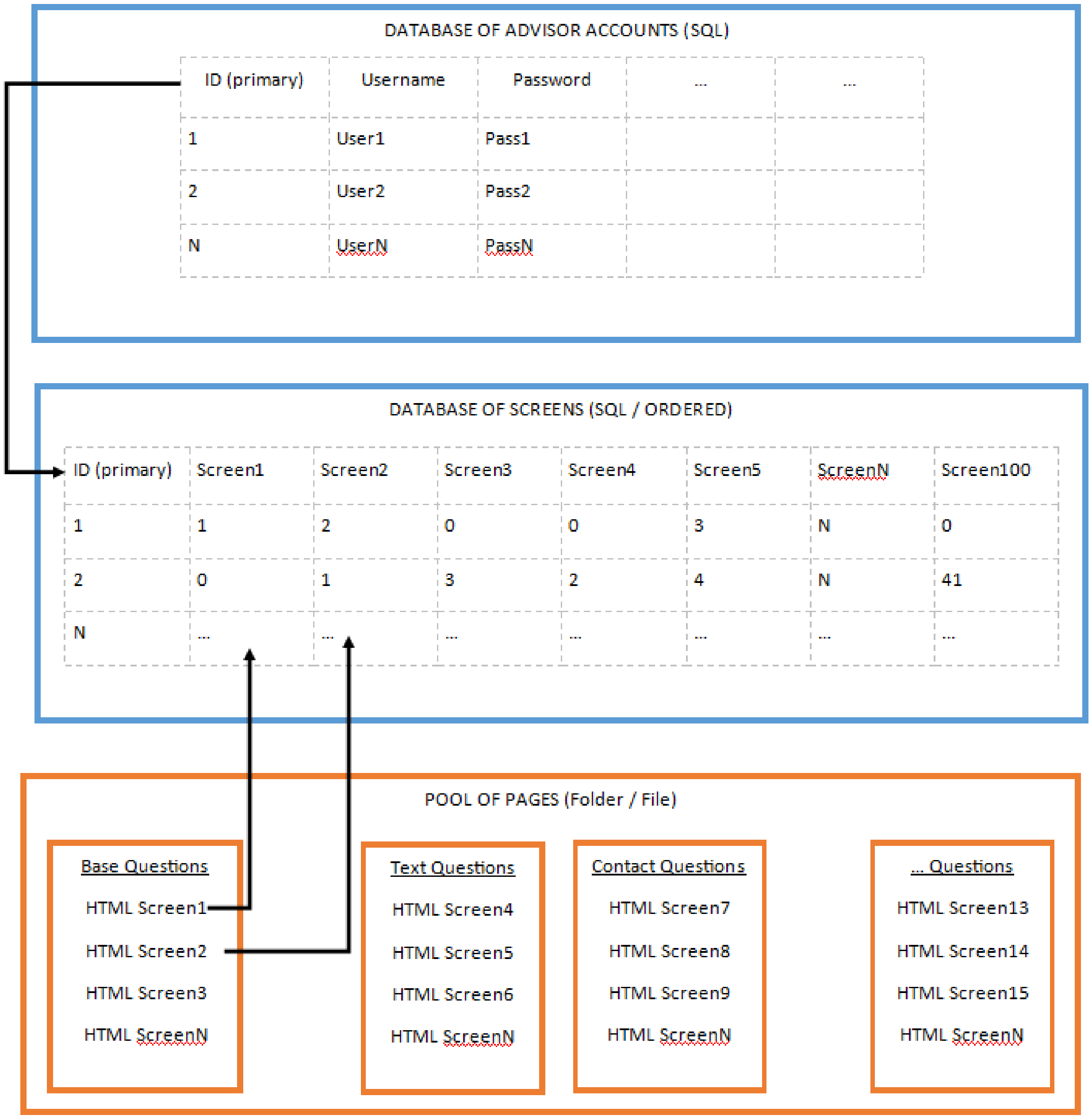
The data was divided into three main components in order to organize the questionnaire screens into a maintainable system.

(1) The first component consists of an SQL database containing the advisors accounts. Each advisor account must have a unique ID (primary key), username, and password. Of course, other columns could be added such as date of creation, profile image, etc.

(2) The second component is an SQL database containing the desired questionnaire screens to be shown and their order. Each row represents an advisor (matched by comparing unique/primary keys) and holds the order to display each questionnaire screen. In this table, a zero is used to show that a questionnaire screen should not be shown, a one means display the corresponding screen first, a two means display the screen second, etc. Each column in this table represents a questionnaire screen and the column title matches the questionnaire html file name.

(3) The third component is the actual questionnaire screen data. Each questionnaire screen is a HTML file containing one or more questions for the user to answer. Each advisor will be able to choose what screens to show (by changing values in the SQL table in component 2 above), however, all of the questionnaire screens will still need to be saved locally. This component holds each HTML questionnaire screen (separated into folders for human readability) so it can be referenced by the SQL database.

A diagram of this structure can be seen in Fig. 1 below.



ID (primary)	Username	Password	...	...
1	User1	Pass1		
2	User2	Pass2		
N	UserN	PassN		

ID (primary)	Screen1	Screen2	Screen3	Screen4	Screen5	ScreenN	Screen100
1	1	2	0	0	3	N	0
2	0	1	3	2	4	N	41
N	...	...	...	...	...	...	...

Figure 1: Diagram of structure of SQL databases and referenced questionnaire screens.

Advantages, Disadvantages, and Variations:

Advantages:

For this project, the questionnaire screens are stored as files and referenced by the SQL database, rather than storing the actual HTML code in the database. This is clearly the more efficient approach both in efficiency and clarity. The Advisor Accounts SQL database was separated from the Screens SQL database for ease of use and possible performance increases. The Screens Database is fairly large, containing a column per screen, and should be loaded as little as possible. Separating this table from the Advisor Accounts database removes the overhead of querying the advisors' database and separates the advisors' critical data (username, password, etc.) from optional/additional data.

Disadvantages:

The proposed Screens SQL table contains a column for every possible questionnaire screen. In the case where there are many (thousands) of possible screens this table would become large and possibly unwieldy. This disadvantage was disregarded because it is unlikely the Riskalyze Questionnaire would ask the users thousands of questions.

Since the Screens SQL table contains a column for every possible questionnaire screen, there is a potential for many zero values and thus wasted space. For example, suppose there where 100 possible questionnaire screens, also suppose an advisor wants to ask his clients only 10 questions to determine their risk number. In this case there would be 90 zero values for this advisor.

Implementing this system revealed additional disadvantages to the proposed approach. The Screens SQL database contains an ordered list that must be reordered every time it is changed. Even if the order of two items are changed, all the items after them must be re indexed. This only occurs when the advisor updates their questionnaire screens, so it is not major issue.

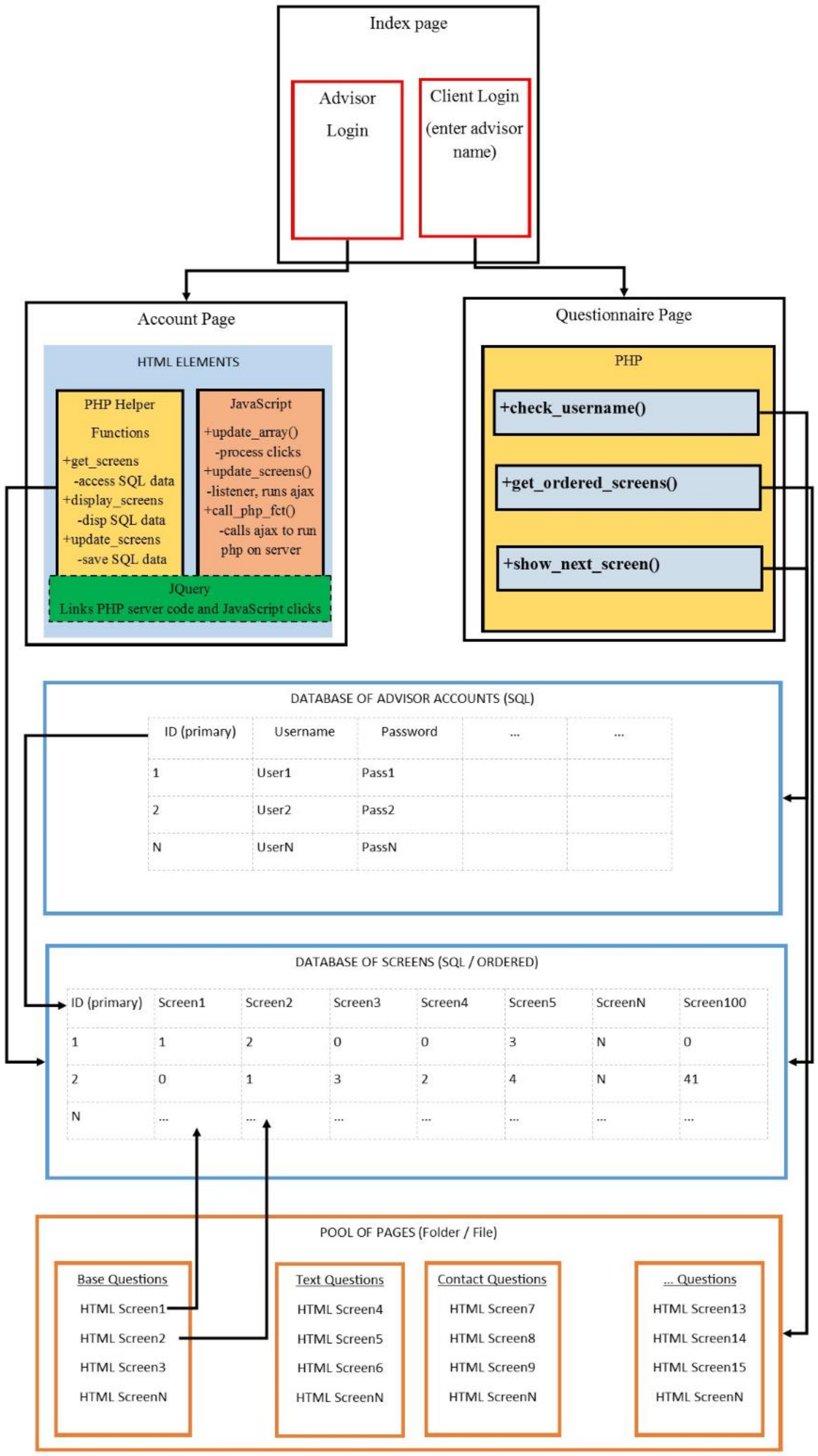
Variations:

There are many different ways this database could be structured. The two SQL databases could be combined, the HTML pages could be created dynamically rather than stored statically, swapping the rows/cols, etc. The proposed system could be made more efficient, but doing so would likely lose the simplicity of the proposal. For the scope of this project, the proposed solution achieves a adequate balance between efficiency and simplify.

Implementation:

In order to demonstrate a complete understanding of the proposed structure, the entire system was implemented on a personal server. The main focus of this project is on the back end development, however, some front end development was required in order to develop a functioning system. The implemented system was written with PHP for the server side code, used SQL databases for holding the backend information, HTML/JavaScript for the front end development, and AJAX to link the front end to the backend when needed. A diagram of the entire system is shown in Fig. 2 below. Figures 3, 4, and 5 show the front end interface for reference.

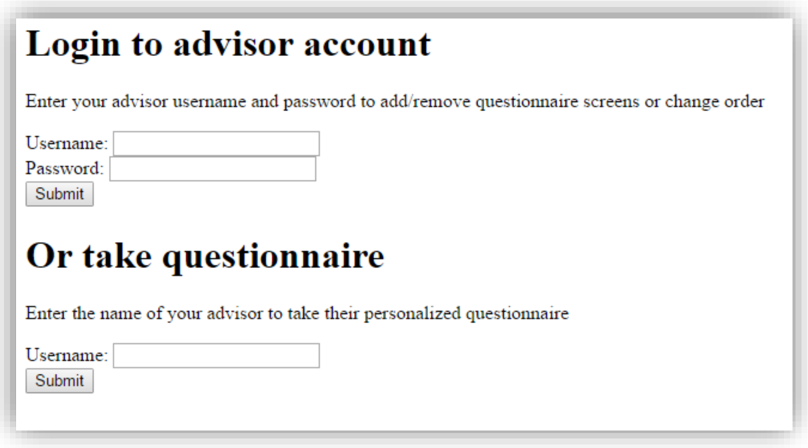
You can run the demo by navigating to: https://www.fastfishdevelopment.com/riskalyze



ID (primary)	Username	Password	...	...
1	User1	Pass1		
2	User2	Pass2		
N	UserN	PassN		

ID (primary)	Screen1	Screen2	Screen3	Screen4	Screen5	ScreenN	Screen100
1	1	2	0	0	3	N	0
2	0	1	3	2	4	N	41
N	...	...	...	...	...	...	...

Figure 2: Diagram of both front end and backend of system.


Login to advisor account

Enter your advisor username and password to add/remove questionnaire screens or change order

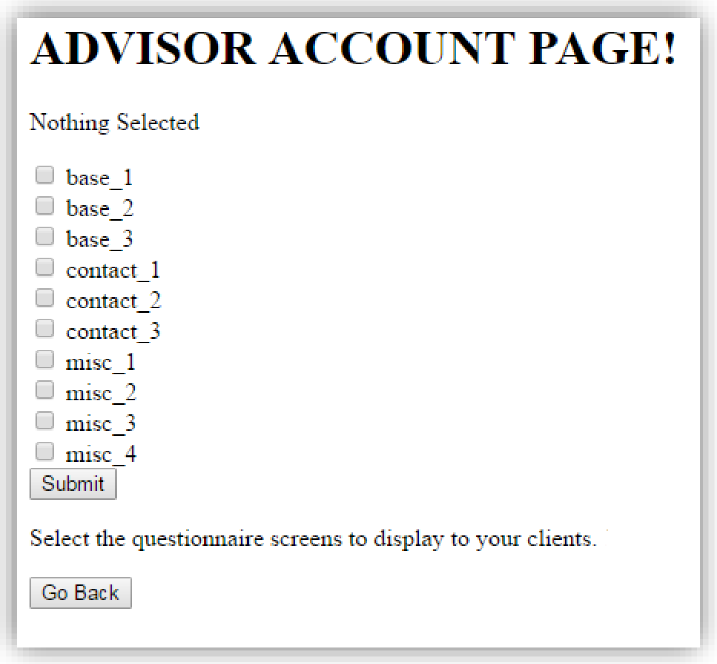
Username:
Password:
Submit

Or take questionnaire

Enter the name of your advisor to take their personalized questionnaire

Username:
Submit

Figure 3: Index Page – Allows user to login as advisor or take questionnaire as client.


ADVISOR ACCOUNT PAGE!

Nothing Selected

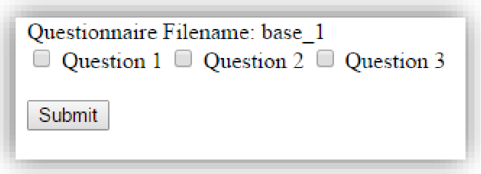
- [ ] base_1
- [ ] base_2
- [ ] base_3
- [ ] contact_1
- [ ] contact_2
- [ ] contact_3
- [ ] misc_1
- [ ] misc_2
- [ ] misc_3
- [ ] misc_4

Submit

Select the questionnaire screens to display to your clients.

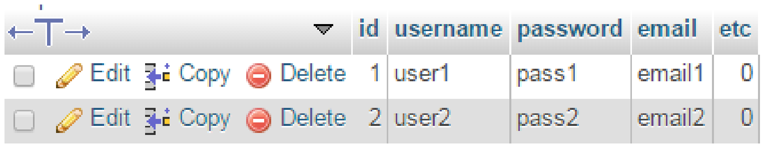
Go Back

Figure 4: Advisor Account Page – Allows an advisor to change order and number of screens.


Questionnaire Filename: base_1
☐ Question 1 ☐ Question 2 ☐ Question 3

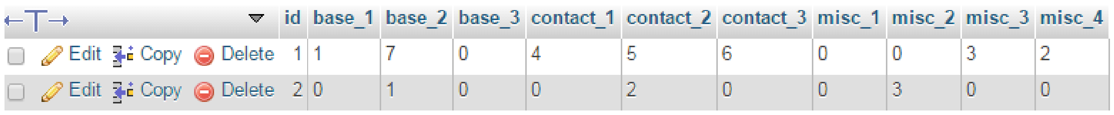
Submit

Figure 5: Questionnaire Page – Iterates over each desired questionnaire screen as specified by the advisor (dummy HTML pages, since recording responses is not required).



	id	username	password	email	etc
Edit Copy Delete	1	user1	pass1	email1	0
Edit Copy Delete	2	user2	pass2	email2	0

Figure 6: phpmyadmin SQL database of advisors.



	id	base_1	base_2	base_3	contact_1	contact_2	contact_3	misc_1	misc_2	misc_3	misc_4
Edit Copy Delete	1	1	7	0	4	5	6	0	0	3	2
Edit Copy Delete	2	0	1	0	0	2	0	0	3	0	0

Figure 7: phpmyadmin SQL database of order of screens.

Source Code:

The source code for this proposal is available to view on Github at the following address: https://github.com/alexwardCS480/riskalyze.git

The system used by going to the following address: https://www.fastfishdevelopment.com/riskalyze

Dreamweaver was used to write the PHP, HTML, JavaScript, and jQuery code. The SQL databases were created in phpMyAdmin as shown by Fig. 6 and 7.

Note, there are currently 2 advisors saved to the database. To login as an advisor use:
Username: user1
Password: pass1
OR
Username: user2
Password: pass2

To view a questionnaire, enter an advisors username (either user1 or user2).

Security:

Vulnerabilities:

There are known security vulnerabilities in the proposed system. The password to the SQL databases is stored in plaintext in the PHP files. Instead, the password should be stored in a config file protected by htaccess. There are also cases where POST and SESSION data are not properly escaped, this leaves the database vulnerable to SQL injections.

Precautions:

There are cases where the data is correctly escaped for proof of concept. The server is behind SSL and mod_security is implement on the server for an IDS.

Conclusion:

The system proposed to store questionnaire data provides a balance between efficiency and simplicity. An SQL database is used to save the desired screens to display for each advisor. This SQL database holds the file names of each HTML questionnaire screen so the correct screens can be displayed to each user. A second SQL database is used to hold the advisors critical account data and used to link to the prior database. The actual questionnaire screens are saved locally as HTML files.