

Online Self-Learning Software: Main Interface Design Specifications

Final Draft
April 27, 2011

By Robert S. LaForge
Instructor Janusz Zalewski

Networks Project
Florida Gulf Coast University
Fort Myers, Florida
Spring 2011

Table of Contents

1.	Introduction	3
2.	Requirements Specification	4
2.1	General Requirements	4
2.2	Static Representation	24
2.2.1	Overview	24
2.2.2	Registration, Log In, Log Out	24
2.2.3	Basic Program User Navigation	25
2.2.4	Basic Program Administrator Navigation	26
2.3	Dynamic Representation	27
2.3.1	Overview	27
2.3.2	Site Registration	28
2.3.3	Other Dynamic Representations	28
3.	Design Description	29
3.1	Overview	29
3.2	Database Design	29
3.2.1	Users Table	30
3.2.2	Game Table	30
3.2.3	Question Table	30
3.2.4	Genre Table	31
3.2.5	GameLinePair Table	31
3.2.6	LiveGame Table	31
4.	Implementation	33
4.1	Main Page	33
4.2	Header Files	34
4.3	Footer File	34
4.4	Log in Page	35
4.5	Play Games Page	36
4.6	Continue Games Page	38
4.7	Previous Games Page	39
4.8	Data Input Pages	39
4.9	Data View Pages	41
4.10	Modify Data Pages	42
5.	Conclusion	44
	References	46
	Appendix A: Code	47
	Appendix B: User Manual	97

1. Introduction

The purpose of this document is to provide details on the structure of the online self-learning software being developed as a group by the Spring 2011, CNT 4104 Computer Network Programming students. In general, this is a client/ server model software design containing a web interface, server, and database. This interface shall allow users to build a dataset of information that may be used in various game environments designed to help the user memorize the information. To access the web server through the web interface, the interface shall allow users to create user accounts and log in to the site. Once logged in, the interface shall provide multiple options.

First and foremost, the software shall allow users to interact with a database of information for use in the various game environments that it provides. The software allows the user to select a category for the information being entered and then requires the user to provide word/description pairs. The software also allows users to select game types, play games, and view high scores. The software allows the user to add additional data by category at any time. During web interface connections, the software provides a means for remembering previous session states in case of connection loss or users choosing to exit game play either because of pausing or exiting currently active games.

The purpose of this particular document is to provide descriptive details regarding the software's interface as it relates to site access, navigation, session states, and data management. These descriptions shall be detailed in the following sections both through diagrammed and written explanations.

2. Requirements Specification

2.1 General Requirements

Listed below are the functional requirements for the online self-learning software:

2.1.1 Registration Screen Requirements:

- 2.1.1.1 The interface shall allow new users to register by requiring a first name, last name, username, password, and email address.
- 2.1.1.2 The program shall insert the newly created user into the database.
- 2.1.1.3 The interface shall provide a button by which the users may click to create an account.
- 2.1.1.4 The interface shall provide a link by which the user may click to be redirected to the log in page.

Create A New Account

[Return to Log In](#)



The form consists of five text input fields stacked vertically, each preceded by a label: 'First Name:', 'Last Name:', 'Username:', 'Password:', and 'Email:'. Below these fields is a rectangular button with the text 'Create user'.

(Figure 2.1.1) Create an Account interface

2.1.2 Log In Screen Requirements:

- 2.1.2.1** The interface shall allow users to login to access all web pages on the site.
- 2.1.2.2** The interface allows users to log in by typing in a user name and password.
- 2.1.2.3** The interface shall provide a button by which the users may click to log in.
- 2.1.2.4** The program shall check the database to determine whether the username and password are a valid combination.
- 2.1.2.5** The program shall display an error message at the top of the log in text boxes stating that the combination is incorrect if the username and password combination are invalid.
- 2.1.2.6** The program will redirect the user to the main page of the site if the username and password combination are valid.
- 2.1.2.7** The program will create a unique user session that is maintained while the user is logged in if the username and password combination are valid.
- 2.1.2.8** The program will link all related data stored in the database to the user id.
- 2.1.2.9** The interface shall provide a link by which the user may click to be redirected to the 'create an account' page.

Welcome to the FGCU Online Learning Game Website



Username:

Password:

[Create An Account](#)

(Figure 2.1.2) Log in interface

2.1.3 Main Page Screen Requirements:

- 2.1.3.1** The program shall display a link to log in or create an account if

the user is not logged in.

Already a member? Please Login

or **Sign up for a Free Account**

(Figure 2.1.3) Main page interface when not logged in

- 2.1.3.2 The program shall display a Welcome note displaying the username when the user is logged in.
- 2.1.3.3 The program shall provide a 'log out' link if the user is logged in.
- 2.1.3.4 The program shall provide an 'update profile' link if the user is logged in.

Welcome slaforge!

Have fun playing games and uploading Word/Description Pairs.

Click [here to logout](#) | [Update Profile](#)

(Figure 2.1..4) Main page interface when logged in

2.1.4 Update Profile Screen Requirements:

- 2.1.4.1 The interface shall allow users to update their profile by entering data into the text boxes.
- 2.1.4.2 When the screen loads, the text boxes shall be populated by the current profile information stored in the database.
- 2.1.4.3 The interface shall provide a button that, when selected, populates the database with the new information filled into the text boxes.
- 2.1.4.4 The interface shall display a message stating that the profile was updated successfully.
- 2.1.4.5 The interface shall provide a link, that when selected, will redirect the user to the home page.
- 2.1.4.6 The interface shall display an error message when the update is not

successfully inserted into the database.

Update Profile

[Return to Home](#)

First Name:	<input type="text" value="Scott"/>
Last Name:	<input type="text" value="LaForge"/>
User Name:	<input type="text" value="slaforge"/>
Password:	<input type="text" value="1panther"/>
Email:	<input type="text" value="rslafor@eagle.fgcu.edu"/>
<input type="button" value="Update"/>	

(Figure 2.1.5) Update Profile interface

2.1.5 Play Game Screen Requirements:

- 2.1.5.1 The interface shall allow users to select a game currently hosted on the website.
- 2.1.5.2 The interface shall provide links to each of the game pages.
- 2.1.5.3 The interface shall display the name of each game as a link.
- 2.1.5.4 The interface shall display a description of each game.
- 2.1.5.5 The program shall create an instance (a unique row designated by an ID in the 'live game' table) of the game in the database that holds the game ID, the user ID, and the current date and time.
- 2.1.5.6 The program shall pass the ID of the instance just created in the database as a parameter when the game link is selected.

Choose a Game to Play!

Title	Description
Burn and Learn	Race your car faster and faster by answering questions quickly. Otherwise your game may come to a complete stop!
Crossword Madness!	Try your best at these insane crossword clues!
Dynamic Multiple Choice	Take time-limited multiple choice quizzes on a variety of topics, with randomized correct and incorrect choices.
Peril	A dangerous round of questions where answering incorrectly could cost you a lot of points!
Quizzes	Three rounds of increasing difficulty, can you make it all the way through with 80% correct or better?

(Figure 2.1.5) Play Game interface

2.1.6 Continue Game Screen Requirements:

- 2.1.6.1 The interface shall allow users to continue a game they have previously started and have not finished yet.
- 2.1.6.2 The interface shall display all the current live games as a list.
- 2.1.6.3 The interface shall provide links to access these games in progress.
- 2.1.6.4 The interface shall display the name of each game as a link.
- 2.1.6.5 The interface shall display the number of questions correct out of the total number of questions.
- 2.1.6.6 The interface shall displaying a link titled, 'delete game', that allows the user to delete a current game from the database.
- 2.1.6.7 The program shall pass the ID of the live game as a parameter to the game page that is being accessed.
- 2.1.6.8 The program shall display a message stating that the game was successfully deleted when the user selects the delete game page.
- 2.1.6.9 The program shall display an error message when the program is unsuccessful at deleting the game from the database.

Continue A Game!

The Game you selected has been successfully deleted.

Continue Burn and Learn	16 out of 20 questions answered	Current Score: 70.00	Delete Game
Continue Peril	7 out of 22 questions answered	Current Score: 31.82	Delete Game
Continue Dynamic Multiple Choice	0 out of 0 questions answered	Current Score: 0.00	Delete Game

(Figure 2.1.6) Continue Game interface

2.1.7 Previous Game Screen Requirements:

- 2.1.7.1 The interface shall allow users to review a list of games they have previously finished.
- 2.1.7.2 The interface shall provide links to access these previously played

games in detail.

- 2.1.7.3 The interface shall display the name of each game as a link.
- 2.1.7.4 The interface shall display the number of questions correct out of the total number of questions.
- 2.1.7.5 The interface shall display the final score of the game.

Previous Game Info

Crossword Madness! 13 out of 15 questions answered correctly. Final Score: 86.67

(Figure 2.1.7) Previous Game interface

2.1.8 Input Data Screen Requirements:

- 2.1.8.1 The interface shall allow users to add new word/description pairs to the database.
- 2.1.8.2 The interface shall allow the users to select a category to describe the word/description pair they are adding.
- 2.1.8.3 The interface shall provide a button titled, 'add', that saves the word description pairs into the database.
- 2.1.8.4 The interface shall provide a link to the specific word/description pairs the user has added.

Input word description pairs.

[View List](#)

Word:

Description:

Genre:

(Figure 2.1.8) Input Data interface

2.1.9 My Data Screen Requirements:

- 2.1.9.1** The interface shall allow users to view the word/description pairs that they have added into the database.
- 2.1.9.2** The interface shall have a link directing them to a page where they may modify their word/description pair.
- 2.1.9.3** The interface shall have a link allowing them to delete their word/description pair.
- 2.1.9.4** The interface shall display the word, description and genre of each listing.
- 2.1.9.5** The interface shall display a message telling the user that the word/description pair was successfully deleted when the user selects the delete link.
- 2.1.9.6** The interface shall display an error message when the program is unsuccessful at deleting the word/description pair.

View, Modify, and Delete Word/Description Pairs

Modify	Delete	Word	Description	Genre
Modify	Delete	Accretion	The process by where dust and gas accumulated into larger bodies such as stars and planets.	Astronomy
Modify	Delete	Java	An object-oriented programming language developed by Sun Microsystems. It resembles C++, but was designed to avoid some of C++'s most notorious flaws.	Computer Science
Modify	Delete	Search Engine	A computer system designed to help find information over a computer network such as the World Wide Web, inside a corporate or proprietary network or a personal computer.	Computer Science

(Figure 2.1.9) My Data interface

2.1.10 Modify My Data Screen Requirements:

- 2.1.10.1** The interface shall allow users to modify a selected word/description pair.
- 2.1.10.2** The interface shall display a form with the current data stored in the database.
- 2.1.10.3** The program shall allow the user to change the word, description, and genre of the particular pair.
- 2.1.10.4** The program shall update the database with the new information when the save button is selected.
- 2.1.10.5** The program shall display a message stating that the pair was successfully updated when the 'save' button is selected.
- 2.1.10.6** The interface shall display an error message when the program was unsuccessful at updating the database.
- 2.1.10.7** The interface shall contain a link that shall direct the user to the entire list of the user's word/ description pairs.

Modify Word/Description Pairs

[Back to list](#)

Update and save	
Name:	<input type="text" value="Accretion"/>
Description:	<div>The process by where dust and gas accumulated into larger bodies such as stars and planets.</div>
Genre:	<input type="text" value="Astronomy"/>
<input type="button" value="Save"/>	

(Figure 2.1.10) Modify My Data interface

2.1.11 Administration Home Page Screen Requirements:

- 2.1.11.1 The interface shall display a welcome message displaying the user name.

Adminstration Area

Welcome, lilapolarski.

Here you can view and delete the word description pairs, current users, and current games.

(Figure 2.1.11) Modify My Data interface

2.1.12 Admin View Data Screen Requirements:

- 2.1.12.1 The interface shall allow administrators to view the word/description pairs that have been added into the database.

- 2.1.12.2 The interface shall have a link directing them to a page where they may modify these word/description pair.
- 2.1.12.3 The interface shall have a link allowing them to delete any word/description pair.
- 2.1.12.4 The interface shall display the word, description, and genre of each listing.
- 2.1.12.5 The interface shall display a message telling the administrator that the word/description pair was successfully deleted when the administrator selects the delete link.
- 2.1.12.6 The interface shall display an error message when the program is unsuccessful at deleting the word/description pair.

View, Modify, and Delete Word/Description Pairs

Modify	Delete	Word	Description	Genre
Modify	Delete	Accretion	The process by where dust and gas accumulated into larger bodies such as stars and planets.	Astronomy
Modify	Delete	Algorithm	A component that improves performance by transparently storing data such that future requests for that data can be served faster.	Computer Science
Modify	Delete	Assembly Language	A type of computer CPU programming language, which consists entirely of numbers and are almost impossible for humans to read and write.	Computer Science
Modify	Delete	AVL Tree	A self-balancing binary search tree, and it is the first such data structure to be invented. In this tree, the heights of the two child subtrees of any node differ by at most one.	Computer Science
Modify	Delete	Cache	A component that improves performance by transparently storing data such that future requests for that data can be served faster.	Computer Science
Modify	Delete	Computer	A device doing calculations.	Computer Science
Modify	Delete	Computer Science	The study of the theoretical foundations of information and computation, and of practical techniques for their implementation and application in computer systems.	Computer Science
Modify	Delete	Constellation	An internationally defined area of the celestial sphere.	Astronomy
Modify	Delete	Critical Section	A piece of code or a set of instructions that can only be executed by one process or thread at a time.	Computer Science

(Figure 2.1.12) Admin View Data interface

2.1.13 Admin Modify Data Screen Requirements:

- 2.1.13.1 The interface shall allow administrators to modify a selected word/description pair.
- 2.1.13.2 The interface shall display a form with the current data stored in the database.
- 2.1.13.3 The program shall allow the administrator to change the word, description, and genre of the particular pair.
- 2.1.13.4 The program shall update the database with the new information

when the save button is selected.

- 2.1.13.5** The program shall display a message stating that the pair was successfully updated when the 'save' button is selected.
- 2.1.13.6** The interface shall display an error message when the program was unsuccessful at updating the database.
- 2.1.13.7** The interface shall contain a link that shall direct the administrator to the entire list of word/ description pairs.

Modify Word/Description Pairs

[Back to list](#)

Update and save	
Name:	<input type="text" value="Accretion"/>
Description:	<div>The process by where dust and gas accumulated into larger bodies such as stars and planets.</div>
Genre:	<div>Astronomy ▼</div>
<div>Save</div>	

(Figure 2.1.13) Admin Modify Data interface

2.1.14 Admin View Genre Screen Requirements:

- 2.1.14.1** The interface shall allow administrators to view the genres that have been added into the database.
- 2.1.14.2** The interface shall have a link directing them to a page where they may modify each genre.

- 2.1.14.3 The interface shall have a link allowing them to delete any genre.
- 2.1.14.4 The interface shall provide a link to go to an add new genre page.
- 2.1.14.5 The interface shall display the name and description of the genre.
- 2.1.14.6 The interface shall display a message telling the administrator that the genre was successfully deleted when the administrator selects the delete link.
- 2.1.14.7 The interface shall display an error message when the program is unsuccessful at deleting the genre from the database.

View, Add, Modify, and Delete Genres

[Add A New Genre](#)

Modify	Delete	Name	Description
Modify	Delete	Astronomy	The branch of physics that studies celestial bodies and the universe as a whole.
Modify	Delete	Computer Science	The study of the theoretical foundations of information and computation, and of practical techniques for their implementation and application in computer systems.
Modify	Delete	Horticulture	The industry and science of plant cultivation including the process of preparing soil for the planting of seeds, tubers, or cuttings.
Modify	Delete	Misc	A miscellaneous selection of questions. A perfect place to put a question when you are not certain where it should go, or the category does not exist at this time.
Modify	Delete	Movies	Film encompasses individual motion pictures, the field of film as an art form, and the motion picture industry.
Modify	Delete	Video Games	Any of various games played using a microcomputer with a keyboard and often joysticks to manipulate changes or respond to the action or questions on the screen.

(Figure 2.1.14) Admin View Genres interface

2.1.15 Admin Modify Genre Screen Requirements:

- 2.1.15.1 The interface shall allow administrators to modify a selected genre.
- 2.1.15.2 The interface shall display a form with the current data stored in the database.
- 2.1.15.3 The program shall allow the administrator to change the name and description of a particular genre.
- 2.1.15.4 The program shall update the database with the new information when the save button is selected.
- 2.1.15.5 The program shall display a message stating that the genre was successfully updated when the 'save' button is selected.
- 2.1.15.6 The interface shall display an error message when the program was unsuccessful at updating the database.

- 2.1.15.7** The interface shall contain a link that shall direct the administrator to the entire list of genres.

Modify Genre

[Back to list](#)

Update and save	
<input type="text"/>	
Name:	<input type="text" value="Computer Science"/>
Description:	<div>The study of the theoretical foundations of information and computation, and of practical techniques for their implementation and application in computer systems.</div>
<input type="button" value="Save"/>	

(Figure 2.1.15) Admin Modify Genre interface

2.1.16 Admin Add New Genre Screen Requirements:

- 2.1.16.1** The interface shall allow administrators to add new genres to the database.
- 2.1.16.2** The interface shall allow the administrators to input a name and description of the new Genre.
- 2.1.16.3** The interface shall provide a button titled, 'add', that saves the Genre into the database.
- 2.1.16.4** The interface shall provide a link to the entire set of Genres in the database.

Add A New Genre

[Return to List](#)

Name:

Description:

(Figure 2.1.16) Admin Add New Genre interface

2.1.17 Admin View Users Screen Requirements:

- 2.1.17.1 The interface shall allow administrators to view the users that have been added into the database.
- 2.1.17.2 The interface shall have a link directing them to a page where they may modify each user.
- 2.1.17.3 The interface shall have a link allowing them to delete any user.
- 2.1.17.4 The interface shall display the username and email of all the users in the database.
- 2.1.17.5 The interface shall display a message telling the administrator that the user was successfully deleted when the administrator selects the delete link.
- 2.1.17.6 The interface shall display an error message when the program is unsuccessful at deleting the user from the database.

View, Modify, and Delete the Online Users

Modify	Delete	Username	Email
Modify	Delete	amcolema	amcolema@eagle.fgcu.edu
Modify	Delete	capitalreef	capital@reef.com
Modify	Delete	jocanas	
Modify	Delete	justinHodnett	jrhodnet@eagle.fgcu.edu
Modify	Delete	lilapolarski	lilapolarski@yahoo.com
Modify	Delete	ncalteen	ncalteen@gmail.com
Modify	Delete	slaforge	rslaforg@eagle.fgcu.edu
Modify	Delete	tjyanai	tjyanai@eagle.fgcu.edu
Modify	Delete	Zalewski	ikswelaz@gmail.com

(Figure 2.1.17) Admin View User interface

2.1.18 Admin Modify Users Screen Requirements:

- 2.1.18.1 The interface shall allow administrators to modify a selected user.
- 2.1.18.2 The interface shall display a form with the current data stored in the database.
- 2.1.18.3 The program shall allow the administrator to change the first name, last name, username, password, rank, email address, the activation status, and the administrative privileges of a particular user.
- 2.1.18.4 The program shall update the database with the new information when the save button is selected.
- 2.1.18.5 The program shall display a message stating that the user was successfully updated when the 'update' button is selected.
- 2.1.18.6 The interface shall display an error message when the program was unsuccessful at updating the database.
- 2.1.18.7 The interface shall contain a link that shall direct the administrator to the entire list of users.

Modify Users

[Back to list](#)

Update and save	
First Name:	<input type="text" value="Capital"/>
Last Name:	<input type="text" value="Reef"/>
User Name:	<input type="text" value="capitalreef"/>
Password:	<input type="text" value="reefy"/>
Rank:	<input type="text" value="0"/>
Email:	<input type="text" value="capital@reef.com"/>
Activated:	<input type="text" value="1"/> ▼
Admin Priveleges:	<input type="text" value="0"/> ▼
<input type="button" value="Update"/>	

(Figure 2.1.18) Admin Modify User interface

2.1.19 Admin View Games Screen Requirements:

- 2.1.19.1** The interface shall allow administrators to view the games that have been added into the database.
- 2.1.19.2** The interface shall have a link directing them to a page where they may modify each game.
- 2.1.19.3** The interface shall have a link allowing them to delete any game.
- 2.1.19.4** The interface shall display the title, description, and site address of all the games in the database.
- 2.1.19.5** The interface shall display a message telling the administrator that the game was successfully deleted when the administrator selects the delete link.
- 2.1.19.6** The interface shall display an error message when the program is

unsuccessful at deleting the game from the database.

2.1.19.7 The interface shall provide a link to add new games into the database.

View, Modify, and Delete the Online Games

[Add A New Game](#)

Modify	Delete	Title	Description	Site Address
Modify	Delete	Burn and Learn	Race your car faster and faster by answering questions quickly. Otherwise your game may come to a complete stop!	/~tjyanai/BurnNLearn.php
Modify	Delete	Crossword Madness!	Try your best at these insane crossword clues!	/~ncalteen/main.php
Modify	Delete	Dynamic Multiple Choice	Take time-limited multiple choice quizzes on a variety of topics, with randomized correct and incorrect choices.	/~acoleman/self_study/user_menu.php
Modify	Delete	Peril	A dangerous round of questions where answering incorrectly could cost you a lot of points!	/~jrhodnet/peril.php
Modify	Delete	Quizzes	Three rounds of increasing difficulty, can you make it all the way through with 80% correct or better?	/~jocanas/game.php

(Figure 2.1.19) Admin View Games interface

2.1.20 Admin Modify Games Screen Requirements:

- 2.1.20.1** The interface shall allow administrators to modify a selected Game.
- 2.1.20.2** The interface shall display a form with the current data stored in the database.
- 2.1.20.3** The program shall allow the administrator to update the title, description, and site address of a particular Game.
- 2.1.20.4** The program shall update the database with the new information when the save button is selected.
- 2.1.20.5** The program shall display a message stating that the Game was successfully updated when the 'update' button is selected.
- 2.1.20.6** The interface shall display an error message when the program was unsuccessful at updating the database.
- 2.1.20.7** The interface shall contain a link that directs the administrator to the entire list of Games.

Modify Game

[Back to list](#)

Update and save	
Title:	<input type="text" value="Burn and Learn"/>
Description:	<div>Race your car faster and faster by answering questions quickly. Otherwise your game may come to a complete stop!</div>
Site Address:	<input type="text" value="/~tjyanai/BurnNLearn.php"/>
<input type="button" value="Save"/>	

(Figure 2.1.20) Admin Modify Game interface

2.1.21 Admin Add New Game Screen Requirements:

- 2.1.21.1** The interface shall allow administrators to add new Games to the database.
- 2.1.21.2** The interface shall allow the administrators to input a name and description of the new Game.
- 2.1.21.3** The interface shall provide a button titled, 'add', that saves the Game information into the database.
- 2.1.21.4** The interface shall provide a link to the entire set of Games in the database.

Add A New Game

[Return to List](#)

Title:

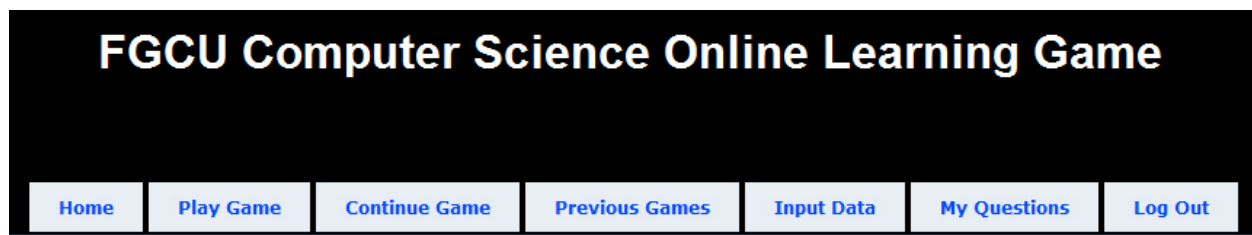
Description:

Site Address:

(Figure 2.1.21) Admin Add New Game interface

2.1.22 User Header File Interface Requirements:

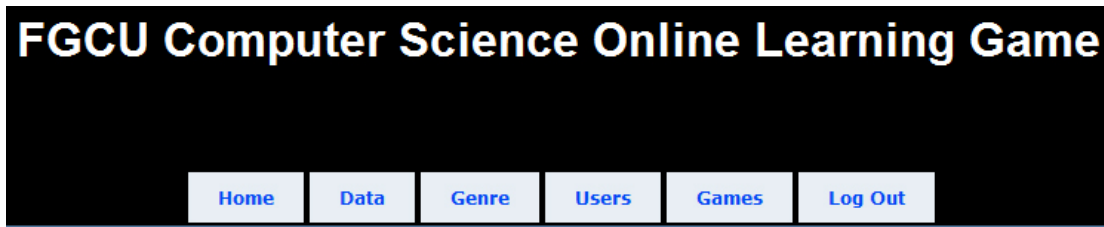
- 2.1.22.1 The interface shall provide buttons for users to the different pages on the site that are accessible to users without administrative privileges.
- 2.1.22.2 The interface shall provide a log out button, allowing the user to stop the current session.
- 2.1.22.3 The interface shall display the title of the site.
- 2.1.22.4 The program shall display a log in button if the user is currently not logged in.



(Figure 2.1.22) User Header File Interface

2.1.23 Admin Header File Interface Requirements:

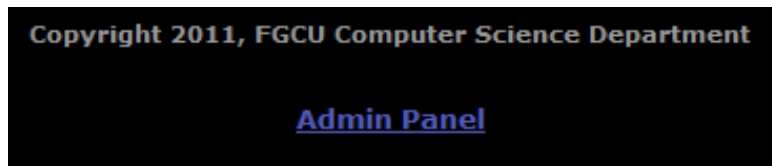
- 2.1.23.1 The interface shall provide buttons for users with administrative privileges to the different pages on the site.
- 2.1.23.2 The interface shall provide a log out button, allowing the administrator to stop the current session.
- 2.1.23.3 The interface shall display the title of the site.
- 2.1.23.4 The program shall display a log in button if the administrator is currently not logged in.



(Figure 2.1.23) Admin Header File Interface

2.1.24 Footer File Requirements:

- 2.1.24.1 The interface shall display the copyright year along with the name of the Computer Science Department.
- 2.1.24.2 A link to the administrative portion of the site shall be displayed when users with administrative privileges are logged in.



(Figure 2.1.24) Footer File Interface

2.2 Static Representation

2.2.1 Overview

The purpose of this section is to provide a static representation of the online self-learning software interface, separated into components, and to provide graphical and written descriptions as they correlate to the functional requirements of the software.

2.2.2 Registration, Login, Logout

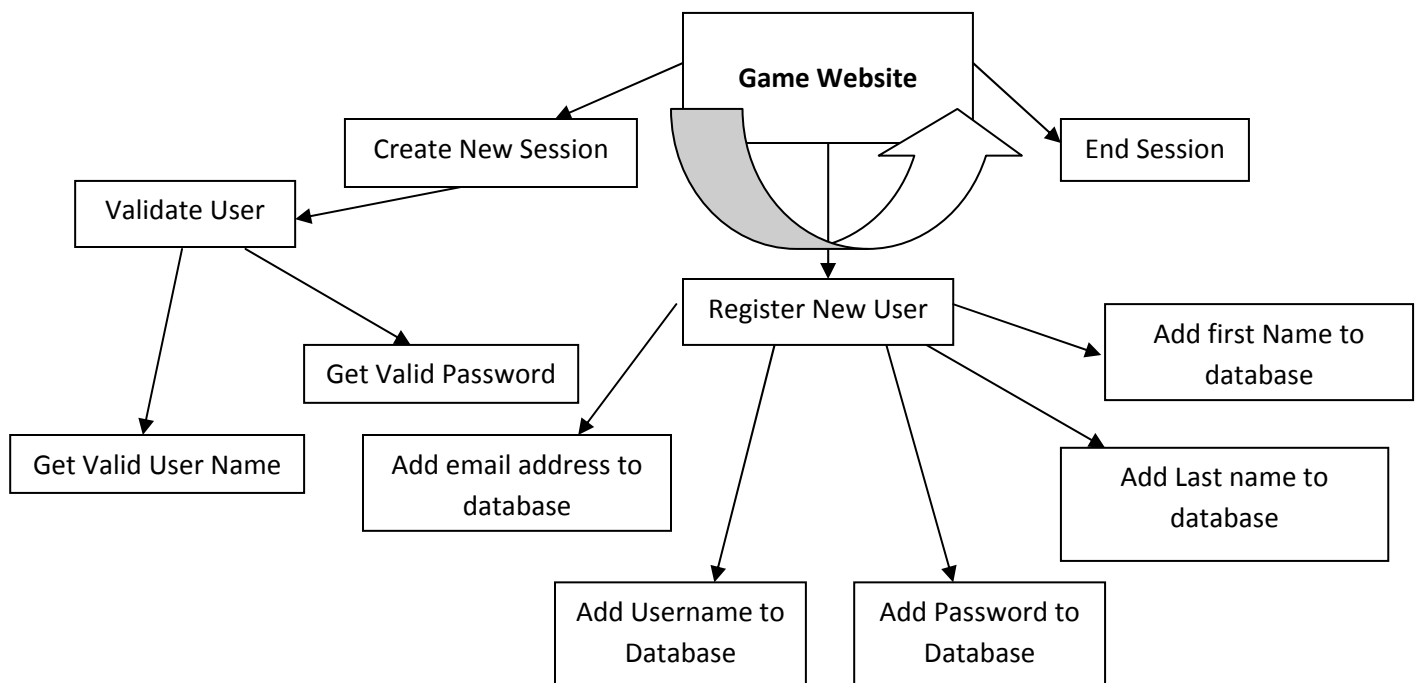


Figure 2.2.1: Registration, Login, Logout

Figure 2.2.1 depicts the general means by which the program allows a user to register, login, and logout. To register, the program requires a first name, last name, username, email address, and password. Once the create account button is clicked, the username/password combination allows the user to login on the log in screen. **(Requirement 2.1.1)**. When the program accepts the user registration, the user may create a new session by logging in. The program validates the user name and password. If the username/password pair is correct, a new session is created. If it is incorrect, the program will display an error message. **(Requirement**

2.1.2). The program allows users to logout by selecting the logout button on the Main Page screen (**Requirements 2.1.22, 2.1.23**).

2.2.3 Basic Program User Navigation

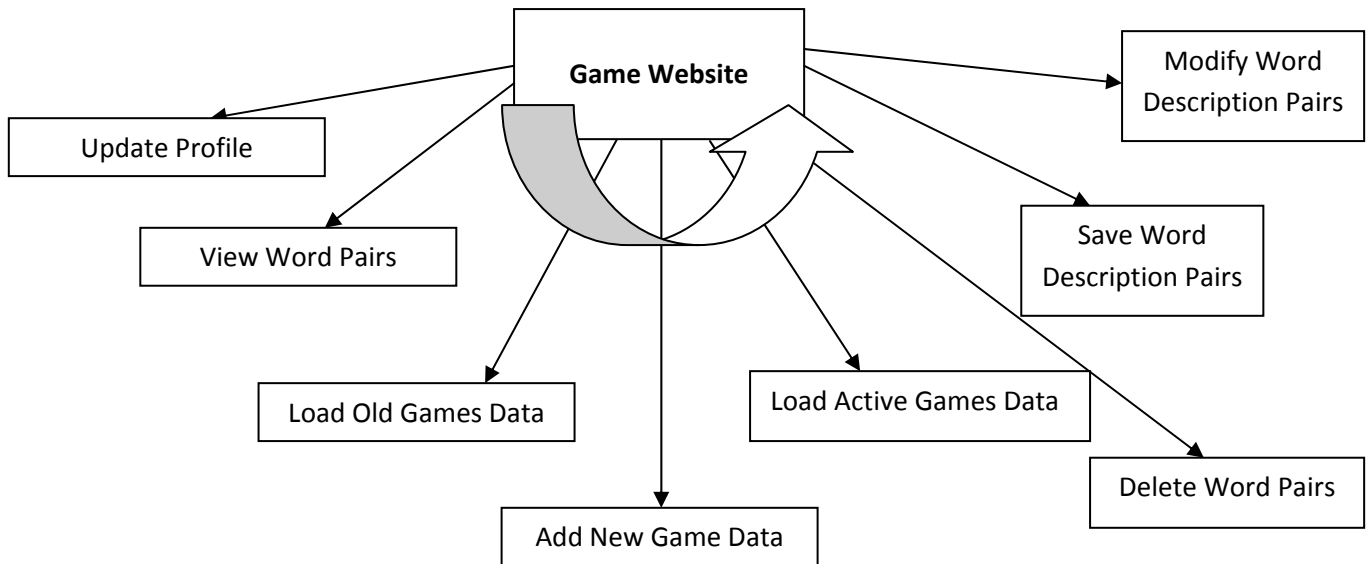


Figure 2.2.2: General User Screen Navigation

Figure 2.2.2 displays the basic navigation that the program allows users, without administrative privileges, within the site. Essentially, the program allows users to update their profiles (**Requirement 2.1.4**), play a new game (**Requirement 2.1.5**), play currently active games (**Requirement 2.1.6**), view a list of previously played games (**Requirement 2.1.7**), develop word/description pairs and select a category (**Requirement 2.1.8**), view word description/pairs that they have previously added (**Requirement 2.1.9**), modify their word/pair data at a later time (**Requirement 2.1.10**).

2.2.4 Basic Program Administrator Navigation

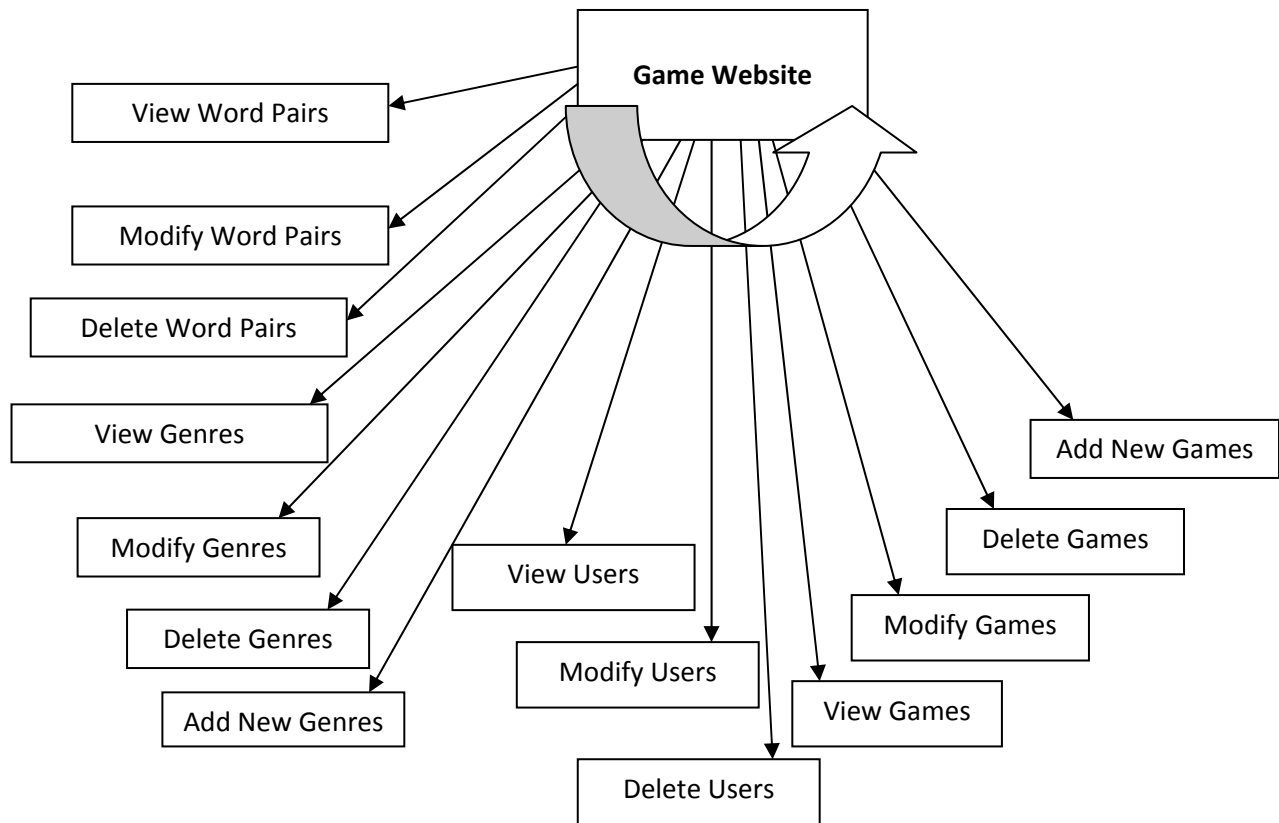


Figure 2.2.2: General User Screen Navigation

Figure 2.2.2 displays the basic navigation that the program allows administrative users within the site. For word pairs in the database, the program allows administrators to view (**Requirement 2.1.12**), delete (**Requirement 2.1.12**), and modify individual pairs (**Requirement 2.1.13**). As well, administrators may view (**Requirement 2.1.14**), delete (**Requirement 2.1.14**), modify (**Requirement 2.1.15**), and add genres into the database (**Requirement 2.1.16**). Administrators may also view (**Requirement 2.1.17**), delete (**Requirement 2.1.17**), and modify users (**Requirement 2.1.18**). Administrators are also responsible for the games on the site. They may view games stored in the database (**Requirement 2.1.19**), delete games (**Requirement 2.1.19**), modify the games (**Requirement 2.1.20**), and add new games into the system (**Requirement 2.1.21**).

2.3 Dynamic Representation

2.3.1 Overview

The purpose of this section is to provide a sample dynamic representation of the online self-learning software interface as it relates to user registration and logging in by providing graphical and written descriptions as they correlate to functional requirements one through four of the software.

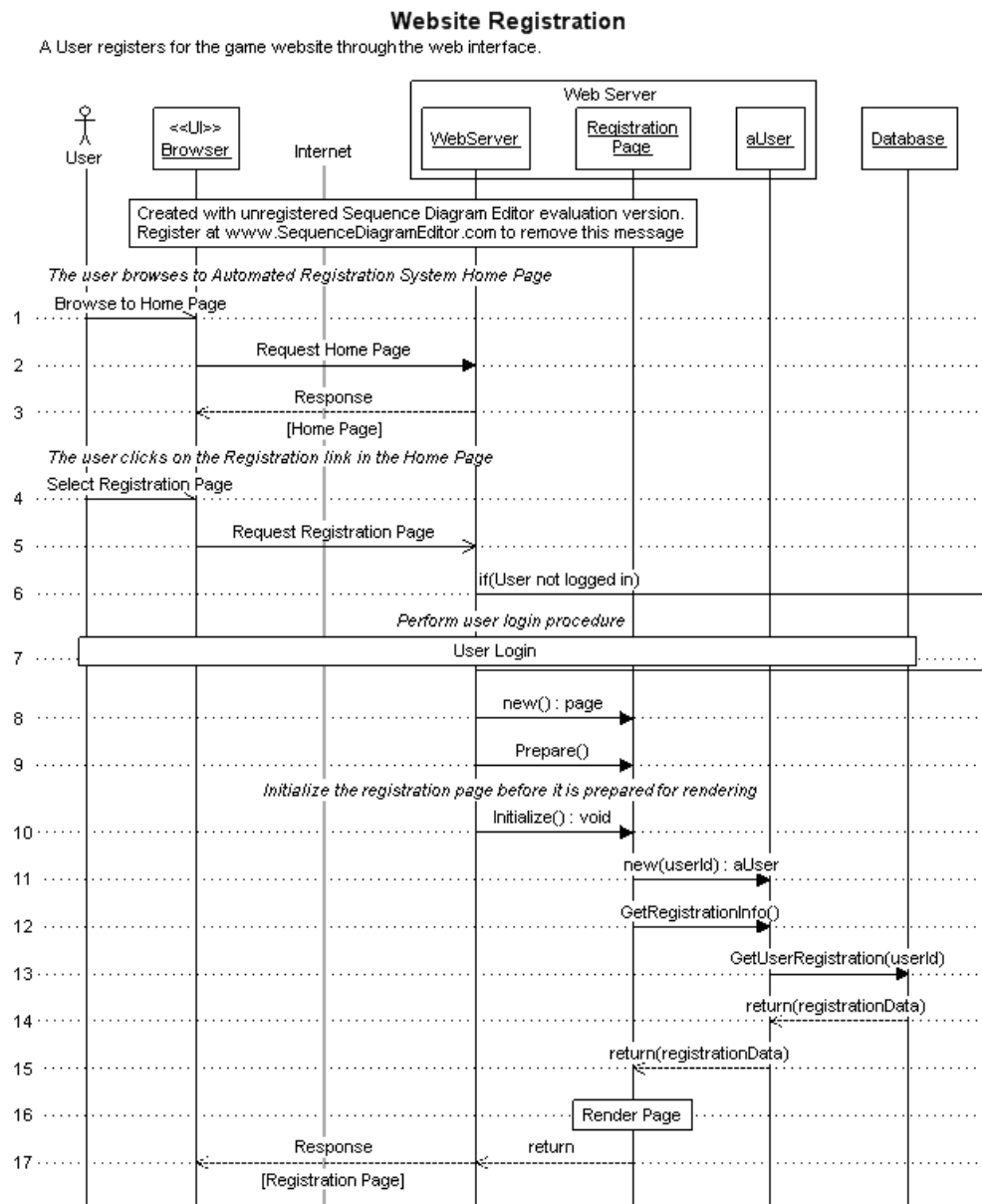


Figure 2.3.1: Sequence Diagram: Website Registration and Login

2.3.2 Site Registration

Figure 2.3.1 portrays the sequencing of event for registering new users and user login (**Requirements 2.1.1 and 2.2.2**). In this diagram, a user views the website through a web browser. The web browser connects through the Internet to the online self-learning software consisting of a web server, web pages, and a database for controlling data population specific to the user. The user types the web address of the game site. This requests the home page from the web server. In response the web server returns the home page screen. The user then clicks on the registration link on the home page. This sends a request to the web server. Since the user is not logged in or registered, a blank registration page is returned by the server. This is then populated by the user and submitted. Upon submission, the request is sent to the web server, the web server receives the filled out page and then prepares it for being rendered to the website's database. The information is initialized, user id and registration information is established as a new user and then this information is passed to the database for storage. The registration information is then returned successfully. The completed registration page is rendered and returned through the web server to the web browser client-side displaying a completed registration page reiterating the sign up information. The sequence diagram depicts the act of logging in as a simple block in the center of the diagram without details.

2.3.3 Other Dynamic Representations

Other dynamic interactions shall follow similarly to the diagram as depicted in the registration process. Through a Browser (User Interface), a user shall make different requests to the web server depending on the specific links that are filled in and selected. This in turn will make accesses to the database, linked to the user's profile, and eventually return responses to the browser. In this manner, all games may be played, additional game data may be entered into the database, and previous games may be viewed.

3. Design Description

3.1 Overview

In general, this project utilizes the Internet to connect the online gaming website to the user. So long as the user and hosting server are on the same network, the pages may be accessed from anywhere. To complete this project, a database is designed to store vital information necessary for site interaction. Then, web pages are developed to provide an interface allowing the user or administrators to navigate the site, play games, and view and modify database information.

The webpages are displayed to the user via HTML and formatted with CSS. PHP is used for all scripts that must generate sessions, maintain variables, or contain SQL queries that access the database. SATNET, one of the FGCU Computer Science servers, is used to house both the webpages and the database.

3.2 Database Design

Figure 3.2.1 shows the database structure for the online gaming website and includes their relationships.

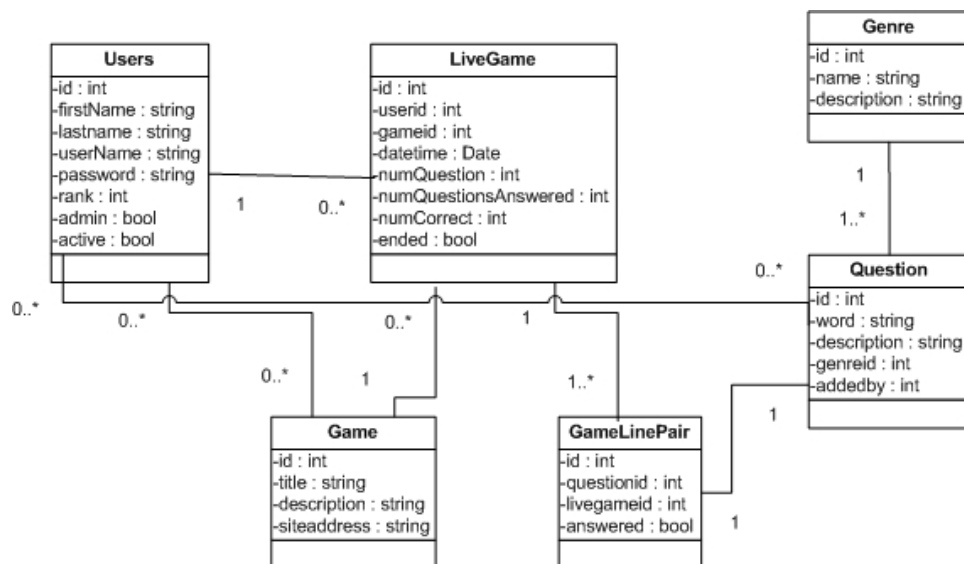


Figure 3.2.1: UML Diagram of the database

3.2.1 Users Table

The 'users' table consists of eight attributes. The 'id' is an integer value and acts as the primary key of the table. It is unique and auto increments. The 'firstname', 'lastname', 'username' and 'password' are all varchars and are self-explanatory. The 'rank' is an integer value that increments as the user adds data pairs. The 'admin' attribute can either be a zero one. If it is zero, the user does not have administrator privileges. If it is a one, the user has administration privileges. The 'active' variable stores whether the user has an active account or not. If it is a one, the account is active.

3.2.2 Game Table

The 'game' table consists of three attributes. The 'id' is an integer value and acts as the primary key of the table. It is unique and auto increments. The 'title' attribute is a varchar and stores the game title. The 'description' attribute is a varchar and stores the description of the game in the database. The 'siteaddress' attribute is a varchar and stores the address of the game on the website.

3.2.3 Question Table

The 'question' table consists of five attributes. The 'id' is an integer value and acts as the primary key of the table. It is unique and auto increments. The 'word' and 'description' attributes are both varchars. The word attribute stores up to 100 characters and the 'description' attribute stores up to 500 characters. The 'genreid' is a foreign key from the 'genre' table and is used to reference the particular genre that the 'question' instance is a part of. The 'addedby' attribute is also a foreign key and references the 'id' from the 'users' table according to the 'user' that created the 'question' instance.

3.2.4 Genre Table

The 'genre' table consists of four attributes. The 'id' is an integer value and acts as the primary key of the table. It is unique and auto increments. The 'name' attribute is a varchar and stores the genre title. The 'description' attribute is also a varchar with a length up to 250 characters and stores the description of the genre in the database.

3.2.5 GameLinePair Table

The 'gamelinepair' table consists of four attributes. The 'id' is an integer value and acts as the primary key of the table. It is unique and auto increments. The 'questionid' and 'livegameid' attributes are integer values. The 'questionid' attribute is a foreign key from the 'question' table and references the 'id' of the question. The 'livegameid' is also a foreign key and references the 'id' of the active or finished game in the 'livegame' table. The 'answered' attribute is a zero or a one. It is set to one if the question The word attribute stores up to 100 characters and the 'description' attribute stores up to 500 characters. The 'genreid' is a foreign key from the 'genre' table and is used to reference the particular genre that the 'question' instance is a part of. The 'addedby' attribute is also a foreign key and references the 'id' from the 'users' table according to the 'user' that created the 'question' instance.

3.2.6 LiveGame Table

The 'livegame' table consists of eight attributes. The 'id' is an integer value and acts as the primary key of the table. It is unique and auto increments. The 'userid' and 'gameid' attributes are integer values. The 'userid' attribute is a foreign key from the 'users' table and references the 'id' of the user. The 'gameid' is also a foreign key and references the 'id' of the game in the 'game' table. The 'datetime' attribute is a timestamp and stores the time when the game was last updated. The 'numquestion' is an integer attribute and is populated with the total number questions for that particular game instance. The 'numquestionsanswered' attribute is

also an integer and stores the total number of questions a user has answered for a particular game. The 'numcorrect' attribute is an integer that stores the total number of questions the user answered correctly while playing the game. The 'ended' attribute is an integer that can be either a zero or a one. If it is a zero, it means that the game has not completed yet. If it is a one, then the game was successfully completed.

4. Implementation

This section discusses some of the key programming features that are necessary to make the website functional.

4.1 Main Page (index.php)

This is the main entrance point into the site. When a user arrives at the site, the PHP script checks to see whether a session currently exists. If there is no session, the code provides a message directing the visitor to create an account or log in. Links to the prospective pages are written in HTML. If the visitor is logged in, a message is displayed showing the username and what the user may participate in on the site. It also shows two links allowing the user to update their profile or logout.

```
<?
if(isset($_SESSION['id'])){
    echo "<center><font face='Verdana' size='2' ><br><h3>Welcome $_SESSION[username]!";
    echo "</h3></br></br><b>Have fun playing games and uploading Word/Description Pairs.";
    echo "</b></br></br> Click <a href=logout.php>here to logout</a> &nbsp;| &nbsp;";
    echo "<a href=modifyprofile.php?id=$pid>Update Profile</a><br></center></font>";
    echo mysql_error();}
else {
    echo "<center><font face='Verdana' size='2' ><a href=login.php><b>Already a member? ";
    echo "Please Login</b></a></br></br> </center></font>";
    echo "<center><font face='Verdana' size='2' >or <a href=new_user.php><b>";
    echo "Sign up for a Free Account</b></a> </center></font>";
}
?>
```

4.2 Header Files (head.php and head_admin.php)

These PHP files are linked to the top of all web pages. The head.php file is included on all pages accessible only to users and the head_admin.php page is included on all pages accessible to administrators. Both of these files check to see whether a session exists or not. If a session does not exist, it prevents the user from accessing any pages except the home page (index.php). Other than the main page, the user will be redirected to the login page. If a session does exist, the user may navigate to the different pages on the site.

Each PHP file also contains a set of links specific to whether the user is accessing the administrative or general user side of the site. These links provide navigation to the majority of pages on the site as shown in figures 2.1.22 and 2.1.23.

The code to redirect the user to the login page is presented immediately below.

```
<?
if(isset($_SESSION['id'])){
    echo "<li><a href=logout.php>Log Out</a></li>";
    echo mysql_error();
}
else{
    echo "<li><a href=login.php>Log In</a></li>";
}
?>
```

4.3 Footer File (footer.php)

The footer file always displays the copyright message using html. PHP checks to see whether a session exists. If so, a database query is made checking to see whether the user has administrative privileges. If so, it allows a link to the administrative portion of the site to be displayed and clickable.

```

<?
if(isset($_SESSION['id'])){
    $u=$_SESSION['user_id'];
    $v=1;
    $adminquery="select * from users where id=$u";
    $adminresult=mysql_query($adminquery);
    $t=mysql_result($adminresult, 0, "admin");

    if($t==1) {
        echo "<center><font face='Verdana' size='2' ><br><a href=staff.php>Admin
        Panel</a><br></center></font>";
        echo mysql_error();
    }
}
?>

```

4.4 Log In Page (login.php)

The login page allows users to log in and have access to the content on the website. When the sign in button is clicked the program calls the same page again. The page, on load, checks to see whether a post is made. If so, the program first prepares the data input from the user before making a query. It then makes a query to the database to see whether the username and password combination are paired correctly or exist at all. If the combination is correct, a session is created that will be maintained while the user is on the site. The program then redirects the user to the home page. If the combination is incorrect, an error message is displayed on the page.

```

// START FORM PROCESSING
if (isset($_POST['submit'])) { // Form has been submitted.
    $errors = array();

    // perform validations on the form data
    $required_fields = array('username', 'password');
    $errors = array_merge($errors, check_required_fields($required_fields, $_POST));
}

```

```

$fields_with_lengths = array('username' => 30, 'password' => 30);
$errors = array_merge($errors,
    check_max_field_lengths($fields_with_lengths, $_POST));

$username = trim(mysql_prep($_POST['username']));
$password = trim(mysql_prep($_POST['password']));

if ( empty($errors) ) {
    // Check database to see if username and the hashed password exist there.
    $query = "SELECT id, username ";
    $query .= "FROM users ";
    $query .= "WHERE username = '{ $username }' ";
    $query .= "AND hashed_password = '{ $password }' ";
    $query .= "LIMIT 1";
    $result_set = mysql_query($query);
    confirm_query($result_set);
    if (mysql_num_rows($result_set) == 1) {
        // username/password authenticated
        // and only 1 match
        $found_user = mysql_fetch_array($result_set);
        $_SESSION['id']=session_id();
        $_SESSION['user_id'] = $found_user['id'];
        $_SESSION['username'] = $found_user['username'];

        redirect_to("~/slaforge/index.php");
    } else {
        // username/password combo was not found in the database
        $message = "Username/password combination incorrect.<br />
            Please make sure your caps lock key is off and try again.";
    }
} else {
    if (count($errors) == 1) {
        $message = "There was 1 error in the form.";
    } else {
        $message = "There were " . count($errors) . " errors in the form.";
    }
}
}
?>

```

4.5 Play Games Page (games.php)

When this page loads, the program checks to see whether parameters have been passed to itself. If parameters have not been passed, the program makes a query to the database requesting

all the game data. It then generates a list of all games, displaying each title as an html link that includes the game id as a parameter and a description of each game.

The program then calls the same page when a link to a particular game is selected. Now, on load, the page recognizes the request parameter and inserts a new instance of the particular game into the database in the 'livegame' table. It then redirects the user to the game they have selected and passes the 'livegame' ID as a parameter so that the particular game will have the necessary information to recognize the user and repopulate the database later when the user either finishes a game or chooses to continue it later.

One note, because Andon Coleman's game is set up differently, the program checks to see whether Andon's game is going to be played. If it is, it sends the user ID instead of the live game ID as a parameter.

```
if($id==6)
{
    $queryAndon = "Select siteaddress from game where id=$id";
    $resultAndon=mysql_query($queryAndon);
    $gaddress=mysql_result($resultAndon, 0, "siteaddress");
    if ($resultAndon) {
        echo '<META HTTP-EQUIV="Refresh" Content="0; URL='.$gaddress.'?uid='.$uid.'">';
        exit;
    }
}
$result = mysql_query("INSERT INTO livegame (userid, gameid, datetime, numquestion,
numquestionsanswered, ended, numcorrect)
VALUES ('$uid', $id, FROM_UNIXTIME($phpdate), 0, 0, 0, 0)");

$query2 = "Select siteaddress from game where id=$id";
$result2=mysql_query($query2);
$siteaddress=mysql_result($result2, 0, "siteaddress");

$query3 = "Select id, UNIX_TIMESTAMP(datetime) from livegame where
UNIX_TIMESTAMP(datetime)=$phpdate";
$result3=mysql_query($query3);
$id=mysql_result($result3, 0, "id");

if ($result3) {
    echo '<META HTTP-EQUIV="Refresh" Content="0; URL='.$siteaddress.'?id='.$id.'">';
    exit;
} else {
    $message = "There was a problem redirecting to the Game Site";
    $message .= "<br />" . mysql_error();
}
```

4.6 Continue Games Page (continue.php)

When this page loads, the program checks to see whether parameters have been passed to itself. If parameters have not been passed, the program makes a query to the database requesting all 'livegame' table data where the 'userid' equals the session userid and where the 'ended' attribute equals zero, meaning the game has not been completed. It then generates a list of all unfinished games, displaying each title as an html link to continue the game that includes the 'livegame' id appended to it, the number of questions answered out of the total number, the current score, and an html link also including the 'livegame' id to delete the game if they do not wish to continue it.

When a user selects the 'continue game' link the program navigates to the game page selected with the 'livegame' id appended. If the 'delete game' link is selected, the page reloads itself. Now, on load, the page recognizes the request parameter. It then deletes the game instance from the database in 'livegame' table. It then displays a message stating that the game was successfully deleted. The code below details the process by which a game instance is deleted from the 'livegame' table.

One note, the program also checks to see whether the row being added to the list is Andon Coleman's game. If it is, it sets the link so that it has two parameters to pass to his game, the user ID and the live game ID.

Below is the code for deleting a 'continue game' database entry.

```
<?php
if(isset($_REQUEST['id']))
{
    $id=$_REQUEST['id'];
    $query = "DELETE FROM livegame WHERE id=$id";
    $result=mysql_query($query);
    if ($result) {
        $message = "<h3>The Game you selected has been successfully deleted.</h3>";
    } else {
        $message = "The Game could not be deleted.";
        $message .= "<br />" . mysql_error();
    }
}
```

```
if (!empty($message)) {echo "<p class=\"message\">" . $message . "</p>";}
if (!empty($errors)) { display_errors($errors); }
```

4.7 Previous Games Page (previous.php)

This page is simpler than the ‘continue’ or ‘play games’ pages. When it loads, it does not look for any requests. Instead, it makes a query to the database retrieving all ‘livegame’ table data where the ‘userid’ equals the session id and that has a one in the ‘ended’ attribute. It then creates a list on the page displaying all of the previously played games. It displays the title, the number of questions answered correctly out of the total number, and the final score.

4.8 Data Input Pages

(addgame.php, addgenre.php, inputdata.php, new_user.php)

All of these PHP files work similarly. On load, the PHP script checks to see whether a post is being submitted. After, the page will simply load the regular page along with appropriate form boxes to be populated by the user. When the form is posted, the same page receives the information and inserts a new row in the appropriate table. Before it inserts, the page checks to see whether the information has been posted correctly. If it has not, it will display an error message. If the information is correct and was successfully inserted into the database, a message stating that the element was successfully added will be displayed.

The addgame.php file is available to administrators and allows them to insert a new game into the ‘game’ table. The addgenre.php file is also only available to administrators and allows them to insert a new genre into the ‘genre’ table. The inputdata.php file allows all users to insert new word description pairs into the ‘question’ table. The new_user.php file allows unregistered users to create an account in order to have access to the complete site.

Below shows code from inputdata.php for inserting a new word/description pair into the ‘question’ table in the database.

```

// START FORM PROCESSING
if (isset($_POST['submit'])) { // Form has been submitted.
    $errors = array();

    // perform validations on the form data
    $required_fields = array('word', 'description');
    $errors = array_merge($errors, check_required_fields($required_fields, $_POST));

    $fields_with_lengths = array('word' => 100, 'description' => 500);
    $errors = array_merge($errors,
        check_max_field_lengths($fields_with_lengths, $_POST));

    $word = trim(mysql_prep($_POST['word']));
    $description = trim(mysql_prep($_POST['description']));
    $genre = trim(mysql_prep($_POST['genre']));
    if ( empty($errors) ) {
        $query = "INSERT INTO question (word, description, genreid, addedby)
            VALUES ('{$word}', '{$description}', '{$genre}',
                '{$_SESSION['user_id']}')";
        $result = mysql_query($query, $connection);
        if ($result) {
            $message = "<h3>The word pair has been successfully added.</h3>";
            $word = "";
            $description = "";
            $query = "UPDATE users SET rank=rank+1 WHERE
                id='{$_SESSION['user_id']}'";
            $result = mysql_query($query, $connection);

        } else {
            $message = "The word pair could not be added.";
            $message .= "<br />" . mysql_error();
        }
    } else {
        if (count($errors) == 1) {
            $message = "There was 1 error in the form.";
        } else {
            $message = "There were " . count($errors) . " errors in the form.";
        }
    }
} else { // Form has not been submitted.
    $word = "";
    $description = "";
}
}

```

?>

4.9 Data View Pages

(mydata.php, viewdata.php, viewgenres.php, viewusers.php, viewgames.php)

All of these PHP files work similarly. On load, the PHP script checks to see whether a request is being made. If there is, the program will attempt to delete the specific record that was requested from the appropriate table in the database. If the row is successfully deleted, a message will be displayed along the top of the page. If there is an error while attempting to delete the record, an error message will be displayed.

The page then makes a database query retrieving all the data necessary to populate a list of items to display on the screen. The program runs through a 'while loop' and displays the appropriate records. Each row also includes a 'delete' and modify' link that attaches the ID of the record as a parameter. In this way, if a user chooses to modify or delete a record, the program will know which record to update. When 'delete' is selected, the page returns to itself and then deletes the record. When 'modify' is selected, the program navigates to the appropriate page and passed the record ID.

The mydata.php file is navigable by all user types and allows the user to view a list of all word/pairs they have added into the database. It displays the word, description, and genre. The viewdata.php file is like the mydata.php file but it is only available to administrators and allows them to view the entire set of word/description pairs that have been added by all users. The viewgenres.php file is available only to administrators and allows them to view all genre types in the database. It displays the name and description of the genre. This page also includes a link where the administrator can navigate to a page where they may add a new genre. The viewusers.php page is only available to administrators and displays a list of all users, displaying the username and email address. The viewgames.php page is only available to administrators and displays a list of all the games in the database. The display includes the title, description, and site address for the game.

4.10 Modify Data Pages

(modifyuserpair.php, modifyprofile.php, modifypair.php, modifygenre.php, modifyuser.php, modifygame.php)

All of these PHP files work similarly. On load, the page checks to see what parameter was passed as a request. It then populates the form on the page with data from the proper table in the database according to the parameter that it receives. If no parameter is passed, it displays a blank form and displays an error. The program then allows the user to update the information and click the update button. The page then reloads and checks to see whether the page is receiving a post. If it is, it accesses the database table and updates the information. If the database is successfully updated, a message is displayed at the top letting the user know that the database was updated successfully. Otherwise, the page displays an error message stating that the information could not be updated. At the top of each page is also a hyperlink, allowing the user to return to the page to view all the data of that table.

The modifyuserpair.php and modifyprofile.php files are accessible to all users while the other four files are only available to users with administrative privileges. The modifyprofile.php page displays the first name, last name, user name, password, and email address of the user. The modifyuser.php file includes the same information as the modifyprofile.php page but also includes list boxes where the administrator can change administrator privileges and whether the account is activated or not. The modifyuserpair.php file displays the word, description, and a list box of all available genres. The modifypair.php page is identical except that it includes the head_admin.php file to display links appropriate to administrators. The modifygenre.php file displays the name and description. The modifygame.php page displays the title, description, and site address of the game.

```
<?php
if(isset($_REQUEST['id']))
{
    $id=$_REQUEST['id'];
    $query= "SELECT * FROM question WHERE id='$id'";
    $result=mysql_query($query);
    if($result && $feed=mysql_fetch_object($result))
    {
        $id=$feed->id;
```

```

        $word=$feed->word;
        $description=$feed->description;
        $genre=$feed->genreid;
    }
}
else
{
    $id=NULL;
    $word="";
    $description="";
    $genre="";
    $msg =$msg." Wrong access. ID is null. <a href='mydata.php'>Try again</a><br>";
}

?>
<?php if (!empty($message)) {echo "<p class=\"message\">" . $message . "</p>";} ?>
<?php if (!empty($errors)) { display_errors($errors); } ?>

```

5. Conclusion

To reiterate, this document provides insight into the structural layout of the online self-learning software being developed by the Spring 2011, CNT 4104 Computer Network Programming students. It follows a standard client/ server model containing a web interface, server, and database where users can log in, insert data sets, and play games related to memorization. The functional requirements specifications designate the requirements necessary to support the site's accessibility, navigation, session states, and data management. Both the static and dynamic design diagrams and descriptions thoroughly incorporate these functional requirements and map how they fit together to package a complete software. The implementation section describes in detail, how the code work to create a complete user experience.

For future students who continue this project, there is still much to do regarding the database, error checking, and the graphical user interface. For the database, there currently exists no protection when a table row is deleted. If a row in one table is referenced in another table. It will be deleted without question, yet leave the other record that references it untouched. This should adjusted to prevent this from happening, or to also delete all records associated with it from the database. Also, the 'livegame' table rows expand rapidly. Every time a user clicks a link to play a game, a new instance is added. This is fine normally, because when a game is being continued, or is finished a record of the game is maintained for each user. However, should a user decide they no longer want to continue a game, and instead delete the record, a gap in the numbering system now will exist. A future student may want to create an additional table that stores the ID's of all deleted records. Then, when a new game is being clicked, if a record exists in deleted ID's table, the program could reuse one of these ID's in the 'livegame' table.

In terms of error checking, the program works correctly in most regards. It also ensures that there are no errors on log in and logout and protects against leaving a field blank. However, the program does not protect against leaving a field blank when a new record is being inserted or

updated in other areas of the site. This should be thoroughly investigated and fixed so that data sets are complete in the database.

Lastly, the graphical user interface is extremely simplified. To achieve the goals of this project, only minimal thought went into it and the focus was completely on functionality. To enhance the user experience, careful thought should go into developing a dynamic site that entertains the user.

References

- [1] Ambler, Scott. "UML 2 Sequence Diagrams." AgileModeling.com. 10 Feb 2011.
< <http://www.agilemodeling.com/artifacts/sequenceDiagram.htm>>.
- [2] Borysowich, Craig. "Observations from a Tech Architect: Enterprise Implementation Issues & Solutions." Toolbox.com. 25 May 2007. 10 Feb 2011.
< <http://it.toolbox.com/blogs/enterprise-solutions/using-structure-charts-16477>>.
- [3] "MLA Format for Websites." Roanstate.edu. 15 Feb 2011.
< <http://www.roanestate.edu/owl/MLA-Format4Websites.htm>>.
- [4] "SQL Tutorial." W3Schools.com. 12 April 2011.
< <http://www.w3schools.com/sql/default.asp>>.
- [5] " PHP Login and logout script and example." Plus2Net.com. 14 April 2011.
< http://www.plus2net.com/php_tutorial/php_login_logout.php>.
- [6] " PostgreSQL & PHP Tutorials – Deleting from your Database." DesignMagick.com. 13 April 2011. < <http://www.designmagick.com/article/12/Starting-Out/Deleting-Data> >.
- [7] " PHP Server Variables." PCTalkNet.com. 15 April 2011.
< <http://www.pctalknet.com/php/serverVariables.php> >.

Appendix A: Code

Public_html Folder

Addgame.php

```
<?php require_once("includes/session.php"); ?>
<?php require_once("includes/connection.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php confirm_logged_in(); ?>
<?php include("includes/head_admin.php"); ?>
<?php
    include_once("includes/form_functions.php");

    // START FORM PROCESSING
    if (isset($_POST['submit'])) { // Form has been submitted.
        $errors = array();

        // perform validations on the form data
        $required_fields = array('title', 'description', 'siteaddress');
        $errors = array_merge($errors,
                                check_required_fields($required_fields, $_POST));

        $fields_with_lengths = array('title' => 100, 'description' => 250,
                                      'siteaddress' => 100);
        $errors = array_merge($errors,
                                check_max_field_lengths($fields_with_lengths, $_POST));

        $title = trim(mysql_prep($_POST['title']));
        $description = trim(mysql_prep($_POST['description']));
        $siteaddress = trim(mysql_prep($_POST['siteaddress']));

        if ( empty($errors) ) {
            $query = "INSERT INTO game (title, description, siteaddress)
                        VALUES ('{$title}', '{$description}', '{$siteaddress}')";
            $result = mysql_query($query, $connection);
            if ($result) {
                $message = "<h3>The game has been successfully added.</h3>";
                $title = "";
                $description = "";
                $siteaddress = "";

            } else {
                $message = "The game could not be added.";
                $message .= "<br />" . mysql_error();
            }
        } else {
            if (count($errors) == 1) {
                $message = "There was 1 error in the form.";
            } else {
```

```

        $message = "There were " . count($errors) . " errors in the form.";
    }
}
} else { // Form has not been submitted.
    $title = "";
    $description = "";
    $siteaddress = "";
}

?>
<div id="main">

<h2>Add A New Game</h2>
<a href="viewgames.php">Return to List</a></br>

<table id="structure">
    <tr>
        <td id="page">

            <?php if (!empty($message)) {echo "<p class='message'>" . $message . "</p>";} ?>
            <?php if (!empty($errors)) { display_errors($errors); } ?>
            <form action="addgame.php" method="post">
                <table>
                    <tr>
                        <td>Title:</td>
                        <td><input type="text" name="title" maxlength="100"
                            value="<?php echo htmlentities($title); ?>"></td>
                    </tr>
                    <tr>
                        <td>Description:</td>
                        <td><textarea name="description" cols=40 rows=5 maxlength="250"
                            value="<?php echo htmlentities($description); ?>"
                        ></textarea></td>
                    </tr>
                    <tr>
                        <td>Site Address:</td>
                        <td><input type="text" name="siteaddress" maxlength="100"
                            value="<?php echo
                            htmlentities($siteaddress); ?>" ></td>
                    </tr>
                    <tr>
                        <td colspan="2"><input type="submit" name="submit" value="Add"
                        ></td>
                    </tr>
                </table>
            </form>
        </td>
    </tr>
</table>
<?php include("includes/footer.php"); ?>

```

Addgenre.php


```

<?php require_once("includes/session.php"); ?>
<?php require_once("includes/connection.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php confirm_logged_in(); ?>
<?php include("includes/head_admin.php"); ?>
<?php
    include_once("includes/form_functions.php");

    // START FORM PROCESSING
    if (isset($_POST['submit'])) { // Form has been submitted.
        $errors = array();

        // perform validations on the form data
        $required_fields = array('name');
        $errors = array_merge($errors, check_required_fields($required_fields, $_POST));

        $fields_with_lengths = array('name' => 100, 'description' => 250);
        $errors = array_merge($errors,

        check_max_field_lengths($fields_with_lengths, $_POST));

        $name = trim(mysql_prep($_POST['name']));
        $description = trim(mysql_prep($_POST['description']));

        if ( empty($errors) ) {
            $query = "INSERT INTO genre (name, description) VALUES ('{$name}',
                                '{$description}')";
            $result = mysql_query($query, $connection);
            if ($result) {
                $message = "<h3>The Genre has been successfully added.</h3>";
                $name = "";
                $description = "";

            } else {
                $message = "The Genre has not be added.";
                $message .= "<br />" . mysql_error();
            }
        } else {
            if (count($errors) == 1) {
                $message = "There was 1 error in the form.";
            } else {
                $message = "There were " . count($errors) . " errors in the form.";
            }
        }
    } else { // Form has not been submitted.
        $name = "";
        $description = "";
    }

?>

<div id="main">

    <h2>Add A New Genre</h2>
    <a href='viewgenres.php'>Return to List</a>

```

```

<table id="structure">
    <tr>
        <td id="page">

            <?php if (!empty($message)) {echo "<p class='message'>" . $message . "</p>";} ?>
            <?php if (!empty($errors)) { display_errors($errors); } ?>
            <form action="addgenre.php" method="post">
                <table>
                    <tr>
                        <td>Name:</td>
                        <td><input type="text" name="name" maxlength="100"

value="<?php echo htmlentities($name); ?>" ></td>
                    </tr>
                    <tr>
                        <td>Description:</td>
                        <td><textarea name="description" cols='40' rows='8'
maxlength="250" value="<?php echo htmlentities($description); ?>"
></textarea></td>
                    </tr>
                    <tr>
                        <td colspan="2"><input type="submit" name="submit" value="Add"
></td>
                    </tr>
                </table>
            </form>
        </td>
    </tr>
</table>
<?php include("includes/footer.php"); ?>

```

Continue.php

```

<?php require_once("includes/session.php"); ?>
<?php require_once("includes/connection.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php confirm_logged_in(); ?>
<?php include("includes/head.php"); ?>
<?php
    include_once("includes/form_functions.php");

?>
    <div id="main">

        <h2>Continue A Game!</h2>

        <table id="structure">
            <tr>
                <td id="page">
                    <div id="gamescontainer">
                        <ul id="gamesmenu">
<?php

```

```

if(isset($_REQUEST['id']))
{
    $id=$_REQUEST['id'];
    $query = "DELETE FROM livegame WHERE id=$id";
    $result=mysql_query($query);
    if ($result) {
        $message = "<h3>The Game you selected has been successfully deleted.</h3>";
    } else {
        $message = "The Game could not be deleted.";
        $message .= "<br />" . mysql_error();
    }
}
if (!empty($message)) {echo "<p class='message'>" . $message . "</p>";}
if (!empty($errors)) { display_errors($errors); }

$id=$_SESSION['user_id'];
$result = mysql_query("select g.id, g.title, g.siteaddress, lg.id, lg.userid, lg.numquestion,
    lg.numquestionsanswered, lg.numcorrect, lg.ended from game as g inner join
    livegame as lg on lg.gameid=g.id where lg.ended=0 && lg.userid='$id'");

$num=mysql_numrows($result);
$i=0;
while($i < $num)
{
    $lid=mysql_result($result, $i, "lg.id");
    $gid=mysql_result($result, $i, "g.id");
    $uid=mysql_result($result, $i, "lg.userid");
    $siteaddress=mysql_result($result, $i, "g.siteaddress");
    $title=mysql_result($result, $i, "g.title");
    $numquestions=mysql_result($result, $i, "lg.numquestion");
    $numquestionsanswered=mysql_result($result, $i, "lg.numquestionsanswered");
    $numcorrect=mysql_result($result, $i, "lg.numcorrect");

?>
<tr><b><td>
<?
    // checks to see if this is Andon's game and send appropriate variables
    if($gid==6) {
        echo '<a href="'.$siteaddress.'?uid='.$uid.'&id='.$lid.'"'>' ?> Continue
        <? echo $title ?></a></b></td><td></td><?
    }
    // otherwise sends the standard variables
    else {
        echo '<a href="'.$siteaddress.'?id='.$lid.'"'>' ?> Continue
        <? echo $title ?></a></b></td><td></td><?
    }?>
</td>
<?
    $percent=100;
    $d= ($numcorrect / $numquestions) * $percent;

?>
    <? echo $numquestionsanswered?> out of <? echo $numquestions ?> questions answered
    </td><td>Current Score: <? echo number_format ($d, 2); ?></td>
<b><td>

```

```

        <? echo '<a href="continue.php?id='.$lid.'"> ?> Delete Game</a></b></td>
</tr></b>
<?
$i++;
}
?>
        </td>
</tr>
</table>

        </ul>
</div>
        </td>
</tr>
</table>
<?php include("includes/footer.php"); ?>

```

Games.php

```

<?php require_once("includes/session.php"); ?>
<?php require_once("includes/connection.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php confirm_logged_in(); ?>
<?php include("includes/head.php"); ?>
<?php
    include_once("includes/form_functions.php");

?>
    <div id="main">

        <h2>Choose a Game to Play!</h2>

<table id="structure">
    <tr>
        <td id="page">
            <div id="gamescontainer">
                <ul id="gamesmenu">
                    <!-- <? echo '<a href="'.$siteaddress.'?id='.$_SESSION['user_id'].'">' ?> -->

<?php
if(isset($_REQUEST['id']))
{
    $uid = $_SESSION['user_id'];
    $id=$_REQUEST['id'];

    $date = date("Y-m-d H:i:s",time());
    $phpdate = strtotime( $date );

    // checks to see if this is Andon's game. If it is, it sends the userid.
    if($id==6)
    {
        $queryAndon = "Select siteaddress from game where id=$id";
    }
}

```

```

$resultAndon=mysql_query($queryAndon);
$gaddress=mysql_result($resultAndon, 0, "siteaddress");
if ($resultAndon) {
    echo '<META HTTP-EQUIV="Refresh" Content="0; URL='.$gaddress.'?uid='.$Suid.'">';
    exit;
}
}
$result = mysql_query("INSERT INTO livegame (userid, gameid, datetime, numquestion,
numquestionsanswered, ended, numcorrect)
VALUES ('$uid', $id, FROM_UNIXTIME($phpdate), 0, 0, 0, 0)");

$query2 = "Select siteaddress from game where id=$id";
$result2=mysql_query($query2);
$siteaddress=mysql_result($result2, 0, "siteaddress");

$query3 = "Select id, UNIX_TIMESTAMP(datetime) from livegame where
UNIX_TIMESTAMP(datetime)=$phpdate";
$result3=mysql_query($query3);
$lid=mysql_result($result3, 0, "id");

if ($result3) {
    echo '<META HTTP-EQUIV="Refresh" Content="0; URL='.$siteaddress.'?id='.$lid.'">';
    exit;
} else {
    $message = "There was a problem redirecting to the Game Site";
    $message .= "<br />" . mysql_error();
}
}
if (!empty($message)) { echo "<p class=\"message\">" . $message . "</p>";}
if (!empty($errors)) { display_errors($errors); }

echo "<table><tr><td><b>Title</b></td><td><b>Description</b></td></tr>";
$result1 = mysql_query("SELECT * FROM game ORDER BY title ASC");
$num=mysql_numrows($result1);
$i=0;
while($i < $num)
{
    $gid=mysql_result($result1, $i, "id");
    $title=mysql_result($result1, $i, "title");
    $description=mysql_result($result1, $i, "description");
    $siteaddress=mysql_result($result1, $i, "siteaddress");

?>
<tr><b><td>
<!-- <? echo '<a href="'.$siteaddress.'?id='.$_SESSION['user_id'].'">' ?> -->
<? echo '<a href="games.php?id='.$gid.'">' ?>
<? echo $title ?></a>

</td></b><td>
<? echo $description ?></td></tr>
<?
$i++;
}

```

```

?>
        </td>
    </tr>
</table>

        </ul>
    </div>
    </td>
</tr>
</table>
<?php include("includes/footer.php"); ?>

```

Index.php

```

<?php require_once("includes/session.php"); ?>
<?php require_once("includes/connection.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php include("includes/head.php"); ?>
<?php include("includes/welcome.php"); ?>
<?php include("includes/footer.php"); ?>

```

Inputdata.php

```

<?php require_once("includes/session.php"); ?>
<?php require_once("includes/connection.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php confirm_logged_in(); ?>
<?php include("includes/head.php"); ?>
<?php
    include_once("includes/form_functions.php");

    // START FORM PROCESSING
    if (isset($_POST['submit'])) { // Form has been submitted.
        $errors = array();

        // perform validations on the form data
        $required_fields = array('word', 'description');
        $errors = array_merge($errors, check_required_fields($required_fields, $_POST));

        $fields_with_lengths = array('word' => 100, 'description' => 500);
        $errors = array_merge($errors,

        check_max_field_lengths($fields_with_lengths, $_POST));

        $word = trim(mysql_prep($_POST['word']));
        $description = trim(mysql_prep($_POST['description']));
        $genre = trim(mysql_prep($_POST['genre']));
        if ( empty($errors) ) {
            $query = "INSERT INTO question (word, description, genreid, addedby)
                VALUES ('{$word}', '{$description}', '{$genre}', '{$_SESSION['user_id']}')";

```

```

$result = mysql_query($query, $connection);
if ($result) {
    $message = "<h3>The word pair has been successfully added.</h3>";
    $word = "";
    $description = "";
    $query = "UPDATE users SET rank=rank+1 WHERE
id='{$_SESSION['user_id']}'";
    $result = mysql_query($query, $connection);

    } else {
        $message = "The word pair could not be added.";
        $message .= "<br />" . mysql_error();
    }
} else {
    if (count($errors) == 1) {
        $message = "There was 1 error in the form.";
    } else {
        $message = "There were " . count($errors) . " errors in the form.";
    }
}
} else { // Form has not been submitted.
    $word = "";
    $description = "";
}

?>
<div id="main">

<h2>Input word description pairs.</h2>
<a href='mydata.php'>View List</a>

<table id="structure">
    <tr>
        <td id="page">

            <?php if (!empty($message)) {echo "<p class='message'>" . $message . "</p>";} ?>
            <?php if (!empty($errors)) { display_errors($errors); } ?>
            <form action="inputdata.php" method="post">
                <table>
                    <tr>
                        <td>Word:</td>
                        <td><input type="text" name="word" maxlength="100"
value="<?php
echo htmlentities($word); ?>"></td>
                    </tr>
                    <tr>
                        <td>Description:</td>
                        <td><textarea name="description" cols='40' rows='8'
maxlength="500" value="<?php echo
htmlentities($description); ?>">
                    </td>
                    </tr>
                    <tr>
                        <td>Genre: </td><td>
                        <?php
$query2="SELECT id, name from genre order by name";

```

```

$result2=mysql_query($query2);

echo "<select name=genre value=''>Name</option>";
// printing the list box select command

while($nt=mysql_fetch_array($result2)){//Array or records stored in $nt
if($nt[id]==1) {
    echo "<option value=$nt[id] selected='selected'>$nt[name]</option>";
}
else {
    /* Option values are added by looping through the array */

    echo "<option value=$nt[id]>$nt[name]</option>";
}
/* Option values are added by looping through the array */

}
echo "</select>";// Closing of list box
?>
</tr><tr><td></td>

                                <td colspan="2">
                                    <input type="submit" name="submit" value="Add" >
                                </td>
                            </tr>
                        </table>
                    </form>
                </td>
            </tr>
        </table>
<?php include("includes/footer.php"); ?>

```

Login.php

```

<?php require_once("includes/session.php"); ?>
<?php require_once("includes/connection.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php include("includes/head.php"); ?>
<?php

    if (logged_in()) {
        redirect_to("staff.php");
    }

    include_once("includes/form_functions.php");

    // START FORM PROCESSING
    if (isset($_POST['submit'])) { // Form has been submitted.
        $errors = array();

        // perform validations on the form data
    }

```



```

$required_fields = array('username', 'password');
$errors = array_merge($errors, check_required_fields($required_fields, $_POST));

$fields_with_lengths = array('username' => 30, 'password' => 30);
$errors = array_merge($errors,

check_max_field_lengths($fields_with_lengths, $_POST));

$username = trim(mysql_prep($_POST['username']));
$password = trim(mysql_prep($_POST['password']));

if ( empty($errors) ) {
    // Check database to see if username and the hashed password exist there.
    $query = "SELECT id, username ";
    $query .= "FROM users ";
    $query .= "WHERE username = '{ $username }' ";
    $query .= "AND hashed_password = '{ $password }' ";
    $query .= "LIMIT 1";
    $result_set = mysql_query($query);
    confirm_query($result_set);
    if (mysql_num_rows($result_set) == 1) {
        // username/password authenticated
        // and only 1 match
        $found_user = mysql_fetch_array($result_set);
        $_SESSION['id']=session_id();
        $_SESSION['user_id'] = $found_user['id'];
        $_SESSION['username'] = $found_user['username'];

        redirect_to("~/~slaforge/index.php");
    } else {
        // username/password combo was not found in the database
        $message = "Username/password combination incorrect.<br />
        Please make sure your caps lock key is off and try again.";
    }
} else {
    if (count($errors) == 1) {
        $message = "There was 1 error in the form.";
    } else {
        $message = "There were " . count($errors) . " errors in the form.";
    }
}

} else { // Form has not been submitted.
    if (isset($_GET['logout']) && $_GET['logout'] == 1) {
        $message = "You are now logged out.";
    }
    $username = "";
    $password = "";
}

?>

<div id="main">

<h2>Welcome to the FGCU Online Learning Game Website</h2>

<table id="structure">

```

```

<tr>
  <td id="page">
    <?php if (!empty($message)) {echo "<p class='message'>" . $message . "</p>";} ?>
    <?php if (!empty($errors)) { display_errors($errors); } ?>
    <form action="login.php" method="post">
      <table>
        <tr>
          <td>Username:</td>
          <td><input type="text" name="username" maxlength="30"
            value="<?php echo
htmlentities($username); ?>"></td>
        </tr>
        <tr>
          <td>Password:</td>
          <td><input type="password" name="password" maxlength="30"
            value="<?php echo
htmlentities($password); ?>"></td>
        </tr>
        <tr>
          <td colspan="2"><input type="submit" name="submit"
            value="Login"></td>
          <td><a href="new_user.php">Create An Account</a></td>
        </tr>
      </table>
    </form>
  </td>
</tr>
</table>
<?php include("includes/footer.php"); ?>

```

Logout.php

```

<?php require_once("includes/functions.php"); ?>
<?php
    // Four steps to closing a session
    // (i.e. logging out)

    // 1. Find the session
    session_start();

    // 2. Unset all the session variables
    $_SESSION = array();

    // 3. Destroy the session cookie
    if(isset($_COOKIE[session_name()])) {
        setcookie(session_name(), "", time()-42000, '/');
    }

    // 4. Destroy the session
    session_destroy();

    redirect_to("login.php?logout=1");
?>

```

Modifygame.php

```
<?php require_once("includes/session.php"); ?>
<?php require_once("includes/connection.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php confirm_logged_in(); ?>
<?php include("includes/head_admin.php"); ?>
<?php
    include_once("includes/form_functions.php");

?>

<div id="main">

    <h2>Modify Game</h2> <a href='viewgames.php'>Back to list</a>

<?php
if (isset($_POST['submit'])) { // Form has been submitted.
    $errors = array();

    // perform validations on the form data
    $required_fields = array('title', 'description', 'siteaddress');
    $errors = array_merge($errors,
                           check_required_fields($required_fields, $_POST));

    $fields_with_lengths =
        array('title' => 100, 'description' => 250, 'siteaddress' => 150);
    $errors = array_merge($errors,
                           check_max_field_lengths($fields_with_lengths,
$_POST));

    $id=$_POST['id'];
    $title=$_POST['title'];
    $description=$_POST['description'];
    $siteaddress=$_POST['siteaddress'];

    if ( empty($errors) ) {
        $query = "UPDATE game SET title='$title', description='$description',
                                                                    siteaddress='$siteaddress'
WHERE id='$id'";

        $result = mysql_query($query, $connection);
        if ($result) {
            $message = "<h3>The Game has been successfully updated.</h3>";
        } else {
            $message = "The Game could not be updated.";
            $message .= "<br />" . mysql_error();
        }
    } else {
        if (count($errors) == 1) {
            $message = "There was 1 error in the form.";
        } else {
            $message = "There were " . count($errors) . " errors in the form.";
        }
    }
}
```

```

        } else { // Form has not been submitted.
            $name = "";
            $description = "";
        }

if(isset($_REQUEST['id']))
{
    $id=$_REQUEST['id'];
    $query= "SELECT * FROM game WHERE id='$id'";
    $result=mysql_query($query);
    if($result && $feed=mysql_fetch_object($result))
    {
        $id=$feed->id;
        $title=$feed->title;
        $description=$feed->description;
        $siteaddress=$feed->siteaddress;
    }
}
else
{
    $id=NULL;
    $title="";
    $description="";
    $siteaddress="";
    $msg =$msg." Wrong access. ID is null. <a href='viewgames.php'>Try again</a><br>";
}
?>
<?php if (!empty($message)) {echo "<p class='message'>" . $message . "</p>";} ?>
<?php if (!empty($errors)) { display_errors($errors); } ?>

<form name="updated" action="modifygame.php" method="post">
<?php echo "<font color='red'>$msg</font>";?>
<table border=1>
<tr><td colspan="2" align="center">Update and save</td></tr>
<tr>
<?php
echo"<td><input type='hidden' name='id' value='$id'></td>";
?>
</tr><tr>
<td>Title: </td>
<?php
echo"<td><input type='text' name='title' value='$title'></td>";
?>
</tr><tr>
<td>Description: </td>
<?php
echo"<td><textarea name='description' cols='40' rows='8'>$description";
echo "</textarea></td>";
?>
</tr><tr>
<td>Site Address: </td>
<?php
echo"<td><input type='text' name='siteaddress' value='$siteaddress'></td>";
?>
</tr><tr>
<td colspan="2" align="center"><input type="submit" name="submit"

```

```

        value="Save"></td>
</tr>
</table>
</form>

<?php include("includes/footer.php"); ?>

```

Modifygenre.php

```

<?php require_once("includes/session.php"); ?>
<?php require_once("includes/connection.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php confirm_logged_in(); ?>
<?php include("includes/head_admin.php"); ?>
<?php
    include_once("includes/form_functions.php");

?>

<div id="main">

    <h2>Modify Genre</h2> <a href='viewgenres.php'>Back to list</a>

<?php
if (isset($_POST['submit'])) { // Form has been submitted.
    $errors = array();

    // perform validations on the form data
    $required_fields = array('name', 'description');
    $errors = array_merge($errors,
        check_required_fields($required_fields, $_POST));
    $fields_with_lengths = array('name' => 100, 'description' => 250);
    $errors = array_merge($errors,
        check_max_field_lengths($fields_with_lengths, $_POST));

    $id=$_POST['id'];
    $name=$_POST['name'];
    $description=$_POST['description'];

    if ( empty($errors) ) {
        $query = "UPDATE genre SET name='$name', description='$description'

        WHERE id='$id'";
        $result = mysql_query($query, $connection);
        if ($result) {
            $message = "<h3>The Genre has been successfully updated.</h3>";
        } else {
            $message = "The Genre could not be updated.";
            $message .= "<br />" . mysql_error();
        }
    } else {
        if (count($errors) == 1) {
            $message = "There was 1 error in the form.";

```

```

        } else {
            $message = "There were " . count($errors) . " errors in the form.";
        }
    }
} else { // Form has not been submitted.
    $name = "";
    $description = "";
}

?>

<?php
if(isset($_REQUEST['id']))
{
    $id=$_REQUEST['id'];
    $query= "SELECT * FROM genre WHERE id='$id'";
    $result=mysql_query($query);
    if($result && $feed=mysql_fetch_object($result))
    {
        $id=$feed->id;
        $name=$feed->name;
        $description=$feed->description;
    }
}
else
{
    $id=NULL;
    $name="";
    $description="";
    $msg =$msg." Wrong access. ID is null. <a href='viewgenres.php'>Try again</a><br>";
}
?>

<?php if (!empty($message)) {echo "<p class='message'>" . $message . "</p>";} ?>
<?php if (!empty($errors)) { display_errors($errors); } ?>

<form name="updated" action="modifygenre.php" method="post">
<?php echo "<font color='red'>$msg</font>";?>
<table border=1>
<tr><td colspan="2" align="center">Update and save</td></tr>
<tr>
<td><input type='hidden' name='id' value='$id'></td>";
?>
</tr><tr>
<td>Name: </td>
<td><input type='text' name='name' value='$name'></td>";
?>
</tr><tr>
<td>Description: </td>
<td><input type='text' name='description' cols='40' rows='8'>$description</td>";
?>
</tr><tr>
<td colspan="2" align="center"><input type="submit" name="submit" value="Save"></td>
</tr>
</table>
</form>

```

```
<?php include("includes/footer.php"); ?>
```

Modifypair.php

```
<?php require_once("includes/session.php"); ?>
<?php require_once("includes/connection.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php confirm_logged_in(); ?>
<?php include("includes/head_admin.php"); ?>
<?php
    include_once("includes/form_functions.php");

?>
    <div id="main">

        <h2>Modify Word/Description Pairs</h2> <a href='viewdata.php'>Back to list</a>

<?php
if (isset($_POST['submit'])) { // Form has been submitted.
    $errors = array();

    // perform validations on the form data
    $required_fields = array('word', 'description');
    $errors = array_merge($errors,
                           check_required_fields($required_fields, $_POST));
    $fields_with_lengths = array('word' => 100, 'description' => 500);
    $errors = array_merge($errors,
                           check_max_field_lengths($fields_with_lengths, $_POST));

    $id=$_POST['id'];
    $word=$_POST['word'];
    $genreid=$_POST['genre'];
    $description=$_POST['description'];

    if ( empty($errors) ) {
        $query = "UPDATE question SET word='$word', genreid='$genreid',
                                description='$description' WHERE
id='$id'";

        $result = mysql_query($query, $connection);
        if ($result) {
            $message = "<h3>The Word/Description Pair has been
successfully updated.</h3>";
        } else {
            $message = "The Word/Description Pair could not be updated.";
            $message .= "<br />" . mysql_error();
        }
    } else {
        if (count($errors) == 1) {
            $message = "There was 1 error in the form.";
        } else {
            $message = "There were " . count($errors) . " errors in the form.";
        }
    }
}
```

```

        }
    } else { // Form has not been submitted.
        $word = "";
        $description = "";
    }
?>

<?php
if(isset($_REQUEST['id']))
{
    $qid=$_REQUEST['id'];
    $query= "SELECT * FROM question WHERE id='$qid'";
    $result=mysql_query($query);
    if($result && $feed=mysql_fetch_object($result))
    {
        $qid=$feed->id;
        $word=$feed->word;
        $description=$feed->description;
        $genre=$feed->genreid;
    }
}
else
{
    $qid=NULL;
    $word="";
    $description="";
    $genre="";
    $msg =$msg." Wrong access. ID is null. <a href='viewdata.php'>Try again</a><br>";
}

?>

<?php if (!empty($message)) {echo "<p class='message'>" . $message . "</p>";} ?>
<?php if (!empty($errors)) { display_errors($errors); } ?>

<form name="updated" action="modifypair.php" method="post">
<?php echo "<font color='red'>$msg</font>";?>
<table border=1>
<tr><td colspan="2" align="center">Update and save</td></tr>
<tr>
<?php
echo"<td><input type='hidden' name='id' value='$qid'></td>";
?>
</tr><tr>
<td>Name: </td>
<?php
echo"<td><input type='text' name='word' value='$word'></td>";
?>
</tr><tr>
<td>Description: </td>
<?php
echo"<td><textarea name='description' cols='40' rows='8'>$description</textarea></td>";
?>
</tr><tr>
<td>Genre: </td><td>
<?php
$query2="SELECT id, name from genre order by name";

```



```

$result2=mysql_query($query2);

echo "<select name=genre value=>Name</option>";
// printing the list box select command

while($nt=mysql_fetch_array($result2)){//Array or records stored in $nt
if($genre==$nt[id]) {
    echo "<option value=$nt[id] selected='selected'>$nt[name]</option>";
}
else {
    /* Option values are added by looping through the array */

    echo "<option value=$nt[id]>$nt[name]</option>";
}
/* Option values are added by looping through the array */

}
echo "</select>";// Closing of list box
?>
</tr><tr>

<td colspan="2" align="center"><input type="submit" name="submit" value="Save"></td>
</tr>
</table>
</form>
<?php include("includes/footer.php"); ?>

```

Modifyprofile.php

```

<?php require_once("includes/session.php"); ?>
<?php require_once("includes/connection.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php confirm_logged_in(); ?>
<?php include("includes/head.php"); ?>
<?php
    include_once("includes/form_functions.php");
    ?>
    <div id="main">
        <h2>Update Profile</h2>
        <a href='index.php'>Return to Home</a>

    <?php
if (isset($_POST['submit'])) { // Form has been submitted.
    $errors = array();

    // perform validations on the form data
    $required_fields = array('firstname', 'lastname', 'username', 'password',
                                'email');
    $errors = array_merge($errors, check_required_fields($required_fields, $_POST));
    $fields_with_lengths = array('firstname' => 100, 'lastname' => 100,

```

```

'username' => 100, 'password' => 40, email
=> 100);
$errors = array_merge($errors,
    check_max_field_lengths($fields_with_lengths,
$_POST));
$id = $_SESSION['user_id'];
$firstname = trim(mysql_prep($_POST['firstname']));
$lastname = trim(mysql_prep($_POST['lastname']));
$username = trim(mysql_prep($_POST['username']));
$password = trim(mysql_prep($_POST['password']));
$email = trim(mysql_prep($_POST['email']));

if ( empty($errors) ) {
    $query = "update users set firstname='$firstname', lastname='$lastname',
        username='$username', hashed_password='$password',
        email='$email' where id = '$id'";
    $result = mysql_query($query, $connection);
    if ($result) {
        $message = "<h3>Your profile has been successfully updated.</h3>";
        $_SESSION['username'] = $username;

    } else {
        $message = "Your profile could not be updated.";
        $message .= "<br />" . mysql_error();
    }
} else {
    if (count($errors) == 1) {
        $message = "There was 1 error in the form.";
    } else {
        $message = "There were " . count($errors) . " errors in the form.";
    }
}
} else { // Form has not been submitted.
    $firstname = "";
    $lastname = "";
    $username = "";
    $password = "";
    $email = "";
}
?>

```

```

<?php
// START FORM PROCESSING
if(isset($_REQUEST['id']))
{
    $id=$_REQUEST['id'];
    $query= "SELECT * FROM users WHERE id='$id'";
    $result=mysql_query($query);
    if($result && $feed=mysql_fetch_object($result))
    {
        $uid=$feed->id;
        $firstname=$feed->firstname;
        $lastname=$feed->lastname;
        $username=$feed->username;
        $password=$feed->hashed_password;
    }
}

```

```

        $email=$feed->email;    }
    }
    ?>

<table id="structure">
    <tr>
        <td id="page">
            <?php if (!empty($message)) {echo "<p class=\"message\">" . $message . "</p>";} ?>
            <?php if (!empty($errors)) { display_errors($errors); } ?>
            <form name="updated" action="modifyprofile.php" method="post">
                <table>
                    <tr>
                        <td>First Name:</td>
                        <td><?php
                            echo "<input type='text' name='firstname' maxlength='100'";
                            echo " value='$firstname'>";
                        ?>
                    </td>
                    </tr>
                    <tr>
                        <td>Last Name:</td>
                        <td><?php
                            echo "<td><input type='text' name='lastname'
                                maxlength='100'";
                                echo " value='$lastname'></td>";
                            ?>
                        </td>
                    </tr>
                    <tr>
                        <td>User Name:</td>
                        <td><?php
                            echo "<td><input type='text' name='username'
                                maxlength='100'";
                                echo " value='$username'></td>";
                            ?>
                        </td>
                    </tr>
                    <tr>
                        <td>Password:</td>
                        <td><?php
                            echo " <td><input type='text' name='password'
                                maxlength='40'";
                                echo " value='$password'></td>";
                            ?>
                        </td>
                    </tr>
                    <tr>
                        <td>Email:</td>
                        <td><?php
                            echo "<td><input type='text' name='email' maxlength='100'";
                            echo " value='$email'></td></tr><td></td>";
                            ?>
                        </td>
                    </tr>
                    <tr>
                        <td colspan="2"><input type="submit" name="submit"
                            id="submit" value="Update"></td>
                    </tr>
                </table>
            </form>
        </td>
    </tr>
</table>

```

```

                </table>
            </form>
        </td>
    </tr>
</table>
<?php include("includes/footer.php"); ?>

```

Modifyuser.php

```

<?php require_once("includes/session.php"); ?>
<?php require_once("includes/connection.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php confirm_logged_in(); ?>
<?php include("includes/head_admin.php"); ?>
<?php
    include_once("includes/form_functions.php");

?>
    <div id="main">

        <h2>Modify Users</h2> <a href='viewusers.php'>Back to list</a>

<?php
if (isset($_POST['submit'])) { // Form has been submitted.
    $errors = array();

    // perform validations on the form data
    $required_fields = array('id', 'firstname', 'lastname', 'username',
                             'password', 'rank', 'email', 'active', 'admin');
    $errors = array_merge($errors,
                           check_required_fields($required_fields,
$_POST));
    $fields_with_lengths = array('firstname' => 100, 'lastname' => 100,
                                'username' => 100, 'password' => 40, email
=> 100);
    $errors = array_merge($errors,
                           check_max_field_lengths($fields_with_lengths,
$_POST));

    $id= $_POST['id'];
    $firstname=trim(mysql_prep($_POST['firstname']));
    $lastname = trim(mysql_prep($_POST['lastname']));
    $username = trim(mysql_prep($_POST['username']));
    $password = trim(mysql_prep($_POST['password']));
    $rank = trim(mysql_prep($_POST['rank']));
    $email = trim(mysql_prep($_POST['email']));
    $active = trim(mysql_prep($_POST['active']));
    $admin=trim(mysql_prep($_POST['admin']));

    if ( empty($errors) ) {
        $query = "UPDATE users SET firstname='$firstname', lastname='$lastname',
                                username='$username', hashed_password='$password', rank='$rank',
                                email='$email', active='$active', admin='$admin' WHERE id='$id'";

```

```

        $result = mysql_query($query, $connection);
        if ($result) {
            $message = "<h3>The profile has been successfully updated.</h3>";
        } else {
            $message = "The profile could not be updated.";
            $message .= "<br />" . mysql_error();
        }
    } else {
        if (count($errors) == 1) {
            $message = "There was 1 error in the form.";
        } else {
            $message = "There were " . count($errors) . " errors in the form.";
        }
    }
} else { // Form has not been submitted.
    $firstname = "";
    $lastname = "";
    $username = "";
    $password = "";
    $email = "";
}

?>

<?php

if(isset($_REQUEST['id']))
{
    $id=$_REQUEST['id'];
    $query= "SELECT * FROM users WHERE id='$id'";
    $result=mysql_query($query);
    if($result && $feed=mysql_fetch_object($result))
    {
        $uid=$feed->id;
        $firstname=$feed->firstname;
        $lastname=$feed->lastname;
        $username=$feed->username;
        $password=$feed->hashed_password;
        $rank=$feed->rank;
        $email=$feed->email;
        $active=$feed->active;
        $admin=$feed->admin;
    }
}
else
{
    $uid=NULL;
    $firstname="";
    $lastname="";
    $username="";
    $password="";
    $rank="";
    $email="";
    $active="";
    $admin="";
    $msg = $msg." Wrong access. ID is null. <a href='viewusers.php'>Try again</a><br>";
}

```

```

}
?>
<?php if (!empty($message)) {echo "<p class='message'>" . $message . "</p>";} ?>
<?php if (!empty($errors)) { display_errors($errors); } ?>

<form name="updated" action="modifyuser.php" method="post">
<?php echo "<font color='red'>$msg</font>";?>
<table border=1>
<tr><td colspan="2" align="center">Update and save</td></tr>
<tr>
<?php
echo"<td><input type='hidden' name='id' value='$uid'></td>";
?>
</tr><tr>
<td>First Name: </td>
<?php
echo"<td><input type='text' name='firstname' value='$firstname'></td>";
?>
</tr><tr>
<td>Last Name: </td>
<?php
echo"<td><input type='text' name='lastname' value='$lastname'></td>";
?>
</tr><tr>
<td>User Name: </td>
<?php
echo"<td><input type='text' name='username' value='$username'></td>";
?>
</tr><tr>
<td>Password: </td>
<?php
echo"<td><input type='text' name='password' value='$password'></td>";
?>
</tr><tr>
<td>Rank: </td>
<?php
echo"<td><input type='text' name='rank' value='$rank'></td>";
?>
</tr><tr>
<td>Email: </td>
<?php
echo"<td><input type='text' name='email' value='$email'></td>";
?>
</tr><tr>
<td>Activated: </td><td>
<?php
$query2="SELECT id, active from users where id='$id'";
$result2=mysql_query($query2);
$out=mysql_result($result2, 0, "active");

echo "<select name=active value='>Name</option>";
// printing the list box select command
$e=0;
if($out==$e) {
    echo "<option value=0 selected='selected'>0</option>";
    echo "<option value=1>1</option>";
}

```

```

    }
    else {
        echo "<option value=0>0</option>";
        echo "<option value=1 selected='selected'>1</option>";
    }
    echo "</select>"; // Closing of list box
    ?>
</tr><tr>
<td>Admin Priveleges: </td><td>
<?php
$query3="SELECT id, admin from users where id='$id'";
$result3=mysql_query($query3);
$out2=mysql_result($result3, 0, "admin");

echo "<select name=admin value='>Name</option>";
// printing the list box select command

if($out2==$e) {
    echo "<option value=0 selected='selected'>0</option>";
    echo "<option value=1>1</option>";
}
else {
    echo "<option value=0>0</option>";
    echo "<option value=1 selected='selected'>1</option>";
}
echo "</select>"; // Closing of list box
?>
</tr><tr>
<td colspan="2" align="center"><input type="submit" name="submit" id="submit"

        value="Update"></td>
</tr>
</table>
</form>

<?php include("includes/footer.php"); ?>

```

Modifyuserpair.php

```

<?php require_once("includes/session.php"); ?>
<?php require_once("includes/connection.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php confirm_logged_in(); ?>
<?php include("includes/head.php"); ?>
<?php
    include_once("includes/form_functions.php");

?>

<div id="main">

    <h2>Modify Word/Description Pairs</h2> <a href='mydata.php'>Back to list</a>

<?php

```

```

if (isset($_POST['submit'])) { // Form has been submitted.
    $errors = array();

    // perform validations on the form data
    $required_fields = array('word', 'description');
    $errors = array_merge($errors,
                           check_required_fields($required_fields, $_POST));
    $fields_with_lengths = array('word' => 100, 'description' => 500);
    $errors = array_merge($errors,
                           check_max_field_lengths($fields_with_lengths, $_POST));

    $id=$_POST['id'];
    $word=$_POST['word'];
    $genreid=$_POST['genre'];
    $description=$_POST['description'];

    if ( empty($errors) ) {
        $query = "UPDATE question SET word='$word', genreid='$genreid',
                                description='$description' WHERE id='$id'";
        $result = mysql_query($query, $connection);
        if ($result) {
            $message = "<h3>Your Word/Description Pair has been
successfully updated.</h3>";
        } else {
            $message = "Your Word/Description Pair could not be updated.";
            $message .= "<br />" . mysql_error();
        }
    } else {
        if (count($errors) == 1) {
            $message = "There was 1 error in the form.";
        } else {
            $message = "There were " . count($errors) . " errors in the form.";
        }
    }
} else { // Form has not been submitted.
    $word = "";
    $description = "";
}
}

```

?>

```

<?php
if(isset($_REQUEST['id']))
{
    $id=$_REQUEST['id'];
    $query= "SELECT * FROM question WHERE id='$id'";
    $result=mysql_query($query);
    if($result && $feed=mysql_fetch_object($result))
    {
        $id=$feed->id;
        $word=$feed->word;
        $description=$feed->description;
        $genre=$feed->genreid;
    }
}
else

```



```

{
    $id=NULL;
    $word="";
    $description="";
    $genre="";
    $msg =$msg." Wrong access. ID is null. <a href='mydata.php'>Try again</a><br>";
}

?>
<?php if (!empty($message)) {echo "<p class='message'>" . $message . "</p>";} ?>
<?php if (!empty($errors)) { display_errors($errors); } ?>

<form name="updated" action="modifyuserpair.php" method="post">
<?php echo "<font color='red'>$msg</font>";?>
<table border=1>
<tr><td colspan="2" align="center">Update and save</td></tr>
<tr>
<?php
echo"<td><input type='hidden' name='id' value='$id'></td>";
?>
</tr><tr>
<td>Name: </td>
<?php
echo"<td><input type='text' name='word' value='$word'></td>";
?>
</tr><tr>
<td>Description: </td>
<?php
echo"<td><textarea name='description' cols='40' rows='8'>$description</textarea></td>";
?>
</tr><tr>
<td>Genre: </td><td>
<?php
$query2="SELECT id, name from genre order by name";
$result2=mysql_query($query2);

echo "<select name=genre value='>Name</option>";
// printing the list box select command

while($nt=mysql_fetch_array($result2)){//Array or records stored in $nt
if($genre==$nt[id]) {
    echo "<option value=$nt[id] selected='selected'>$nt[name]</option>";
}
else {
    /* Option values are added by looping through the array */

    echo "<option value=$nt[id]>$nt[name]</option>";
}
}
/* Option values are added by looping through the array */

}
echo "</select>";// Closing of list box
?>
</tr><tr>
<td colspan="2" align="center"><input type="submit" name="submit" value="Save"></td>

```

```

</tr>
</table>
</form>
<?php include("includes/footer.php"); ?>

```

Mydata.php

```

<?php require_once("includes/session.php"); ?>
<?php require_once("includes/connection.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php confirm_logged_in(); ?>
<?php include("includes/head.php"); ?>
<?php
    include_once("includes/form_functions.php");

?>
    <div id="main">

        <h2>View, Modify, and Delete Word/Description Pairs</h2>

<?php
if(isset($_REQUEST['id']))
{
    $id=$_REQUEST['id'];
    $query = "DELETE FROM question WHERE id=$id";
    $result=mysql_query($query);
    if ($result) {
        $message = "<h3>The Word/Description Pair has been successfully deleted.</h3>";
    } else {
        $message = "The Word/Description Pair could not be deleted.";
        $message .= "<br />" . mysql_error();
    }
}
if (!empty($message)) { echo "<p class='message'>" . $message . "</p>"; }
if (!empty($errors)) { display_errors($errors); }
echo "<table><tr><td><b>Modify</b></td><td><b>Delete</b></td><td><b>Word</b></td>";
echo "<td><b>Description</b></td><td><b>Genre</b></td></tr>";
$id=$_SESSION['user_id'];
$query="select q.id, q.word, q.genreid, q.addedby, q.description, g.id, g.name, u.id
        from question as q inner join genre as g on q.genreid=g.id inner
        join users as u on q.addedby=u.id where q.addedby=$id ORDER BY word asc";
$result = mysql_query($query);

$num=mysql_numrows($result);
$i=0;
while($i < $num)
{
    $qid=mysql_result($result, $i, "q.id");
    $word=mysql_result($result, $i, "q.word");
    $description=mysql_result($result, $i, "q.description");
    $genre=mysql_result($result, $i, "g.name");

?>

```

```

<tr><b><td>

<? echo '<a href="modifyuserpair.php?id='.$_GET['id'].'>Modify</a>'; ?></td><td>
<? echo '<a href="mydata.php?id='.$_GET['id'].'>Delete</a>'; ?></td><td>
<? echo $word ?></td></b><td>
    <? echo $description ?></td><td>
    <? echo $genre ?></td></tr>
<?
$i++;
}
?>

        </td>

    </tr>
</table>
<?php include("includes/footer.php"); ?>

```

New_user.php

```

<?php require_once("includes/session.php"); ?>
<?php require_once("includes/connection.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php include("includes/head.php"); ?>

<?php
    include_once("includes/form_functions.php");

    // START FORM PROCESSING
    if (isset($_POST['submit'])) { // Form has been submitted.
        $errors = array();

        // perform validations on the form data
        $required_fields = array('firstname', 'lastname', 'username',
            'password', 'email');
        $errors = array_merge($errors,
                                check_required_fields($required_fields,
$_POST));

        $fields_with_lengths = array('firstname' => 100, 'lastname' => 100,
                                     'username' => 100, 'password' => 40, 'email' => 100);
        $errors = array_merge($errors,
                                check_max_field_lengths($fields_with_lengths, $_POST));

        $firstname = trim(mysql_prep($_POST['firstname']));
        $lastname = trim(mysql_prep($_POST['lastname']));
        $username = trim(mysql_prep($_POST['username']));
        $password = trim(mysql_prep($_POST['password']));
        $email = trim(mysql_prep($_POST['email']));

        if ( empty($errors) ) {
            $query = "INSERT INTO users ( firstname, lastname, username, hashed_password,
rank, email, active, admin) VALUES ('{$firstname}', '{$lastname}',
 '{$username}', '{$password}', 0, '{$email}', 1, 0)";

```

```

$result = mysql_query($query, $connection);
if ($result) {

    $message = "Your account was successfully created.";
    $firstname = "";
    $lastname = "";
    $username = "";
    $password = "";
    $email = "";

} else {
    $message = "Your account could not be created.";
    $message .= "<br />" . mysql_error();
}
} else {
    if (count($errors) == 1) {
        $message = "There was 1 error in the form.";
    } else {
        $message = "There were " . count($errors) . " errors in the form.";
    }
}
} else { // Form has not been submitted.
    $firstname = "";
    $lastname = "";
    $username = "";
    $password = "";
    $email = "";
}
?>

<div id="main">

<h2>Create A New Account</h2><a href="login.php">Return to Log In</a></br>

<p>
<table id="structure">
    <tr>
        <td id="page">
            <?php if (!empty($message)) {echo "<p class='message'>" . $message . "</p>";} ?>
            <?php if (!empty($errors)) { display_errors($errors); } ?>
            <form action="new_user.php" method="post">
                <table>
                    <tr>
                        <td>First Name:</td>
                        <td><input type="text" name="firstname" maxlength="100"
                            value="<?php echo htmlentities($firstname);
?>"></td>

                    </tr>
                    <tr>
                        <td>Last Name:</td>
                        <td><input type="text" name="lastname" maxlength="100"
                            value="<?php echo htmlentities($lastname);
?>"></td>

                    </tr>
                    <tr>
                        <td>Username:</td>

```

```

                                <td><input type="text" name="username" maxlength="100"
                                    value="<?php echo
htmlentities($username); ?>"></td>
                                </tr>
                                <tr>
                                <td>Password:</td>
                                <td><input type="password" name="password" maxlength="40"
                                    value="<?php echo
htmlentities($password); ?>"></td>
                                </tr>
                                <tr>
                                <td>Email:</td>
                                <td><input type="text" name="email" maxlength="150"
                                    value="<?php echo
htmlentities($email); ?>"></td>
                                </tr>
                                <tr>
                                <td colspan="2"><input type="submit" name="submit" value="Create
user"></td>
                                </tr>
                                </table>
                                </form>
                                </td>
                                </tr>
                                </table>
                                </p>
                                <?php include("includes/footer.php"); ?>

```

Previous.php

```

<?php require_once("includes/session.php"); ?>
<?php require_once("includes/connection.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php confirm_logged_in(); ?>
<?php include("includes/head.php"); ?>
<?php
    include_once("includes/form_functions.php");

?>
    <div id="main">

        <h2>Previous Game Info</h2>

        <table id="structure">
            <tr>
                <td id="page">
                    <div id="gamescontainer">
                        <ul id="gamesmenu">
<?php
$id=$_SESSION['user_id'];
$result = mysql_query("select g.id, g.title, g.siteaddress, lg.numquestion,
                        lg.numquestionsanswered, lg.ended, lg.numcorrect from game as g

```

```

        inner join livegame as lg on lg.gameid=g.id where lg.userid='$id'
        && lg.ended=1");

$num=mysql_numrows($result);
$i=0;
while($i < $num)
{

    $title=mysql_result($result, $i, "g.title");
    $numquestions=mysql_result($result, $i, "lg.numquestion");
    $numanswered=mysql_result($result, $i, "lg.numquestionsanswered");
    $numcorrect=mysql_result($result, $i, "lg.numcorrect");

    ?>
    <tr><b><td>
    <? echo '<a href="'. $siteaddress.'?id='.$_SESSION['user_id'].'' '>' ?>
    <? echo $title ?></a></b></td><td></td>
    <td>
    <?
        $percent=100;
        $d= ($numcorrect / $numquestions) * $percent;

    ?>
        <? echo $numcorrect?> out of <? echo $numquestions ?> questions answered
        correctly.</td><td>Final Score: <? echo number_format ($d, 2); ?></td>
    </tr></b>
    <?
    $i++;
    }

    ?>
        </td>
    </tr>
</table>

        </ul>
    </div>
    </td>
</tr>
</table>
<?php include("includes/footer.php"); ?>

```

Staff.php

```

<?php require_once("includes/session.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php confirm_logged_in(); ?>
<?php include("includes/head_admin.php"); ?>
    <div id="main">

    <h2>Administration Area</h2>

    <h3>Welcome, <?php echo $_SESSION['username']; ?>.</h3>

```

```

        <p> <b>Here you can view and delete the word description pairs,
                                current users, and current games.</b>
    </p>

    <?php include("includes/footer.php"); ?>

```

Viewdata.php

```

<?php require_once("includes/session.php"); ?>
<?php require_once("includes/connection.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php confirm_logged_in(); ?>
<?php include("includes/head_admin.php"); ?>
<?php
    include_once("includes/form_functions.php");

?>

    <div id="main">

        <h2>View, Modify, and Delete Word/Description Pairs</h2>

<?php
if(isset($_REQUEST['id']))
{
    $id=$_REQUEST['id'];
    $query = "DELETE FROM question WHERE id=$id";
    $result=mysql_query($query);
    if ($result) {
        $message = "<h3>The Word/Description Pair has been successfully deleted.</h3>";
    } else {
        $message = "The Word/Description Pair could not be deleted.";
        $message .= "<br />" . mysql_error();
    }
}
if (!empty($message)) { echo "<p class='message'>" . $message . "</p>"; }
if (!empty($errors)) { display_errors($errors); }

echo "<table><tr><td><b>Modify</b></td><td><b>Delete</b></td><td><b>Word</b></td>";
echo "<td><b>Description</b></td><td><b>Genre</b></td></tr>";
$result = mysql_query("SELECT q.id, q.word, q.genreid, q.description, g.id, g.name
FROM question as q INNER JOIN genre as g where q.genreid=g.id ORDER BY q.word asc");
$num=mysql_numrows($result);
$i=0;
while($i < $num)
{
    $id=mysql_result($result, $i, "q.id");
    $word=mysql_result($result, $i, "q.word");
    $description=mysql_result($result, $i, "q.description");
    $genre=mysql_result($result, $i, "g.name");
    ?><tr><b><td><? echo '<a href="modifypair.php?id='.$id.'">Modify</a>'; ?></td><td>
<? echo '<a href="viewdata.php?id='.$id.'">Delete</a>'; ?></td><td>
<? echo $word ?></td><b><td>
<? echo $description ?></td><td>
<? echo $genre ?></td></tr>

```

```

        <?
        $i++;
        }
        ?>

                </td>

        </tr>

</table>
<?php include("includes/footer.php"); ?>

```

Viewgames.php

```

<?php require_once("includes/session.php"); ?>
<?php require_once("includes/connection.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php confirm_logged_in(); ?>
<?php include("includes/head_admin.php"); ?>
<?php
        include_once("includes/form_functions.php");
?>
        <div id="main">

                <h2>View, Modify, and Delete the Online Games</h2>
                <a href="addgame.php">Add A New Game</a><br></br>

<?php
if(isset($_REQUEST['id']))
{
    $id=$_REQUEST['id'];
    $query = "DELETE FROM game WHERE id=$id";
    $result=mysql_query($query);
    if ($result) {
        $message = "<h3>The Game has been successfully deleted.</h3>";
    } else {
        $message = "The Game could not be deleted.";
        $message .= "<br />" . mysql_error();
    }
}
if (!empty($message)) { echo "<p class='message'>" . $message . "</p>";}
if (!empty($errors)) { display_errors($errors); }

echo "<table><tr><td><b>Modify</b></td><td><b>Delete</b></td><td><b>Title</b></td>";
echo "<td><b>Description</b></td><td><b>Site Address</b></td></tr>";

$result = mysql_query("SELECT * FROM game ORDER BY title asc");

$num=mysql_numrows($result);
$i=0;
while($i < $num)
{
    $id=mysql_result($result, $i, "id");
    $title=mysql_result($result, $i, "title");
    $description=mysql_result($result, $i, "description");
    $siteaddress=mysql_result($result, $i, "siteaddress");
}
}

```



```

?><tr><b><td><? echo '<a href="modifygame.php?id='.$id.'">Modify</a>'; ?>
</td><td>
<? echo '<a href="viewgames.php?id='.$id.'">Delete</a>'; ?></td><td>
<? echo $title?></td></b><td>
<? echo $description ?></td><td><? echo $siteaddress ?></td></tr>
<?
$i++;
}
?>

</td>

</tr>
</table>
<?php include("includes/footer.php"); ?>

```

Viewgenres.php

```

<?php require_once("includes/session.php"); ?>
<?php require_once("includes/connection.php"); ?>
<?php require_once("includes/functions.php"); ?>
<?php confirm_logged_in(); ?>
<?php include("includes/head_admin.php"); ?>
<?php
    include_once("includes/form_functions.php");

?>

<div id="main">

    <h2>View, Add, Modify, and Delete Genres</h2>
    <a href="addgenre.php">Add A New Genre</a></br></br>

<?php
if(isset($_REQUEST['id']))
{
    $id=$_REQUEST['id'];
    $query = "DELETE FROM genre WHERE id=$id";
    $result=mysql_query($query);
    if ($result) {
        $message = "<h3>The Genre has been successfully deleted.</h3>";
    } else {
        $message = "The Genre could not be deleted.";
        $message .= "<br />" . mysql_error();
    }
}
if (!empty($message)) {echo "<p class='message'>" . $message . "</p>";}
if (!empty($errors)) { display_errors($errors); }

echo "<table><tr><td><b>Modify</b></td><td><b>Delete</b></td><td><b>Name</b>";
echo "</td><td><b>Description</b></td></tr>";
$result = mysql_query("SELECT * FROM genre ORDER BY name asc");
$num=mysql_numrows($result);
$i=0;

```

```

while($i < $num)
{
    $id=mysql_result($result, $i, "id");
    $name=mysql_result($result, $i, "name");
    $description=mysql_result($result, $i, "description");
    ?><tr><b><td><? echo '<a href="modifygenre.php?id='.$id.'">Modify</a>'; ?></td><td>
<? echo '<a href="viewgenres.php?id='.$id.'">Delete</a>'; ?></td><td>
<? echo $name ?></td></b><td>
<? echo $description ?></td></tr>
<?
$i++;
}
?>

</td>

</tr>
</table>
<?php include("includes/footer.php"); ?>

```

Includes Folder

Connection.php

```

<?php
require("constants.php");

// 1. Create a database connection
$connection = mysql_connect(DB_SERVER,DB_USER,DB_PASS);
if (!$connection) {
    die("Database connection failed: " . mysql_error());
}

// 2. Select a database to use
$db_select = mysql_select_db(DB_NAME,$connection);
if (!$db_select) {
    die("Database selection failed: " . mysql_error());
}
?>

```

Constants.php

```

<?php

// Database Constants
define("DB_SERVER", "localhost");
define("DB_USER", "slaforge");
define("DB_PASS", "");
define("DB_NAME", "slaforgedb");

?>

```

Footer.php

```
</div>
                <div id="footer">Copyright 2011, FGCU Computer Science Department</div>
<div>
<?
if(isset($_SESSION['id'])){
    $u=$_SESSION['user_id'];
    $v=1;
    $adminquery="select * from users where id=$u";
    $adminresult=mysql_query($adminquery);
    $t=mysql_result($adminresult, 0, "admin");

    if($t==1) {
        echo "<center><font face='Verdana' size='2' ><br><a href=staff.php>Admin
Panel</a><br></center></font>";
    echo mysql_error();
    }
}

?>
</div>
    </body>
</html>
<?php
    // 5. Close connection
    mysql_close($connection);
?>
```

Form_functions.php

```
<?php
function check_required_fields($required_array) {
    $field_errors = array();
    foreach($required_array as $fieldname) {
        if (!isset($_POST[$fieldname]) || (empty($_POST[$fieldname]) && $_POST[$fieldname] != 0)) {
            $field_errors[] = $fieldname;
        }
    }
    return $field_errors;
}

function check_max_field_lengths($field_length_array) {
    $field_errors = array();
    foreach($field_length_array as $fieldname => $maxlength ) {
        if (strlen(trim(mysql_prep($_POST[$fieldname]))) > $maxlength) { $field_errors[] = $fieldname;
    }
    }
    return $field_errors;
}
```

```

}

function display_errors($error_array) {
    echo "<p class=\"errors\">";
    echo "Please review the following fields:<br />";
    foreach($error_array as $error) {
        echo " - " . $error . "<br />";
    }
    echo "</p>";
}

?>

```

Functions.php

```

<?php
// This file is the place to store all basic functions

function mysql_prep( $value ) {
    $magic_quotes_active = get_magic_quotes_gpc();
    $new_enough_php = function_exists( "mysql_real_escape_string" ); // i.e. PHP >= v4.3.0
    if( $new_enough_php ) { // PHP v4.3.0 or higher
        // undo any magic quote effects so mysql_real_escape_string can do the work
        if( $magic_quotes_active ) { $value = stripslashes( $value ); }
        $value = mysql_real_escape_string( $value );
    } else { // before PHP v4.3.0
        // if magic quotes aren't already on then add slashes manually
        if( !$magic_quotes_active ) { $value = addslashes( $value ); }
        // if magic quotes are active, then the slashes already exist
    }
    return $value;
}

function redirect_to( $location = NULL ) {
    if ($location != NULL) {
        header("Location: {$location}");
        exit;
    }
}

function confirm_query($result_set) {
    if (!$result_set) {
        die("Database query failed: " . mysql_error());
    }
}

function get_all_subjects($public = true) {
    global $connection;
    $query = "SELECT *
              FROM subjects ";
    if ($public) {
        $query .= "WHERE visible = 1 ";
    }
}

```

```

$query := "ORDER BY position ASC";
$subject_set = mysql_query($query, $connection);
confirm_query($subject_set);
return $subject_set;
}

function get_pages_for_subject($subject_id, $public = true) {
    global $connection;
    $query = "SELECT *
              FROM pages ";
    $query .= "WHERE subject_id = {$subject_id} ";
    if ($public) {
        $query .= "AND visible = 1 ";
    }
    $query .= "ORDER BY position ASC";
    $page_set = mysql_query($query, $connection);
    confirm_query($page_set);
    return $page_set;
}

function get_subject_by_id($subject_id) {
    global $connection;
    $query = "SELECT * ";
    $query .= "FROM subjects ";
    $query .= "WHERE id=" . $subject_id . " ";
    $query .= "LIMIT 1";
    $result_set = mysql_query($query, $connection);
    confirm_query($result_set);
    // REMEMBER:
    // if no rows are returned, fetch_array will return false
    if ($subject = mysql_fetch_array($result_set)) {
        return $subject;
    } else {
        return NULL;
    }
}

function get_page_by_id($page_id) {
    global $connection;
    $query = "SELECT * ";
    $query .= "FROM pages ";
    $query .= "WHERE id=" . $page_id . " ";
    $query .= "LIMIT 1";
    $result_set = mysql_query($query, $connection);
    confirm_query($result_set);
    // REMEMBER:
    // if no rows are returned, fetch_array will return false
    if ($page = mysql_fetch_array($result_set)) {
        return $page;
    } else {
        return NULL;
    }
}

function get_default_page($subject_id) {
    // Get all visible pages

```

```

$page_set = get_pages_for_subject($subject_id, true);
if ($first_page = mysql_fetch_array($page_set)) {
    return $first_page;
} else {
    return NULL;
}
}

function find_selected_page() {
    global $sel_subject;
    global $sel_page;
    if (isset($_GET['subj'])) {
        $sel_subject = get_subject_by_id($_GET['subj']);
        $sel_page = get_default_page($sel_subject['id']);
    } elseif (isset($_GET['page'])) {
        $sel_subject = NULL;
        $sel_page = get_page_by_id($_GET['page']);
    } else {
        $sel_subject = NULL;
        $sel_page = NULL;
    }
}

function navigation($sel_subject, $sel_page, $public = false) {
    $output = "<ul class=\"subjects\">";
    $subject_set = get_all_subjects($public);
    while ($subject = mysql_fetch_array($subject_set)) {
        $output .= "<li";
        if ($subject["id"] == $sel_subject['id']) { $output .= " class=\"selected\""; }
        $output .= "><a href=\"edit_subject.php?subj=" . urlencode($subject["id"]) .
            "\">{$subject["menu_name"]}</a></li>";
        $page_set = get_pages_for_subject($subject["id"], $public);
        $output .= "<ul class=\"pages\">";
        while ($page = mysql_fetch_array($page_set)) {
            $output .= "<li";
            if ($page["id"] == $sel_page['id']) { $output .= " class=\"selected\""; }
            $output .= "><a href=\"content.php?page=" . urlencode($page["id"]) .
                "\">{$page["menu_name"]}</a></li>";
        }
        $output .= "</ul>";
    }
    $output .= "</ul>";
    return $output;
}

function public_navigation($sel_subject, $sel_page, $public = true) {
    $output = "<ul class=\"subjects\">";
    $subject_set = get_all_subjects($public);
    while ($subject = mysql_fetch_array($subject_set)) {
        $output .= "<li";
        if ($subject["id"] == $sel_subject['id']) { $output .= " class=\"selected\""; }
        $output .= "><a href=\"index.php?subj=" . urlencode($subject["id"]) .
            "\">{$subject["menu_name"]}</a></li>";
        if ($subject["id"] == $sel_subject['id']) {
            $page_set = get_pages_for_subject($subject["id"], $public);
            $output .= "<ul class=\"pages\">";
        }
    }
}

```

```

        while ($page = mysql_fetch_array($page_set)) {
            $output .= "<li";
            if ($page["id"] == $sel_page["id"]) { $output .= " class=\"selected\""; }
            $output .= "><a href=\"index.php?page=" . urlencode($page["id"]) .
                "\">{$page["menu_name"]}</a></li>";
        }
        $output .= "</ul>";
    }
    $output .= "</ul>";
    return $output;
}
?>

```

Head.php

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>FGCU Computer Science Online Games</title>
<link href="stylesheets/Site.css" rel="stylesheet" type="text/css" />
</head>

<body>
    <div class="page">
        <div id="header">
            <table width="100%">
                <tr>
                    <td>

                        <div id="title">
                            <h1>FGCU Computer Science Online Learning Game</h1>
                        </div>

                    </td>
                </tr>
                <tr>
                    <td colspan="3">
                        <div id="menucontainer">
                            <ul id="menu">
                                <li><a href="index.php">Home</a></li>
                                    <li><a href="games.php">Play Game</a></li>
                                <li><a href="continue.php">Continue Game</a></li>
                                    <li><a href="previous.php">Previous Games</a></li>
                                <li><a href="inputdata.php">Input Data</a></li>
                                    <li><a href="mydata.php">My Questions</a></li>

                                if(isset($_SESSION['id'])){
                                    echo "<li><a href=logout.php>Log Out</a></li>";
                                    echo mysql_error();
                                }
                            </ul>
                        </div>
                    </td>
                </tr>
            </table>
        </div>
    </body>
</html>

```

?>

Head_admin.php

 $\leq ?$


```
</div>
```

Session.php

```
<?php
    session_start();

    function logged_in() {
        return isset($_SESSION['user_id']);
    }

    function confirm_logged_in() {
        if (!logged_in()) {
            redirect_to("login.php");
        }
    }
?>
```

Welcome.php

```
<div>
<?php
    $con = mysql_connect("localhost","slaforge","");
    if (!$con)
    {
        die('Could not connect: ' . mysql_error());
    }

    mysql_select_db("slaforgedb", $con);
    $pid=$_SESSION['user_id']

?>
<div id="main">
<?
if(isset($_SESSION['id'])){

echo "<center><font face='Verdana' size='2' ><br><h3>Welcome $_SESSION[username]!";
echo "</h3></br></br><b>Have fun playing games and uploading Word/Description Pairs.";
echo "</b></br></br> Click <a href=logout.php>here to logout</a> &nbsp;|&nbsp; ";
echo "<a href=modifyprofile.php?id=$pid>Update Profile</a><br></center></font>";
echo mysql_error();}
else{
echo "<center><font face='Verdana' size='2' ><a href=login.php><b>Already a member? ";
echo "Please Login</b></a></br></br> </center></font>";

echo "<center><font face='Verdana' size='2' >or <a href=new_user.php><b>";
echo "Sign up for a Free Account</b></a> </center></font>";

}
?>
```

Stylesheets Folder

Site.css

```
/*-----  
The base color for this template is #5c87b2. If you'd like  
to use a different color start by replacing all instances of  
#5c87b2 with your new color.  
-----*/  
body  
{  
    background-color:Black;  
    font-size: .75em;  
    font-family: Verdana, Helvetica, Sans-Serif;  
    margin: 0;  
    padding: 0;  
    color: #696969;  
    min-width: 900px;  
}  
  
a:link  
{  
    color: #034af3;  
    text-decoration: underline;  
}  
a:visited  
{  
    color: #505abc;  
}  
a:hover  
{  
    color: #1d60ff;  
    text-decoration: none;  
}  
a:active  
{  
    color: #12eb87;  
}  
  
p, ul  
{  
    margin-bottom: 20px;  
    line-height: 1.6em;  
}  
  
/* HEADINGS  
-----*/  
h1, h2, h3, h4, h5, h6  
{  
    font-size: 1.5em;  
    color: #000;  
    font-family: Arial, Helvetica, sans-serif;  
}
```

```

h1
{
    font-size: 2em;
    padding-bottom: 0;
    margin-bottom: 0;
}
h2
{
    padding: 0 0 10px 0;
}
h3
{
    font-size: 1.2em;
}
h4
{
    font-size: 1.1em;
}
h5, h6
{
    font-size: 1em;
}

/* this rule styles <h2> tags that are the
first child of the left and right table columns */
.rightColumn > h1, .rightColumn > h2, .leftColumn > h1, .leftColumn > h2
{
    margin-top: 0;
}

/* PRIMARY LAYOUT ELEMENTS
-----*/

/* you can specify a greater or lesser percentage for the
page width. Or, you can specify an exact pixel width. */
.page
{
    width: 90%;
    margin-left: auto;
    margin-right: auto;
}

#header
{
    position: relative;
    margin-bottom: 0px;
    color: #000;
    padding: 0;
}

#header h1
{
    font-weight: bold;
    padding: 5px 0;
    margin: 0;
    color: #fff;
}

```

```

border: none;
line-height: 2em;
font-family: Arial, Helvetica, sans-serif;
font-size: 32px !important;
}

#main
{
padding: 30px 30px 15px 30px;
background-color: #fff;
margin-bottom: 30px;
_height: 1px; /* only IE6 applies CSS properties starting with an underscore */
}

#footer
{
color: #999;
padding: 10px 0;
text-align: center;
line-height: normal;
margin: 0;
font-size: .9em;
}

.tab
{
text-indent: 1in;
}

/* TAB MENU
-----*/
ul#menu
{
border-bottom: 1px #5C87B2 solid;
padding: 0 0 2px;
position: relative;
margin: 0;
text-align: center;
}

ul#menu li
{
display: inline;
list-style: none;
}

ul#menu li#greeting
{
padding: 10px 20px;
font-weight: bold;
text-decoration: none;
line-height: 2.8em;
color: #fff;
}

```

```

ul#menu li a
{
    padding: 10px 20px;
    font-weight: bold;
    text-decoration: none;
    line-height: 2.8em;
    background-color: #e8eef4;
    color: #034af3;
}

ul#menu li a:hover
{
    background-color: #fff;
    text-decoration: none;
}

ul#menu li a:active
{
    background-color: #a6e2a6;
    text-decoration: none;
}

ul#menu li.selected a
{
    background-color: #fff;
    color: #000;
}

/* FORM LAYOUT ELEMENTS
-----*/

fieldset
{
    margin: 1em 0;
    padding: 1em;
    border: 1px solid #CCC;
}

fieldset p
{
    margin: 2px 12px 10px 10px;
}

legend
{
    font-size: 1.1em;
    font-weight: 600;
    padding: 2px 4px 8px 4px;
}

input[type="text"]
{
    width: 200px;
    border: 1px solid #CCC;
}

```

```

input[type="password"]
{
    width: 200px;
    border: 1px solid #CCC;
}

/* TABLE
-----*/

table
{
    /*border: solid 1px #e8eef4;*/
    /* border-collapse: collapse;*/
}

table#fullsize
{
    min-width: 900;
}

table td
{
    padding: 5px;
    /*border: solid 1px #e8eef4;*/
}

table th
{
    padding: 6px 5px;
    text-align: left;
    background-color: #e8eef4;
    /*border: solid 1px #e8eef4; */
}

/* MISC
-----*/

.clear
{
    clear: both;
}

.error
{
    color: Red;
}

#menucontainer
{
    margin-top: 40px;
    min-width: 900;
}

div#title
{
    display: block;
    float: center;

```

```

    text-align:center;
}

#logindisplay
{
    font-size:1.1em;
    display:block;
    text-align:center;
    margin:10px;
    color:White;
}

#logindisplay a:link
{
    color: white;
    text-decoration: underline;
}

#logindisplay a:visited
{
    color: white;
    text-decoration: underline;
}

#logindisplay a:hover
{
    color: white;
    text-decoration: none;
}

/* Styles for validation helpers
-----*/
.field-validation-error
{
    color: #ff0000;
}

.field-validation-valid
{
    display: none;
}

.input-validation-error
{
    border: 1px solid #ff0000;
    background-color: #ffebee;
}

.validation-summary-errors
{
    font-weight: bold;
    color: #ff0000;
}

.validation-summary-valid

```

```

{
    display: none;
}

/* Styles for editor and display helpers
-----*/
.display-label,
.editor-label,
.display-field,
.editor-field
{
    margin: 0.5em 0;
}

.text-box
{
    width: 30em;
}

.text-box.multi-line
{
    height: 6.5em;
}

.tri-state
{
    width: 6em;
}

```


Appendix B: User Manual

I. Overview

The purpose of the User Manual is to provide a general understanding of how to utilize the online gaming website. This manual will provide instructions on how to create an account, log in and out, and navigate between pages.

The general idea of the site is to allow you to play games that involve a word or group of words and its definition. You will also have the opportunity to add your own word/description pairs to the database and these will be shuffled into the questions in the various games on the site. These will be included for not only you, but anyone with an account, to interact with. For the word pairs that you add, you will have the ability to modify and delete them, if you desire.

II. Creating an Account

If you do not already have an existing account, you will need to create one. When you first enter the site you will see a welcome message with two hyperlinks. One will direct you to log in, if you already have an account. The other links you to the page where you may create an account. Click the hyperlink that displays, “Sign up for a free account.”

You will now see a form, where you must populate the required data and click the, “Create User” button. If done correctly, a message at the top of the page will let you know that the account was created successfully.

The screenshot shows a web application interface for 'FGCU Computer Science Online Learning Game'. At the top, there is a dark blue header with the title in white. Below the header is a navigation bar with seven buttons: 'Home', 'Play Game', 'Continue Game', 'Previous Games', 'Input Data', 'My Questions', and 'Log In'. The main content area is white and titled 'Create A New Account'. It includes a blue hyperlink 'Return to Log In' and a form with five input fields: 'First Name:', 'Last Name:', 'Username:', 'Password:', and 'Email:'. A 'Create user' button is located at the bottom of the form.

FGCU Computer Science Online Learning Game

[Home](#) [Play Game](#) [Continue Game](#) [Previous Games](#) [Input Data](#) [My Questions](#) [Log In](#)

Create A New Account

[Return to Log In](#)

First Name:

Last Name:

Username:

Password:

Email:

(Figure B1) Create an Account Page

III. Logging in to the Site

From the home page, click either the tab at the top right that says, “Log In”, or the hyperlink in the center of the page that states, “Already a Member? Please log in.” Until you sign in, you will not have access to the site contents.

At the log in page, enter your username and password and click the button titled, “Log In.” If you do not currently have an account, click the hyperlink that states, “Create an Account” next to the form and create an account.

The screenshot shows a web interface for the 'FGCU Computer Science Online Learning Game'. At the top, a black header contains the title in white. Below the header is a navigation bar with seven blue buttons: 'Home', 'Play Game', 'Continue Game', 'Previous Games', 'Input Data', 'My Questions', and 'Log In'. The main content area is white and features a welcome message, followed by login fields for 'Username' and 'Password', a 'Login' button, and a 'Create An Account' link. The footer is black and contains a copyright notice.

FGCU Computer Science Online Learning Game

[Home](#) [Play Game](#) [Continue Game](#) [Previous Games](#) [Input Data](#) [My Questions](#) [Log In](#)

Welcome to the FGCU Online Learning Game Website

Username:

Password:

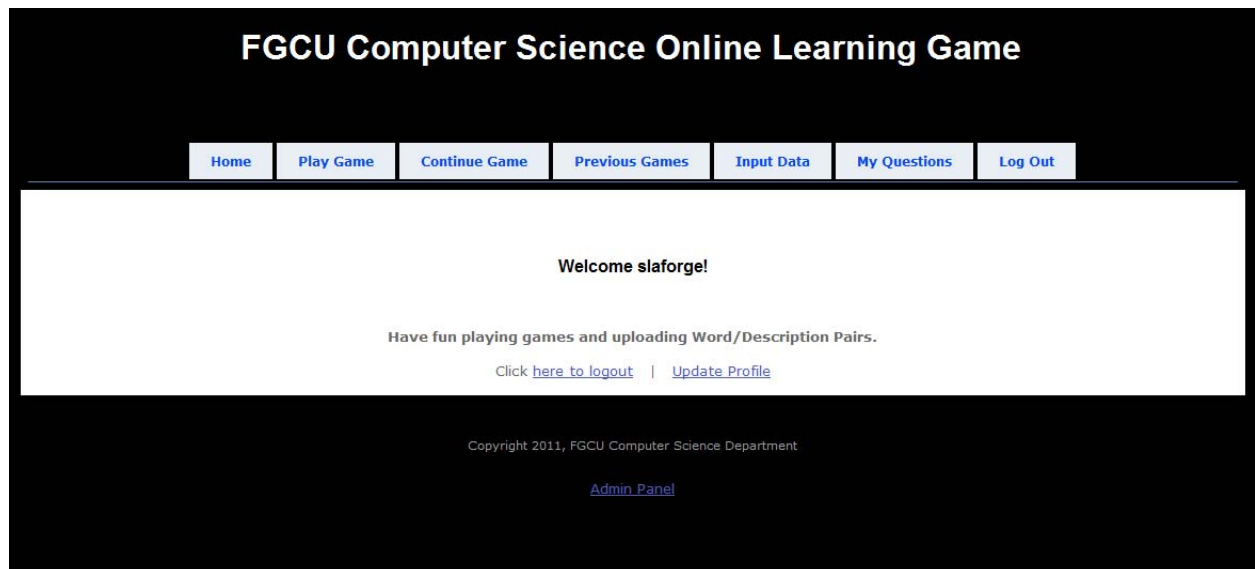
[Create An Account](#)

Copyright 2011, FGCU Computer Science Department

(Figure B2) Log In Page

IV. User Site Navigation

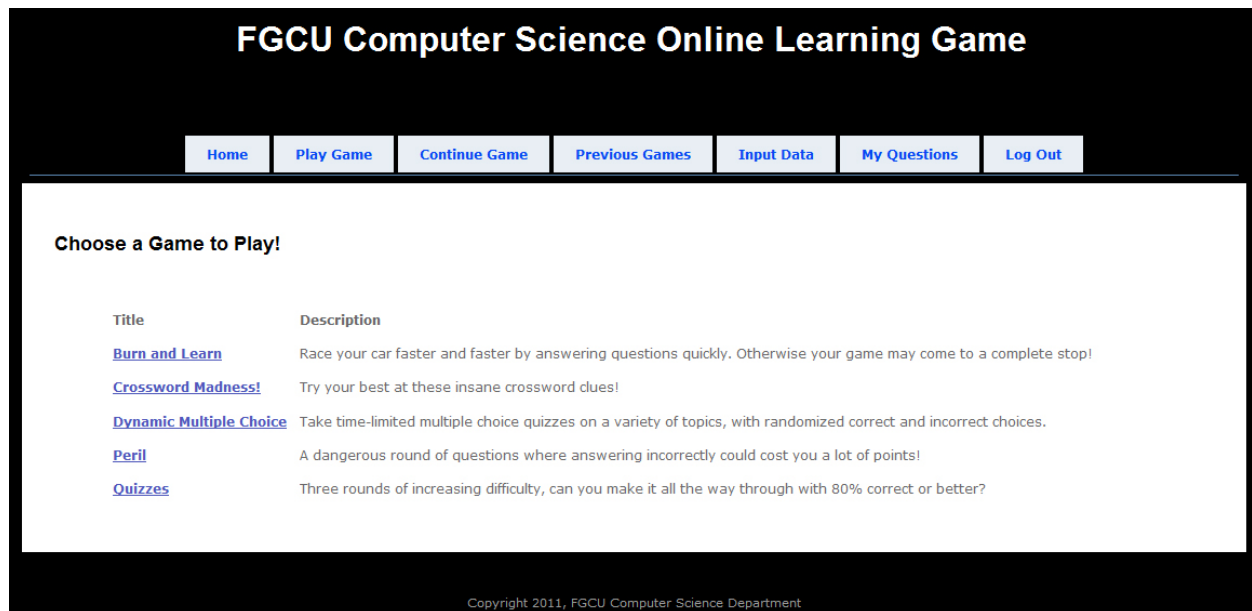
Once you are logged in, you will be redirected to the home page where you will see a welcome message that includes your username. You now have access to the site and may entertain yourself with all that the site has to offer. Across the top of the screen are different tabs that lead you to the different pages of the site. Those users with administrative privileges will see a hyperlink at the bottom of the page, under the copyright, titled, “Admin Panel”, where they may click and be directed to the administrative side of the site. The following subheadings will explain what you may do on each page.



(Figure B3) User Main Page

A. Play Games Page

If you select the “Play Games” tab, you will be directed to the games page. Here you will see a list of games along with a description of what the game is about. To play a game, simply click the title of the game. This is a hyperlink that will direct you to the game you have selected and is colored in blue and underlined.



(Figure B4) Play Game Page

B. Continue Games Page

If you select the “Continue Game” tab, you will be directed to the continue games page. Here you will see a list of games that you have not yet completed, if there are any. This list will show the title of the game along with the number of questions you have answered out of the total number of questions. You will also see your current score.

To continue playing a game, simply click on the title of the game. This is a hyperlink that will take you to the game page and is colored in blue and underlined.

FGCU Computer Science Online Learning Game

Home

Play Game

Continue Game

Previous Games

Input Data

My Questions

Log Out

Continue A Game!

Continue Burn and Learn	16 out of 20 questions answered	Current Score: 70.00	Delete Game
Continue Peril	7 out of 22 questions answered	Current Score: 31.82	Delete Game
Continue Peril	0 out of 0 questions answered	Current Score: 0.00	Delete Game
Continue Peril	0 out of 0 questions answered	Current Score: 0.00	Delete Game
Continue Peril	0 out of 0 questions answered	Current Score: 0.00	Delete Game
Continue Dynamic Multiple Choice	0 out of 0 questions answered	Current Score: 0.00	Delete Game

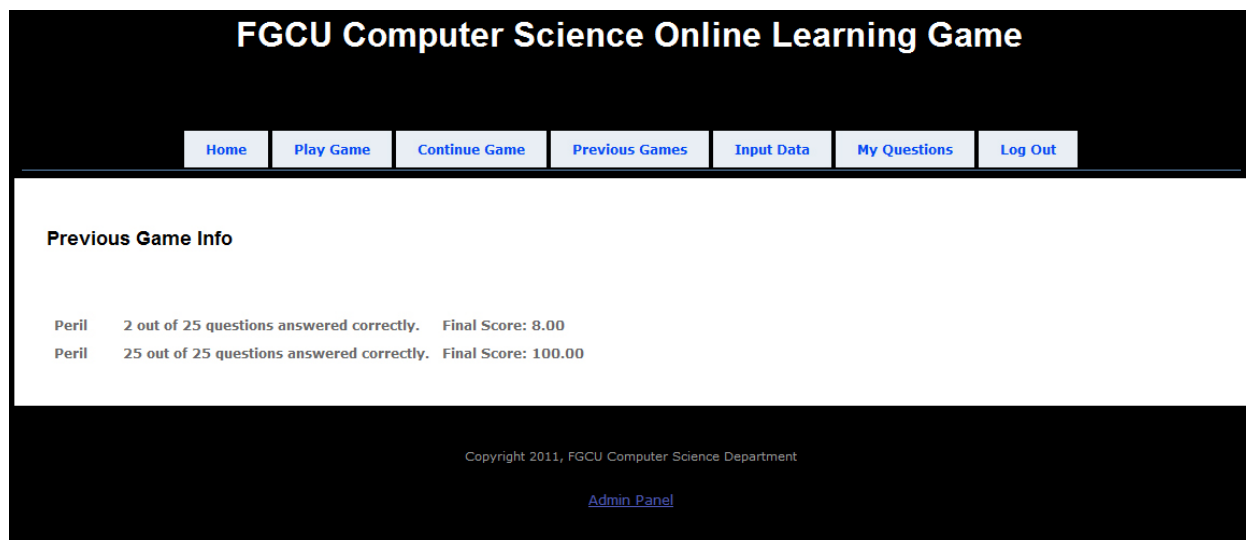
Copyright 2011, FGCU Computer Science Department

[Admin Panel](#)

(Figure B5) Continue Games Page

C. Previous Games Page

If you select the “Previous Games” tab, you will be directed to the previous games page. Here you will see a list of games that you have already completed, if they exist. This list will show the title of the game along with the number of questions you have answered correctly out of the total number of questions. You will also see your final score.



(Figure B6) Previous Games Page

D. Input My Data Page

If you select the “Input Data” tab, you will be directed to the page where you can add a word or group of words, along with its definition, into the database. Simply fill in the different sections of the form. Add a word or group of words up to 100 characters long in the “Word” text box. Add its definition or description in the “Description” text box, select a genre in the “Genre” list box, and then click the “Add” button. If you do not see a category that you feel it should belong in, simply add it to the “Misc” genre. This will store it in a miscellaneous category and will eventually be updated to its own category when enough pairs exist in the database that are of a similar genre. A message at the top of the page, above the form, will inform you that the word/description pair was successfully added to the database. If there was an error, an error message will display instead. To view the word/description pairs that you have added, you may either click the tab at the top left that says, “View List” or select the tab titled, “My Questions.”

FGCU Computer Science Online Learning Game

[Home](#) [Play Game](#) [Continue Game](#) [Previous Games](#) [Input Data](#) [My Questions](#) [Log Out](#)

Input word description pairs.

[View List](#)

Word:

Description:

Genre:

(Figure B7) Input data Page

E. My Data Page

If you select the “My Data” tab, you will be directed to the page where you can view all the words or small groups of words that you have added. This will be displayed as a list that shows the word, its, description, and what genre it belongs to. You will also have the option to modify or delete these word/description pairs by selecting either the “modify” or “delete” hyperlinks that correspond to the particular line.

Home			Play Game			Continue Game			Previous Games			Input Data			My Questions			Log Out		
View, Modify, and Delete Word/Description Pairs																				
Modify	Delete	Word	Description												Genre					
Modify	Delete	Algorithm	A component that improves performance by transparently storing data such that future requests for that data can be served faster.												Computer Science					
Modify	Delete	Assembly Language	A type of computer CPU programming language, which consists entirely of numbers and are almost impossible for humans to read and write.												Computer Science					
Modify	Delete	AVL Tree	A self-balancing binary search tree, and it is the first such data structure to be invented. In this tree, the heights of the two child subtrees of any node differ by at most one.												Computer Science					
Modify	Delete	Cache	A component that improves performance by transparently storing data such that future requests for that data can be served faster.												Computer Science					
Modify	Delete	Computer Science	The study of the theoretical foundations of information and computation, and of practical techniques for their implementation and application in computer systems.												Computer Science					
Modify	Delete	Critical Section	A piece of code or a set of instructions that can only be executed by one process or thread at a time.												Computer Science					
Modify	Delete	Dijkstra	A Dutch computer scientist who, in 1956, conceived a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge path costs, producing a shortest path tree. This algorithm is often used in routing and as a subroutine in other graph algorithms.												Computer Science					
Modify	Delete	Disassembly	The process of taking a binary program and deriving the source code from it.												Computer Science					
Modify	Delete	Distributed Database	A database that is under the control of a central database management system (DBMS) in which storage devices are not all attached to a common CPU.												Computer Science					

(Figure B8) My Data Page

F. Update Profile Page

On the home page of the site, you may choose to update your profile by selecting the hyperlink in the center of the page titled, “Update Profile.” On this page you will see a form populated with all of your user data. Here you may change your first name, last name, username, password, and email address. Once you have finished editing, simply click the button under the form named, “Update.” A message will be posted above the form telling you whether your profile was updated successfully or if there were errors while updating.

Update Profile

[Return to Home](#)

First Name:

Last Name:

User Name:

Password:

Email:

(Figure B9) Update Profile Page

IV. Administrator Site Navigation

FGCU Computer Science Online Learning Game

[Home](#) [Data](#) [Genre](#) [Users](#) [Games](#) [Log Out](#)

(Figure B10) Admin Navigation links

A. Data Page

By clicking on the tab titled, “Data”, you will be directed to a page listing all of the word/description pairs that have been added by users on the site. Each line in the list will display a “modify” and “delete” button along with the word or group of words, its description, and the genre that it belongs to. By selecting the modify button, the administrator will be directed to a page where the pair may be edited. By selecting the “delete” button, the word/description pair will be deleted from the database and removed from the list. If the word/description pair is

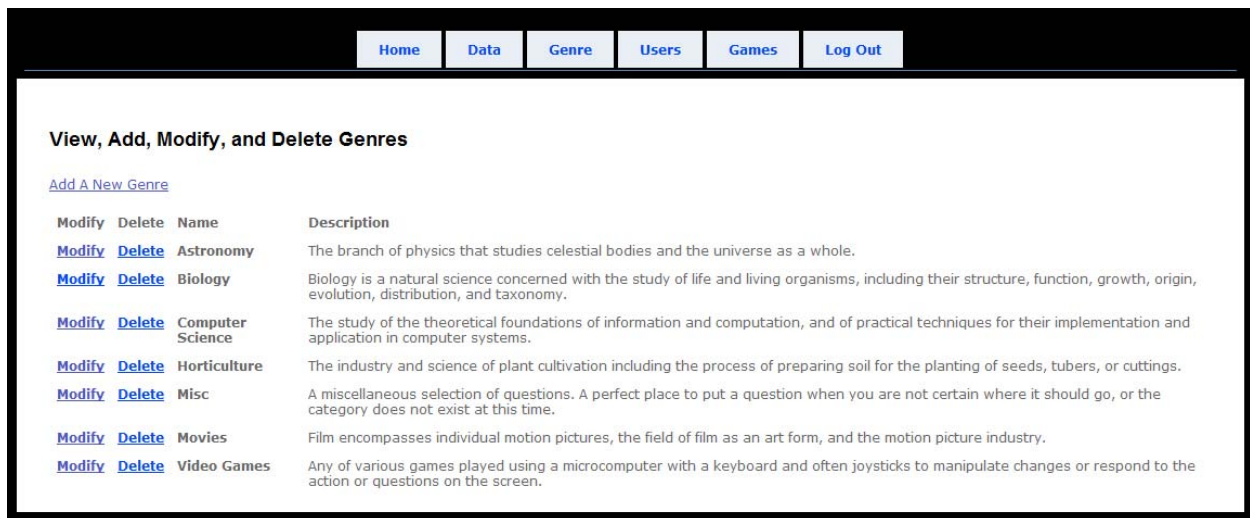
successfully deleted, a message stating that the word/description pair was successfully deleted will be displayed.

Home Data Genre Users Games Log Out				
View, Modify, and Delete Word/Description Pairs				
Modify	Delete	Word	Description	Genre
Modify	Delete	Accretion	The process by where dust and gas accumulated into larger bodies such as stars and planets.	Astronomy
Modify	Delete	Algorithm	A component that improves performance by transparently storing data such that future requests for that data can be served faster.	Computer Science
Modify	Delete	Assembly Language	A type of computer CPU programming language, which consists entirely of numbers and are almost impossible for humans to read and write.	Computer Science
Modify	Delete	AVL Tree	A self-balancing binary search tree, and it is the first such data structure to be invented. In this tree, the heights of the two child subtrees of any node differ by at most one.	Computer Science
Modify	Delete	Cache	A component that improves performance by transparently storing data such that future requests for that data can be served faster.	Computer Science
Modify	Delete	Computer	A device doing calculations.	Computer Science
Modify	Delete	Computer Science	The study of the theoretical foundations of information and computation, and of practical techniques for their implementation and application in computer systems.	Computer Science
Modify	Delete	Constellation	An internationally defined area of the celestial sphere.	Astronomy
Modify	Delete	Critical Section	A piece of code or a set of instructions that can only be executed by one process or thread at a time.	Computer Science
Modify	Delete	Dijkstra	A Dutch computer scientist who, in 1956, conceived a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge path costs, producing a shortest path tree. This algorithm is often used in routing and as a subroutine in other graph algorithms.	Computer Science

(Figure B11) View Data Page

B. Genre Page

By clicking on the tab titled, “Genre”, you will be directed to a page listing all of the genres that currently exist in the database. Each line in the list will display a “modify” and “delete” button along with the genre name and its description. By selecting the “modify” button, the administrator will be directed to a page where the genre may be edited. By selecting the “delete” button, the genre will be deleted from the database and removed from the list. If the genre is successfully deleted, a message stating that the genre was successfully deleted will be displayed. Also, above the list of genres will reside a hyperlink titled, “Add New Genre.” By clicking this link, the administrator will be directed to a page where a new genre may be added to the database.



(Figure B12) View Genre Page

C. Add New Genre Page

If you select the “Add New Genre” hyperlink on the “Genre” page, you will be directed to a page where you may add new genres into the database. Simply fill in the different sections of the form. Add a new genre name up to 100 characters long in the “Name” text box. Add its definition or description in the “Description” text box. Then click the “Add” button. A message at the top of the page, above the form, will inform you that the genre was successfully added to the database. If there was an error, an error message will display instead. To view the genres that have been added, you may either click the hyperlink above the form titled, “View List” or select the tab titled, “Genre.”

Home Data Genre Users Games Log Out

Add A New Genre

[Return to List](#)

Name:

Description:

(Figure B13) Add New Genre Page

D. Modify Genre Page

By clicking on a “modify” hyperlink next to the genre on the “Genre” page, you will be directed to a page that includes a form populated with the data specific to that genre. Here you may change the genre name and its description. Once you have finished editing, simply click the button under the form named, “Update.” A message will be posted above the form telling you whether the genre was updated successfully or if there were errors while updating.

Home Data Genre Users Games Log Out

Modify Genre

[Back to list](#)

Name:

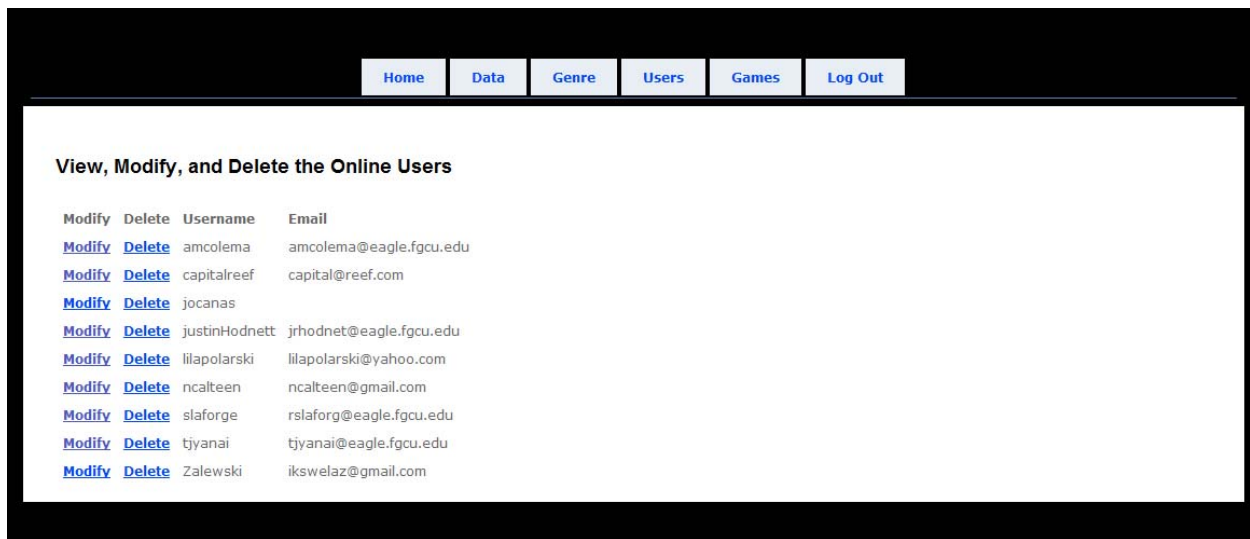
Description:

The branch of physics that studies celestial bodies and the universe as a whole.

(Figure B14) Modify Genre Page

E. Users Page

By clicking on the tab titled, “Users”, you will be directed to a page listing all of the users that currently exist in the database. Each line in the list will display a “modify” and “delete” button along with the username and email of the user. By selecting the “modify” button, the administrator will be directed to a page where the user may be edited. By selecting the “delete” button, the user will be deleted from the database and removed from the list. If the user is successfully deleted, a message stating that the user was successfully deleted will be displayed.

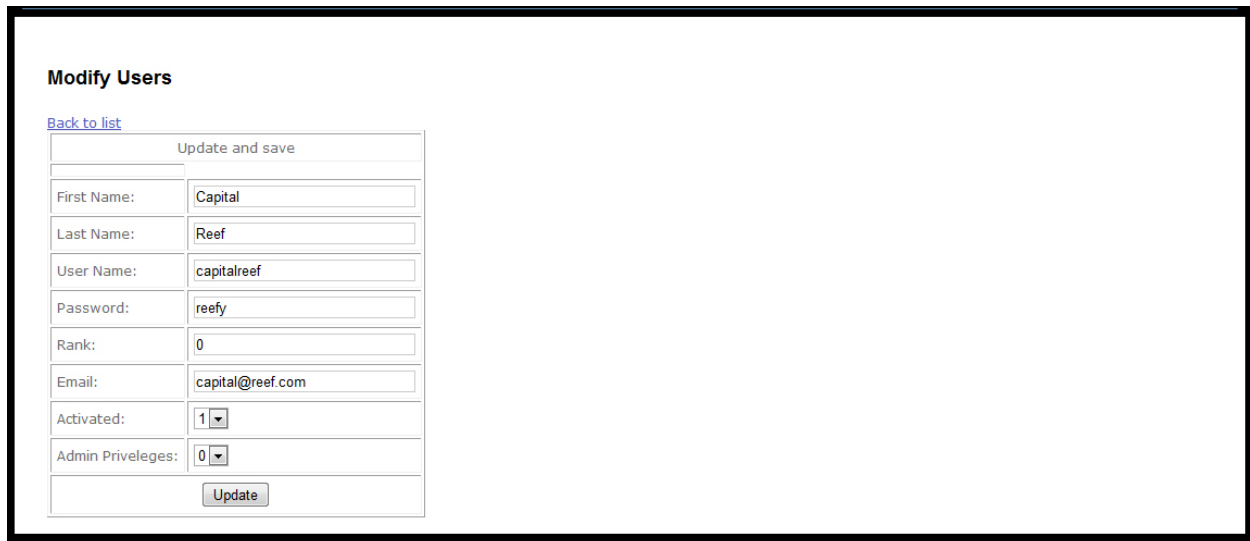


(Figure B15) View Users Page

F. Modify User Page

By clicking on a “modify” hyperlink next to the user on the “Users” page, you will be directed to a page that includes a form populated with the data specific to that user. Here you may change the user’s first name, last name, username, password, email address, active status, and their administrative privileges. A zero, for either, means that they are either inactive or without administration privileges. A one denotes that the user has an active status or that the user has administrative privileges. Once you have finished editing, simply click the button under

the form named, “Update.” A message will be posted above the form telling you whether the user was updated successfully or if there were errors while updating.



The screenshot shows a web form titled "Modify Users". At the top left, there is a blue hyperlink labeled "Back to list". Below this, a light gray box contains the text "Update and save". The form consists of several input fields: "First Name:" with the value "Capital", "Last Name:" with the value "Reef", "User Name:" with the value "capitalreef", "Password:" with the value "reefy", "Rank:" with the value "0", "Email:" with the value "capital@reef.com", "Activated:" with a dropdown menu showing "1", and "Admin Priveleges:" with a dropdown menu showing "0". At the bottom of the form is a gray button labeled "Update".

(Figure B16) Modify User Page

G. Game Page

By clicking on the tab titled, “Game”, you will be directed to a page listing all of the games that currently exist in the database. Each line in the list will display a “modify” and “delete” button along with the game name, its description, and its site address. By selecting the “modify” button, the administrator will be directed to a page where the game may be edited. By selecting the “delete” button, the game will be deleted from the database and removed from the list. If the game is successfully deleted, a message stating that the game was successfully deleted will be displayed. Also, above the list of games will reside a hyperlink titled, “Add New Game.” By clicking this link, the administrator will be directed to a page where a new game may be added to the database.

Home Data Genre Users Games Log Out				
View, Modify, and Delete the Online Games				
Add A New Game				
Modify Delete	Title	Description	Site Address	
Modify Delete	Burn and Learn	Race your car faster and faster by answering questions quickly. Otherwise your game may come to a complete stop!	/~tjyanai/BurnNLearn.php	
Modify Delete	Crossword Madness!	Try your best at these insane crossword clues!	/~ncalteen/main.php	
Modify Delete	Dynamic Multiple Choice	Take time-limited multiple choice quizzes on a variety of topics, with randomized correct and incorrect choices.	/~acoleman/self_study/user_menu.php	
Modify Delete	Peril	A dangerous round of questions where answering incorrectly could cost you a lot of points!	/~jrhodnet/peril.php	
Modify Delete	Quizzes	Three rounds of increasing difficulty, can you make it all the way through with 80% correct or better?	/~jocanas/game.php	

(Figure B17) View Games Page

H. Add New Game Page

If you select the “Add New Game” hyperlink on the “Game” page, you will be directed to a page where you may add new games into the database. Simply fill in the different sections of the form. Add a new game name up to 100 characters long in the “Title” text box. Add its description in the “Description” text box. Add its site address on the server in the “Site Address” text box. Then click the “Add” button. A message at the top of the page, above the form, will inform you that the game was successfully added to the database. If there was an error, an error message will display instead. To view the games that have been added, you may either click the hyperlink above the form titled, “View List” or select the tab titled, “Game.”

Home Data Genre Users Games Log Out

Add A New Game

[Return to List](#)

Title:

Description:

Site Address:

(Figure B18) Add New Game Page

I. Modify Game Page

By clicking on a “modify” hyperlink next to the game on the “Game” page, you will be directed to a page that includes a form populated with the data specific to that game. Here you may change the game title, its description, and its site address. Once you have finished editing, simply click the button under the form named, “Update.” A message will be posted above the form telling you whether the game was updated successfully or if there were errors while updating.

Home Data Genre Users Games Log Out

Modify Game

[Back to list](#)

Update and save

Title:

Description:

Site Address:

(Figure B19) Modify Game Page