Web UI Team

Final Report

Members:
Aryn Grause
Garrett Dieckmann
Katie Mueller
Kyle Monnett

December 14, 2011

Contents

1 End Product Description

As the Web User Interface team for VirPong, our end product is a direct portal through which users of VirPong can watch VirPong games (both live and past matches) and access game information including high scores and current rankings for VirPong players. The website will also serve as a front end to VirPong, allowing new users to explore the VirPong world and become active in it. Therefore, our goal with the VirPong website is to provide an attractive and intuitive portal into the world of VirPong.

In producing the VirPong website, we will be working on various aspects that are essential to the success of the VirPong experience. A portion of the website will be dedicated to viewing VirPong games, whether these games have already been played by VirPong users or are happening in real time, by using the VirPong mobile applications. Another part of the website will be dedicated to the user experience, providing a space for users to update their personal information. No sports experience would be complete without access to top scores, as well as following our favorite player, so there will also be a section of the VirPong website that will showcase high scoring games, as well as statistics about certain VirPong power players.

In constructing the VirPong platform, we also want to create an Open Source project that will break into the competitive sports industry, thus we want to focus on creating a solution that will be adaptable for other developers. To achieve this goal, our team will work on VirPong knowing that other developers will be using our code, and will provide in-line comments as well as documentation to make VirPong easy to redeploy. Specifically, we are maintaining install guides that document every technology, as well as the install instructions for these technologies, so that other developers can quickly redeploy VirPong by following a set of instructions tailored explicitly for the VirPong platform. To make the VirPong project easy to download, all of the source code has been pushed to GitHub and the VirPong team will continue to update VirPong code to include more features, as well as update VirPong to conform to current standards of the web.

2 Functional Requirements

2.1 Use Cases

Registering for an Account

Title: Successful Registration for the Game
Actors: User, Data Repository, System
Preconditions:

- Computer System (in its on state).
- User has the ability to navigate on the computer.

Post Conditions:

- User is successfully registered for the site.

Main Success Scenario:

1. User fills out appropriate information on Systems form.

2. System connects to Data Repository.

3. System checks for duplicate information.

4. Requests differnt information if duplicate found.

5. System stores information in Data Repository.

6. System disconnects from Data Repository.

7. System reports successful registration.

Extensions:

1. User navigates to the System.

1a. System is down.

1. Be sad.
2. Fix system.

2. User fills out appropriate information on Systems form

2a. Users fills out information incorrectly (e.g. enters street name instead of birthday)

1. Validate form before accepting
2. Report failed validation (e.g. incorrect information for the following field)

2b. Username information already exists

1. Check Data Repository for identical usernames
2. Inform User that the username is already taken
3. Prompts User to register with a unique username

3. System connects to Data Repository

3a. Failed connection

1. Ask user to refresh system

3b. Second failed connection

1. Ask user to contact VirPong and report an error
2. Provide user with a link to the Contact Us page

4. System stores information in Data Repository

4a. Information is not stored in repository due to repository going down

1. Report an error stating that this service is not available and to return at another time

4b. Repository throws an error

1. Ask user to try again

4c. Repository throws an error a second time

1. Ask user to contact VirPong and report an error
2. Provide user with a link to the Contact Us page

5. System disconnects from Data Repository

5a. Disconnection failed

1. Terminate through the use of exception handling

Watch Live

Title: Successful Live Game Viewing
Actors: User, Game Manager, System
Preconditions:

- Computer System (in its on state)
- User has the ability to navigate on the computer

Post Conditions:

- User sucessfully watches game without errors

Main Success Scenario:

1. User navigates to the System

2. System display options to user.

3. User selects the game information to view

4. System connects to the Game Manager

5. System informs Game Manager of which game was selected by user

6. Game Manager supplies information on the selected game to the system

7. System displays information

8. Repeat 4, 5, 6, and 7 until information stream ends from Game Manager

9. System closes connection to the Game Manager

10. System informs User that the game is over

Extensions:

1. User navigates to the System

1a. System is down

1. Report 404 Error that provides information to the User

2. User selects the game information to view

2a. User isn't doing much

1. Suggest a current game to view

2b. No games are available

1. Print an apologetic message stating that no current games are available
2. Suggest some historical games to view

3. System connects to the Game Manager

3a. Unsuccessful connection

1. Ask user to refresh system

3b. Second unsuccessful connections

1. Ask user to contact VirPong and report an error
2. Provide user with a link to the Contact Us page

4. System informs Game Manager of which game was selected by user

4a. System does not inform the Game Manager within allotted time

1. Restart connection with the Game Manager

4b. System fails to inform the Game Manager a second time

1. Ask user to refresh system

4c. System continues to fail

1. Ask user to contact VirPong and report an error
2. Provide user with a link to the Contact Us page

5. Game Manager supplies information on the selected game to the system

5a. Information on the wrong game is provided

1. Provide the user with the ability to leave a game at any moment

5b. No information is supplied

1. Ask user to refresh system

5c. No information was provided a second time

1. Ask user to contact VirPong and report an error
2. Provide user with a link to the Contact Us page

7. System displays information

7a. System is not supported by the users browser

1. Check which browser the user is using

2. Provide suggestion for a compatible browser

8. Repeat 4, 5, 6, and 7 until information stream ends from Game Manager

8a. No new event has occurred in an allotted time period

1. Close connection with the Game Manager

9. System closes connection to the Game Manager

9a. Connection is not closed in an allotted time

1. Reattempt to close connection

10. System informs User that the game is over

10a. System does not inform user

1. Provide navigation to other services

Signing-In

Title: Successful Sign-In Attempt
Actors: User, Data Repository, System
Preconditions:

- Computer System (in its on state)
- Ability to Navigate to the system

Post Conditions:

- User is signed in to the system

Main Success Scenario:

1. User navigates to the System
2. User fills out appropriate information on Systems sign-in

3. System connects to Data Repository
4. System confirms user input with information in Data Repository
5. System disconnects from Data Repository
6. System reports successful sign-in

Extensions:

1. User navigates to the System

1a. System is down

1. Report 404 Error that provides information to the User

3. System connects to Data Repository

3a. Failed connection

1. Ask user to refresh system

3b. Failed connection a second time

1. Ask user to contact VirPong and report an error
2. Provide user with a link to the Contact Us page

4. System confirms user input with information in Data Repository

4a. User input is incorrect and does not match repository

1. Report an error message stating unable to log-in
2. Prompt the user to try authenticating again

4b. User does not exist

1. Report an error message stating unable to log-in
2. Prompt the user to try authenticating again

4c. Repository throws an error

1. Ask user to refresh system

4d. Repository throws an error a second time

1. Ask user to contact VirPong and report an error
2. Provide user with a link to the Contact Us page

5. System disconnects from Data Repository

5a. Disconnection failed

1. Terminate through the use of exception handling

Change Account Settings

Title: Successful Modification of Account Settings
Actors: User, Data Repository, System Preconditions:

- Computer System (in its on state)
- User has the ability to navigate on the computer

Post Conditions:

- User successfully changes settings of their account

Main Success Scenario:

1. User navigates to the System
2. User authenticates into the System
3. User changes information
4. System connects to Data Repository
5. System provides updated feedback to Data Repository
6. System Disconnects from Data Repository
7. System relays successful changes

Extensions:

1. User authenticates into the System

1a. User is not registered to use system

1. Report an error message stating unable to log-in
2. Prompt the user to try authenticating again

1b. Authentication fail

1. Report an error message stating unable to log-in

2. Prompt the user to try authenticating again

2. User navigates to the System

2a. System is down

1. Report 404 Error that provides information to the User

3. User changes information

3a. User provides wrong type of information (ex. a name instead of a date for the birthday field

1. Validate form upon submission

2. Report failed validation issues (aka. incorrect information for the following field)

4. System connects to Data Repository

4a. Repository throws an error

1. Ask user to refresh system

4b. Repository throws an error a second time

1. Ask user to contact VirPong and report an error

2. Provide user with a link to the Contact Us page

5. System provides updated feedback to Data Repository

5a. Information does not get passed to the Data Repository

1. Notify user that the information was not updated

2. Navigates user to Step 3

5b. Connection breaks

1. Ask user to refresh system

5c. Connection breaks a second time

1. Ask user to contact VirPong and report an error

2. Provide user with a link to the Contact Us page

6. System Disconnects from Data Repository

6a. Disconnection failed

1. Terminate through the use of exception handling

Look Up Player History

Title: Successful Viewing of Player History
Actors: User, Data Repository, System
Preconditions:

- Computer System (in its on state)
- User has the ability to navigate on the computer

Post Conditions:

- User successfully views their player history

Main Success Scenario:

1. User navigates to the System
2. User authenticates into the System
3. System connects to Data Repository
4. System asks for information from Data Repository
5. Data Repository sends requested information to System
6. System Disconnects from Data Repository
7. System displays information
8. User reads their game information

Extensions:

1. User authenticates into the System

1a. User is not registered to use system

1. Report an error message stating unable to log-in
2. Prompt the user to try authenticating again

1b. Authentication fail

1. Report an error message stating unable to log-in
2. Prompt the user to try authenticating again

2. User navigates to the System

2a. System is down

1. Report 404 Error that provides information to the User

3. System connects to Data Repository

3a. Repository throws an error

1. Ask user to refresh system

3b. Repository throws an error a second time

1. Ask user to contact VirPong and report an error

2. Provide user with a link to the Contact Us page

4. System asks for information from Data Repository

4a. Request for information does not get made

1. Report an error stating that this service is not available and to return at another time

4b. Connection breaks

1. Report an error stating that this service is not available and to return at another time

4c. Repository throws an error

1. Ask user to refresh system

4d. Repository throws an error a second time

1. Ask user to contact VirPong and report an error

2. Provide user with a link to the Contact Us page

5. Data Repository sends requested information to System

5a. Source sends wrong information

1. Check information for proper format

2. Display an error message

3. Ask user to refresh system

5b. Source sends wrong information a second time

1. Ask user to contact VirPong and report an error

2. Provide user with a link to the Contact Us page

5c. Source sends no information

1. Check information for proper format
2. Display an error message
3. Ask user to refresh system

5d. Source sends no information a second time

1. Ask user to contact VirPong and report an error
2. Provide user with a link to the Contact Us page

6. System Disconnects from Data Repository

6a. Disconnection fails

1. Terminate through the use of exception handling

7. System displays information

7a. System cannot interpret the information

1. Check information for proper format
2. Display an error message
3. Ask user to refresh system

Player Plays in a Tournament

Title: Successful Participation in Tournament
Actors: Company, User, Data Repository, System
Preconditions:

- Computer System (in its on state)
- User has the ability to navigate on the computer

Post Conditions:

- User successfully plays in tournament

Main Success Scenario:

1. Company decides to host a tournament

2. User authenticates into the System
3. System displays tournament Registration
4. User navigates to the System
5. User requests System they would like to register for a tournament
6. System connects to Data Repository
7. System registers User on Data Repository
8. System Disconnects from Data Repository
9. System confirms registration
10. System sends a reminder to User
11. User navigates to game system
12. User successfully plays a game

Extensions:

3. User authenticates into the System

3a. User is not registered to use system

1. Report an error message stating unable to log-in
2. Prompt the user to try authenticating again

3b. Authentication fails

1. Report an error message stating unable to log-in
2. Prompt the user to try authenticating again

4. User navigates to the System

4a. System is down

1. Report 404 Error that provides information to the User

6. System connects to Data Repository

6a. Repository throws an error

1. Ask user to refresh system

6b. Repository throws an error a second time

1. Ask user to contact VirPong and report an error
2. Provide user with a link to the Contact Us page

7. System registers User on Data Repository

7a. Request for information does not get made

1. Report an error stating that this service is not available and to return at another time

7b. Connection breaks

1. Report an error stating that this service is not available and to return at another time

7c. Repository throws an error

1. Ask user to refresh system

7d. Repository throws an error a second time

1. Ask user to contact VirPong and report an error
2. Provide user with a link to the Contact Us page

8. System Disconnects from Data Repository

8a. Disconnection failed

1. Terminate through the use of exception handling

9. System confirms registration

9a. Response is not interpreted correctly

1. Check information for proper format
2. Display an error message
3. Ask user to refresh system
4. Return to step 3

9b. Response is not interpreted correctly a second time

1. Ask user to contact VirPong and report an error
2. Provide user with a link to the Contact Us page

9c. Registration was unsuccessful

1. Ask user to refresh system

9d. Registration was unsuccessful a second time

1. Ask user to contact VirPong and report an error
2. Provide user with a link to the Contact Us page

10. System sends a reminder to User

10a. No contact information is available for User

1. Remove user from database
2. User must re-register

10b. User's browser does not have capabilities to do so

1. Check which browser the user is using
2. Provide suggestion for a compatible browser

10c. System is not prompted to sent reminder

1. Display an error message informing the user of error

11. User authenticates into the System on Tournament Day

11a. User is not registered to use system

1. Report an error message stating unable to log-in
2. Prompt the user to try authenticating again

11b. Authentication fail

1. Report an error message stating unable to log-in
2. Prompt the user to try authenticating again

12. User navigates to the System on Tournament Day

12a. System is down

1. Report 404 Error that provides information to the User

13. User successfully plays a game

13a. Not tournament day

1. Inform the user that this a tournament game
2. Inform the user of the correct tournament date

Records and High Score

Title: Successful Viewing of Records and High Score
Actors: User, Source, System
Preconditions:

- Computer System (in its on state)
- User has the ability to navigate on the computer

Post Conditions:

- User successfully views records and high scores

Main Success Scenario:

1. User navigates to the System
2. System connects to Data Repository
3. System asks for information from Data Repository
4. Data Repository sends requested information to System
5. System Disconnects from Data Repository
6. System displays information
7. User reads records and high scores

Extensions:

1. User navigates to the System

1a. System is down

1. Report 404 Error that provides information to the User

2. System connects to Data Repository

2a. Repository throws an error

1. Ask user to refresh system

2b. Repository throws an error a second time

1. Ask user to contact VirPong and report an error

2. Provide user with a link to the Contact Us page

3. System asks for information from Data Repository

   3a. Request for information does not get made

      1. Report an error stating that this service is not available and to return at another time

   3b. Connection breaks

      1. Report an error stating that this service is not available and to return at another time

   3c. Repository throws an error

      1. Ask user to refresh system

   3d. Repository throws an error a second time

      1. Ask user to contact VirPong and report an error
      2. Provide user with a link to the Contact Us page

4. Data Repository sends requested information to System

   4a. Source sends wrong information

      1. Check information for proper format
      2. Display an error message
      3. Ask user to refresh system

   4b. Source sends wrong information a second time

      1. Ask user to contact VirPong and report an error
      2. Provide user with a link to the Contact Us page

   4c. Source sends no information

      1. Check information for proper format
      2. Display an error message
      3. Ask user to refresh system

   4d. Source sends no information a second time

      1. Ask user to contact VirPong and report an error
      2. Provide user with a link to the Contact Us page

5. System Disconnects from Data Repository

5a. Disconnection failed

1. Terminate through the use of exception handling

6. System displays information

6a. System cannot interpret the information

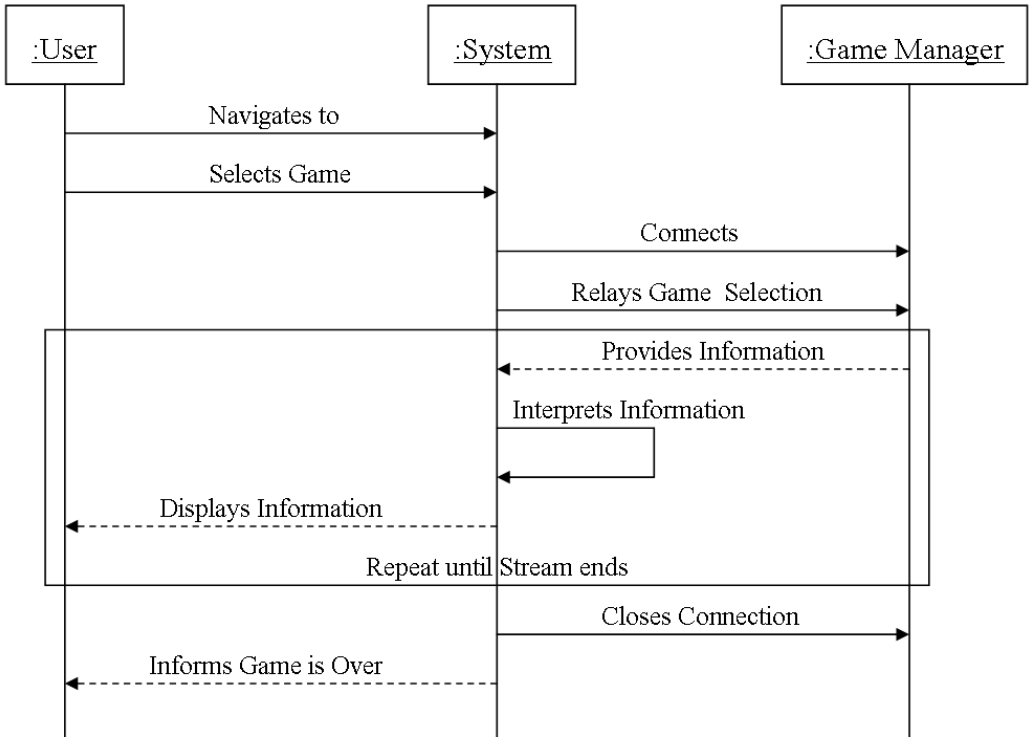
1. Check information for proper format

2. Display an error message

3. Ask user to refresh system

2.2 System Sequence Diagram

2.2.1 Diagram



2.2.2 Description

Within our System Sequence Diagram, we have three actors: the User, the System, and the Game Manager. The User navigates to the game selection

and selects a game to watch from the System, and the System connects to the Game Manager. The System then informs the Game Manager which game the user has selected. With a game selected, the Game Manager will provide game information to the System. The System will interpret the game information and then display this information to the User. These three steps will be repeated until the stream of game information ends. At this point, the System will close the connection with the Game Manager. Once the connection is closed, the System will notify the User that the current game is now over.

3 Nonfunctional Requirments

3.1 Functionality

- Passwords in the database repository should be encrypted to prevent breach in user privacy.
- All forms that write or read from the database should use javascript validation to prevent SQL injection.
- All hyperlinks should transfer the user to the indicated page.
- The design, implementation, and support of the website, and any systems used by the website, will operate on a budget of $0.00 per year.

3.2 Usability

- The website will be displayed in one of the multiple color schemes ("skins") designed for aesthetic pleasantry.
- Users may select a skin of their choosing, otherwise the default skin will be displayed.
- A single color scheme will not include colors that are difficult to distinguish for those with common forms of color blindness.
- All documents and code will comply with the determined style guide.
- Fonts should be readable from 2 feet away by an individual with average vision on a screen of 800x600 or larger resolution.
- Hyperlink text should appropriately represent the information to which the hyperlink leads.

3.3 Reliability

- User authentication will consistently log the correct user into the correct account without error.
- In the case of scheduled down time, the website will display a notification at least 48 hours prior to the down time.
- The database repository will be backed up every two weeks.

3.4 Performance

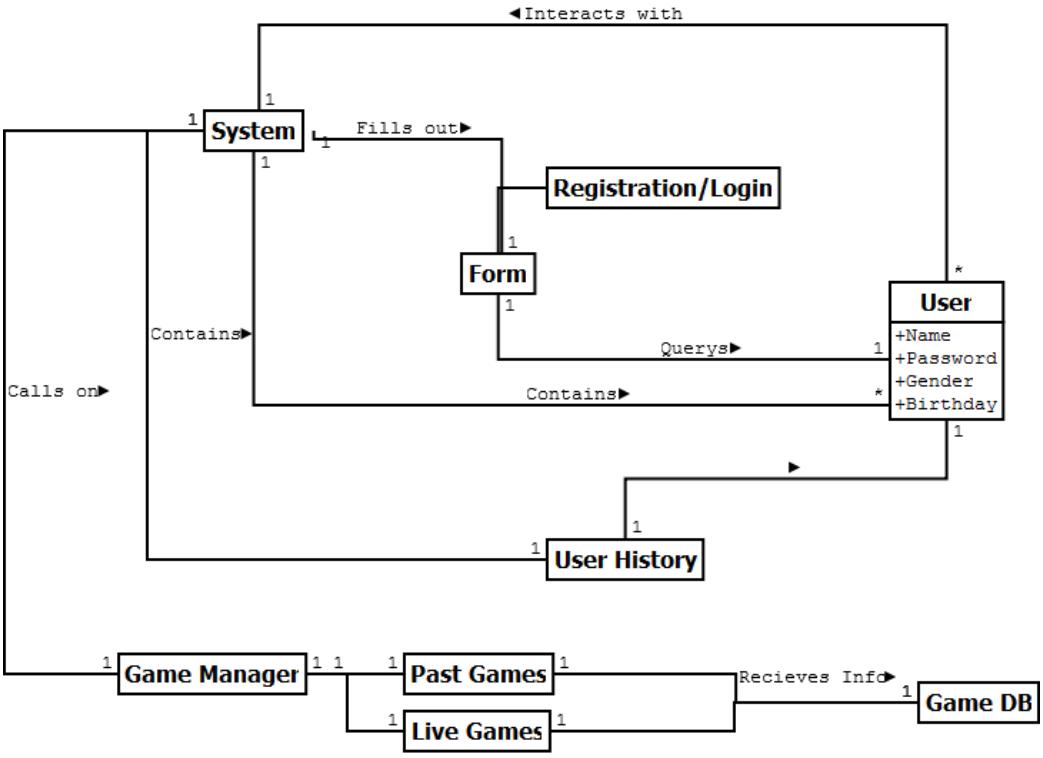
- Page loading should be minimized as much as possible, so page size should be kept as small as possible.

3.5 Supportability

- The website and the database schema will be thoroughly documented to facilitate troubleshooting by all administrators.
- The website will provide a means for users to contact site administrators to report site error or issue.
- In the case of an error, the website will provide a friendly, helpful message telling what happened, calming the user and providing a way to inform site administrators of the problem.

4 Domain Analysis

4.1 Diagram

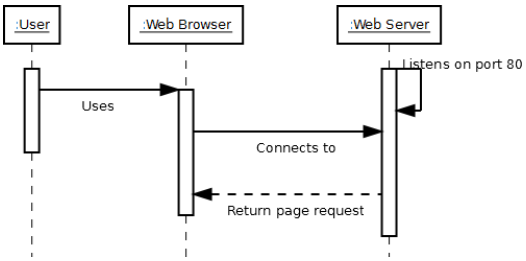


4.2 Description

A user is in control of operating the system, doing such things as interating with the Game Manager and looking up user history. When a person wants to register or sign-in to the site, they are interacting with the Form, not directly to the database. The form is in control of handing the entered info(either login info or registration info) to the database. When the user requests to see a game, The Game Manager is pinged to pull up specific information, either Past Games or Live Games, recieving information from a Games database. the actual game data is presented from the databases maintained by the server group and outside the scope of this paradigm.

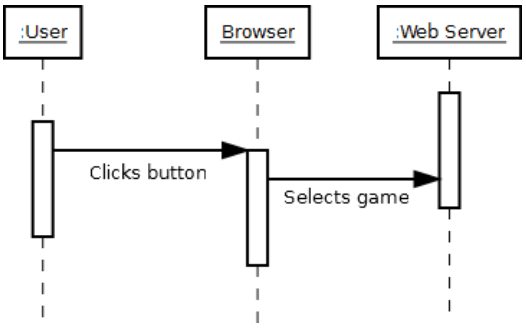
5 Interaction Diagrams

5.1 Navigates to System



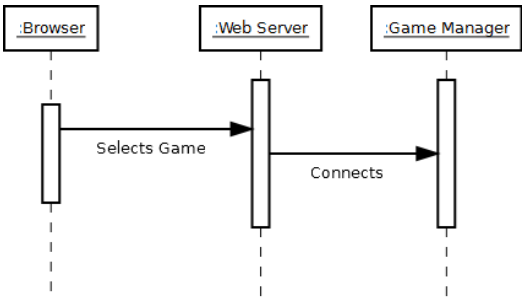
This is a low coupling principle. The user uses their web browser which then interacts with our web server to display any information we intend to send.

5.2 Select Game



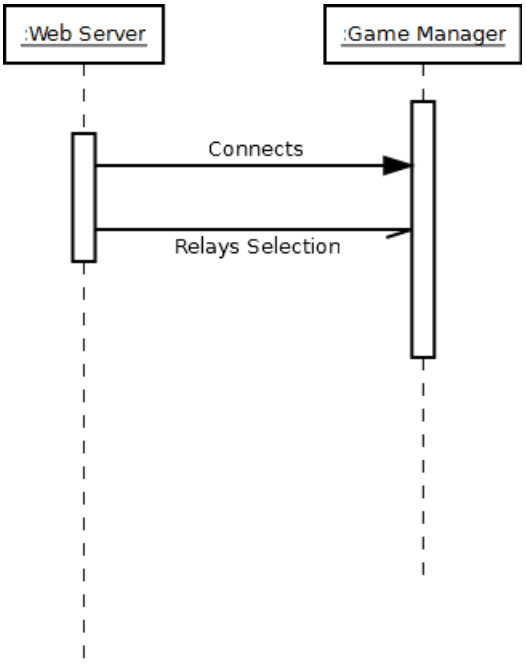
Same principle as above. The user slects a game.

5.3 Connects



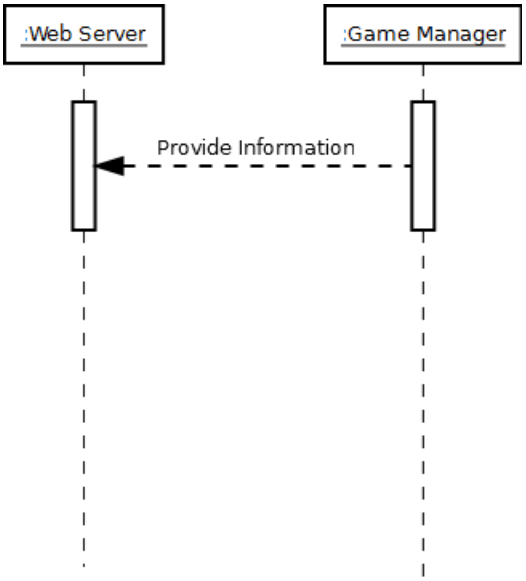
High cohesion, The web server only deals with sending info to the browser, the game manager is responsible for all else.

5.4 Relays game selection



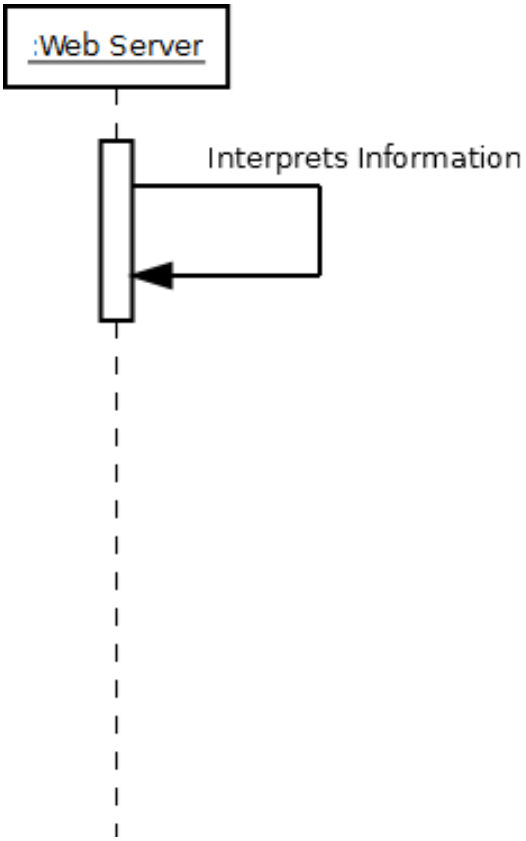
This would full under the controller principle. Web server connected to the game manager and then passed alone the information that game manager needs to continue.

5.5 Provides information



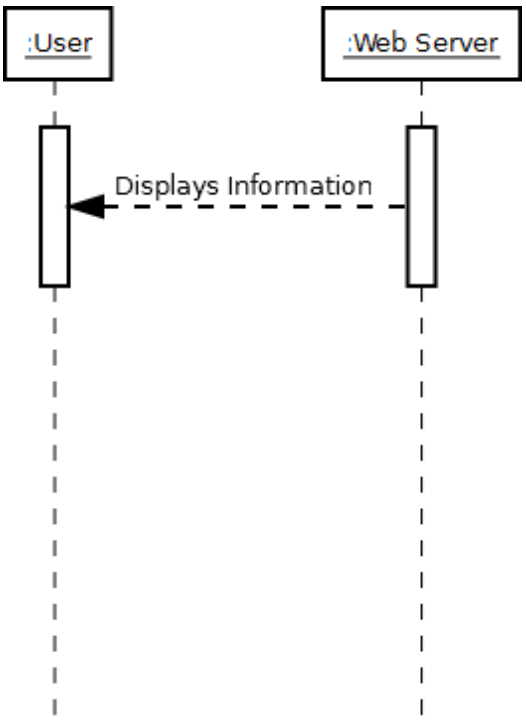
Game manager provides the requested information to the web server to pass alone to user.

5.6 Interpret



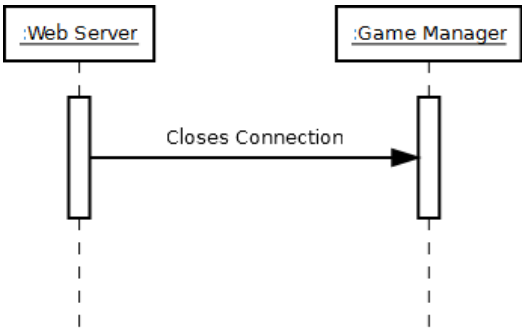
It is now the web servers job to interpret the data that was passed to it.

5.7 Display



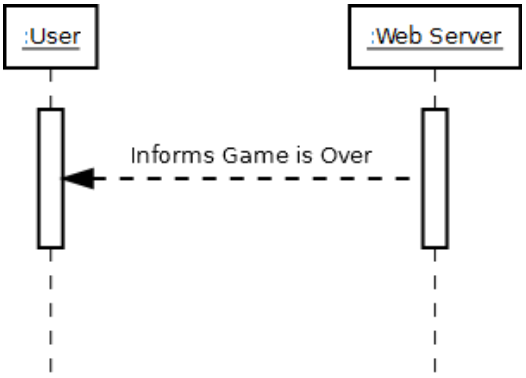
Web server sends the translated information to the users browser to be displayed.

5.8 Close



The task is complete so the web server closes the connetion to the game manager.

5.9 Inform



Web server tells the user it has closed game session.

6 Implementation

6.1 Style Guide

6.1.1 Indentation, Line length and Whitespace

- Use tabs, not spaces, for indentation. Set tabs to equal four spaces.
- Braces should appear on the line below their preceding argument, indented to the same level. All code contained within the braces should be indented by one additional tab.
- The contents of <php? ?> tags should always be indented. The contents of HTML and JavaScript tags may or may not be indented depending on their scope. For example, in HTML the contents of a should be indented while the contents of a should not. In JavaScript, the contents of a <form> should be indented while the contents of an <option> should not. The properties of a CSS selector should be indented.
- Line length should not exceed 85 characters. When using HTML tags to display text, the contents of the paragraph do not need to comply with line length limits. If the line exceeds 85 characters, the tag should be placed on the line below the textual content of the paragraph.
- Methods containing more than 30 lines of code are discouraged. If a method is over 30 lines of code, it can probably be written more efficiently and/or broken down into multiple methods.
- Readability is a priority over small file sizes; use of whitespace is encouraged where appropriate.

6.1.2 Naming Conventions

- Filenames should not contain capital letters. Underscores are only used for pages that pass information from the client to the server. These pages should have a shared, meaningful name followed by _form or _post to indicate whether the page is collecting information (_form) or processing information (_post).

- CSS specifications should not contain capital letters or underscores. Use of dashes is discouraged, though it is allowed if appropriate. ID and class names should refer to the contents rather than the appearance of an element. For example, "errormsg" is preferable to "smallredtext" in the interests of maintainability.

- Variable names in PHP and JavaScript should use camel casing. The names of boolean variables should begin with "is."

6.1.3 Structure of Pages

- All displayable pages on the VirPong website will begin and end with PHP tags to include the header and footer.

- All textual content will be placed within HTML <p> tags, with the exception of headers which will be within the header tag of the appropriate level (e.g. <h1>).

- All HTML and CSS code will comply with the W3C validators in an effort to maximize cross-browser compatibility.

6.1.4 Documentation Conventions

- HTML comments should be formatted like so:
  <!– This is a comment. –>

  CSS, PHP, and Javascript comments may be formatted in either of the following two ways:
  /**
  ∗ This is a comment.
  ∗ Generally used at the beginning of a block of code to explain
  ∗ its overall function in plain English.
  ∗/
  or
  // This is a comment.
  // Generally used within blocks of code to explain how certain
  // elements are being used.
  // Generally does not take up more than one line – if you
  // have that much to say, consider a block comment.

6.1.5 HTML Specific

- All HTML tags and attributes will be entirely in lowercase.
- All opening HTML tags will be accompanied by a corresponding closing tag. When using multiple tags, all elements will be properly nested (e.g. text rather than text). All singleton tags will be closed with a space and an ending slash at the end of the tag (e.g.
).
- Attributes inside of HTML tags should be contained within double quotes. There should be no spaces between the attribute, the equals sign, and the definition. When defining multiple attributes, they should be separated by a single space.
- All image tags will contain an alt attribute.
- Div layers are generally preferable to tables and iframes. Divs will be used to display the main site layout. Tables should only be used in select cases, such as to display the contents of the database. Iframes should never be used.

6.1.6 CSS Specific

- All CSS will be contained in the external stylesheet rather than included as inline style.
- When defining styles for specific elements, think carefully about whether the selector should be an ID or a class. Elements that are only used once on each page (e.g. the content div) should use IDs; elements that may be repeated (e.g. the errormsg) should use classes.
- All text sizes should be specified using ems, not pixels.

6.1.7 PHP Specific

- Control statements (if, for, while, switch, etc.) should have one space between the control keyword and the opening parenthesis, to distinguish them from function calls.

- Long if statements may be split onto several lines to comply with line length limits. The conditions should be indented by one additional tab with the logical operators (e.g. &&) at the beginning of the line. The first condition may be indented to align with the others. The closing parenthesis and opening brace get their own line at the end of the conditions.
- Literal strings should be contained within singled quotes. When a literal string contains apostrophes, it should instead be contained within double quotes. SQL statements may be contained within double quotes whether or not they contain apostrophes.
- String concatenation will be done using the"." operator with spaces on both sides. When string concatenation exceeds line length limits, the statement should be broken up such each successive line begins with the "." operator aligned under the initial "=" operator.

6.1.8 JavaScript Specific

- Any JavaScript code that defines functions should be kept in an external .js file.

6.1.9 Sources

- http://na.isobar.com/standards/
- http://pear.php.net/manual/en/standards.php

6.2 Design Patterns

6.2.1 Home Link

Intent

To always provide the user with a simple path to the website's home page.

Implementation

The VirPong logo in the upper left corner of the layout is a hyperlink to the VirPong home page. This is present on every page.

Consequences
Allows users to navigate to the home page no matter what page they start at. This is especially helpful if users arrive at the website via a search engine or an emailed link to a specific page.

Related Patterns
Main Navigation, Footer Bar

6.2.2 Main Navigation

Intent
To always provide the user with hyperlinks to all areas of the website.

Implementation
The horizontal bar at the top of the layout is a menu containing drop-downs with links to different pages of the VirPong website. This is present on every page.

Consequences
Allows users to navigate to a page of their choosing no matter what page they start at.

Related Patterns
Home Link, Footer Bar

6.2.3 Footer Bar

Intent
To always provide the user with hyperlinks to important informational content regarding the use of the system.

Implementation
The centered bar at the bottom of the layout is a footer containing links to important information regarding the services that VirPong provides. These links are: about us, contact us, privacy policy, terms of use, and code of conduct. This is present on every page.

Consequences
Allows users to directly access information about the service no matter what page they start at.

Related Patterns
Home Link, Main Navigation

6.2.4 Account Registration

Intent
To only display protected content to users who have registered with the service. To store user information which can later be used to enhance the user experience.

Implementation
Users must fill out the registration form before they can play VirPong games, participate in tournaments, or use the chat feature.

Consequences
Allows for personalization of the VirPong experience, including display of personal player history information, the ability to chat with other users, email notifications regarding tournament participation, and happy birthday emails. Trade-offs include potential loss of users who are discouraged by the registration form.

Related Patterns
Lazy Registration, Log In, Input Feedback

6.2.5 Lazy Registration

Intent
To allow users to become familiar with the service before requiring them to register.

Implementation
Users can watch VirPong matches (both live and past matches) and view high score information without being registered or signed in to an account. Users can also access all informational content (about us, contact us, privacy policy, terms of use, code of conduct, rules, system requirements, news) without being registered or signed in to an account.

Consequences
Allows users to try out the system with little immediate commitment, making registration seem like more of a choice than a chore. Allows

users to be already invested in VirPong by the time they register for an account, making it more likely that our registered users are active uses. Trade-offs include potential loss of registered accounts by users who only utilize features that do no require registration.

Related Patterns
Account Registration

6.2.6 Log In

Intent
To identify a registered user in order to properly personalize their experience.

Implementation
The main navigation contains a link to the log in form, allowing registered users to authenticate into the system whenever they desire. If an unauthenticated user attempts to access a page that is only available to registered users, it will display a prompt asking the user to log in before proceeding.

Consequences
Allows for personalization of the VirPong experience, including unlocking access to certain pages that are only available to registered users.

Related Patterns
Account Registration

6.2.7 Input Feedback

Intent
To communicate with users about information they are submitting to the service.

Implementation
All improperly filled out forms will generate specific error messages, whether through Javascript alerts or inline text. All properly filled out forms will lead to a success message, confirming that the user's submission was successful. This design pattern applies to the registration, log in, and account settings forms.

Consequences
Prevents improper information submission by alerting the user immediately and directly of any errors. Allows the user to feel confident upon receipt of success message that their information has been passed to the system.

Related Patterns
Account Registration, Good Defaults

6.2.8 Good Defaults

Intent
To prevent users from having to type any more keystrokes than strictly necessary by prefilling certain form values to anticipate the user input.

Implementation
The account settings form automatically fills in the user's current first name, last name, email address, birthday, and gender. The user can then edit any fields he wants to change and leave any fields he wants to remain the same.

Consequences
Eliminates the need for the user to reenter all of their information in order to change any field, resulting in significantly increased convenience for the user. Trade-offs include reminding users that we have their personal information and we know how to use it.

Related Patterns
Input Feedback

6.3 Installation

There are a few requirements to installing and using our application. Our development platform is Ubuntu and the following are requirements for the installation:

Install a LAMP Server
A LAMP server is composed of a Linux environment, and installations of Apache, MySQL and PHP. However, this step is outside the scope

of our tutorial, but you may follow this guide:
http://www.howtoforge.com/ubuntu_debian_lamp_server.

Install git
Both our application and Node.js are hosted on git, so it is important to install git on your machine. Installation of git will be covered in the next requirement.

Install Node.js
This is how we are going to run our Javascript on the server. You can follow the directions here to install Node.js:
http://howtonode.org/how-to-install-nodejs.

Install NPM
NPM is a package manager and has become the standard for installing node libraries. We are going to use this to install socket.io, and you can install NPM by entering the following command in your terminal window 'curl http://npmjs.org/install.sh — sh'.

Install Socket.io
This is how we are going to create bidirectional communication between the server and browser. To install, enter the following command in your terminal window: 'npm install socket.io'.

Prior to installing the VirPong web application, the database needs to be configured to match the database used with the VirPong game. Once the MySQL database is installed, do the following:

1. Log into your MySQL database. From your terminal window:
   (a) enter 'mysql -u <your_username> -p'
   (b) press enter
   (c) enter '<your_password>'
   (d) press enter
2. Now that you're logged into MySQL, the database needs to be created. From the MySQL prompt, enter the following command:
   'CREATE DATABASE db2'

3. This creates the database, but tables must be created within the database. To create the three tables that are used in the VirPong game, enter the following three commands into the MySQL prompt

   (a) create table GamesPlayed (gameID INT(10) NOT NULL AUTO_INCREMENT, username1 VARCHAR(50) NOT NULL, username2 VARCHAR(50) NOT NULL, score1 TINYINT NOT NULL, score2 TINYINT NOT NULL, win TINYINT NOT NULL, INDEX index1(gameID), CONSTRAINT FK_customer FOREIGN KEY(username1) REFERENCES customer(username), CONSTRAINT FK_customer_2 FOREIGN KEY(username2) REFERENCES customer(username));

   (b) create table Customer (username VARCHAR(50) NOT NULL, password VARCHAR(50) NOT NULL, firstname VARCHAR(50) NOT NULL, lastname VARCHAR(50) NOT NULL, email VARCHAR(256) NOT NULL, birthday DATE NULL, gender TINYINT NULL, PRIMARY KEY(username));

   (c) create table Registration (gameID INT NOT NULL AUTO_INCREMENT, username1 VARCHAR(50) NOT NULL, username2 VARCHAR(50) NOT NULL, gameTime DATETIME NOT NULL, tournamentID INT NULL, INDEX index1(gameID), CONSTRAINT FK_customer FOREIGN KEY(username1) REFERENCES customer(username), CONSTRAINT FK_customer_2 FOREIGN KEY(username2) REFERENCES customer(username));

4. With all three tables created, exit the MySQL prompt to continue the installation process, by entering:
   'exit'

With all of the requirements installed as well as the MySQL database, we can begin running the application. To get the source code for our application:

1. Navigate in your terminal window to your Apache root folder (by default, this is /var/www):
   'cd /./var/www'

2. Clone our source code by issuing the following command in your terminal window:
   'git clone git://github.com/VirPong/human-pong.git'

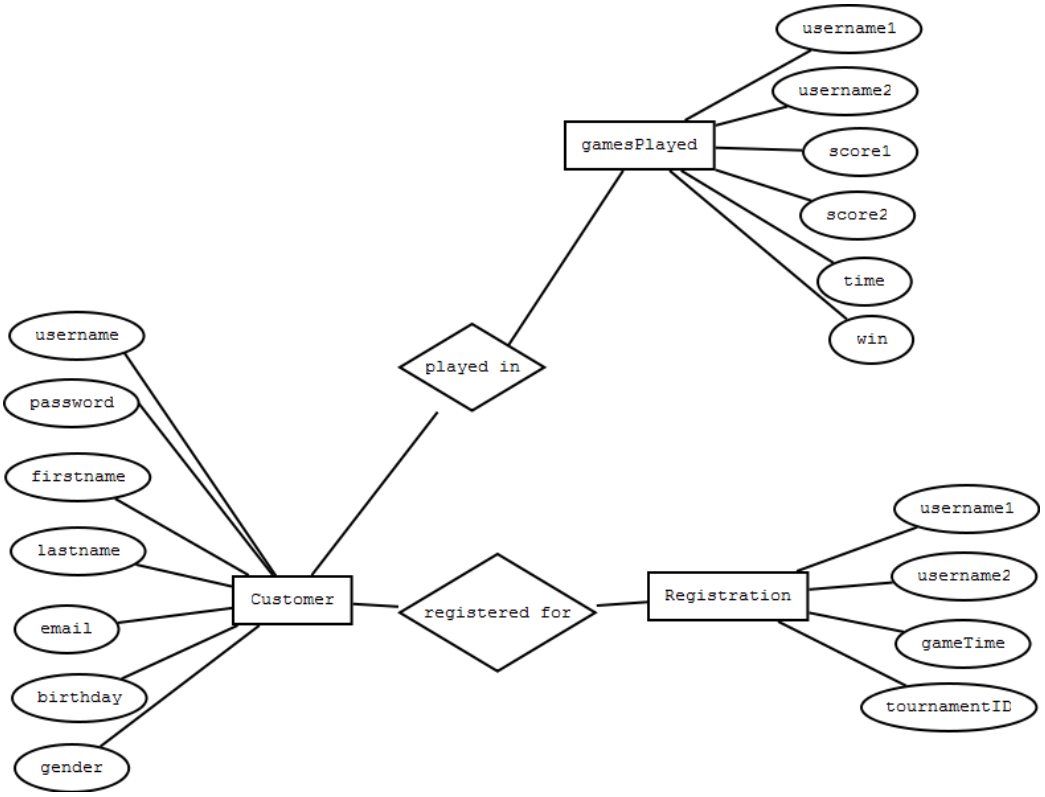
3. Navigate to into the Communication folder:
   'cd human-pong/WebUI/site/http/communication'

4. Run the VirPong web server by issuing command in your terminal window:
   'node server.js'
   Note: The client.js code may need to be changed depending on your system configuration. Simply change open the client.js file and change the IP address to the address of your server. ex: 'socket = io.connect('xxx.xxx.xxx.xxx');' With your server's address in the parenthesis

5. To view the application in use, point your web browser to: 'http://localhost/human-pong/WebUI/site/http/communication/index.html'

To incorporate this HTML file into your own application, simply follow the steps above and create a link from your website to the Live Game page by making a link in your HTML file to the index.html file.
To find further instructions on how to install the VirPong platform, or to find other resources that pertain to VirPong, please visit our repository and wiki page at: https://github.com/VirPong/human-pong

6.4 ER Diagram



6.5 Data Storage

Within the website's database, we have established three different classes of information that we will be storing: information provided by the client that is related to the world outside of VirPong, information specified by the client that will identify the client within the VirPong world, and information recorded about game play or game statistics. Among the information in the personal class, is their full name, their email address, their birthday, and their gender. Our website's database second class stores information that is unique to our clients and is just as sensitive as their personal information. Among this class of information is our client's username, their password, the games that they have played, and the times that they have played these games at. The rest of the data that we store in the website's database makes up a third class of information, information that pertains to VirPong games.

The data associated with this class includes unique game ID's, the game times, and scores of games, and who won a specific game. Although we have three specific classes of data that we are storing, some of the data exists in multiple classes, such as the times that users have played games, and their specified username and passwords.

6.5.1 Database Schema:

Table: Customer twice

Field	Type	Null	Key	Default	Extra
username	varchar(50)	NO	PRI	NULL	
password	varchar(50)	NO		NULL	
firstname	varchar(50)	NO		NULL	
lastname	varchar(50)	NO		NULL	
email	varchar(256)	NO		NULL	
birthday	date	YES		NULL	
gender	tinyint(4)	YES		NULL	

username: This is the primary key for the Customer database and therefore cannot be null. This will store the user names specified by our customers.

password: This field stores the password for the customer. This field cannot be null because it is important for us to maintain the ability to authenticate all users.

firstname: This field stores the user's first name. This field cannot be null because we would like to identify our users by sending them personal messages.

lastname: This field stores the user's last name. This field also cannot be null because we would like to also include the last name of our customer in our personal messages.

email: This field stores the email address for the user. We will be sending notifications of future games and must have the ability to contact our customers, therefore this field cannot be null.

birthday: This field will store the birthday of our customer. This field can be null if the user would like.

gender: This field stores the gender of the customer. This field can be null if the user would like.

Table: GamesPlayed

Field	Type	Null	Key	Default	Extra
gameID	int(10)	NO	MUL	NULL	auto_increment
username1	varchar(50)	NO	MUL	NULL	
username2	varchar(50)	NO	MUL	NULL	
score1	tinyint(4)	NO		NULL	
score2	tinyint(4)	NO		NULL	
win	tinyint(4)	NO		NULL	

gameID: This is the key for the GamesPlayed table and it allows us to uniquely identify each game played. This field is automatically incremented by the database.

username1: This stores the user name of one of the customers that played in this match. This will never be null since we know the users that played in the game.

username2: This stores the user name of one of the customers that played in this match. This will never be null since we know the users that played in the game.

score1: This will store username1?s final score. This field will not be null since we need to know the player?s scores.

score2: This will store username2?s final score. This field will not be null since we need to know the player?s scores.

win: This field will denote the winner, with either a 1 or a 2 depending on the number of the user that won, of the game played.

Table: Registration

Field	Type	Null	Key	Default	Extra
gameID	int(11)	NO	MUL	NULL	auto_increment
username1	varchar(50)	NO	MUL	NULL	
username2	varchar(50)	NO	MUL	NULL	
gameTime	datetime	NO		NULL	
tournamentID	int(11)	YES		NULL	

gameID: This is the key for the Registration table and it allows us to uniquely identify each upcoming game. This field is automatically incremented by the database.

username1: This stores the user name of one of the customers playing in this match. This will never be null since these are established future game with two users.

username2: This stores the user name of one of the customers playing in this match. This will never be null since these are established future game with two users.

gameTime: This field stores the date and time for the future match to occur.

tournamentID: This field is used to denote whether a game is part of a tournament. By allowing a null value for tournamentID, we are able to show that a game isn?t a tournament game, and when the game is a tournament game, we can specify that it is a tournament game, as well as which tournament game, by inserting that tournament's numeric ID.

Reasoning for this design choice: We designed our database this way so that we have tables that are specific to the role that the tables are playing. This path was chosen because we did not want to have all of our data stored in a single table, which would cause inconsistencies in inserting data into the table. The Customer table is created with information that is specific to identifying our client, as well as personal information that will benefit their VirPong experience. The GamesPlayed table contains information about games that have already been played, including the final score of the game, and who won the game. The Registration table contains data about upcoming games.

In our Customer table, we have elected to store a client's identifying information such as their first name, last name, and user name. We also store a password, because we want to maintain that people that are logged into our site are registered users. We also store information such as email addresses, to notify our clients of upcoming events or registered games, as well as client birthdays to wish clients happy birthdays. The email field is also 256 characters long, because this is the longest possible length of an email address.
The Games Played table holds information specific to games that have been played in the past. Within the table, we store the ID of the game that has been played, so we can uniquely identify games. We also store the names of both users so we can display the players in a certain game, as well as the win field, which references the winning player, so we can determine game statistics. The table also stores the score for each player so we can follow the scores of certain players, as well as the scores for an individual game.
The last table, Registration holds data that is used to plan a game in advance. We use a game ID to uniquely identify the games in question, but also require the use of a Game Time so that we can remind players when a game is upcoming, as well as to have absolute times for when games are to occur. We also store the names of both players, which we can use to reference the Customer table, to send notifications to the players, as well as advertise which players are about to play a game. The last field, the tournament ID is also used to tell whether or not a certain game is part of a tournament. The field allows an input of null, so when a certain game is not part of a tournament, we can specify null.

6.6 Testing and Verification

The testing and verification for the VirPong website will be through validators and various kinds of testing. All the HTML and CSS code will be ran through the W3C validators available online. They will check for the indicated doctype syntax compliance of all of our scripts and compare them against the XHTML 1.1 standard. For all other types of scripts we will be using Googles developer tools in Google Chrome to see resource load times and any errors thrown by the scripts.

We plan on performing four types of testing upon our code: unit, integration, system, and regression. For the unit testing we are going to use a program called QUnit. We will then uses integration testing across multiple

departments of code. Then we will do some system testing against all of our requirements. In this stage of testing we will test scripts in the latest versions of Microsofts Internet Explorer, Mozillas Firefox, Apples Safari, and Googles Chrome on a Windows 7 machine and a Ubuntu distribution of Linux. We will also report on each functionalities success on each platform and browser to report to the user. Lastly we will perform regression testing when we make additional releases of the software to make sure they implement all functions.

7 Reflections

7.1 Challenges

7.1.1 Server/Client Communication

One of the main challenges that we faced in creating a method of viewing VirPong games was in the actual communication between the server and the client. In the past, there have only been a handful of ways that developers have tackled the issues of browser communication including the use of making a connection via a port opening by Flash, or through Reverse AJAX. However, both of these techniques had disadvantages that we wanted to avoid, so we elected to use a newer technology, WebSockets, specifically the Socket.io framework for Javascript. The challenge that came along with this route is the newness of Socket.io as well as Node.js, which we use in conjunction of Socket.io. Looking for online tutorials, we found many tutorials that used both of these technologies in different situations, as well as samples of code that used the two technologies in conjunction with other frameworks, making it difficult to get an understanding of how Node.js and Socket.io worked independently of these other frameworks. To overcome the issue of adopting these two new technologies, our team focused on trying to re-implement tutorial code as we found it, and upon finding working tutorial code, we would manipulate the code in attempts to break the code down into more simplistic forms. Going through this process, our team was able to achieve very simple recreations of tutorial code that we had found, providing us with an understanding of what is required in using Node.js and Socket.io.
While this technique worked for our team, and we were able to break tutorial code down into simplified forms, and from these simplified forms determine the best way to structure our own code, we believe that we may have saved some time by investigating our options further. During this process, we explored many different methods of communication between the client and the server, however, we were eager to begin coding and would often try to implement a piece of code before pursuing alternatives. We believe that if we had pursued other examples or pieces of tutorial code, we could have shortened the amount of time, and potentially frustration, that we used to develop a working implementation of server to client communication.

7.1.2 Large Workload

One challenge that we anticipated in the Initial Plan was the sheer number of tasks that lay before us in creating a functional, attractive web user interface for VirPong. Identifying this issue early on was a key to the way we have handled the workload. By always keeping in mind the magnitude of our task, we stay motivated to produce results each and every week.

In hindsight, this challenge may have been easier to deal with if we had planned our initial schedule more carefully. One of our proposed strategies for tackling the workload was to stick tightly to our schedule, but this proved unrealistic. Often our features relied on the existence of features from other departments – for example, we cannot code and test pages that read information from the database of past games until we have discussed with the server department who is responsible for writing this information to the database and what the schema of the database will be. Had we been more aware of the realities of this situation, we would have placed all such features that rely on other departments further down the schedule, and we would have communicated our needs to other departments before the submission of the initial plan so that they might consider them when designing their own schedules.

Because our schedule was at times unhelpful, rather than following it to the letter we have taken care to be proactive about getting things done. In particular, Kyle and Katie often found that the tasks assigned to them were unfeasible at the present time. Rather than shrugging it off and making no progress for the entire week, their response was to seek out alternate tasks that could be completed. In this way, we implemented Javascript validation of forms and a complete site layout more quickly than anticipated in the original schedule.

Our other main strategy in conquering our heavy workload was splitting our team into pairs that worked together on shared tasks. In general, Aryn and Garrett have been configuring the database and working toward communication with the server while Katie and Kyle have been writing pages that access the database and focusing on the aesthetics and organization of the website. We have been happy with the buddy system, as it simultaneously holds us accountable for working on our assigned tasks and gives us another bright mind to bounce ideas off of. Working in pairs has also been quite practical, as it allows our team to tackle two problems at once. Since many of our meetings are with only the pair as opposed to the whole group, they are easier to schedule into our busy lives.

8 References

CSS - http://w3schools.com/css
Cascading Style Sheets is a style sheet language that can be applied to HTML. We will be using CSS to create a consistent, appealing layout across all our pages that include the colors and fonts used plus the menuing system for the entire site.

CSS menuing system - http://www.grc.com/menu2/invitro.htm
For implimenting the CSS menuing system we used the work done by Steven Gibson to help us. The goal of his project was to avoid the use of any javascript and try and maintain browser and platform compatability.

Github - http://github.com
Github is a hosting system for the version control system Git. We will be using Github to collaborate with each other and with the other Vir-Pong development teams.

GDocs - http://docs.google.com
Google Docs is a web-based, collaborative document editing service. We will be using GDocs to share information between team members.

HTML - http://w3schools.com/html
HyperText Markup Language is the standard markup language for web pages. We will be using HTML in conjunction with CSS and PHP to create the layout of our website.

HTML5 - http://w3schools.com/html5
HyperText Markup Language 5 is a revision of the HTML standard that includes new multimedia content. We will be using HTML5 in conjunction with JavaScript to display games of Vir-Pong on the web.

HTTP/HTTPS
Hyper Text Transfer Protocol/Hyper Text Transfer Protocol Secure is the protocol for sending web-pages or other information across a network. HTTPS is the SSL encrypted version, providing secure communication between the web user and the server when sensitive data is being handled.

JavaScript - http://w3schools.com/js
JavaScript is a dynamic, client-side scripting language. We will be using JavaScript in our forms, including validating those forms to enhance Vir-Pong's security. We will also use JavaScript in conjunction with the Canvas element in HTML5 to produce our game display.

JWebSockets - http://jwebsocket.org/
JWebSockets is both a Java and a JavaScript implementation of HTML5's WebSockets. The use of WebSockets allows for full-duplex communications between a server and a set of clients, including websites, as well as mobile devices. We will be using JWebSockets to push information from the server to the website clients, which will include features such as live streaming of Vir-Pong games, as well as the potential for a client chat system.

Coda-Slider - http://www.ndoherty.biz/2009/10/coda-slider-2/
For the cool sliding effect on the websites home page.

LaTeX- http://latex-project.org
LaTeXis a document markup language for TeXtypesetting. We will be using LaTeXto format our printed documents, such as weekly manager reports and this final report.

MySQL - http://dev.mysql.com/downloads/mysql/
MySQL is type of SQL database that stores relational information well. We are using a MySQL database to store all of our information we collect from the user, game information, and future game times.

Photoshop - http://photoshop.com
Adobe Photoshop is a graphics editing program. We will be using Photoshop to produce mockups of our site layout. We will also use Photoshop to render the Vir-Pong logo and any other graphics used on our website.

PHP - http://php.net
PHP: Hypertext Processor is a server-side scripting language that integrates smoothly with HTML. We will be using PHP on all of our web pages. We will use PHP to insert the header and footer of our layout on each page. We will also use PHP to make database queries, such

as when registering a new member with our system or retrieving user information (e.g. high scores, account settings).

PHP Designer - http://mpsoftware.dk/phpdesigner.php
PHP Designer is a text editor designed specifically for writing PHP, HTML, CSS, and JavaScript. It contains useful tools for testing and debugging code in all four of these languages. We will be using PHP Designer to write much of our code for this project.

Powerpoint - http://office.microsoft.com/en-us/powerpoint
Microsoft Powerpoint is a commercial presentation program that is part of the Microsoft Office suite. We will be using Powerpoint for our weekly manager presentations in order to provide the rest of the company with a visual representation of our progress.

Various browsers -
Internet Explorer: http://microsoft.com/ie
Mozilla Firefox: http://mozilla.org/firefox
Google Chrome: http://google.com/chrome
Safari: http://apple.com/safari
Web browsers allow users to navigate the World Wide Web. We will be using browsers to test our website for display and usability. For this reason, it is important that we test in all of the most popular browsers currently in use. This enables us to be sure that our site displays properly to all users, thereby keeping our client base as wide as possible.