Humans vs. Zombies
Game Management System

Connor Kelly

Final Report

8 December 2009

Table of Contents

Application Development

Project Concept and Description

(Revised 2009-09-26 for grammar.)

- This project is set up to allow for the management of a game of Humans vs. Zombies. The reasoning behind the creation of such a project is that the management of a game of Humans vs. Zombies requires a lot of overhead when it comes to keeping track of all that is happening simultaneously and there are very few, if any, viable solutions currently available that make the tracking of this information feasible.
- The goal of this project is to provide an easily customizable and deployable application that can be installed on a web server with PHP and MySQL to provide a way to easily manage and maintain a game of Humans vs. Zombies.
- Expected Functionality:
  - Allow registration of players.
  - Manage players (change player information and possibility of banning unruly players).
  - Tag reporting for the game.
  - Configuration of settings including stun times, conversion times, starvation times, time zones, etc.
  - Administration interface into all this information.
  - A “Game-Flow” mechanic to change the game phases.
  - Allow the altering of game rules.
  - I currently have a working Game Management System (hereafter called GMS) that provides some of the above functionality, but it is built specifically for Berea College and is not customizable in the least. Goucher College provides their own GMS for schools around the nation but provides only minimal functionality and customization (can change rules, main page, edit players, track tags and change game flow. That's it. No real customization).
- As stated already, there is at least Goucher's system available on the internet (the only one that could be located at the time of this writing) that is widely available for schools. Therefore, this project is not entirely original. However, the amount of customization that will be built into the system would be a great improvement over what is currently available.
- If time permits, this project does provide further development opportunities beyond the course. Possibly in the future, support could be added for other database systems, Drupal connectivity, or the integration of blog/forum/chat support into the system to allow players to directly communicate with each other.
- The construction of a *secure* system is expected to be very challenging. This means preventing SQL injection attacks, password/cookie stealing, and general illegal access. So the GMS must be able to parse and secure all data entered on forms along with encrypting any sensitive information (passwords) *prior* to sending that information across the internet.
- Measures of Success:
  - If this project can be installed on a server and used to run and manage a quick mock game of Humans vs. Zombies without any problems and users can navigate the site effectively and efficiently, then the project has been a success.
- Possible project extensions (time-permitting):

- Blog support. This would allow for an updating news feed for the application.
- Forum support. This would give users a place to communicate with each other over topics.
- Chat support. This would give users the ability to chat in real-time with each other.

Vision and Scope

(Revised 2009-10-10. Removed unneeded sentences.)

- As the world of Humans vs. Zombies stands right now, game administrators have very few options over how to manage hundreds or even thousands of people playing the game, leaving the administrators at the mercy of the players. With the release of the Humans vs. Zombies Game Management System (GMS), this will be turned completely around. Administrators will finally have the control that they need to be able to manage a group of people that vastly outnumber their own. The Humans vs. Zombies GMS will enable the spread of the Humans vs. Zombies game as it will be easy for anyone to be able to start managing their own game.
- The Humans vs. Zombies GMS will be a standalone web application that will allow for the efficient management of a game of Humans vs. Zombies. It is designed to keep track of player information and all tags reported during the game along with allowing for the updating of players, changing of game information and settings, and the registration of players. Outside of these essential functions, the Humans vs. Zombies GMS may eventually include support for databases outside of MySQL; a built-in forum, blog, and/or chat interface; and possibly a way to integrate the GMS into a Drupal installation.

Motivation

- This project is important to me as I have been an administrator of a game of Humans vs. Zombies for a couple years now. I wrote the system that Berea College currently uses to manage their game and it has been a lifesaver. Without it, it would have been near impossible to keep the game running here. So I felt that if I could build something that was extremely customizable and flexible and provide it to other schools, they would also benefit from having such a system, thus making it easier for new games of Humans vs. Zombies to be set up around the nation.
- This project will help me immensely with my career path and my own professional growth. Through this project, I will doing full comment documentation within the code, something that would be important at any job writing software and applications. I will also be learning programming aspects to web security, something that is important for writing web applications. I will also be continuing to improve my PHP and MySQL abilities, which are two things that are commonly used within the world of online application development (others include ASP.NET and MSSQL).
- The intended audience of this project is students of colleges around the nation wishing to start their own game of Humans vs. Zombies. Users will benefit from this project by being able to set up and keep track of their own game of Humans vs. Zombies in very little time, allowing for more administrative time being devoted towards improving the game and coming up with mission ideas for players. They will also benefit by having an easy way to report tags for the game.

Resources

(Revised 2009-10-17. Added MD5 script.)

- The tools that will be used for the Humans vs. Zombies GMS are as follows:
  - PHP. The base language that the system will be written in.
  - MySQL. The database system that will drive all storage of data.
  - JavaScript. Used for certain functions that PHP can do but only with a page submission, which adds extra server load.
  - Notepad++. This program will be used to write the code for the Humans vs. Zombies GMS.
  - Doxygen. This program will be used to pull the comments from the code and create a documentation guide of the uses of all functions and classes within the Humans vs. Zombies GMS. That way in the event that I do not get around to working on the project for a while or if someone else wants to help with the project, they can see what is already implemented and how everything works.
  - WAMP. This is a program that will provide a local server testing environment so that I can see how the website would actually appear and operate on a real web server.
  - MD5 JavaScript implementation – JavaScript has no built-in MD5 checksum but it is needed for the GMS to function properly, so Paul Johnston's script is used under the BSD license.

Preliminary Software Requirements Specifications

(Revised 2009-10-10. Added FR-12.)

- Functional Requirements (FR):
  1. GMS must be able to keep track of a variety of configuration options that can be changed by an administrator.
     - Evaluation: If there exists a variable or configuration page in the administration section of the GMS and it allows for any configuration option stored in the database to be changed, then this requirement is fulfilled.
     - Dependency: FR-7
     - Priority: Essential
  2. GMS must be able to keep track of a variety of information regarding players.
     - Evaluation: If the GMS has pages that display all of the information regarding players (a full display in the administration interface and a partial display in the player and main interfaces), then this requirement is fulfilled.
     - Dependency: FR-7
     - Priority: Essential
  3. GMS must be able to change player information as needed.
     - Evaluation: If the GMS has pages that allow for the editing/deleting of specific players in the system, then this requirement is fulfilled.
     - Dependency: FR-2, FR-7
     - Priority: Essential
  4. GMS must be able to keep track of all tags that players make and report.
     - Evaluation: If the GMS has pages that show all of the tags that a player has made and a page that allows for players to report their tags, then this requirement is fulfilled.
     - Dependency: FR-3, FR-7
     - Priority: Essential
  5. GMS must automatically update player information when tags are reported.
     - Evaluation: If the GMS contains specific functions that change the information of players whenever they submit a tag so as to show that a player made a tag and that another player

was tagged, then this requirement is fulfilled.
    - Dependency: FR-3, FR-7
    - Priority: Essential
6. GMS must automatically update player information when specific time requirements are met by the player.
    - Evaluation: If the GMS contains specific functions that change the information of players whenever specific time requirements have been met, then this requirement is fulfilled.
    - Dependency:FR-3, FR-7
    - Priority: Essential
7. GMS must be divided into at least 3 sections: Administration, Main, and Players.
    - Evaluation: If the GMS contains at least three sections, specifically a section for administrators, a section for players, and a section for those people who are not logged in, then this requirement is fulfilled.
    - Dependency: None.
    - Priority: Essential
8. GMS must provide an easily navigated interface into all of the pages.
    - Evaluation: If a user can sit down and effectively navigate the site with little or no help whatsoever, then this requirement is fulfilled.
    - Dependency: None.
    - Priority: Essential
9. GMS must provide a "Game Flow" mechanic that allows administrators to change which game phase is the current one.
    - Evaluation: If the GMS contains a page that allows for the alteration of the game phase by the administrator, then this requirement is fulfilled.
    - Dependency: FR-1
    - Priority: Essential.
10. GMS must provide mechanisms for validating user input.
    - Evaluation: If the GMS contains a class or similar functions that allow for the validation of data through Regular Expressions, then this requirement is fulfilled.
    - Dependency: None.
    - Priority: Essential.
11. GMS must provide security measures to protect the database and user input.
    - Evaluation: If the GMS contains classes or similar functions that allow for the securing of the database from SQL injections and that encrypts important user information, such as passwords, then this requirement is fulfilled.
    - Dependency: FR-3
    - Priority: Essential.
12. GMS must provide a mechanism to allow players to register.
    - Evaluation: If the GMS can register players on its own, this requirement is fulfilled.
    - Dependency: FR-3, FR-10

- Non-Functional Requirements (NFR):
    1. Must be able to install and run on a server running Apache, PHP, and MySQL.
        - Evaluation: If the GMS can be uploaded and installed successfully on a server with this setup, then this requirement has been fulfilled.
        - Dependency: None.
        - Priority: Essential.
    2. Must appear and act exactly the same in Firefox as in Internet Explorer so as be cross-browser

compliant.

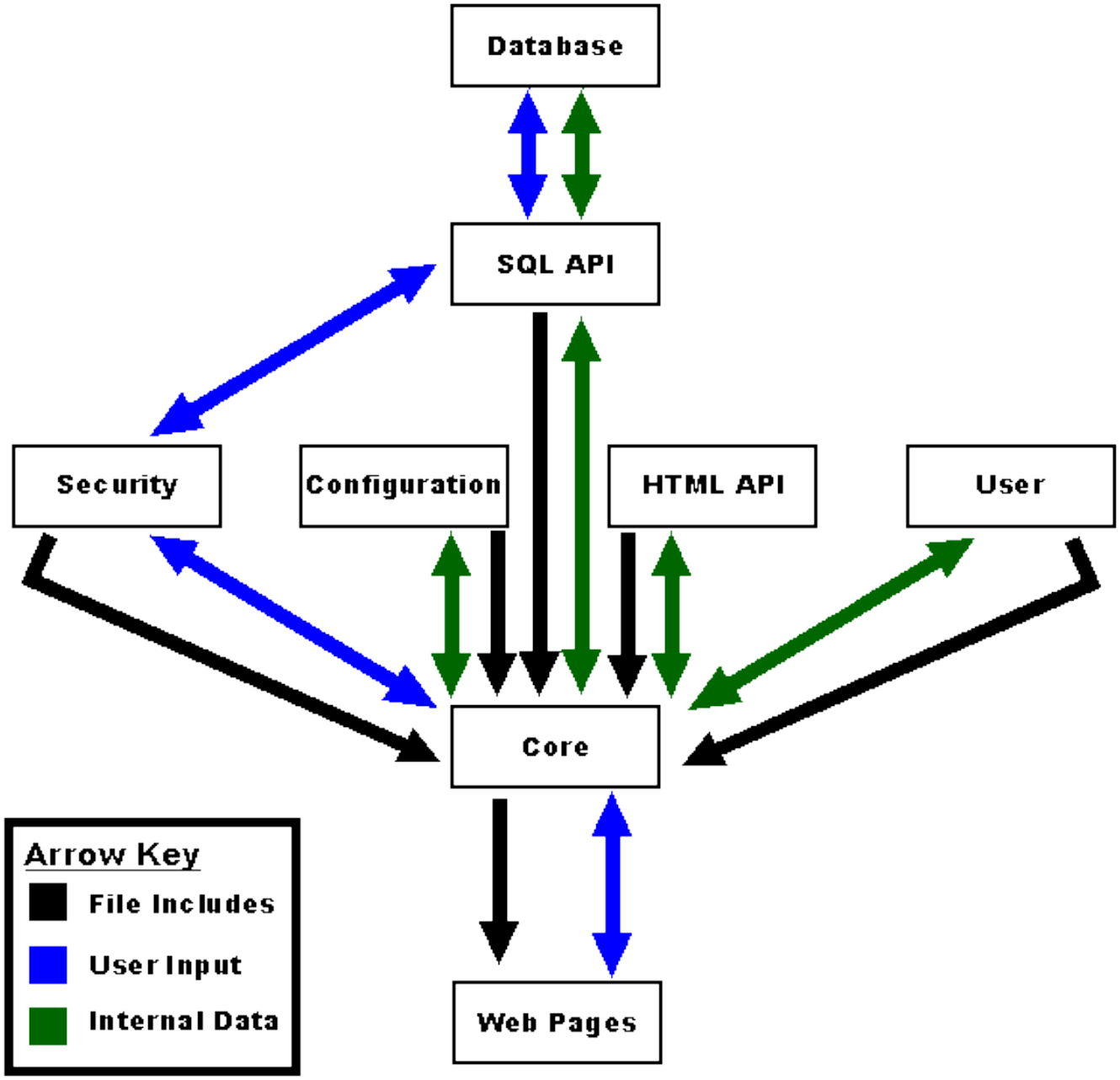
- Evaluation: Test an installed setup in both Firefox and IE. If they work the same, this requirement has been met.
- Dependency: None.
- Priority: Essential.

System Design and Architecture

- The Humans vs. Zombies GMS will be a complete system that will do the following:
  - Control user data: The GMS will be able to control and manipulate user data as needed. This includes updating player information and registering players.
  - Secure data: The GMS will encrypt data across connections, securing any information that could be important (such as passwords).
  - Dynamic page generation: The GMS will be able to construct its pages based on predefined functions, thus making the addition of new pages to the GMS simple.
- The entire Humans vs. Zombies GMS is broken down into the following categories of classes and pages:
  - SQL API: Controls all data flow to and from the database.
  - Security: Deals with all aspects of logging in and securing data flow between the user and the GMS, preventing unwanted access and data attacks.
    - Security Class: This class and its supporting JavaScript files are responsible for the encryption of sensitive data that crosses network connections, mainly passwords.
    - Validation Class: This class is used to validate that data received from the user conforms to specific data types that the system asks for. This is to prevent the user from entering malicious data into a form.
  - Configuration: Deals with all manner of configuration options for the system.
    - System Class: This class is responsible for the configuration of the system and for the logging of errors that occur during SQL queries.
    - Config Class: This class is responsible for the actual loading of the system configuration from the database.
    - Miscellaneous Class: This class contains other configuration functions that do not fit into any other class.
  - HTML API: Deals with the construction of all of the actual pages that users visit.
  - User: Deals with any information regarding users of the GMS.
    - User Class: This contains all functions regarding the manipulation of data involving a user.
    - Email Class: This class is responsible for the sending out of emails based on user input and information.
    - Session Class: This class is responsible for the logging of user sessions so that users can log in and be able to stay logged in while navigating through the site.

- Core: Pulls all of the above together into one place. All data from the user and the database flows through this section of the GMS. However, the data flow is passed off to the Security classes before going to the database and vice versa. So even though the Configuration classes or User classes deal with data from the database, its data flow actually goes through the Core and then through the Security before going to the database (though the security for the internal classes is lighter than the user input).
- Web Pages: These are the actual pages that are viewed and interacted with by the user.

- Below can be found a graphical representation of the categories. The black arrows represent the cascading structure that is used to include all of the files together for use. The blue double-arrows show how data flows throughout the application framework.



Implementation

(revised 2009-11-15 to add pages)

- README.txt (*updated 2009-11-27*) – This file holds information about the GMS and instructions on how to install it.
- api.html.php (*updated 2009-11-15*) – This is the HTML API. It is responsible for the outputting of any and all information to the screen.
- api.sql.php (*completed 2009-10-11*) – This is the SQL API. It is responsible for controlling all data flow into and out of the database. It is also responsible for the prevention of SQL injection attacks.
- class.config.php (*completed 2009-09-24*) – This is the Config class definition. This class is responsible for loading the configuration for the GMS from the database into a global variable that will be accessible throughout the entire system.
- class.email.php (*completed 2009-10-03*) – This is the Email class definition. It is responsible for the construction and sending of all emails through the GMS. In conjunction with the Validate class, it will only send emails to addresses that are in a valid format.
- class.image.php (*updated 2009-10-28*) – This is the Image class definition. This is used by the Image script to select strings of text and colors for the image itself.
- class.misc.php (*completed 2009-10-03*) – This is the Miscellaneous class definition. It is responsible for containing all functions that do not fit into any other class.
- class.security.php (*completed 2009-10-12*) – This is the Security class definition. It is responsible for ensuring that sensitive data is encrypted properly across connections and to validate that data. In conjunction with the Security and MD5 scripts, it allows for the passing of passwords across connections.
- class.session.php (*updated 2009-10-21*) – This is the Session class definition. It is responsible for the construction and destruction of user sessions, thus allowing users to login and logout of the GMS and to have their login persist across pages.
- class.sys.php (*completed 2009-09-24*) – This is the Sys class definition. It is responsible for the logging of any errors from the database along with the calling of the Config class to load the configuration.
- class.user.php (*updated 2009-10-24*) – This is the User class definition. It is responsible for all user data manipulation.
- class.validate.php (*completed 2009-09-20*) – This is the Validate class definition. It is used to ensure that all user input conforms to specific standards before being used for emails or database submissions.
- config.image.php (*completed 2009-10-12*) – This if the configuration file for the Image script. The Image script is what is used to generate the title graphics of each page within the GMS and this file is responsible for defining the size and color of the image along with defining the image's font and the colors used for the font itself.
- config.sql.php (*completed 2009-09-20*) – This is the configuration file for the SQL API. It contains all necessary information that is needed to connect to the database and also for the structure of the database itself.

- config.validate.php (*completed 2009-09-20*) – This is the configuration file for the Validate class. It contains all of the RegEx strings used by the Validate class to ensure that user data conforms to certain parameters.
- core.css (*completed 2009-09-20*) – This is the main CSS file for the GMS. It is used for all of the default styling of the site. If an administrator wishes to change the appearance of the site for their own needs, this would be the place to start.
- core.php (*updated 2009-11-01*) – This file is responsible for tying all other components together for use on the actual GMS pages that are created.
- expander.js (*added 2009-10-28*) – This file is responsible for the special display of tags made by players during the game.
- image.php (*completed 2009-09-24*) – This is the Image script. It is responsible for actually generating the title image of any page within the system. image.php actually *becomes* the title image for any page, so the `src` of the `img` tag on a page is actually this script.
- md5.js (*added 2009-10-05*) – This is the MD5 JavaScript to allow for actually encrypting passwords prior to sending them across internet connections. If this wasn't done, then user passwords would be sent as plaintext across an internet connection, which isn't secure at all. This file is used courtesy of Paul Johnston and is licensed under a BSD license.
- security.js (*completed 2009-10-08*) – This is the Security script. It is uses the MD5 JavaScript along with the Security class to ensure the encryption of such things as passwords prior to being sent across connections. It is mainly responsible for adding a salt to the encrypted password so that it doesn't fit the length of a regular MD5 sum and to help the encryption stay secure.
- Pages within the system:
  - Public Pages:
    - contact.php (*updated 2009-11-06*)
    - forgotten.php (*added 2009-11-06*)
    - index.php (*updated 2009-11-06*)
    - login.php (*updated 2009-11-06*)
    - logout.php (*added 2009-11-03*)
    - register.php (*updated 2009-11-06*)
    - regvalidate.php (*added 2009-11-06*)
    - rules.php (*updated 2009-11-06*)
    - stats.php (*updated 2009-11-06*)
  - Player Pages:
    - contact.php (*added 2009-11-06*)
    - index.php (*added 2009-11-06*)
    - informer.php (*added 2009-11-14*)
    - report.php (*added 2009-11-06*)

        - rules.php (*added 2009-11-06*)
        - stats.php (*added 2009-11-06*)
    - Admin Pages:
        - adminadd.php (*added 2009-11-14*)
        - admindelete.php (*added 2009-11-14*)
        - adminupdate.php (*added 2009-11-14*)
        - banadd.php (*added 2009-11-14*)
        - bandelete.php (*added 2009-11-14*)
        - banlist.php (*added 2009-11-14*)
        - banupdate.php (*added 2009-11-14*)
        - flow.php (*added 2009-11-10*)
        - index.php (*added 2009-11-06*)
        - informer.php (*added 2009-11-14*)
        - playeradd.php (*added 2009-11-14*)
        - playerdelete.php (*added 2009-11-14*)
        - playerlist.php (*added 2009-11-11*)
        - playerupdate.php (*added 2009-11-14*)
        - report.php (*added 2009-11-06*)
        - stats.php (*added 2009-11-10*)

Known Bugs and Other Issues in HvZ GMS 1.0 β

(revised 2009-11-15 to reflect current bugs/issues)

- Current configuration options and account settings are not being used, namely the “Player Cap” configuration setting and the “Access_Level” account setting for administrative accounts. (resolved)
- install.php hasn't been tested and could cause issues. (resolved)
- core.php doesn't force the system to be installed for pages to load, which could lead to PHP errors from NULL variables. (resolved)
- Pre-submit encryption hasn't been added to any forms yet, so all data submitted through forms is currently plain-text only. (resolved)
- Possible missing displays of information from the global variable $_HVZ_CONFIG due to function declarations removing the global from it's scope (due to a missing global declaration). All functions that use this global need the global declaration in it to eliminate this issue. (resolved)
- Current public pages do not use all required HTML API functions to build the pages properly.

However, they are using enough of the functions to show proof-of-concept on how the API works. (resolved)

Preliminary Test Plan and Preliminary Test Cases

(revised 2009-11-15 to add cases and reorganize structure)

- **The Plan**
  - The test plan for the Humans vs. Zombies GMS is rather simple and straightforward. What better way to test it for its purpose than to run a simple simulation of a game of Humans vs. Zombies? I actually have used this approach before on the deprecated system and it turned out to be a great way to test the system. So the general approach will work like this: I'll take the GMS and do a fresh install on a computer that doesn't have it installed yet. I will then proceed to set up the the configuration of the system and then register a few (10) players and some admins to make sure that all login information works. I'll then run through all of the steps that a standard game would run through to ensure that the GMS works properly.
- **Primary System Test Cases and Integration Testing**
  - Install Case
    - Requirement(s) Being Tested: NFR-1
    - Preconditions: none
    - Steps:
      - Unzip GMS to a directory on a web server.
      - Follow GMS install instructions
    - Desired Result: The system installs properly with no errors and the database is populated with default information.
  - Configuration Case
    - Requirement(s) Being Tested: FR-1
    - Preconditions: Install Case
    - Steps:
      - Log in to the GMS using the administrator account created during Install Case.
      - Navigate to the Configuration page.
      - Edit Configuration values and submit changes.
      - Reload page and verify that the changes have indeed taken effect.
    - Desired Result: The configuration options change successfully without any errors.
  - “The Flow” Case
    - Requirement(s) Being Tested: FR-9
    - Preconditions: Configuration Case

- Steps:
    - Log in to the GMS using the administrator account created during Install Case.
    - Navigate to the “The Flow” page.
    - Change “The Flow” to “Registration Open”
- Desired Result: The registration for the current install of the GMS is opened to allow player registrations.

- Register Case
    - Requirement(s) Being Tested: FR-12, FR-11, FR-10
    - Preconditions: “The Flow” Case
    - Steps:
        - Go to the main site of the GMS and navigate to the Register page.
        - Fill out the form to register a player in the system.
            - First try with data that doesn't match requirements (such as passwords that don't match) to ensure error detection and validation is set up.
            - Then try with data that could be used as an SQL injection attack, such as “;--DROP TABLE players” as a first name to ensure that it doesn't actually drop the table. This will create a player with that ridiculous name of course though. We can use it later.
            - Then create a player using legitimate information
        - Follow all directions following registration to validate the registration.
    - Desired Result: A couple players are created in the system and they should be able to log in now through the Login page.
- Gameplay Case
    - Requirement(s) Being Tested: FR-4, FR-5, FR-6, FR-2
    - Preconditions: Register Case, one player selected as OZ, “The Flow” set to “Game Start”
    - Steps:
        - Log in with the OZ player.
        - Go to the Report Kills page.
        - Report another player dead with any given time needed.
        - Submit the report.
        - Go to the Game Stats page to verify the kill was reported and to check to make sure that it updated the current player as well.
        - Wait the turn time specified in the system (1 hour by default).
        - Check the Game Stats page again to verify that the reported player has changed from a Dead player to a Zombie player.
    - Desired Result: The GMS changes the player's information based on tags reported and

based on time constraints met and displays these values properly in the Game Stats page.

- Information Case
  - Requirement(s) Being Tested: FR-3
  - Preconditions: Register Case
  - Steps:
    - Log in to the GMS with the administrator account set up during installation.
    - Navigate to the Player List page.
    - Choose a player and hit the “Edit” button.
    - On the new page that pops up, change information regarding the player.
    - Hit submit when done.
    - Go back to the player's edit page and verify that the changes kept.
  - Desired Result: The player's information has been changed through the administrative interface, thus allowing for the manipulation of player data.

- **Unit Test Cases**
  - Data Encryption
    - Requirement(s) Being Tested: Sensitive data should be encrypted *prior* to being submitted
    - Preconditions: none
    - Steps:
      - Write a small form that has a field for text input.
      - Attach the `encrypt_password` function to the form.
      - Attach another small function that alerts the value of the field.
      - Submit the form.
    - Desired Result: The alert box should show an encrypted string for the value. If it's the original value, then the function did not operate properly.
  - Email Function
    - Requirement(s) Being Tested: RegEx for email submissions.
    - Preconditions: none
    - Steps:
      - Use the contact moderator form
      - Fill out the form with improper email addresses, such as “hello world” or “hello@world.program”
      - Submit form.
    - Desired Result: The form should return an error that the emails were invalid. If the emails were valid, it would send out an actual email though.

- Validate Function
  - Requirement(s) Being Tested: Validate Class
  - Preconditions: none
  - Steps:
    - Write a simple page that calls each of the functions contained in the Validate function.
    - Pass two data strings into each function: one that is valid and one that is invalid.
    - Print the results of each function call.
  - Desired Result: Each function call that has valid data should print true and the other ones should print false.
- Current URL
  - Requirement(s) Being Tested: Miscellaneous Class
  - Preconditions: none
  - Steps:
    - Write a very short page that simply calls the `current_url` function of the class.
    - Load the page.
  - Desired Result: The page should display the current URL that is in the browser's address bar.

Software Demos

- Demo covering the main pages of the site made using the Humans vs. Zombies GMS and the pages that a player will see when logged into the system. (http://www.youtube.com/watch_private?v=RDCUxqX3wtk&sharing_token=tMpXVkmNiqbEo-k3erjgFw==).

Final Codebase and Documentation Access

- Code has not been publically released yet but is available in the .zip file that was uploaded to Moodle as part of this report. The README file can be found in the root directory that contains the codebase (the folder named hvzgms).

About the Author

- Connor is a fairly laid-back guy who excels at programming, software design, database management, Dance Dance Revolution, Rock Band, and generally having fun. He is always looking for new and interesting programming opportunities to join in on. He also prefers simple layouts to really busy layouts. Connor has also contributed code to the Drupal Content Management System and has also worked on other management systems for things such as Dungeons and Dragons.

Executive Memos

To: Jan Pearce (Project Director)
Project: Humans vs. Zombies Game Management System.
Logo:

Author: Connor Kelly
Date: 15 September 2009

The last week of work on the Humans vs. Zombies Game Management System (GMS) consisted of the following being developed or designed:

- Documentation and coding standards were finalized (1 hour). All code can now be formatted in a similar manner as to provide consistency throughout the entire code base of the project. Also, this allows for block comments to be written for functions that can be used with the Doxygen program to generate full documentation pages for the code that the comments pertain to.
- SQL API was written (4 hours). This Application Programming Interface (API) is a subset of the MySQL functions built into PHP that have been rewritten to be specifically used with the Humans vs. Zombies GMS. These functions are universal to the system are used for all query calls to the database.
- SQL config file was written (15 minutes). This file contains global constants used by the SQL API to connect to the database and to select the correct database. It also contains a RegEx expression that is used by the SQL API to allow for a dynamically expanding query function. That function is designed to prevent SQL injection attacks.
- System class was developed (1 hour 30 minutes). The System class is responsible for logging all SQL query errors that occur and is also responsible for loading the configuration. This class will more than likely be expanded to encompass **all** error reporting before the system is complete.
- Config class was developed (1 hour). The Config class is responsible for loading and updating the configuration of the GMS within the database. The `load()` function is called by the System class to load the configuration.

The total time spent on work this week is 7 hours 45 minutes. The total time spent since the project began is 7 hours 45 minutes.

Project goals for Week 2:

- Write the Core file. This file will pull all components together and then will be loaded onto pages to actually give access to GMS features. This file cannot be fully completed until project is done, but can be built up with each new class that is added to the GMS.
- Write test Install script. This file will be a test install script that will use the SQL API to construct the tables within the database and then populate the tables with default information. Another portion of the test install script will be to load the Core file and check to make sure that the System and Config classes are operating properly.
- Time permitting, the next phase of development will begin by writing some non-dependent (does not depend upon or depends very little on other files/classes) classes for the GMS, such as the Email class to provide email functionality.

To: Jan Pearce (Project Director)
Project: Humans vs. Zombies Game Management System.
Logo:

Author: Connor Kelly
Date: 22 September 2009

The last week of work on the Humans vs. Zombies Game Management System (GMS) consisted of the following being developed or designed:

- Core.php file was created and development started on it (2 hours). This file cannot be finished until the rest of the system has been defined as it ties all of the files together. However, I was able to define a basic structure for this file and begin tying the current files together into it.
- Wrote a basic install script to test the current features (1 hour 30 minutes). This file was a **very** basic file that pulled in the files that I've written so far and tested some of the features that I've created, such as changing the SQL configuration, constructing the database for the GMS, and testing the error logging features of the GMS along with loading the configuration from the database (which is empty at this point, but it was good to make sure that the function was able to execute with no SQL errors).
- Wrote the Email class (1 hour). This class is used to send emails out to users through the GMS. The helper function of the function that sends out the emails is a RegEx checker that ensures that the email addresses that are provided are of a valid format. If **any** of the email addresses are of an invalid format, then the function refuses to send the email as it would cause an error to do so.

The total time spent on work this week is 4 hours 30 minutes. The total time spent since the project began is 12 hours 15 minutes.

Project goals for Week 3:

- Write a Validation class. This class will be responsible for validating all user input for correctness. It will also pull out the RegEx validation used in the Email class so as to separate validation functions specifically into this class.
- Write the Image class. This class is used to dynamically construct the title images of the pages.
- Time permitting, I will begin work on writing the User class, which is responsible for keeping track of very basic information about a person logged in, such as the fact that they are logged in, what their ID is (if any), and what level of site permissions the user has.
- Also time permitting, I will begin work on the Session class. This class works in tandem with the User class to allow for users to be logged in and to allow for their logged in state to persist between pages in the GMS.

To: Jan Pearce (Project Director)
Project: Humans vs. Zombies Game Management System.
Logo:

Author: Connor Kelly
Date: 29 September 2009

The last week of work on the Humans vs. Zombies Game Management System (GMS) consisted of the following being developed or designed:

- Validation class created (1 hour). This class uses RegEx and functions to parse through any manner of input that the GMS will take and makes sure that it is of the format that is specified. This allows for the prevention of entering an email address that is not formatted like an email address and entering a string of characters for somewhere that should be a number.
- Image class created (2 hours). This class constructs title images for pages based on variables passed to the class. This is useful as we only need one file to generate all of the possible header images for the site without having to actually create the images by hand. This is also nice because then the headers can be changed dynamically based on what school is using the GMS.
- User class created (3 hours 15 minutes). This class deals with all user information in the database. It allows for the selecting, adding, updating, and deleting of any player or administrator in the database. This class will be used by the GMS for pages that display user information or pages used to edit user information.

The total time spent on work this week is 6 hours 15 minutes. The total time spent since the project began is 18 hours 30 minutes.

Project goals for Week 4:

- Build a Session class. This class will be used to allow for users to log in and to stay logged in between pages.
- Build the Miscellaneous class. This class holds functions that do not belong in any other class.
- Time permitting, work will be done on the Security class. This class and the related script file that will go with it are responsible for encrypting sensitive information for transport across network connections. This is to prevent stealing of passwords. It will also be responsible for the "salt", which is what is added to a password to make sure that it is a valid call. The salt is changed at each login, so in the event that an encrypted password is skimmed from the network connection, the encrypted version is less useful as it does not contain the exact password, but the password + salt.
- Time permitting, work will begin on the HTML API. This API is responsible for actually displaying pages in the GMS. Once this API has been constructed, actual pages can be built within the GMS as this file contains all necessary functions to construct pages.

To: Jan Pearce (Project Director)
Project: Humans vs. Zombies Game Management System.
Logo:

Author: Connor Kelly
Date: 13 October 2009

The last two weeks of work on the Humans vs. Zombies Game Management System (GMS) consisted of the following being developed or designed:

- Session class created (1 hour). This class creates and destroys session information. This session information is used to determine whether the user is logged in or not.
- Miscellaneous class created (30 minutes). This class holds an functions that do not belong anywhere else in the system. There are currently very few functions that fit this category, but more could come up in the future.
- Security class created (7 hours). This class is responsible for the encryption of passwords across connections. This includes a JavaScript file for encrypting **prior** to sending the password as well. This also includes an open-source JavaScript adaptation of the MD5 checksum so that this can work. The Security class uses two different salts to help ensure that the data is correct. The first salt is generated on the client side prior to sending the encrypted password across an internet connection. The second salt is generated on the server and added to the one sent across a connection to ensure that the data was freshly sent to the server and is not stale information.

The total time spent on work this week is 8 hours 30 minutes. The total time spent since the project began is 27 hours.

Project goals for Week 6:

- Build the HTML API. This is the largest section of the system as it is used to build the pages that will be displayed to the user. It will incorporate bits and pieces of each class to achieve this goal. Once the HTML API has been finished, the actual GMS pages can begin to be constructed.

To: Jan Pearce (Project Director)
Project: Humans vs. Zombies Game Management System.
Logo:

Author: Connor Kelly
Date: 20 October 2009

The last week of work on the Humans vs. Zombies Game Management System (GMS) consisted of the following being developed or designed:

- HTML API (6 hours 30 minutes). This file is still incomplete as there are a few more functions that I would like to add to it. However, it is almost done and has all the basic functionality that I desire out of the file. I simply need to add some extended functionality to it to help remove some code out of the actual pages when those are created

The total time spent on work this week is 6 hours 30 minutes. The total time spent since the project began is 33 hours 30 minutes.

Project goals for Week 7:

- Finish building the HTML API.
- If time permits, build a finalized installation script so that the GMS can be set up properly upon distribution.
- Also if time permits, begin work on the actual pages of the GMS. This will allow for the final stages of the system to take shape. If the pages can be built and successfully working within the next few weeks and they are free of errors and bugs, then time will be taken to begin work on non-essential features to improve upon the current GMS.

To: Jan Pearce (Project Director)
Project: Humans vs. Zombies Game Management System.
Logo:

Author: Connor Kelly
Date: 27 October 2009

The last week of work on the Humans vs. Zombies Game Management System (GMS) consisted of the following being developed or designed:

- HTML API (12 hours 15 minutes). The API is still incomplete but is nearing completion for the public and player pages of the site. There are approximately 3-4 functions left to code completely to get it ready for use for those pages.

The total time spent on work this week is 12 hours 15 minutes. The total time spent since the project began is 45 hours 45 minutes.

Project goals for Week 8:

- Finish building the HTML API for public and player pages.
- Write an installer script so that the system is now deployable.
- If time permits, begin construction on the public and player pages so as to test out the majority of the work on the HTML API. Once these pages have been constructed, work can continue on the HTML API to include the functions that will be needed for the administration pages of the site.

To: Jan Pearce (Project Director)
Project: Humans vs. Zombies Game Management System.
Logo:

Author: Connor Kelly
Date: 3 November 2009

The last week of work on the Humans vs. Zombies Game Management System (GMS) consisted of the following being developed or designed:

- HTML API (10 hours). The HTML API now includes all necessary functions to build the Public and Player pages within the system with the exception of possibly one or two functions. These functions haven't been tested for functionality yet but they do pass syntax checking. Also, some of the functions that are used on these pages are usable on the Admin pages and have been coded to take this into account so that more functions that did duplicate work would be unneeded.

The total time spent on work this week is 10 hours. The total time spent since the project began is 55 hours 45 minutes.

Project goals for Week 9:

- Test the installer and make sure that it works properly (remove the first issue from known issues).
- Force the system to have been installed before loading any pages (remove issue #2 from the known issues).
- Add encryption to the forms that are submitted in the system so that necessary data is encrypted properly (remove issue #3 from the known issues).
- Enforce the global variable $_HVZ_CONFIG across all functions that use it (remove issue #4 from the known issues).
- Finish building the Public pages so that they operate fully and correctly (remove issue #5 from the known issues).
- Build and test the Player pages so that they operate fully and correctly.
- Begin adding in the functions necessary for building the remaining Admin pages that currently aren't taken into account to the HTML API.

To: Jan Pearce (Project Director)
Project: Humans vs. Zombies Game Management System.
Logo:

Author: Connor Kelly
Date: 10 November 2009

The last week of work on the Humans vs. Zombies Game Management System (GMS) consisted of the following being developed or designed:

- Resolved all issues described in the last report (5 hours) - I have successfully resolved all of the issues presented in the last report, including testing to make sure that these issues were indeed resolved.
- Built all of the Player pages (3 hours) - I constructed all of the Player pages for the GMS. They have also passed the initial testing as to whether they function correctly or not. Final testing on them will be done once the system is in its final stages of testing.
- Began work on Admin pages (1 hour) - I began work on the construction of the Admin pages, with the creation of three of the pages being done already. More pages still need to be done, but I have to write the requisite HTML API functions for those pages first.

The total time spent on work this week is 5 hours. The total time spent since the project began is 64 hours 45 minutes.

Project goals for Week 10:

- Continue working on getting all issues resolved as they come up.
- Work on the HTML API for Admin pages. This is going to be the bulk of the work. It is currently coming down to about 5-8 functions. With these functions done, then the Admin pages can be successfully constructed and finished, thus leading into the final stages of testing.

To: Jan Pearce (Project Director)
Project: Humans vs. Zombies Game Management System.
Logo:


Author: Connor Kelly
Date: 17 November 2009

The last week of work on the Humans vs. Zombies Game Management System (GMS) consisted of the following being developed or designed:

- Wrote most of the remaining HTML API functions (9 hours 30 minutes) – Most of the remaining functions for the HTML API have been written for the system. Only a couple more remain before the code is considered complete and the testing of the full system begins.
- Built some of the remaining pages in the Admin section (1 hour 15 minutes) – With the new HTML API functions having been written, I was able to build some of the remaining Admin pages. Unfortunately, these pages cannot be tested until I finish the HTML API, build the remaining pages, and run my full system test.

The total time spent on work this week is 10 hours 45 minutes. The total time spent since the project began is 75 hours 30 minutes.

Project goals for Week 11:

- Continue working on getting all issues resolved as they come up.
- Finish HTML API, build remaining pages, and begin a full system test.

To: Jan Pearce (Project Director)
Project: Humans vs. Zombies Game Management System.
Logo:


Author: Connor Kelly
Date: 1 December 2009

The last two weeks of work on the Humans vs. Zombies Game Management System (GMS) consisted of the following being developed or designed:

- Touching up the code through debugging (10 hours) – Did an extensive debugging test to make sure that I caught as many of the bugs as possible and then worked in solutions for all bugs found and/or fixed code that was causing issues.
- Ran a simple simulated game to make sure all functions worked as expected (7 hours 30 minutes) – As this says, I ran a simple simulation to make sure that the GMS worked as expected when tags were reported or when timers elapsed.

The total time spent on work these last two weeks was 17 hours 30 minutes. The total time spent since the project began is 93 hours.

Project goals for Week 13:

- Finish up documentation on the code and write the readme file for the GMS.
- Work on a finalized project demo video along with a poster presentation.

To: Jan Pearce (Project Director)
Project: Humans vs. Zombies Game Management System.
Logo:

Author: Connor Kelly
Date: 8 December 2009

This last week of work on the Humans vs. Zombies Game Management System (GMS) consisted of the following:

- Creating the poster for the poster session (4 hours).
- Creating and refining the demo video to go along with the poster presentation (3 hours).
- Touching up the code and changing small details in text where changes were made (2 hours).

The total time spent on work this last week was 9 hours. The total time spent since the project began is 102 hours.

My experience with this project this term has been a rather good one. I got to learn more about making access to websites more secure and to protect against things like SQL injection attacks. It was also a nice experience to work with PHP as an application framework and development language rather than simply as a way to add nifty features to a website. There was always the frustration of coding something just to find out that it wasn't working, but it was still thrilling when it all finally came together and just worked.