Game Server Control Panel

Computer Sceince Project

By Diogo Moura

Student ID: 12353680
Course: ECSC699.Y Computer Science Project
Route: Software Engineering with Foundation
School: Electronics and Computer Science
Submission Date: 28th April 2013

Supervisor: Philip Trwoga

This report is submitted in partial fulfilment of the requirements for the BEng (Hons) Software Engineering with Foundation Degree at the University of Westminster.

ABSTRACT

This project looks at the creation of a Game Server Control Panel; it explores problems, which span several environments and languages.

Overall the solution given addressed the issues head on, found creative ways to tackle the problem and indulged into emerging technologies.

Table of Contents

Table of Figures

Table of Snippets

1 Introduction

Today some of the most popular pc games on the market have the ability to connect friends and players all over the word together, allowing for dynamic gameplay that was not possible before. There are two main ways to connect players together, peer-to-peer - where there is no central host and the game is synchronized between the players, and the client-server model – where a central server exists in which players connect to partake in the same multiplayer session, the latter being the more popular in modern games as it allows for better performance. Game developers providing the client-server model of multiplayer would also usually provide Game Server software. Whilst it is possible to run this software on a regular machine with non-commercial Internet access, there are many benefits of running these on a Dedicated Server machine, which is usually hosted within a Data Center, such as the ability to maintain the server online 24/7 with a guaranteed IP address and a stable upload connection. A single Dedicated Server is able to run multiple Game Server software, providing it has sufficient IP/ports, resources (such as processing power and memory) and bandwidth available.

1.1 Problem Statement

The companies that offer the 24/7 Game Server Rentals are called Game Server Providers (GSP for short). Clients, typically clan leaders, or just players who want to run their own Game Servers will visit a GSP's website and purchase a server. In the past server rental has been very primitive, clients would purchase a server and wait for the administrator to manually install the servers. Once installed the GSP would assign clients FTP for managing files and SSH credentials to start/restart their server, and in some cases only ftp was given and restarts would need to be requested. The GSP's also had no way to keep track of their infrastructure. Things quickly got out of hand as the companies grew. A Game Server Control Panel Is a solution to this problem, it will allow the administrators to keep track of their Game Servers, Clients and Dedicated Servers which would prove to be a nightmare previously, and at the same time would give their clients control they need over their server.

1.2 Project Aim

The aim of this project is to produce a functional starting point for a Game Server control panel, which will be a foundation to a larger project in the future.

1.3 Objectives

- To develop a deep understanding of web technologies.
- To gain experience in the full software life cycle.
- To experiment with new, interesting and emerging web technologies.
- To explore ideas surrounding the problem space and providing adequate solutions.

1.4 Methodologies

Methodologies are techniques or a set of proven steps that can be taken to ensure the best chance of a projects success. It is important to pick a suitable methodology for this software project.

1.4.1 Waterfall

The waterfall model is often considered as the classic approach, it consists of five main stages and its most important principle is that each stage must be completed before the next is started.

The five stages of the waterfall model are as follows:

1. **Requirements**
   In this stage, requirements are gathered via the various requirements elicitation techniques. The requirements gathered are collated and a specification is formed, usually these are split into functional and non-functional requirements.
2. **Design**
   Each requirement in the pervious stage is reviewed, the technologies required to achieve these requirements are reviewed, UML modeling can be used to represent each aspect of the system. It is important to cover all angles in this stage, as it cannot be revisited once finalized.
3. **Implementation**
   The system is coded as per the design, and documentation is written so that it is maintainable in the future.
4. **Testing**
   The implementation should be tested against the design and requirements to ensure accurate functionality and any bugs found should be documented.
5. **Maintenance**
   There is a handover to the client of the finished software, but support will be offered to tackle any issues that may arise.

The strengths of this methodology lie within its simplicity, it is a linear model, easy to follow and implement. The Waterfall Model also ensure minimal risk of a product not delivering on schedule by enforcing discipline, as each stage cannot be revisited and must be finalized before proceeding.

The model does not come without its drawbacks, as software projects grow in complexity the chances of unforeseen circumstances occurring at later stages increases, and with this particular methodology there is no room for correction once a stage is finalized. This can potentially lead to a final product that differed from the original vision.

1.4.2 Spiral

“The spiral model combines the idea of iterative development (prototyping) with the systematic, controlled aspects of the waterfall model” – [nasa]. This model has an emphasis on risk management, major risks should be identified and solutions to

minimize the risks should be put in place. The model is split into four distinct stages, once a full cycle is complete, the spiral begins again, and the four main stages are:

1. Defining Objectives.
2. Risk Management.
3. Development & Testing.
4. Planning for next iteration.

The spiral model is mainly suited to larger projects, it ensures minimum risk for projects, unlike the waterfall model, stages are revisited after each cycle where improvements can be made and from a business sense clients are able to follow production of the system closely and gain a deeper understanding of the system.

However the spiral model is not so suitable for smaller projects, due to the several iterations and the emphasis on risk management the costs involved are much higher than traditional methodologies.

1.4.3 Incremental Development

The incremental build is an iterative version of the waterfall model. It is usually implemented as a series of increments which after each increment feedback can be taken and the product refined. The incremental build still has the same five stages of the waterfall except that at stages can be revisited as needed.

With the incremental development model, features are usually created in increments allowing for products to be refined if a feature if needed, testing is also carried out after each increment allowing for problems to be identified early on in the development, on a similar note, there is less room for error with testing as there are small changes made per increment meaning it is far less complex.

On the other hand, the incremental development, due to its iterative nature is naturally more expensive to implement than its linear counter-part, the other problem with this methodology is that as feedback is taken, and features are refined or added, problems that were perhaps not applicable earlier in the design may start to show up.

1.4.4 For This Project

Each of the methodologies above has their clear advantages and disadvantages, and careful consideration of the methodologies has been taken to apply to this project.

Due to the size of the project, the spiral model is not suitable as it is designed for larger scale projects and will add unnecessary complexity to this project. The waterfall model is simple but restrictive, although a suitable model for this project.
However the flexibility provided by the iterative development model makes it the stronger candidate and this is the methodology that has been chosen for this project.

2 Background Research

2.1 Current Solutions

2.1.1 TCAdmin

TCAdmin has been in continuous development since 2004; it is now a mature product with several iterations and comes with a large array of features. There are currently two distributions of this software, Version 1 and Version 2. TCAdmin 2 is a ground-up re-write of the control panel and the most noticeable difference being the support for both Linux and windows remote servers, versus TCAdmin 1's windows only support. TCAdmin today is perhaps the most complete solution for game servers and its list of features includes:

- Activity Graphs.
- File Manager.
- Game Specific Tools (Queries, Remote Control).
- Reseller Packages.
- Task Scheduler.
- E-Mail Templates.
- Updates via web interface.
- VPS Support.
- Ticket Based Support System.
- Stop / Start / Restart Game Servers & Dedicated Servers.
- Specific support for VoIP servers.
- Support for sub-users.
- Manage Users, Game Servers, Templates & Dedicated Servers.

TCAdmin is written in ASP .NET, thus requires windows running IIS. In its bare form it is Master Server software, which provides the web interface, but communicates to the remote servers through its remote-monitor software, which gives it precision control and monitoring over the game servers.

2.1.2 Swift Panel

In 2008 Swift Panel entered the market, at the time was one of a hand full of control panels to support Linux (at this point TCAdmin did not have Linux support and the release of this was indefinite), also unlike TCAdmin it did not require remote software as it interacted solely through SSH.

Key Features:

- Customizable, Logic-less Website Templates.
- Custom Game Templates.
- Remote Server Monitoring.
- Install & Rebuild Game Servers.
- Stop / Start / Restart Servers.
- File Browser.
- Manage Users, Gameservers, Templates & Dedicated Servers.

Swift Panel is written in PHP, meaning that it will run on virtually any web server, its only prerequisites are that the SSH2 extension for PHP is installed as it uses SSH as its sole source of communication to the remote server, at the same time this means that no extra software needs to be running on the remote servers.

Swift Panel 2 was scheduled to be released in summer 2009; however there has been no word since and the project is rumored to be discontinued.

2.1.3 GameCP

GameCP was released to the public as beta in 2004, and like TCAdmin has been in continuous development since. At first the control panel only had support for Linux but more recent versions support windows. GameCP has received much criticism early on due to its software being riddle with bugs and its user-friendliness was questionable. Today GameCP is much more stable software which has come a long way. GameCP uses a queue system to offload the heavy duty work to the background. The dashboard provided by GameCP comes with customizable modules which can be dragged into different positions.

GameCP's key features include:

- Game-Changer.
- Config Manager.
- Map Manager.
- VoIP Server Support.
- Sub-users.
- File Manager.
- Automatic game downloader.
- Addon-Control.
- Manage Users, Gameservers, Templates & Dedicated Servers.

2.1.4 GameServers.com

GameServers.com are one of the words biggest Game Server Providers that began life in the early 2000's. Their control panel is in-house and can only be obtained as a third-party if renting their dedicated servers. The main features for the client of this control panel include:

- Start/stop/restart Game Servers
- Install Mods
- Reinstall Server
- Manage FTP Credentials.

3 Analysis

3.1 Stakeholders

3.1.1 Onion Model

An onion model allows us to clearly see the stakeholders involved in the system. The diagram below clearly shows the stakeholders which directly interface with the system (inner ring), stake holders which benefit from usage of the system (middle ring) and external stake holders which may be negative (red) or beneficiaries these outer ring. We can also see the interfaces the systems, which we will need to communicate with, these are marked as a box on the outline of the inner circle.

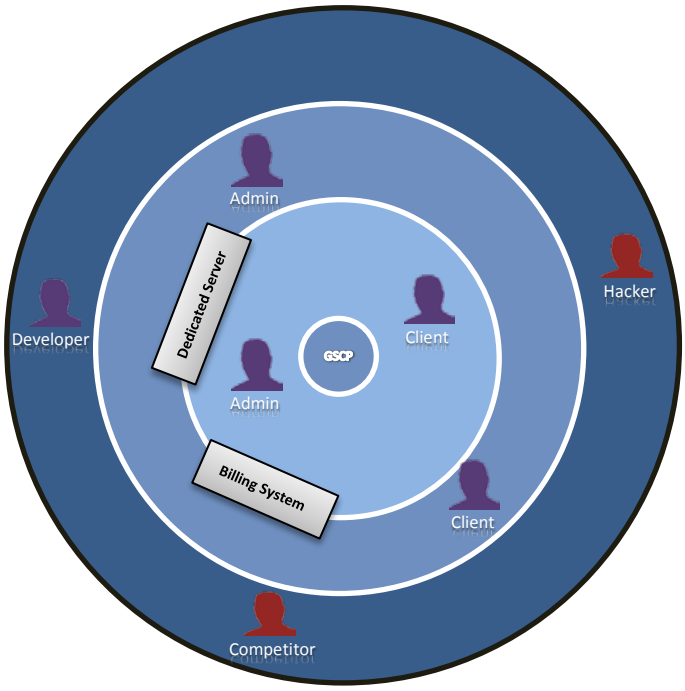


Figure 3.1 Onion Model

3.1.2 Stakeholder Goals

Stakeholder goals are a quick and efficient way to generate obvious requirements, which are different by the goals of the system stakeholders identified in Figure 3.1.

Goal Owner	Goal	Goal Type
Administrator	To Maximize Profits.	Commercial
	To ensure efficient operation of the business.	Functional
Client	To manage their game server.	Functional
	To easily run a game server.	Usability
Hacker	To break into the system.	Ethical
	Damage the system.	Ethical
	Obtain valuable information.	Ethical
Developer	To produce a satisfactory system.	Functional
	To produce a secure system.	Security

Competitor	To produce a reliable system.	Reliability
	To obtain a larger section of the market	Commercial
	To provide better experience to clients	Functional

3.2 Requirements Elicitation

It is important to understand the end users vision of the product, which often easy to misinterpret and this could lead to the failure of the project. To do this a mixture of requirement gathering techniques are used, each one with its benefits to extract a set of requirements, but also to gain a better understanding of how the system will be used, properly undertaking this phase will minimize the time spent of throwing away code and ensure the best chance of the projects success.

3.2.1 Interview

Interviews allow for a more personal perspective from the users of the system, it allows the users to contribute ideas that were perhaps not always evident. A group of clients and administrators were interviewed with the following template.

Administrator Interview Questions

1. Are you already running a game server management control panel for your business?
   It is important to find out whether they are using a competing product, if so we can proceed to question 2, if the administrator is not running software skip to question 5.

2. Which feature would you say is essential and [control panel here]'s biggest selling point?
   This question is aimed at letting the administrator highlight the most important features; it will form a base for our requirements.

3. If you had something bad to say about [control panel here] what would it be?
   To highlight common mistakes that current solutions make so that they can be avoided.

4. Is there anything you wish [control panel here] could do, that it currently cannot?
   Allows the administrator to suggest his or her own features for consideration.

5. Have you ever managed a set of game servers without a control panel (eg. Command line)?
   The question helps gauge the technical knowledge of the administrator.

6. What was the most repetitive task?
   Helps gather requirements on the basis of what is important to automate.

7. Was it ever difficult to remember any of the steps to installing a game server?

Helps to justify the need for automating the process.

8. Did you ever lose track of anything in particular (eg. server passwords)?
   If the administrator identifies any important information here then it can be incorporated in the control panel.

From the responses to the interview questions above, many things became apparent; such as the clear demand for a control panel to manage game servers and, that typically it is easy to lose track of clients, game servers and dedicated servers in a larger scale business. Another thing that was gathered from the interviews is that the most popular existing control panel appeared to be TCAdmin therefore some features could be reverse engineered from this control panel. The biggest concern with existing control panels was the lack of intuitive, user-friendly UI some administrators even said that they would like to occasionally be able to manage their network through a mobile device and most said would like to offer this functionality to clients. Many administrators suggested that they would like some real time feedback on server states, which usually means page refreshes on existing software. The most important highlight of the of the interview though was that administrators main concerns is that the panel is functional, cheap and its easy for their clients to use.

End User (Client) Interview Questions

1. Would you consider purchasing a game server that did not offer a control panel?
   This question is used to kick off the interview and allow the user to bring to light any opinions on the matter; it also affirms that the panel is in demand.

2. How would you describe your understanding of FTP and how to use it?
   FTP is the most common and obvious solution to providing access to the files on the server to modify them; it is likely that many clients will not understand it and this will perhaps justify the need for a web interface to manage files.

3. Do you usually install game mods on your server, and if so how often?
   Continuing the theme of ftp, this will further justify how complicated a potential file manager in browser would need to be.

4. What is the most common setting that you need to change on a particular game server, and how is this setting changed?
   IT is a given that settings are modifiable in various ways, this includes command line changes, configuration files etc... at this stage it is important to identify the most common methods for these to be considered.

5. What is the bare minimum control that you need over a game server?
   Allows the user to define some requirements.

The interviews with clients provided further insight into their expectations, it was clear that most users would not install modifications because it was too complicated

but would consider this if it were made easy for them. Almost all users wanted easy access to modify any configuration files and this pointed towards a file manager in the browser.

3.2.2 Observation

It is common for people to skip steps, particularly when a task become second nature to them, and details that may influence the systems design may be missed with other requirements gathering techniques. Observation may highlight any issues that were previously missed, it allows for better understanding of how operations should be performed.

The main thing noticed was that administrators would constantly lookup install instructions for game servers as it appeared that they handled so many games they could not remember, another noticeable thing was that administrators used password storage programs to connect to their servers.

3.2.3 Reverse Engineering

Reverse engineering requirements, where there are already existing systems is a good place to start as it can save time, existing projects have likely already gone through this process and successful features can be considered. Earlier in the report four separate solutions have been discussed, combined with interview results requirements can be extracted.

A feature that almost all control panels have implemented is sub-users, it is not uncommon for Clients to want to allow friends to manage their servers, sub-users is a secure way of doing this so the client does not have to share he's password.

Another popular feature that has been mentioned in interviews, and is also widely available in current control panels, is the ability to install game maps and mods with ease from the control panel, these are usually a list of mods with an install button.

Some existing solutions provide access to configuration files, and other go as far as allowing users to edit specific configuration variables in a form, this would provide a better user-friendly experience then using ftp or a file manager to navigate to files.

3.2.4 Questionnaire

Questionnaires can often be misleading, it is often not useful to ask open ended questions as collecting this data is difficult, therefore questions are asked in a sort of a biased way, usually used to confirm existing requirements rather than gather new ones.

The questionnaire used for this project is below; these are directed at the clients of the system and are used to obtain a wider opinion on previous research.

1. Is it important when purchasing a server that a control panel is provided?
   yes no

2. On average how many servers do you manage at any point in time?
   less than 3 3-6 6-10 10 or more

3. Do you regularly install modifications on your servers?
   yes no

4. Does your mobile have access to the Internet on the go?
   yes no

5. Would you find a mobile control panel useful for managing your servers?
   yes no

Many things were confirmed using this questionnaire such as the fact that there is demand for a control panel when purchasing a game server. Most people own less than 3 game servers, which may influence the design of the application. There is a demand for an easy option for installing game mods and finally that most users that have access to the internet on their mobile would like support for mobile.

3.3 System Requirements

Information gathered from the previous section should be collated and written into clear requirements that developers can interoperate and begin designing solutions.

For this section the requirements will be split into three categories:

- Essential – these requirements define the system. It is not possible to have the system unless these requirements have been met.
- Desirable – requirements that give the project some shape and appeal, these are not required but are nice to have in the system.
- Future – these are requirements that will not be in the scope of the current project but are future prospects, and areas for extension in the project.

3.3.1 Functional Requirements (Essential & Desirable)

R1. Clients must be able to stop, start and restart game servers that they own.
In the event that a server has crashed, a client needs the ability to start their server to get it back up and running without issues, the client also needs to be able to stop their server in the event that they need to modify files which are being used while the server is running such as game modifications, and to complete the set, clients may be able to modify configuration files without stopping the server however, a restart is needed to apply the changes. **This requirement is Essential.**

R2. Clients may be able to reinstall a server they own.
Occasionally clients damage their server beyond repair, usually by accident, and the usual solution to this problem is to request that an administrator reinstall the server, however clients feel that they should be able to resolve the issue themselves if provided with an intuitive interface to reinstall their servers. **This requirement is Desirable.**

R3. Clients shall be provided with a basic file manager, which will allow them to browse folders and edit configuration files within their browser.
Not all clients are comfortable with ftp, or have access to ftp clients perhaps due to ports being blocked; therefore they should be provided with an in-browser file manager that will allow them to perform basic file management actions. **This requirement is Desirable.**

R4. Clients must be able to view a list of servers they own.
The client shall be able to see details about servers they own at a glance; this will be the connection to extended management features. **This requirement is Essential.**

R5. Clients must be able to view extended information about their game servers.
Aside from the general information provided on the server list, clients should be able to view further details about the server such as their login details for FTP and general connection details. **This requirement is Essential.**

R6. Clients must be able to update their password.
Security is important; in the event that a client's password is compromised they should be able to update their password. **This requirement is Essential.**

R7. Administrators must be able to stop, start and restart game servers on the system.
Administrators need full control over their client's servers to provide support when something goes wrong, to do this they will need access to the stop, start restart functionality provided to clients. **This requirement is Essential.**

R8. Administrators must be able to manage users.
Administrators find it hard to keep track of users, they need to manage users on the system, this include seeing a list of users on the system, creating new users, modifying details for existing users and deleting inactive users from the system. **This requirement is Essential.**

R9. Administrators must be able manage game servers on the system.
When a client purchases a server the admin should be able to create a new game server for the order, if the client requests any changes the admin should also be able to handle these requests and when a client stops paying for the server or the server is no longer being used the administrator shall be able to remove it from the system, these should not only. **This requirement is Essential.**

R10. Administrators must be able to manage dedicated servers on the system.
Game servers run on the dedicated servers, in a company servers are added and removed all the time as they breakdown or demand increases, administrators should be able to add dedicated servers to the system, update details when they change and also remove them from the system. **This requirement is Essential.**

R11. Administrators must be able to manage custom games to the system.
In order to create game servers a game needs to be defined, administrators should be able to create, update and delete custom games on the system. **This requirement is Essential.**

R12. The system shall notify clients when their server status's change.
Clients need to know when their server status changes, this may usually be due to a server crash or even a request, they should be notified of these changes. **This requirement is Desirable.**

R13. The system may assign ftp credentials for a game server on its creation.
FTP is the main way to modify files on the server, the system should automatically issue ftp accounts for each server that has been created. **This requirement is Desirable.**

R14. Administrators may be able to update ftp credentials for servers.
Occasionally ftp credentials may be compromised, the administrator should be able to update these through the control panel. **This requirement is Desirable.**

R15. Administrators may be able to reinstall game servers on the system.
The administrator may have to deal with reinstall requests or troubleshooting, they should be able to schedule a server for reinstall through the control panel. **This requirement is Desirable.**

R16. Administrators may be able to browse files on client's game servers.
Sometimes the administrator will provide support for a client, and this may involve updating a configuration file, therefore the admin should have quick access to the file manager to avoid having to use ftp. **This requirement is Desirable.**

3.3.2 Functional Requirements (Future)

F1. Clients should be able to install popular game mods within the control panel.
Clients have suggested that they would install more game mods if it were made easier, clients should be able to install a game mod with a buttons click, this functionality is currently offered by existing control panels. **This is a Future requirement.**

F2. Clients should be able to install popular maps within the control panel.
Clients should be able to install a map or map pack with a buttons click, this functionality is currently offered by existing control panels. **This is a Future requirement.**

F3. Clients should be able to manage configuration files easily without the file manager.
Clients may be able to view a list of configuration files to easily modify without having to browse them in the file manager, this functionality is currently offered by existing control panels. **This is a Future requirement.**

F4. The system should provide an API for existing systems to trigger new purchase installs.
Many billing systems support automatic setup with third party systems, it is important to provide an API/conform to existing methods for allowing the billing system to trigger automatic installs, this functionality is currently offered by existing control panels. **This is a Future requirement.**

F5. Clients should be able to add sub-users to manage their game servers.
Sub-users allow the client to give permissions on a game sever such as start, stop or restart securely. **This is a Future requirement.**

F6. Clients may be able to categories their game servers.
Some clients have suggested that they own multiple servers, it may be useful for clients to be able to sort servers into categories for easy management. **This is a Future requirement.**

3.3.3 Non-Functional Requirements

NF1. The system shall be web application.
The system should be accessible to everyone, to ensure best coverage the system will be a web application. **This requirement is Essential.**

NF2. The website should feel like an application, page refreshes should be avoided where possible.
To decrease page load times, and increase the user experience the website shall transition between pages seamlessly avoiding refreshes where possible. **This requirement is Desirable.**

NF3. The system shall be cross platform, and particularly it should work on mobile.
Many clients suggested that they would like to see some functionality carried over into a mobile website, therefore the system should attempt to provide this in a mobile form factor. **This requirement is Desirable.**

NF4. The system shall be user friendly.
The most notable complaint about existing systems is the user-friendliness;

to avoid making similar mistakes the system should attempt to provide an intuitive user-friendly experience. **This requirement is Desirable.**

NF5. The system shall be secure.
The web is a hotspot for exploits and hacking, it is important to understand the risks when coding and ensuring that any obvious security holes are patched and that security is tested. **This requirement is Desirable.**

NF6. Actions performed on the system should not cause it to hang.
Some of the tasks required have the potential to hang the user while the request is being processed, the system shall deal with these in a way that it shall not hand the system. **This requirement is Essential.**

4 Design

4.1 Use Case Diagram

A use case diagram show the interaction between actors and the system, below the use case diagrams have been split into two sections that demonstrate both aspects of the sytem.

4.1.1 Game Server / Core Functionality

Figure 4.1 shows the core interactions between actors on the system.

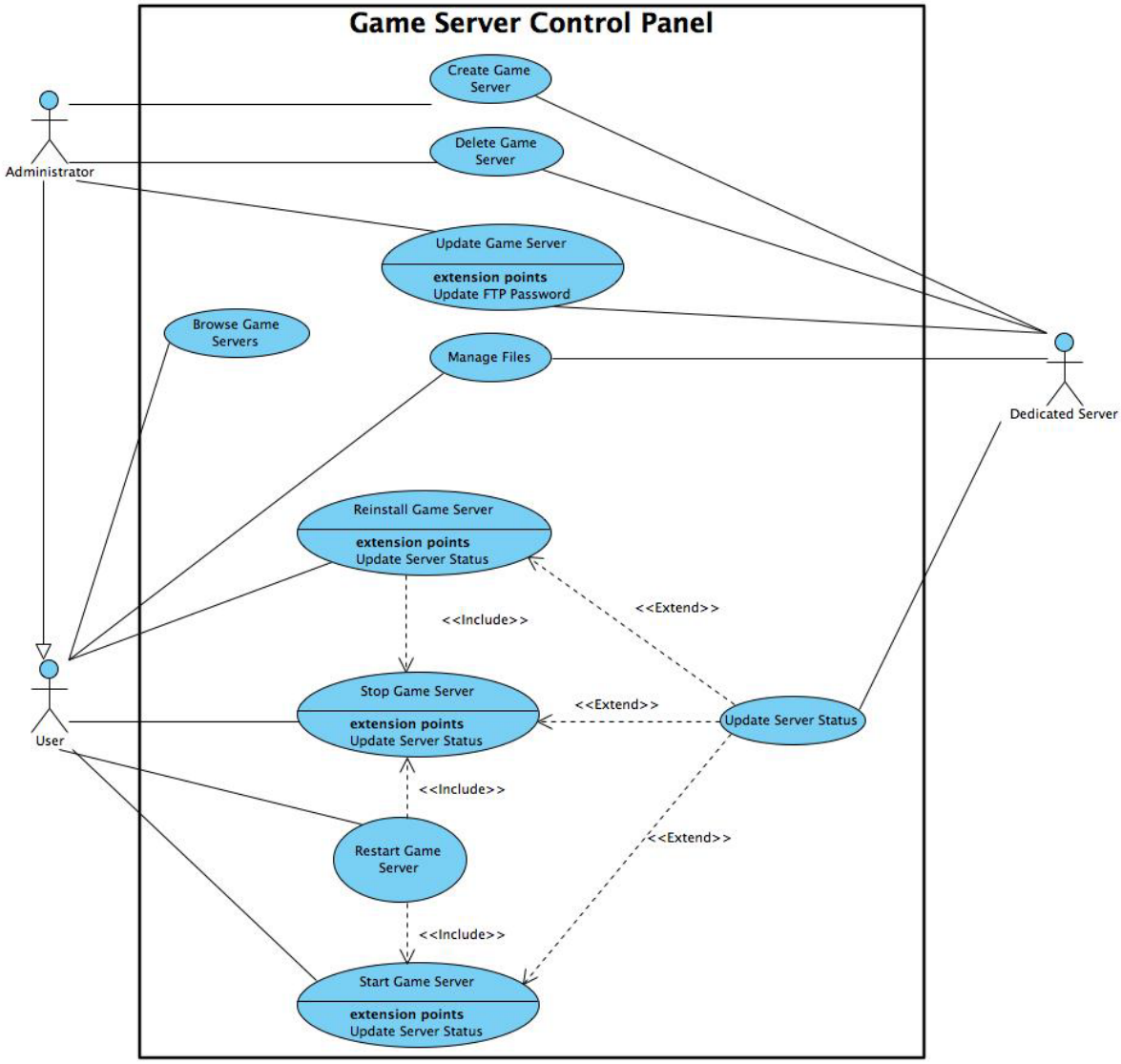


Figure 4.1 Core System Use Case

4.1.2 User, Game and Dedicated Server Management.

The second part to the use case is the admin specific interactions with the system, these are shown in figure 4.2.

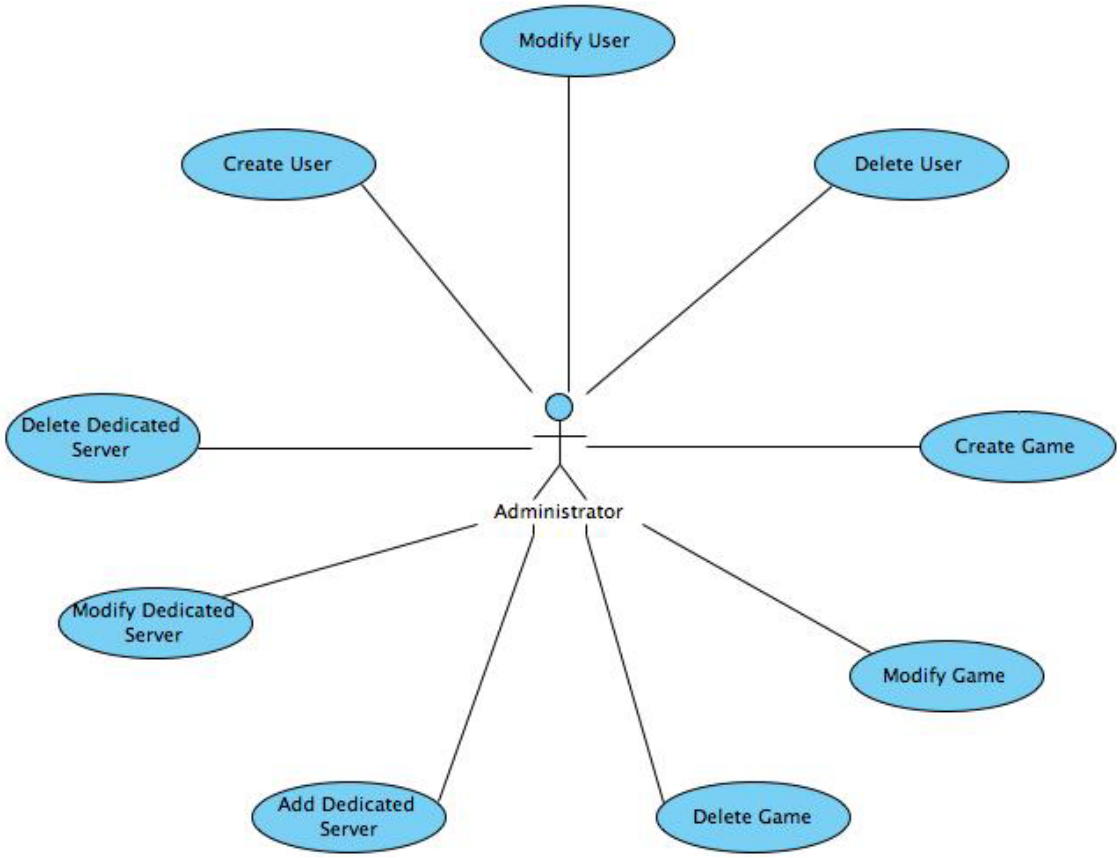


Figure 4.2 Administrator Functionality Use Case

4.2 Use Case Descriptions

Use case descriptions are used to break down the use cases into step by step explanations detailing all the exceptions and alternative flows for the use case, these can be used later in the process for developing the system to ensure that it adheres to the original specification.

The use case descriptions can be found in Appendix 10.3.

4.3 Class Diagrams

4.3.1 Overview

The diagram below shows an overview of the models, views, controllers and data access objects that may be used for the system. The operations and attributes have been omitted in order to clearly show the relationship between the classes.

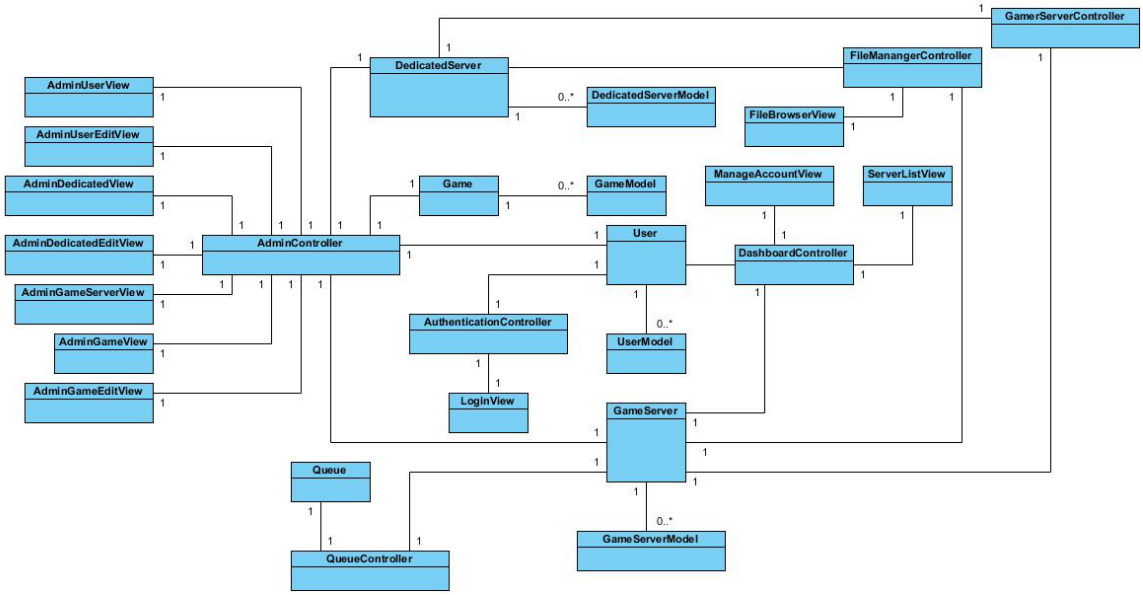


Figure 4.3 Overview Class Diagram

The main thing to note in the diagram above is that generally controllers have a one-to-one relationship with views, and that data access objects generally have a one-to-many relationship with models. The diagram bellow shows the interactions between packages more clearly.

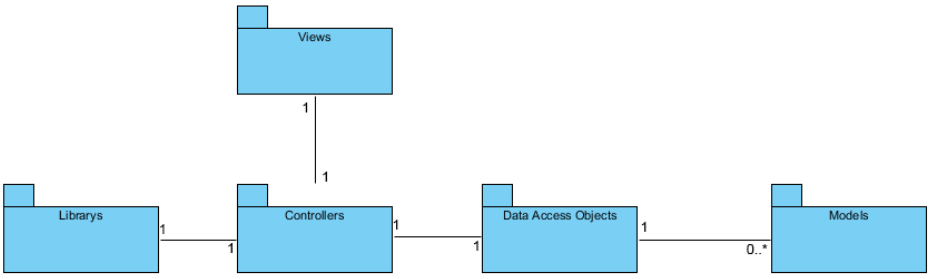


Figure 4.4 Package Relationship Diagram

4.3.2 Controllers

Controllers are used to interpret user input and requests to provide "business logic" to the application, it usually deals with obtaining the correct data from models, sometimes using a data abstraction layer and presenting the appropriate views to the user, it also provides intermediate processing for example in this projects they can queue a request for a game server to restart. The class diagram below further details the controllers expected in the system.

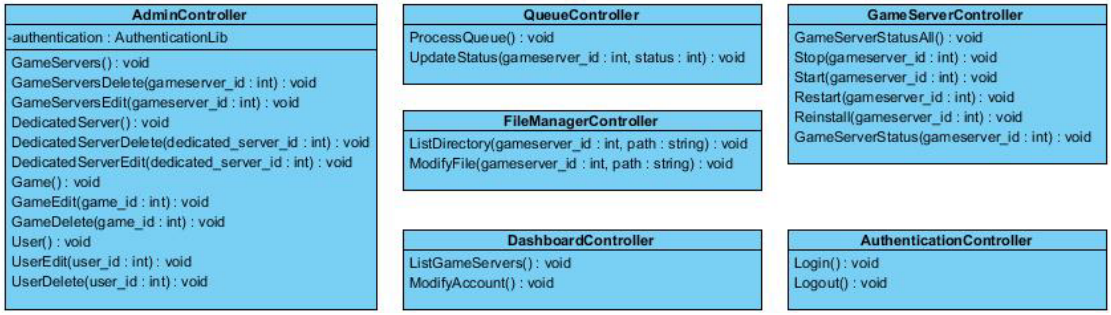


Figure 4.5 Controller Class Diagram

4.3.3 Data Access Objects

Data access objects are a data abstraction layer between models and the database, it allows the system to manipulate data whist having no knowledge of the database used. The diagram below shows the data access objects planned for this system.

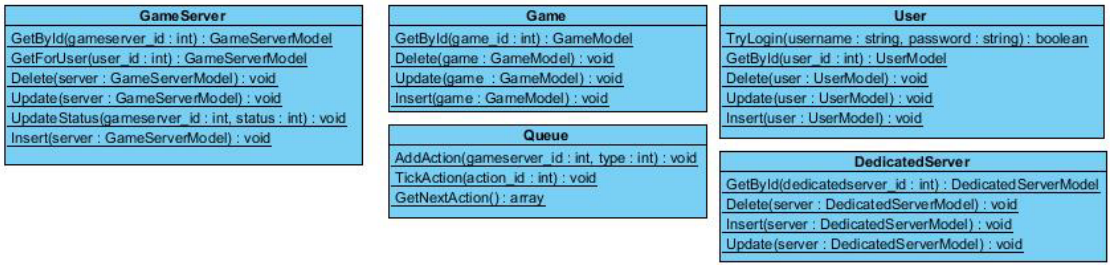


Figure 4.6 Data Access Object Class Diagram

4.3.4 Models

Models represent entities on the system; usually they closely resemble the database entities and are used as a transport of data between views, controllers and the database. Below is a set of provisional models for this project.

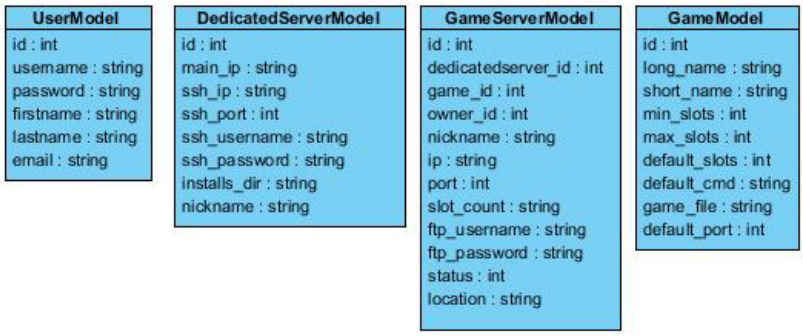


Figure 4.7 Models Class Diagram

4.3.5 Libraries

Libraries provide special utilities to the system, below three core libraries have been identified as required for this project. SSH provides an abstracted object-oriented api to the PHP ssh extension, SSHManager provides an abstraction from Dedicated Servers to SSH instances and finally the game server manager delegates the commands.

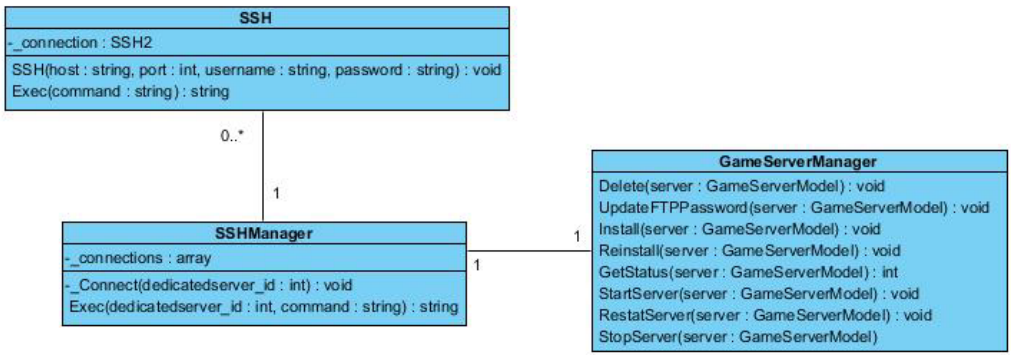


Figure 4.8 Libraries Class Diagram

4.4 Sequence Diagrams

4.4.1 Install Game Server

Figure 4.9 shows the basic flow of messages through the system as an administrator creates a game server and it is installed on the system. This process breaks down into the following steps:

1. The administrator creates a new server, and the server's status is set to pending.
2. The console kicks off the queue processing sub system, which identifies the install request and sets the server status to installing.
3. The subsystem Delegates commands to the dedicated server in order, Create User, Set Password, Create Installation Directory, Unzip Game Files, asks the system to send a curl request back when it has finished.
4. Server is updated to show as offline.

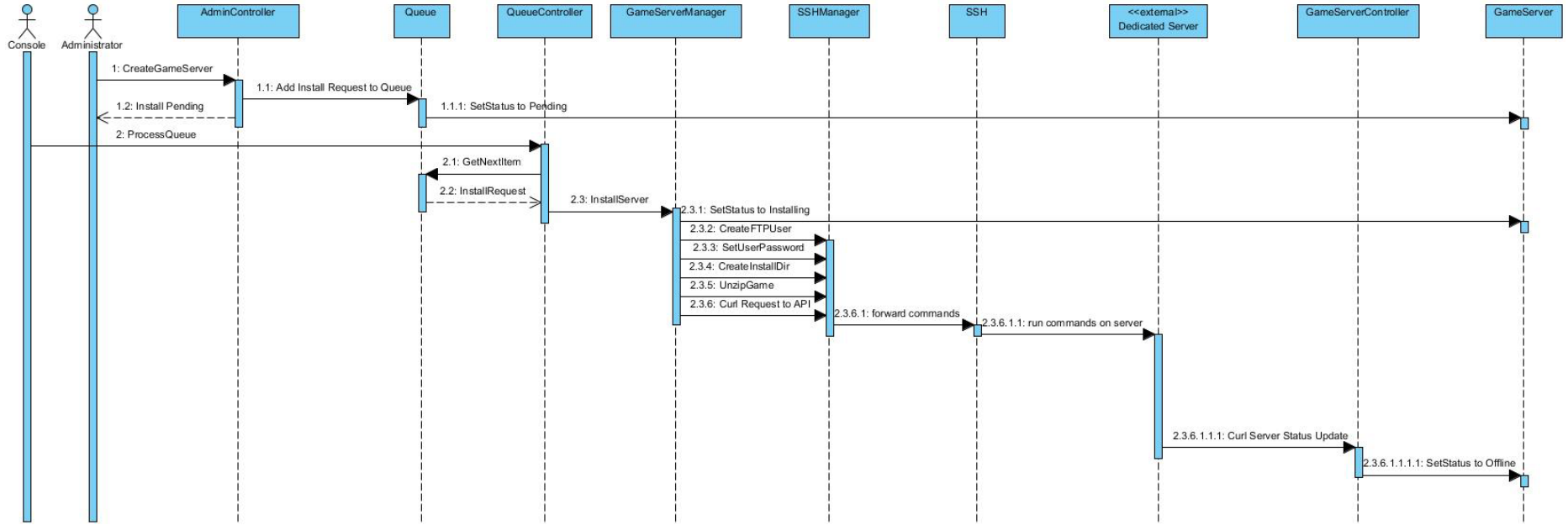


Figure 4.9 Install Game Server Sequence Diagram

4.4.2 Start Game Server

Figure 4.10 shows the basic flow of messages through the system as a client starts a game server on the system.
This process breaks down into the following steps:

1. The client request to start a game server, and the server's status is set to pending.
2. The console kicks off the queue processing sub system, which identifies the start request.
3. The subsystem delegates commands to the dedicated server to start the game server.
4. Server is updated to show as online.

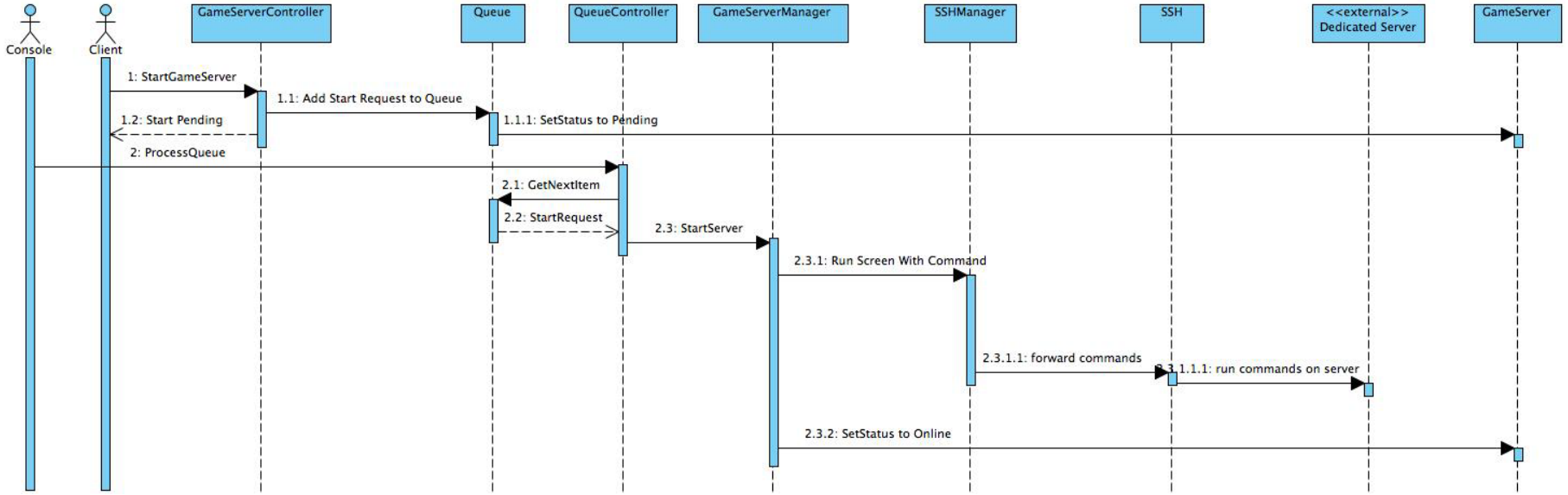


Figure 4.10 Start Game Server Sequence Diagram

4.4.3 Stop Game Server

Figure 4.11 shows the basic flow of messages through the system as a client stops a game server on the system.
This process breaks down into the following steps:

1. The client request to stop a game server, and the server's status is set to pending.
2. The console kicks off the queue processing sub system, which identifies the start request.
3. The subsystem obtains a list of process ids.
4. The process ids are iterated and commands to stop the servers are sent to the dedicated server.
5. Server is updated to show as offline.

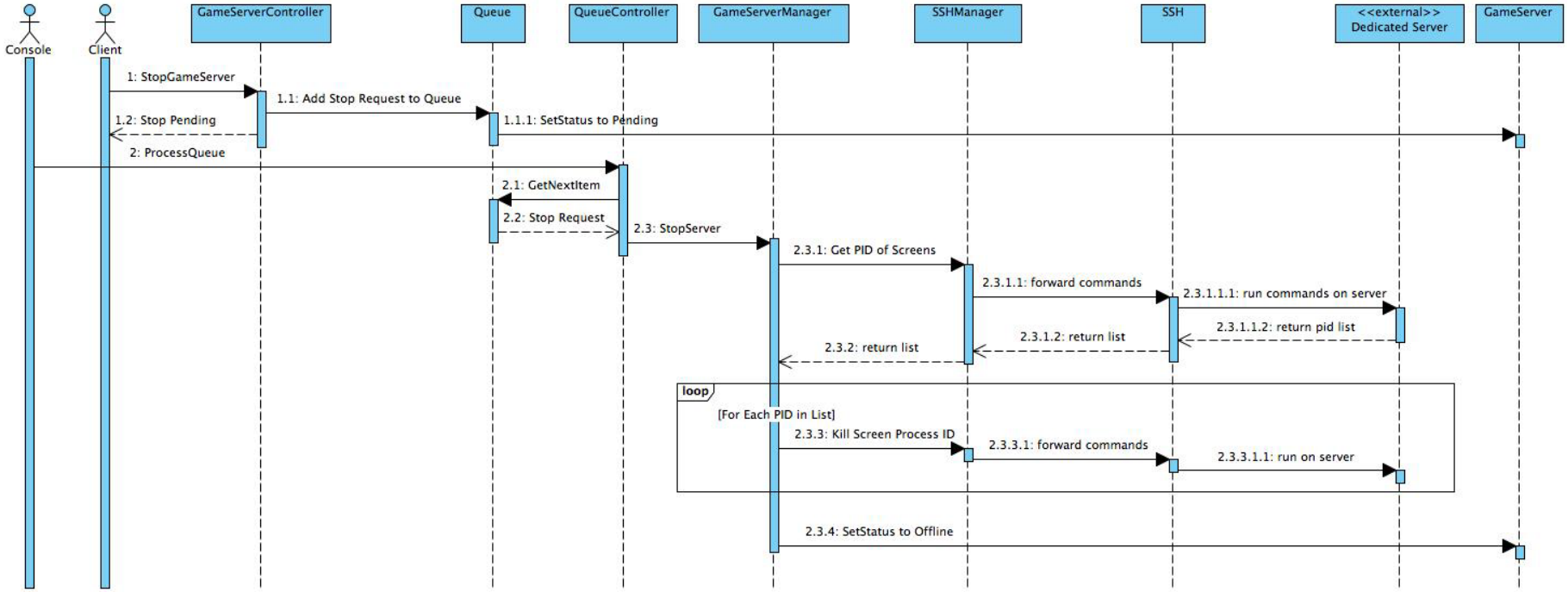


Figure 4.11 Stop Game Server Sequence Diagram

4.4.4 Restart Game Server

Figure 4.10 shows the basic flow of messages through the system as a client restart a game server on the system.
This process breaks down into the following steps:

1. The client request to start a game server, and the server's status is set to pending.
2. The console kicks off the queue processing sub system, which identifies the restart request.
3. The subsystem delegates commands to the dedicated server to kill existing processes and then starts a new process.
4. Server is updated to show as online.

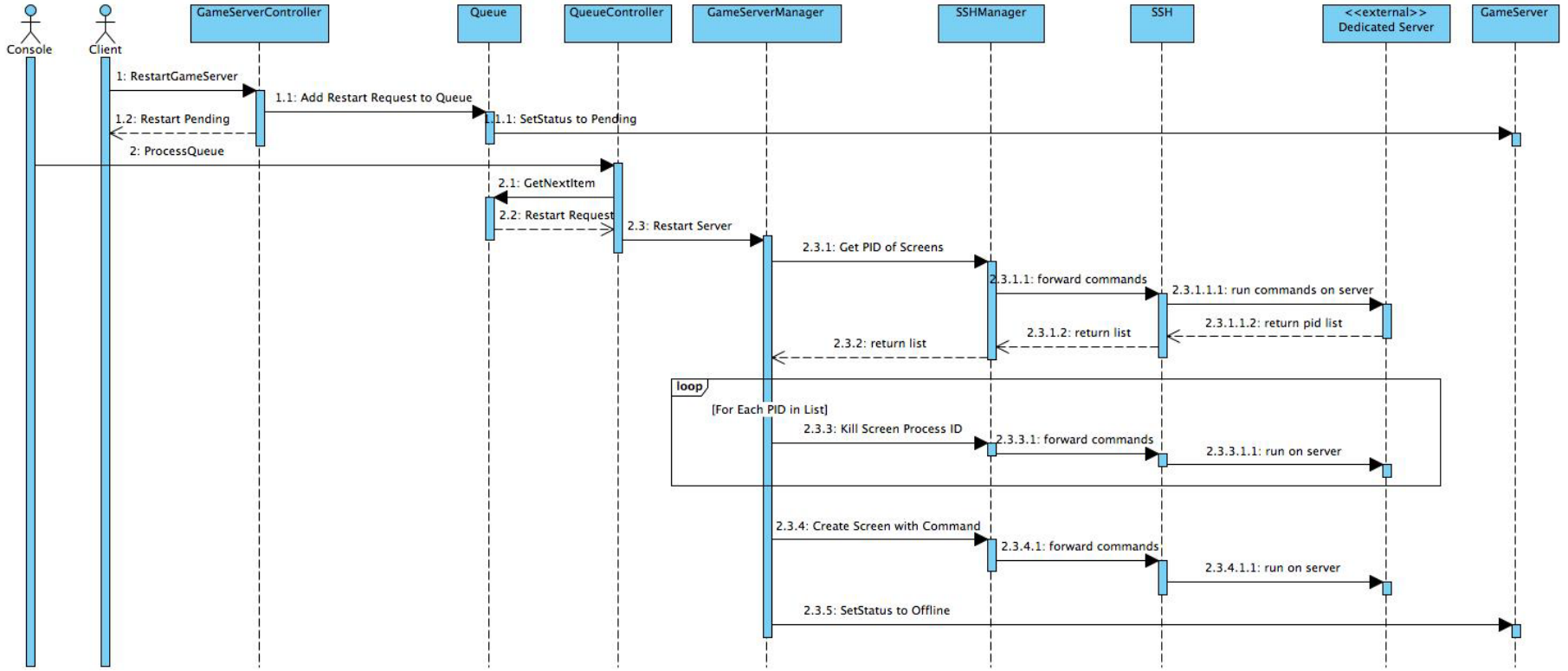


Figure 4.12 Restart Game Server Sequence Diagram

4.5 Activity Diagrams

Activity diagrams are a good way to show has streams of activities can happen in parallel, figure 4.13 shows how the queue sub system is designed, it shows that clients will make requests that are stored in the database and in parallel these requests are handled.

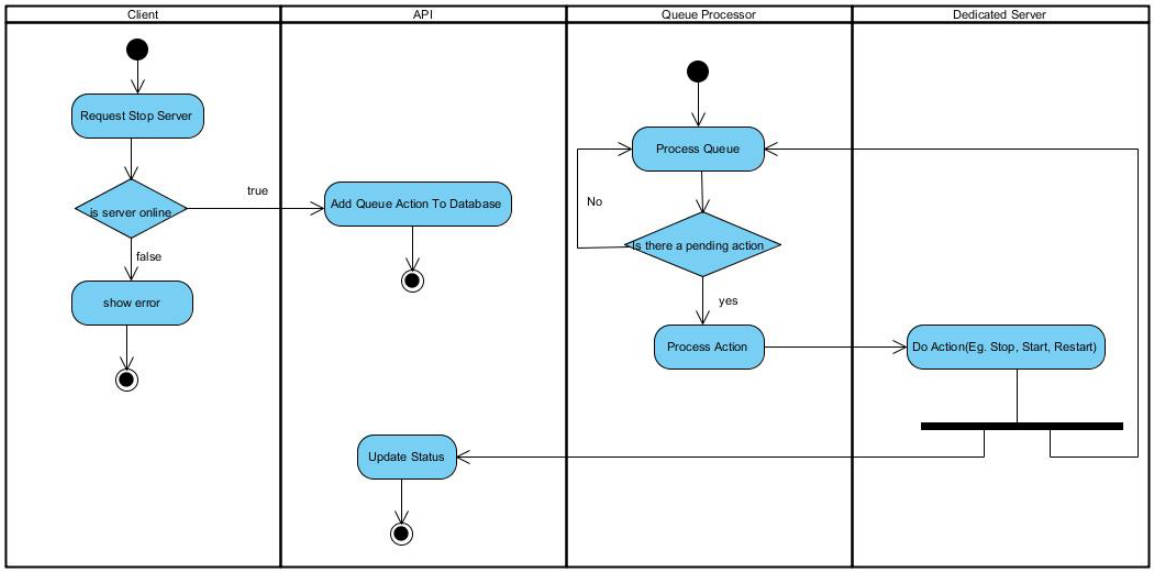


Figure 4.13 Queue Flow Activity Diagram

4.6 Extended Entity Relationship Diagram

Figure 4.14 shows a preliminary data model for the database.

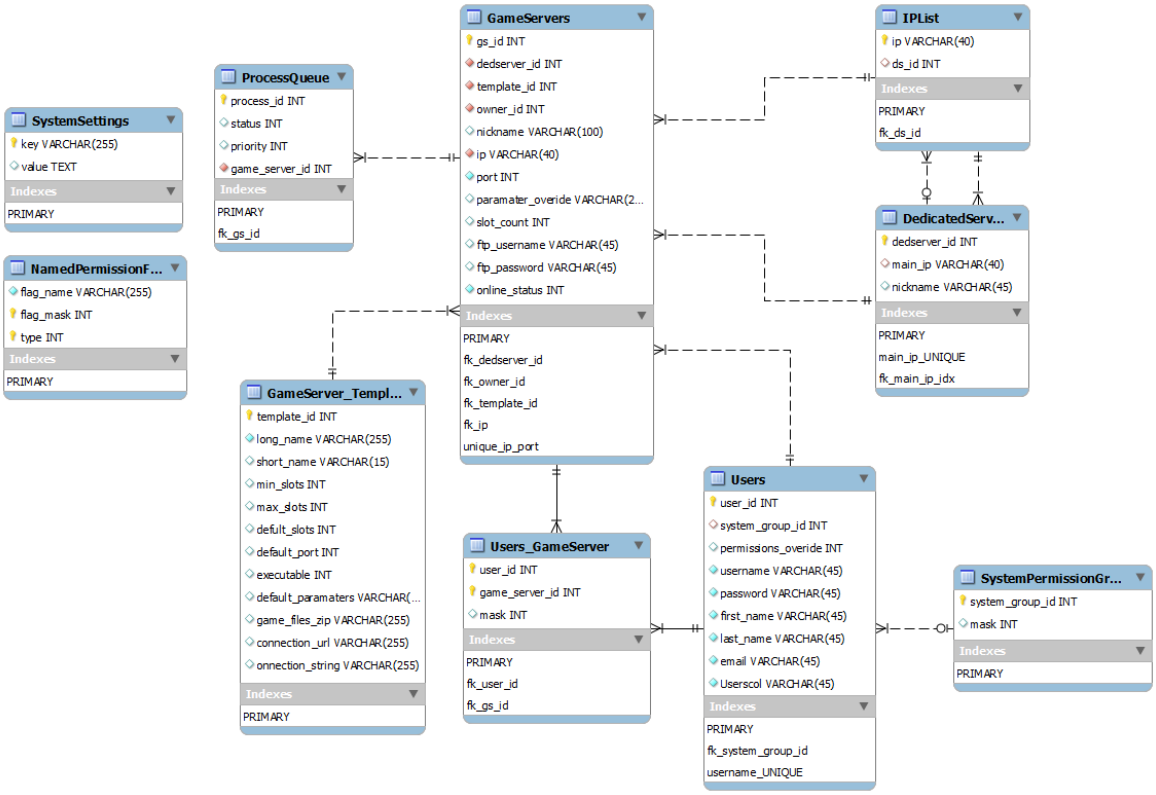


Figure 4.14 Database EERD

4.7 Prototypes

A sitemap for the website has been created and can be seen in Appendix 10.4.

4.7.1 Desktop – Client View

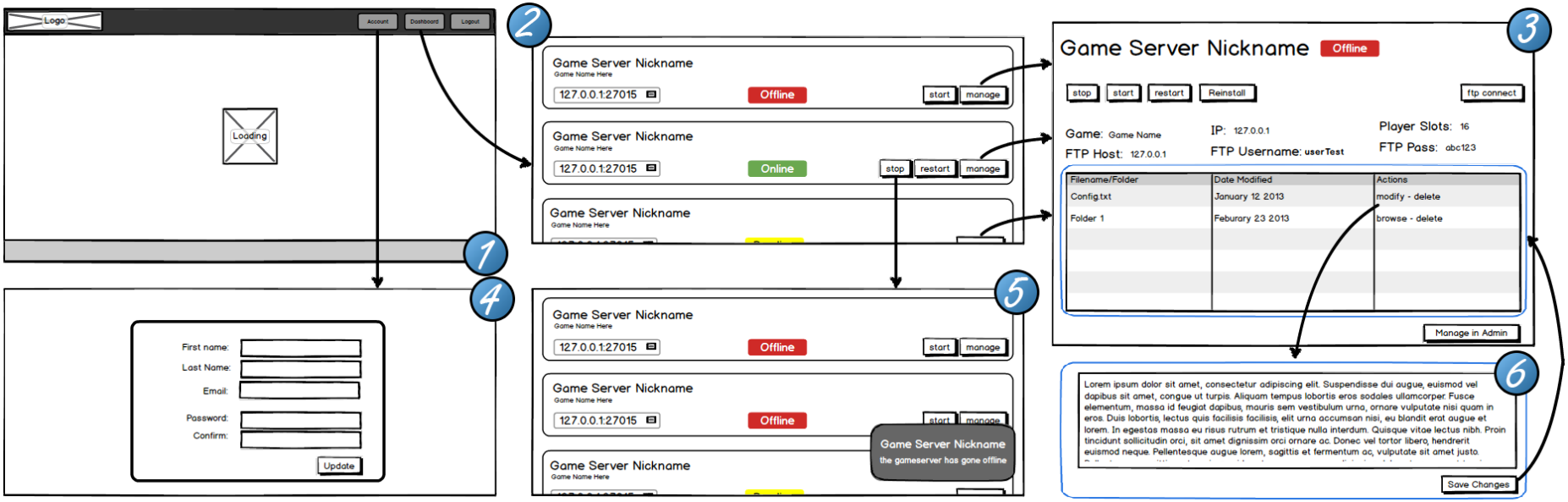


Figure 4.15 Prototype 1 - Client Area

The prototype figure 4.15 shows the interface for each of the screens user (client) who is logged into the system. It encapsulates all of the use cases that are related to the user and also covers many of the requirements. Perhaps the most influential requirement in this design is NF2 as every screen had to be designed around the idea of an application without page refreshes. Screen 1 is a loading state, this will act as a transition for any screen that needs to load, in this screen the permanent headers and footers can also be seen, these will always be the same unless a state such as the user is no longer logged in is triggered. NF2 can also be seen in screen 2 where the user has pressed the stop button for the server and the screen without a refresh notifies the user of the change and updates it state.

Screen 2 encapsulates use cases Start Game Server, Stop Game Server, Restart Game Server and Browse Game Servers with the clear list of game servers and short cut buttons for the actions attached to each one. Screen 5 encapsulates the Update Server Status use case as changes

to the servers are notified to the user. Requirement R5 can -be seen in screen 3, where users have the option to see further details about the game server, and this screen also makes available some more actions which map to use cases such as Manage Files and Reinstall Game Server. For the file manager NF2 is still a major influence to the design as the file manage does not cause page refreshes but updates its state within the screen 5 as seen at screen 6

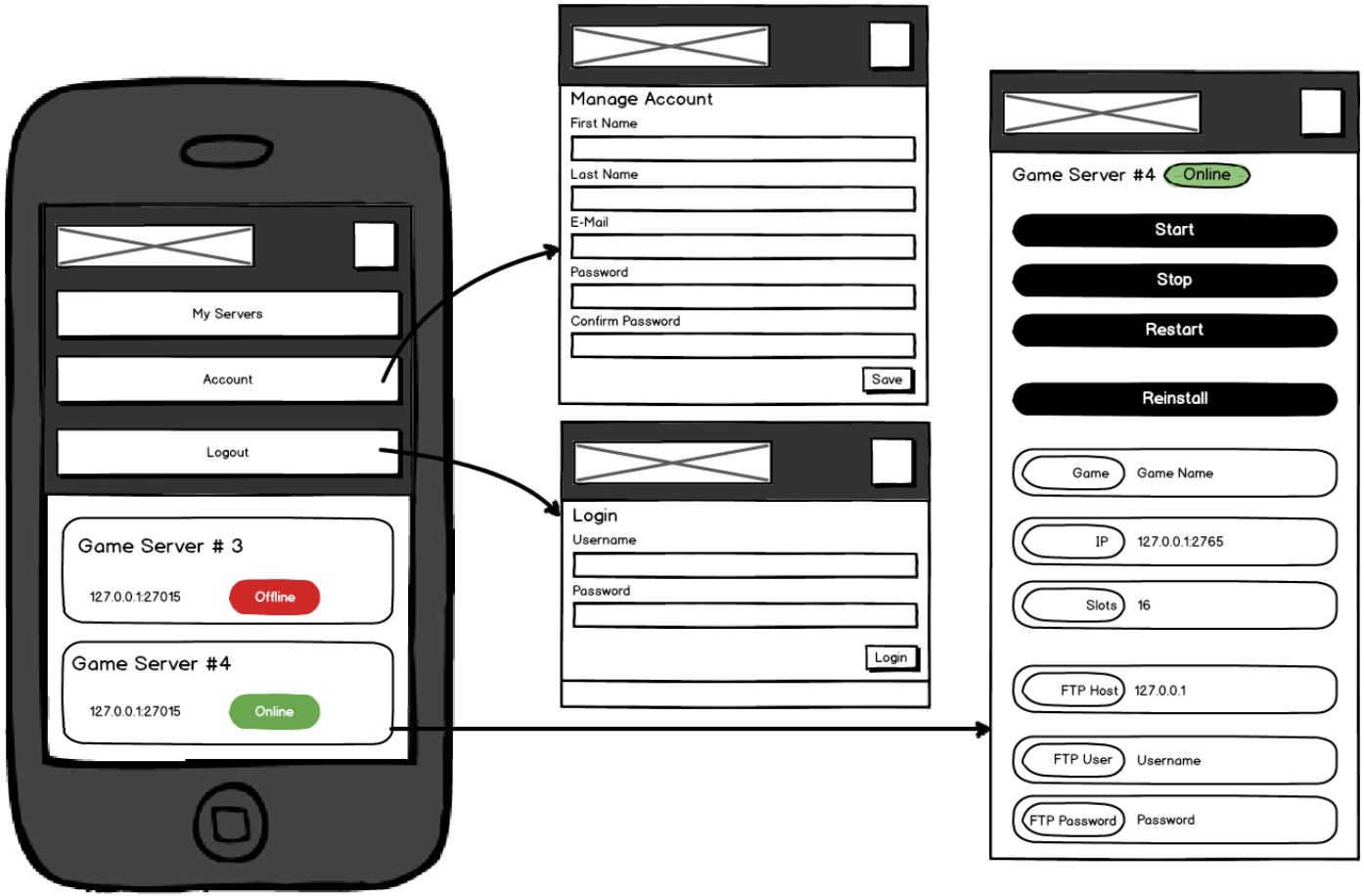


Figure 4.16 Mobile Prototype

5 The Framework

The system has been designed using the model-view-controller pattern, at this stage it may be noted that the language selected for this project is PHP and this choice will be clarified later in the implementation stage. A framework is a reusable underlying piece of software that will provide structure and utilities to an application. It is also important for a framework to be designed in a modular fashion. There are several MVC frameworks that exist for already PHP these include and are not limited to CakePHP, CodeIgniter and Zend, each of theses existing frameworks have been in development for a long time and will be more complete then creating one from the ground up however doing so will provide the flexibility that may not be offered by existing solutions, it will also act as a learning and exploration curve into low level PHP. The framework is a prerequisite to the project and in the following segments an overview of the creation and ideas behind the framework will be discussed.

5.1 Requirements Overview

Requirements for the framework are primarily developer driven with influences from the wider project below, is a summary of the requirements for the framework:

- The frameworks shall support auto loading.
- The framework shall respect the MVC pattern.
- The framework shall provide an object oriented database wrapper.
- The framework shall support “clean” URLs.
- The framework shall support a data abstraction layer (using Data Access Objects).
- The framework shall support database sessions.
- The framework shall be extendable and modular.
- The framework shall support sub-packaging.
- The framework shall support both terminal and browser routing.

5.2 Components Overview

Several components will need to be created to fulfill the requirements stated above, in the following sections these components will be discussed in more depth.

5.2.1 Autoloading

Autoloading is a popular technique used in many frameworks today, it allows classes to define their dependencies but only actually load the files when an object has been instantiated or called. The basic idea is that each class defines its dependencies before the class definition, it will also have the option to specify the namespace of the dependency so that classes with shared names can be differentiated, Snippet 5.1 shows an example of how autoloading has been implemented within the framework.

```
using('UserModel', 'App.Model');
class User extends DAO { }
```

Snippet 5.1 Autoloading in the Framework - PHP

In Snippet 5.1 we can see that the code has defined UserModel as a dependency, this will be added to the autoloaders list of pending objects and only if and when the UserModel class is used (eg. by calling new UserModel()), the autoloader be triggered and the file will be loaded as per the demand.

Also in Snipper 5.1 we can see that the code provides a namespace, this is the second argument of the "using" method. Naturally this namepspace map to the folder **/App/Model***(/UserModel.php)* however the framework supports the option of over-riding namespace locations such as mapping App.Model -> **/ModelRemap**.

5.2.2 URL Routing & Front Facing Controllers

A Front-facing controller is a file through which all other requests are delegated, not only this but it acts as a bootstrap for the entire framework by beginning the loading procedure and setting up any dependencies such as configuration files. The front facing controller should detect its environment such as the console or a browser and pass a destination into the URL Router.

The router shall natively support the "clean" URL formats and delegate the request appropriately.

http://domain.tld/subpackage-controller/method-camel-case/arg1/arg2

With the URL above the router parses and translates this into loading the object Controller found in App.Controllers.Subpackage once loaded it calls the method MethodCamelCase(arg1, arg2) passing forward the parameters shown in the link to the method.

Occasionally controllers will want to route their own methods, the URL router detects whether a controller has defined a method called _ReMap and should pass through the method name and arguments, allowing the controller to override the default routing and handle this itself, this technique has been used in the CodeIgniter framework.

Finally the URL router supports regex evaluation remapping input URL for example a developer can remap http://domain.tld/page1 to actually be processed as if it were http://domain.tld/default/page1 easily through a regex replace array defined in the configuration files.

5.2.3 Database Sessions

Database sessions aim to simulate the behavior of native PHP sessions but also aims to tackle some of the issues that surround it. The biggest benefit that database sessions bring is the ability to maintain a session over several servers whilst a website is load balanced. This is because unlike native PHP sessions where a temporary file on the current machine's file system is used, database sessions store

the data in a centralized database which usually each of the the individual load balanced servers are connected to.

Another problem seen with traditional sessions is that PHP will lock the file whilst a page is loading. What this means for simultaneous Ajax requests or page loads is that as more requests are sent to the server, they are effectively queued up due to PHP only being able to process one session at a time. This usually is not an issue, but when performing tasks that take several seconds will cause the website to stop responding for the same user.

To implement the database session a unique id is generated and used as the key for the session, the users, IP and user agent is also collected as this can be used to validate the authenticity of the session and help to alleviate session hijacking.

5.2.4 Data Access Object

Data access objects provide a data abstraction layer to the system, they are dedicated at handling the transformation from Database to Model and vice versa. Within the framework a data access object is provided with quick access to the database instance.

5.2.5 Views

Views are a core part of the model-view-controller pattern as the name suggest. Views will usually be the markup that is displayed to the user, before a view is rendered the data is required is sent through. In the framework views have and isolated contexts and are only aware of global variables and those, which have been passed to it via the view class, views do not have any prior knowledge to any other classes on the system. To achieve this the view is encapsulated into a class and output buffering is used to render the view. An example of the rendering process for a view can be seen in Snippet 5.2.

```php
public function Render()
{
    ob_start();

    extract($this->_variables);
    include($this->_filePath);

    return ob_get_clean();
}
```

Snippet 5.2 Rendering Views - PHP

5.2.6 Query Builder

Keeping with the nature of object oriented programming, the framework features a query builder that will use object-oriented notation to build dynamic SQL queries. The query builder supports the most common SQL statements, this includes SELECT, UPDATE, INSERT, DELETE, ORDER BY, LIMIT, WHERE and JOIN. An example of the query builder class can be seen in Snippet 5.3.

```php
$query = new Query();
$query->Select(
    array(
        'GS.*',
        'GST.long_name \'game_name\''
    ),
    'GameServers GS')
    ->Join('Users U')
    ->On_NE('U.user_id', 'GS.owner_id')
    ->Join('GameServer_Template GST')
    ->On_NE('GST.template_id', 'GS.template_id')
    ->Where('GS.owner_id', $user_id);

echo($query);

//prints
//SELECT GS.*, GST.long_name 'game_name' FROM GameServers GS
//JOIN Users U ON U.user_id = GS.owner_id
//JOIN GameServer_Template GST ON GST.template_id = GS.template_id
//WHERE GS.owner_id = '1'
```

Snippet 5.3 Query Builder Example – PHP

5.2.7 Error Handling

Errors in the framework are into exceptions allowing developers to take control of them. This will prevent the application from producing garbled output by ending to early and giving the developers a chance to recover the situation.

6 Implementation

6.1 Technologies

6.1.1 Server-Side Languages

Choosing a backend technology can be a challenge sometimes as there are so many available. For this project three potential technologies have been identified; ASP .NET, C++ and PHP. At this stage it is important to state that there are no real benefits of using one language over another until micro-optimization becomes a factor and performance becomes critical and the following can be summarized as the specific requirements of the project and experience of the developer.

However there are certain factors that should be considered whilst making a decision. Firstly environment support, C++ and ASP.net are uncommon to find with webhosting, they will require specialized software and in the case of ASP.net a windows server machine is required this will usually cost more to maintain. On the other hand PHP is very popular in the hosting market, PHP hosting is widely available at a low price and can be installed on existing infrastructure for free as it is an open source project.

Secondly community support can influence the decision on language, C++ has little in terms of web server documentation. Asp .NET has very good documentation from Microsoft but community support is lacking and finally PHP has a wide selection of both community support and good documentation.

Finally where performance might matter, a test performed by Wrensoft shows the potential performance difference between popular web languages.

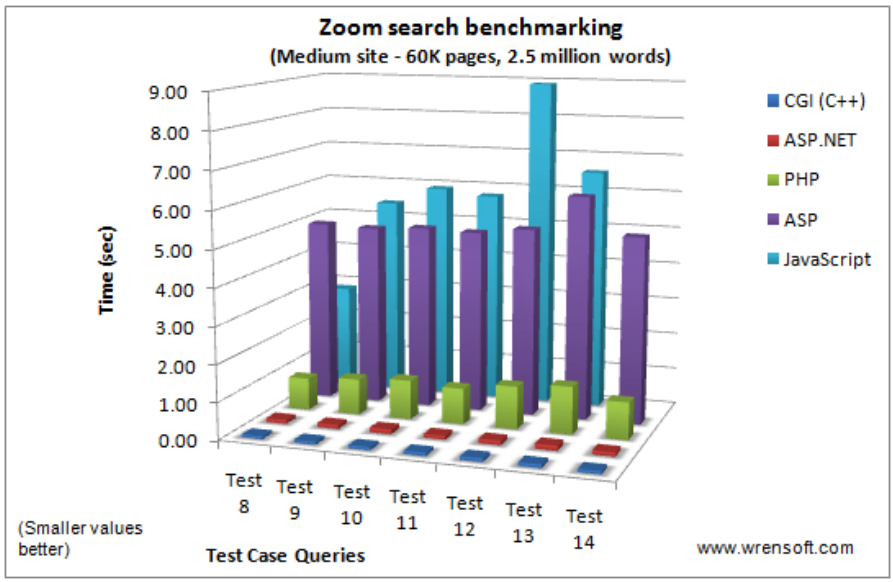


Figure 6.1 Server Side Benchmarking

(Source: http://www.wrensoft.com/zoom/benchmarks.html)

The test consistently shows that there are advantages of using C++ or ASP.net over PHP for larger websites, however all languages obtain less than a second load times which makes each a viable choice unless a website becomes extremely large.

After the above have been considered PHP Is the chosen language, this is mainly due to the author's background knowledge on this as well as the community support, documentation and infrastructure available.

6.1.2 Client Side Languages

In terms of client side implementation there are only really two main approaches, Flash or HTML/CSS/JavaScript. Immediately Flash must be ruled out as an option as there is a requirement to deploy the system on mobile devices. Flash has gained much negative feedback on mobile devices, and received much pressure from Apple because of being omitted on one of the most popular mobile devices has meant that Adobe have ceased development for Mobile Flash.

At the beginning of this document one of the objectives was to explore new and emerging technologies and in honor of this HTML5, CSS3 and JavaScript will be used, Further details will become evident about these choices as the document progresses.

The project will also use jQuery, jQuery's tag line is to write less and do more, and "has changed the way in which millions use JavaScript today". jQuery will be used extensively throughout the project as it will provide flexibility on how JavaScript is applied.

6.1.3 Databases

The choice of database is somewhat influenced by the language selected earlier. As well as this it will be good to look at open source solutions for the project due to availability and in the interest of keeping the system supported widely.

There are three main solutions that are provided by most webhost (or that can be used without a server). These are MySQL, PostgreSQL and SQLite.

SQLite is a relational database that is based on a file, it has great all round support for most queries but has a limited selection of data types. SQLite is very easy to deploy and maintain and is extremely portable making it very useful for mobile applications. However even its documentation suggests that there are better alternatives for databases with large datasets, with high concurrency or client/server applications.

PostgreSQL started life with a focus on functionality, whereas MySQL focused on Performance more recently they are both equally formidable products to use. mainly based on the fact that webhosts have wider support for MySql, and the same is true for PHP the system will use MySQL as its Relational Database Management System.

6.2 Page Transitions

One of the challenges earlier to overcome in this project is the page transitioning. In reference to non-functional requirement NF2, The system should avoid full-page refreshes and behave like a web application. The obvious solution to this is to use AJAX. However this solution alone brings many usability issues such as maintaining the users state on page changes and refreshes. It also doesn't detect page redirects therefore a user may end up with a state they didn't expect such as the login page whilst the system thinks they are in the admin area.

To tackle this problem it is important to remember following criteria:

- The system should trap links and redirect them through a page transition.
- The system should be aware of its current true state.
- The system should update the URL to match the current state.
- If the user refreshes the state should be maintained.
- Users shall be able to navigate backwards through states.

Firstly, to trap links and transition pages a JavaScript layer is required following by a back end counter-part. Snippet 6.1 shows how it is possible to transition the page states using a method with ajax. It is important to note that last_xhr is a global variable so requests are not piled up, and there is one page transition at a time.

```
var ajax_page_load = function(url, error_count)
{
    //hide the page and show the loading screen
    $('#page').hide();
    $('#loading').fadeIn();

    if(last_xhr && last_xhr.readystate != 4)
        last_xhr.abort();

    last_xhr = $.ajax({
        url: url,
        cache: false,
        success: function(page)
        {
            //update the page
            $('#page').html(page);
        },
        complete: function(e, status)
        {
            //fade the page back in unless it was cancled for the next transition
            if (status != "abort")
            {
                $('#loading').hide();
                $('#page').fadeIn();

            }
        }
    });
}
```

Snippet 6.1 Basic Page Loader - JavaScript

The initial method of trapping links that was attempted in this project was $('a').click({}); provided by jQuery, however this quickly lead to issues as newer links appeared on the page. This is because new links are not automatically added to the event listener. To tackle this issue it is important to look at JavaScript's event behavior. JavaScript bubbles events back up the document tree until either it is at the root or a parent traps the event and prevents it from bubbling further. The root for web documents is "document". jQuery provides a .on() method for adding listeners to elements which supports bubbling, it also filters these out by selectors and event type, Snippet 6.2 shows an example of a listener attached to the document, this listener will be when the click event is fired by any anchor (a) element. The other thing to consider with Snippet 6.2 is the fact that it does not cause the page to transition for internal links, such as those which begin with a hash, it also ensures that these links are relative to the website so links which are external do not trigger this.

```
//proxy all links through the ajax transitions.
$(document).on('click', 'a', function(e) {
    var url = $(this).attr('href');
    if(url.length && url[0] != "#" && url[0] == "/") {

        //handle the transition
        ajax_page_load(url);

        //stop the link from loading
        e.preventDefault();
    }
});
```

Snippet 6.2 Redirecting Anchors to Page Loader - JavaScript

Next the server side counter-part needs to be addressed. The server should ignore header and footers when a page is loaded via ajax, but if a user presses refresh a page or requests the page naturally the entire page should be rendered. To do this the controller class will be extended to represent pages that need this behavior. The new controller will detect the mode that a page is requested via and render the correct output. To code below utilizes the _remap functionality mentioned earlier which is provided by the framework.

```
<?php
using('Controller', 'Core');

class PageController extends Controller
{
    public function __construct()
    {
        $this->SetTemplate('default');
    }
    protected function IsAjaxRequest()
    {
        return (!empty($_SERVER['HTTP_X_REQUESTED_WITH']) && strtolower($_SERVER['HTTP_X_REQUESTED_WITH']) == 'xmlhttprequest');
    }
    public function __ReMap($method_name, $args)
    {
        $class_reflection  = new ReflectionClass(get_class($this));
        $method_data = $class_reflection->getMethod($method_name);

        if($method_data->isPublic())
        {
            $ajax_request  = $this->IsAjaxRequest();

            if(!$ajax_request) $this->Display('header');
            call_user_func_array(array($this, $method_name), $args);
            if(!$ajax_request) $this->Display('footer');
        }
        else
        {
            throw new Exception('Method not found...');
        }
    }
}
```

Snippet 6.3 PageController Class - PHP

Next the issue of detecting accurate states regardless of redirection will be reviewed. ajax does not natively provide redirection information, recently this has been a topic of hot debate online and the only real is for the back end to transmit its state through the headers of the request, an example of transmitting a state through a header can be seen in Snippet 6.4.

```php
header('REQUEST_URL: ' . urldecode($_SERVER['QUERY_STRING']));
```

Snippet 6.4 Sending Header for Maintaining State - PHP

The header can then be obtain via JavaScript on the client and can be used to calibrate the real URL/state.

```javascript
if(e.getResponseHeader("REQUEST_URL"))
    url = e.getResponseHeader("REQUEST_URL");
```

Snippet 6.5 Parsing Header on Front-End - JavaScript

If this header is not present, the system can gracefully fallback and assume that it did not have a redirect and the state was the one that was requested.

Finally allowing the user to refresh and maintain the state as well as the ability to browse backwards can all be solved using a new HTML5 feature. The server-side code is prepared to support these criteria however requires that the client keep the URL bar or browser location in synchronous so that when the user presses refresh on their browser, the correct page is reloaded. HTML5 offers a history API targeted as solving this issue, Facebook (as stated by the Facebook Development Team) and many other sites use the history API to achieve similar behavior. The history API provides two main functions, pushState, which is used to update a URL and store state information without reloading the page, and popState, which is an event that is triggered when a user presses the back button on a webpage that has a stack of states that have been pushed.

Each time a page transition happens the state should be pushed, in the ajax request's complete block, this can be seen in Snippet 6.6.

```javascript
if(url != window.location.pathname)
{
    if(!pushed) pushed = true;
    window.history.pushState({}, "Title", url);
}
```

Snippet 6.6 PushState Example - JavaScript

Lastly the client should handle the popState event by adding a listener as seen in Snippet 6.7.

```javascript
//also handle page back
$(window).bind('popstate', function (e) {
    if(pushed && window.location.href[window.location.href.length - 1] != '#' && !(window.location.href.indexOf('#') >= 0))
        ajax_page_load(window.location.pathname);
});
```

Snippet 6.7 PopState Example - JavaScript

One problem found with the initial implementation was that some browsers would trigger a popState on the initial page load, this seemed to be something stated in the

specification for the API and to correct for this a variable should be stored to represent the first pushState has happened and only allow popState to act if this is true.

6.3 Responsive Design

Earlier, clients of the system that were interviewed highlighted that they would like to be able to access the web application via mobile.

There are many ways that this can be achieved, most mobile phones are capable of rendering desktop websites so an option may be to rely on this feature however this does not tie in with the non-functional requirement that the website should be user-friendly, and poses several potential usability issues.

Another conventional approach to this challenge is to create two independent templates, one designed for the desktop and one designed for mobile. This approach will mean that the user is given the best possible experience for the platform that they are on. This method usually relies on JavaScript device detection, to detect if the user is on a mobile platform. However there are also many issues with this approach, this is primarily due to the issue that designs should not target specified devices/devices types, rather the screen capacity available to render, this issue become more and more apparent with the introduction of larger screens on phones and quickly became a problem with tablets. In the past often tablets were served mobile websites because they were recognized as a mobile device, this was regardless of the fact that a tablet has more screen real-estate to offer to the user, this problem can also be seen in smart phones like the Samsung Galaxy Note, this particular phone has more screen space than a conventional phone but pages served to it using this method are usually designed for a smaller screen.

This is where responsive website design comes into play. Responsive design is aimed at solving many of the problems discussed above because it is aimed specifically at brackets of screen size and adaptability of content, effectively supporting a wide array of devices if properly tested. In December 2012, Mashable claimed that 2013 Is the Year of Responsive Web Design and this certainly seems to be the case as recently there has been an explosion of sites which have been updated to support this style, a clear example of this is the University of Westminster's website. Another clear benefit of responsive design is that it reuses the same markup for all devices making maintenance easier and cheaper as changes are powered via fluid-grids, flexible media and media queries.

Snippet 6.8 shows basics to responsive design via media queries, the body font size is calculated in pixels as this is the base measurement for the test, while the primary header is using em, this is a factor of the base. When the screen is resized below 479 pixels and the media query is activated naturally and the primary header size will become smaller as it scales from the base of 9 pixels rather then the base of 12 pixels originally defined, using this technique demonstrates how sizes can be manipulated based on screen size.

```css
body {
    font: 12px Arial;
}

h1 {
    font-size: 2.0em;
}

@media only screen and  (max-width: 479px) {
    body {
        font-size: 9px;
    }
}
```

Snippet 6.8 Media Query Example - CSS

Media queries can also be used to hide content when the screen gets smaller, a common example in web design can be seen in figure 6.2.

Figure 6.2 Responsive Resize Navigation Bar

In figure 6.2 the navigation is hidden and a button becomes visible, there is a JavaScript layer to sliding down the navigation again however the associated media query for the initial transition can be seen in Snippet 6.9.

```css
#sidebar-collapse {
     visibility: hidden;
     display: none;
 }

@media only screen and  (max-width: 479px) {
    #sidebar-collapse {
        display: block;
        visibility: visible;
    }
    ul#navigation {
        display: none;
    }
}
```

Snippet 6.9 Hiding Elements using Media Queries

As mentioned earlier, fluid grids are also a part of responsive design, and this is something that has been incorporated in website.

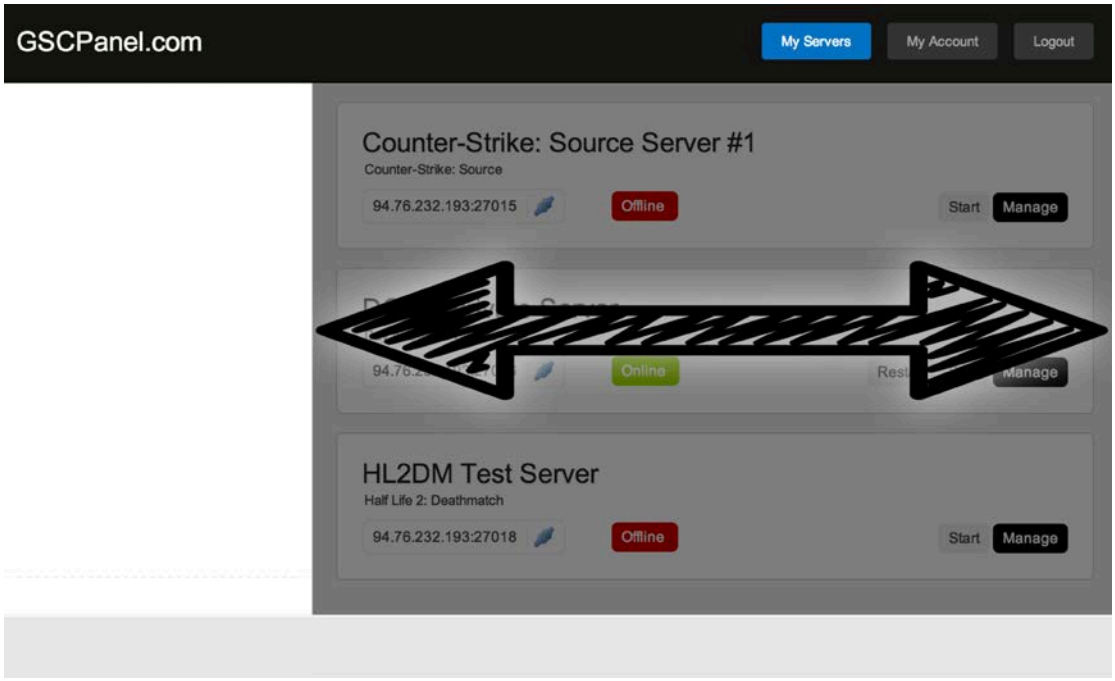


Figure 6.3 Fluid Grid Example

There are several ways to achieve fluid grids, the implementation chosen in figure 6.3 was to fix the position of the main blocks to the screen using the position: fixed; property in CSS. This will give the element a natural adaptive behavior without slowing down performance by using some JavaScript.

6.4 CSS 3 Features & Special Selectors

Throughout the project, several css3 techniques have been used to enhance both the usability and user-friendliness of the website.

6.4.1 Border Radius

A technique that can majorly improve the friendliness of the design is a well-placed border radius, it is better than bloating up the website with more images to download as was the traditional method of achieving this affect and will gracefully fallback on older browsers not tampering with the accessibility of the website.

Beyond this there are several studies supporting the fact that curved corners are easier on humans eyes, it helps the brain process information more clearly.

Figure 6.4 shows several uses of the CSS 3 border radius technique.

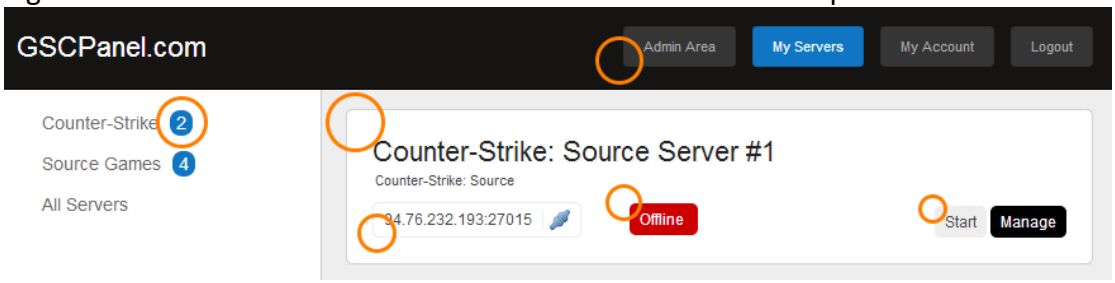


Figure 6.4 Rounded Corners - CSS

6.4.2 Background Gradients

For many of the same reasons that border-radius' enhance design, so do background gradients, background gradients elevate download on images, improve performance by not storing images in memory and also enhance the visuals of a website. A common use of background gradients is for styling buttons like seen in figure 6.5.

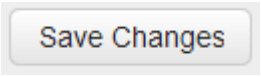


Figure 6.5 Button with background gradient

This is very simple to achieve and maintain, as seen in Snippet 6.10.

```
background: linear-gradient(to bottom, #ffffff 0%,#e5e5e5 100%);
```

Snippet 6.10 Background Gradient - CSS

6.4.3 Box Shadows

Shadows are another common technique to easy transitions on the eye, they are also another feature of CSS3. Like the other features listed above it will fall back gracefully if the feature is not available without compromising the user experience.

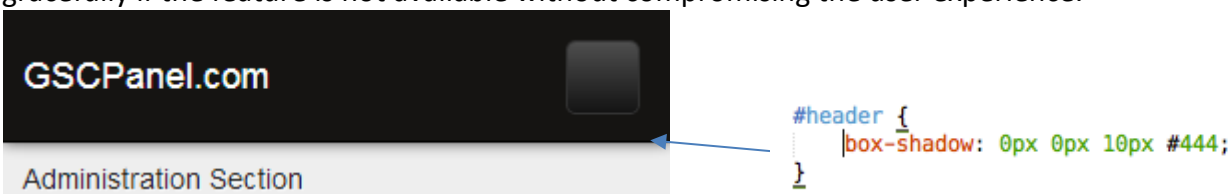


Figure 6.6 Box Shadow Example

6.4.4 Text Shadows

Using text shadows can enhance readability on the website, the example seen in figure 6.7 demonstrates an engraving effect using a 1pixel white shadow on the text.

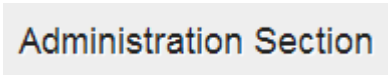


Figure 6.7 Engraved Text using Text Shadow

Figure 6.7 can simply be achieved with the code found in snippet 6.11.

```
.content_pad > h1 {
    text-shadow: 1px 1px 1px #fff;
}
```

Snippet 6.11 Text Shadow Example - CSS

6.4.5 Transitions

Often the classic technique for transitioning or animating elements is through JavaScript, css3 provides an efficient way to transition between two states.

Figure 6.8 Transition from Online to Offline

6.4.6 Hover State

Hover states allow us to provide more usability features, seen in figure 6.9 is an example of a table row being highlighted when the user mouse overs the row, this allows the user to make decisions faster and spend less time working out what row they are on.

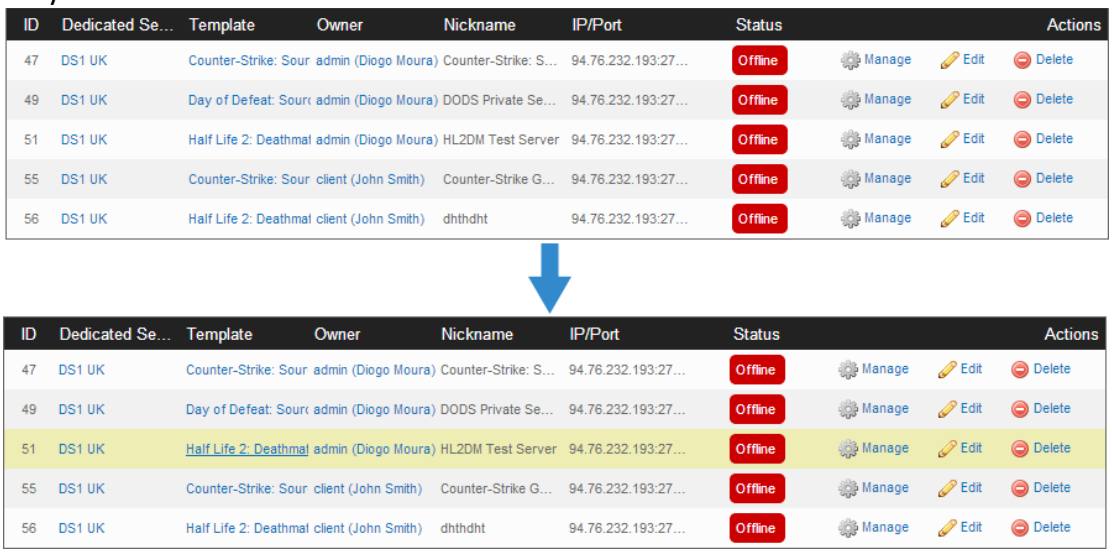


Figure 6.9 Table Hover Example

The example has been achieved with code seen in snippet 6.12.

```
table.admin-table tbody tr:hover {
    background: #EDEDB4 !important;
}
```

Snippet 6.12 Hover State Example – CSS

6.4.7 Attribute Selector

HTML5 the ability to create attributes that are prefixed with data-, this effectively allows DOM elements to transfer data about state or to provide plugin specific data. This can be used in combination with the attribute selector to style elements based on their state, for example take Snippet 6.13, which is markup used for the server listing.

```
<li data-gameserver-id="47" data-status="offline">
    ...

    <span class="gs-status"> </span>

    ...
</li>
```

Snippet 6.13 Data Attribute Example - HTML

Each element in the list contains data about the Game Server it is representing (data-gameserver-id) and data about its status (data-status). Rather than manually

managing the states representation with JavaScript, it is possible to achieve this through the attribute selector.

```
ul#server-list li[data-status='online'] span.gs-status {
    background: #95cc00;
}
ul#server-list li[data-status='offline'] span.gs-status {
    background: #CC0000;
}

ul#server-list li[data-status='restarting'] span.gs-status {
    background: #FF7400;
}

ul#server-list li[data-status='pending'] span.gs-status {
    background: #EDE275;
}

ul#server-list li[data-status='installing'] span.gs-status {
    background: #1075c2;
}
```

Snippet 6.14 Data Selector Example - CSS

This technique can also be used to hide functionality, which is not available based on state with minimal JavaScript interference.

```
ul#server-list li[data-status='pending'] a.server-action[data-action='stop'],
ul#server-list li[data-status='pending'] a.server-action[data-action='restart'],
ul#server-list li[data-status='installing'] a.server-action[data-action='start'],
ul#server-list li[data-status='installing'] a.server-action[data-action='stop'],
ul#server-list li[data-status='installing'] a.server-action[data-action='restart'],
ul#server-list li[data-status='online'] a.server-action[data-action='start']
{
    display: none;
    visibility: hidden;
}
```

Snippet 6.15 Hiding Buttons using Data Selector - CSS

6.4.8 Sibling Selector

Sibling selectors are a powerful but underused tool in CSS. They allow elements to dynamically be adjusted if siblings are detected, a good example of can be seen in snippet 6.16, which enables a sidebar next to the content area, regularly, it would require much more markup to achieve and a different id to be used for the content.

```
<div id="sidebar-tools">
</div>
<div id="sidebar">
</div>
<div id="content">
</div>
```

Snippet 6.16 Markup to be used with sibling selector - HTML

However because of the sibling selector we are able to we can say that if the sidebar is a sibling of content, that the content should shrink to accommodate it.

```
#sidebar ~ #content {
    left: 255px;
}
```

Snippet 6.17 Sibling Selector Example - CSS

JavaScript Layer

An issue that was not apparent in the initial design was how to keep the appropriate JavaScript active whilst the state changes. The idea that some JavaScript is view bound, and should only activate when the correct view is available. To address this issue a JavaScript view controller framework was created to allow modular loading and unloading of modules. The method seen in Snippet 6.18 was added just after a page transition is complete. This method detects the state and handles loading and pausing/playing of view controllers.

```
var manage_state = function (url)
{
    var page = mirror((url + '/').match(/\/?(.*?)\/.*/)[1]);

    if(currentPage != undefined && page != currentPage &&
        ViewControllers[currentPage] != undefined &&
        ViewControllers[currentPage]['ViewPause'] != undefined)
    {
        ViewControllers[currentPage].ViewPause();
    }

    if(page != currentPage &&
         ViewControllers[page] != undefined &&
         ViewControllers[page]['ViewResume'] != undefined)
    {
        ViewControllers[page].ViewResume();
    }
    else if(ViewControllers[page] == undefined)
    {
        require(['viewcontroller/' + page], function(controller) {
            ViewControllers[page] = controller;

            controller.ViewLoad();
        });
    }

    currentPage = page;
}
```

Snippet 6.18 Viewcontroller Implementation - JavaScript

In snippet 6.18 the JavaScript view controllers are directly tied to controllers on the server-side by parsing the URL, an assumption that has been made is that all page controllers will have an associated view controller. In other words the controller Page1 in on the server will map to view controller Page1 on the client.

Using RequireJS as seen in the snippet allows the framework to modularly load view controllers, as they are required, view controller are expected to conform to an interface which contains three methods, ViewLoad, ViewPause and ViewResume seen in Snippet 6.19.

```
define(function() {
    function AuthViewController() {

    }

    AuthViewController.prototype = {

        //ViewController Interface Implementation
        ViewLoad : function ()
        {
            this.ViewResume();
        },
        ViewResume : function ()
        {

        },
        ViewPause : function ()
        {

        },
        ViewUnload : function ()
        {
            //Unimplemented
        }
    };

    return new AuthViewController();
});
```

Snippet 6.19 View Controller interface - JavaScript

6.6 PHP & SSH

Earlier in the design phase SSH was identified as the interfacing system to control the dedicated server. There are many third party libraries available, but in the interest of support PHP SSH2 extension has been chosen for this project. Normally the extension provides a procedural interface to SSH so a wrapper was created for this in an object as seen in Snippet 6.20.

```
class SSH
{
    private $_connection = null;


    public function __construct($host, $port, $username, $password)
    {
        if ( ($this->_connection = ssh2_connect($host, $port)) == false )
            throw new SSHConnectionFailedException($host, $port);

        if ( !ssh2_auth_password($this->_connection, $username, $password) )
        {
            throw new Exception('Authentication rejected by server');
        }
    }

    public function Run($cmd, &$error = "")
    {
        if (!($stream = ssh2_exec($this->_connection, $cmd)))
            throw new Exception('SSH command failed');

        $errorStream = ssh2_fetch_stream($stream, SSH2_STREAM_STDERR);


        stream_set_blocking($stream, true);
        stream_set_blocking($errorStream, true);

        $data = stream_get_contents($stream);
        $error =  stream_get_contents($errorStream);

        return $data;
    }

    private function _Disconnect()
    {
        $this->Run('echo "EXITING" && exit;');
        $this->_connection = null;
    }

    public function __destruct()
    {
        $this->_Disconnect();
    }

}
```

Snippet 6.20 SSH Wrapper - PHP

Next it is important to map individual instances of the SSH objects to actual dedicated servers, to do this the SSHManager has been created which was identified earlier in the design phase.

```php
class SSHManager
{
    private $_connections = array();
    public function __construct()
    {
    }
    private function _Connect($dsid)
    {
        if(!isset($this->_connections[$dsid]))
        {
            $dedicated_server = DedicatedServer::GetModelById($dsid);
            $this->_connections[$dsid] = new SSH($dedicated_server->GetSSHIp(),
                                                 $dedicated_server->GetSSHPort(),
                                                 $dedicated_server->GetSSHUsername(),
                                                 $dedicated_server->GetSSHPassword());

        }
    }
    public function Exec($dsid, $command)
    {
        $this->_Connect($dsid);
        return $this->_connections[$dsid]->Run($command);
    }
}
```

Snippet 6.21 SSHManager Class - PHP

This class will maintain connections that have previously been used, as well as create connections to other dedicated servers on demand.

6.5 Bash Scripting, Screen & Curl

As SSH has been the chosen interfacing system, bash scripting will be required to power the functionality demanded by the system. The first issue that will be addressed is the fact that many of the commands will cause the terminal session to stay alive until they are complete, this might include unzipping a large game server package. To address this issue we must background the process, or the entire chain of commands issued so that PHP can continue to process subsequent tasks.

The easiest way to address this problem is to use bash' built in process back grounding, by appending a "&" sign at the end of a command. This approach is impractical, as it will not allow for diagnosis of problems and if something were to go wrong, and since it is not possible to recover the process, it would not be easy to spot.

Screen is a utility made available by GNU, which is used to run multiple virtual terminals simultaneously. Screen is better categorized as a terminal multiplexer, it acts as a sort of terminal back grounder, however the key difference is that terminals running under screen as able to be recovered and commands can be issued into them. Screen also has the ability to list running virtual terminals.

Basic usage of creating a dethatched (back grounded) screen can be seen in snippet 6.22.

```
screen -A -m -d -S {SCREEN_NAME} {COMMAND}
```

Snippet 6.22 Screen Sample Usage - BASH

This is useful for running a single command, but does not allow us to chain commands and have them work in order, which is typically needed by something like the installation process. This is where the bash command is able to help, bash can run multiple commands using the –c flag shown in Snippet 6.23.

```
bash -c 'command1; command2; command3'
```

Snippet 6.23 Channing Commands - BASH

It in then possible to chain the screen and bash command to achieve back grounded with a queue of processing, this can be seen in Snippet 6.24.

```
screen -A -m -d -S {SCREEN_NAME} bash -c 'command1; command2; command3'
```

Snippet 6.24 Combining Screen and Multiple Commands - BASH

Another command that will be useful for the later implementation stages is the ability to identify running screens; this will be particularly useful for stopping servers and syncing the statuses.

```
[root@game /]# screen -list
There are screens on:
        3228.gscp-49    (Detached)
        3211.gscp-47    (Detached)
        3245.gscp-51    (Detached)
3 Sockets in /var/run/screen/S-root.
```

Snippet 6.25 Screen Listing - BASH

Naming screens with the convention seen in snippet 6.25, for example, gscp-{ID}, will allow the screen to be quickly identified and by filtering the output of the command, the awk utility can be used to obtain the process id alone.

```
screen -ls | awk '/\.gscp-{ID}\t/ {print strtonum($1)}'
```

Snippet 6.26 Using awk to obtain PIDs - Bash

The example seen in snippet 6.26 will return a list of all process ids associated with screens of gscp-{ID}, this way even game servers that have multiple instances running will be detected.

The final prerequisite to the following sections, is ensuring responses to the system via the dedicated server; if a stack of commands have been queued on the dedicated server, the server should communicate back with the system when it is complete. Providing a web service that dedicated servers can use will do this. Curl makes trivial web requests easy through the terminal.

```
curl http://test.gscpanel.com/api/test/1
```

Snippet 6.27 Curl Example - BASH

6.6 Game Server Installation

Earlier in the design phase, a sequence diagram was drawn detailing the process of installing a new game server. This can be broken down into the following steps:

1. Creating a user on the system and setting their password.
2. Unzipping the game files into the new users directory.
3. Updating the ownership of the files, which have been unzipped.
4. Relaying back to the server that the installation has finished.

Firstly adding a new user in Linux is easy. useradd is a utility provided with most Linux distributions for creating users on the system however it is not possible to change passwords using plaintext in a single command line. To overcome this issue useradd provides a –p flag which uses mcrypt encrypted passwords with a two character salt. Generating a password encrypted in this way through in PHP can be seen in Snippet 6.28.

```php
private function  _GeneratePassSalt ()
{
    $chars = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
    return substr(str_shuffle($chars), 0, 2);
}

$password = crypt($ftp_password, $this->_GeneratePassSalt());
```

Snippet 6.28 Generating a password salt for Linux - PHP

One the password is generated it can be fed into the command seen in snippet 6.29.

```
useradd -p {encrypted_pass} -d {gameserver_dir} {ftp_user}
```

Snippet 6.29 useradd Sample - BASH

To unzip the packed server files the Unzip utility can be used, sample usage can be seen in figure 6.30.

```
unzip {zip_dir}/{zip_file} -d {gs_dir}
```

Snippet 6.30 unzip sample usage - BASH

For the user to manage their files the correct ftp permissions are required, to do this we can use the clown command sampled in snippet 6.31.

```
chown -R {USER}:{USER} {gs_dir}/*
```

Snippet 6.31 Change owner example - BASH

Finally the entire process can be coded into the method seen at snippet 6.32.

```php
public function Install(GameServerModel $gameserver)
{
    echo "Installin game server #{$gameserver->GetId()}.\n";
    $password = crypt($gameserver->GetFtpPass(), $this->_GeneratePassSalt());
    $this->_ssh->Exec($gameserver->GetDedicatedServerId(), "useradd -p {$password} -d {$gameserver->GetServerDir()} {$gameserver->GetFtpUser()}");

    GameServer::SetServerStatus($gameserver->GetId(), GameServerStatus::INSTALLING);

    $dedicated_server = DedicatedServer::GetModelById($gameserver->GetDedicatedServerId());
    $gameserver_template = GameServerTemplate::GetModelById($gameserver->GetTemplateId());

    $zip_dir = $dedicated_server->GetInstallDir();
    $zip_file = $gameserver_template->GetZipFilename();

    $gs_dir = $gameserver->GetServerDir();


    $cmd = "screen -A -m -d -S gscp-install-{$gameserver->GetId()} bash -c 'unzip {$zip_dir}/{$zip_file} -d {$gs_dir};
                                                                        chown -R {$gameserver->GetFtpUser()}:{$gameserver->GetFtpUser()} {$gs_dir}/*;
                                                                        curl {$this->_base_url}/api/queue/gameserver-stopped/{$gameserver->GetId()}'";

    $this->_ssh->Exec($gameserver->GetDedicatedServerId(), $cmd);
}
```

Snippet 6.32 Install Server Example - PHP

Reinstalling the same server is a similar process, the difference being that the user is not re-added and the current files are deleted first, as well as any instances of the game server need to be stopped first, the entire process can be seen at snippet 6.33.

```php
public function Reinstall(GameServerModel $gameserver)
{
    echo "Reinstalling server #{$gameserver->GetId()}.\n";
    GameServer::SetServerStatus($gameserver->GetId(), GameServerStatus::INSTALLING);

    $this->StopServer($gameserver, false);

    $dedicated_server = DedicatedServer::GetModelById($gameserver->GetDedicatedServerId());
    $gameserver_template = GameServerTemplate::GetModelById($gameserver->GetTemplateId());

    $zip_dir = $dedicated_server->GetInstallDir();
    $zip_file = $gameserver_template->GetZipFilename();

    $gs_dir = $gameserver->GetServerDir();


    $cmd = "screen -A -m -d -S gscp-install-{$gameserver->GetId()} bash -c 'rm -rf {$gs_dir}/*;
                                                                            unzip {$zip_dir}/{$zip_file} -d {$gs_dir};
                                                                            chown -R {$gameserver->GetFtpUser()}:{$gameserver->GetFtpUser()} {$gs_dir}/*;
                                                                            curl {$this->_base_url}/api/queue/gameserver-stopped/{$gameserver->GetId()}'";

    $this->_ssh->Exec($gameserver->GetDedicatedServerId(), $cmd);
}
```

Snippet 6.33 Reinstall Server Example - PHP

6.7 Game Server Stop & Start

Before starting a server the first thing that needs to be considered is the command line to start the particular server, this is generated rom the database by replacing template variables like {ip} and {port}.

```php
private function _GenerateCommandLine(GameServerModel $gameserver)
{
    $cmd_line = $gameserver->GetCommandLine();
    $cmd_line = str_replace('{port}', $gameserver->GetPort(), $cmd_line);
    $cmd_line = str_replace('{ip}', $gameserver->GetIP(), $cmd_line);
    $cmd_line = str_replace('{slots}', $gameserver->GetSlots(), $cmd_line);

    return $cmd_line;
}
```

Snippet 6.34 Generate Command Line - PHP

Once the command line has been generated it can be used to start the server as described in the bash scripting section earlier, code for this can be seen in snippet 6.35.

```php
public function StartServer(GameServerModel $gameserver)
{
    $cmd = "cd {$gameserver->GetServerDir()};
            screen -A -m -d -S gscp-{$gameserver->GetId()} {$this->_GenerateCommandLine($gameserver)}";

    $this->_ssh->Exec($gameserver->GetDedicatedServerId(), $cmd);
    $this->_ssh->Exec($gameserver->GetDedicatedServerId(), 'curl http://test.gscpanel.com/api/queue/gameserver-started/' . $gameserver->GetId());
}
```

Snippet 6.35 Start Server Example - PHP

Stopping a server is slightly different. This has been documented earlier in the sequence diagram section, and will be used as a basis for the routine. In short a list of game servers running under the id of the game server specified, is obtained using the screen –ls in combination with awk as discussed earlier on. The list of process ids will be iterated through and for each process id the system will perform two commands to try and end the screen, the first is pkill –TERM –P process_id which aims to force the screen closed, and the other is screen –D {pid} –X quit to induce a graceful termination.

```php
public function StopServer(GameServerModel $gameserver, $update = true)
{
    $cmd_get_pids = 'screen -ls | awk \'/\\.gscp-' . $gameserver->GetId() . '\\t/ {print strtonum($1)}\'';
    $screen_ids =  explode("\n", $this->_ssh->Exec($gameserver->GetDedicatedServerId(), $cmd_get_pids));
    array_pop($screen_ids); //get rid of the blank

    $instances = count($screen_ids);

    foreach($screen_ids as $screen_id)
    {
        echo "Terminating Screen With PID $screen_id.\n";
        $this->_ssh->Exec($gameserver->GetDedicatedServerId(), "pkill -TERM -P $screen_id");
        $this->_ssh->Exec($gameserver->GetDedicatedServerId(), "screen -D {$screen_id}.gscp-{$gameserver->GetId()} -X quit");
    }

    if($update)
        $this->_ssh->Exec($gameserver->GetDedicatedServerId(), "curl {$this->_base_url}/api/queue/gameserver-stopped/{$gameserver->GetId()}");
}
```

Snippet 6.36 Stop Server Sample - PHP

6.8 Game Server Status Syncing

In the pervious sections we can see several callbacks to API controllers, and as mentioned a web service will be required to allow dedicated servers to report status updates. An example of the API provided to dedicated servers can be seen in snippet 6.37.

```php
class QueueController extends AjaxController
{
    public function __construct()
    {
        parent::__construct();
    }
    public function Invoke()
    {
    }
    public function GameserverStarted($gsid)
    {
        GameServer::SetServerStatus($gsid, GameServerStatus::ONLINE);
    }
    public function GameserverStopped($gsid)
    {
        GameServer::SetServerStatus($gsid, GameServerStatus::OFFLINE);
    }
    public function GameserverRestarting($gsid)
    {
        GameServer::SetServerStatus($gsid, GameServerStatus::RESTARTING);
    }
}
```

Snippet 6.37 QueueController - PHP

6.9 Queue Requests

Before any of the actions above can be processed the user must add requests to the queue to perform these actions, the first thing we need is the Queue data access object, seen in snippet 6.38.

```php
class Queue extends DAO
{
    public static function GetNextItem()
    {
        ...
    }
    public static function DeleteAllInProgress()
    {
        ...
    }
    public static function AddAction($gs_id, $type, $data = null)
    {
        ...
    }
    public static function InProgress($actid)
    {
        ...
    }
}
```

Snippet 6.38 Queue - PHP

The Queue object will provide quick access to the necessary functionality to operate the data pertaining to queues, it is important to also to avoid magic numbers in the code so to overcome this an enumeration class to represent the queue event type shall be used.

```php
class QueueEvent
{
    const STOP_SERVER = 0;
    const START_SERVER = 1;
    const RESTART_SERVER = 2;
    const REINSTALL_SERVER = 3;
    const INSTALL_SERVER = 4;
    const UPDATE_FTP_PASSWORD = 5;
    const DELETE_SERVER = 6;
}
```

Snippet 6.39 QueueEvent Example - PHP

Next when a user clicks a button to perform an action on a game server, a controller needs to be created to handle these requests via ajax. The controller will add the requests to the queue for processing. Another job of this controller is to verify the current states of the game server and validate the requests. A sample for validating permissions can be seen in snippet 6.40.

```php
private function _ValidateServerPermissions($gsid)
{
    try {
        $this->_ForceAuthenticate();
        if(!($this->auth->GetUser()->Permissions()->Has('admin') || GameServer::GetModelById($gsid)->GetOwnerId() == $this->auth->GetUser()->GetId()))
        {
            $this->Error('Invalid Permissions');
            die();
        }
    }
    catch(Exception $e)
    {
        $this->Error($e->getMessage());
        die();
    }
}
private function _SetStatusPending($gsid)
{
    try{
        GameServer::SetServerStatus($gsid, GameServerStatus::PENDING);
    }
    catch(Exception $e)
    {
        $this->Error($e->getMessage());
    }
}
```

Snippet 6.40 Validate Permissions - PHP

Finally a sample of the API provided to start the server can be seen in snippet 6.41.

```php
public function Start($gsid)
{
    try {
        $this->_ValidateServerPermissions($gsid);

        $currentStatus = GameServer::GetModelById($gsid)->GetStatus();

        if($currentStatus != GameServerStatus::ONLINE && $currentStatus != GameServerStatus::PENDING && $currentStatus != GameServerStatus::INSTALLING)
        {
            $this->_SetStatusPending($gsid);
            Queue::AddAction($gsid, QueueEvent::START_SERVER);

            $this->Out(null);
        }
        else
        {
            $this->Error('Cannot start the server as it is already online or there is an action pending!');
        }
    }
    catch(Exception $e)
    {
        $this->Error($e->getMessage());
    }
}
```

Snippet 6.41 Start Server - PHP

6.10 Cron

The final piece of the puzzle to making the queuing system work is to invoke the system via the terminal on a regular basis, the way that has been chosen is to schedule the tasks is Linux' built in cron system.

The first step to doing this is to make the front facing controller support input from the terminal, which is achieved with snippet 6.42.

```php
$arguments = getopt("q:");

if(defined('STDIN') || isset($arguments['q']))
{
    if(!defined('STDIN')) define('STDIN', true);
    $query_string = $arguments['q'];
}else{
    $query_string =  isset($_SERVER['QUERY_STRING']) ? urldecode($_SERVER['QUERY_STRING']) : '';
}

Router::SetConfig(Config::Get('Router'));
Router::Dispatch($query_string);
```

Snippet 6.42 Running from Terminal - PHP

Snippet 6.42 obtains a list of arguments which are only passed in through the terminal, it also checks if STDIN (standard input) is defined, if so it uses the input from the terminal flag –q as the query string to process the request, otherwise it uses the website URLs query string.

The next step is to extend the base controller and create a cron controller that will only accept console requests, this is as simple as ensuring that STDIN is defined and if not terminating the program as seen in Snippet 6.43.

```php
class CronController extends Controller
{
    public function __construct()
    {
        if(!defined('STDIN'))
            die();
    }
}
```

Snippet 6.43 Cron Controller - PHP

Following this we need to create a controller which to handle pending requests that have been added to the queue. The controller will look at each request in the queue and proses it via the first-in-first-out method. The controller will determine the request type and delegate the request to the game server manager seen in snippet 6.4.

```php
class QueueController extends CronController
{
    private $_gs_man = null;
    public function __construct()  {...}
    public function CleanUp()  {...}
    public function Status() {...}
    public function Invoke() {...}
    private function _NextItem()
    {
        $data = Queue::GetNextItem()->fetch();
        if(!empty($data))
            $this->_ProcesItem($data);
    }
    private function _ProcesItem($item)
    {

        Queue::InProgress($item['action_id']);
        $data = unserialize($item['data']);

        switch($item['type'])
        {
            case QueueEvent::START_SERVER:
                echo "Start Event \n";
                $this->_gs_man->StartServer(GameServer::GetModelById($item['gs_id']));
                break;
            case QueueEvent::STOP_SERVER:
                echo "Stop Event \n";
                $this->_gs_man->StopServer(GameServer::GetModelById($item['gs_id']));
                break;
            case QueueEvent::RESTART_SERVER:
                echo "Restart Event \n";
                $this->_gs_man->RestartServer(GameServer::GetModelById($item['gs_id']));
                break;
                //and so on for remaining events.
        }

        $this->_NextItem();
    }
}
```

Snippet 6.44 Queue Controller - PHP

Finally to trigger these requests the system currently uses a minutely cron job to kick off the process.

Minute	Hour	Day	Month	Weekday	Command
*	*	*	*	*	php /home/gscpanel/websites/test/index.php -q /internal/queue

Figure 6.10 Example Cron Sample

At this stage the entire queue process is ready to be used and has been pulled toether, the system is able stop, start and restart game servers also install them. Administrators have the option to change the polling rate of processing queue requests by changing the frequency of the cron job.

6.11 File Manager

A controller will need to be produced to handle clients browsing their files without an ftp client. PHP offers a dedicated FTP library, however states in the documentation that for basic usage that the file operations provided such as fopen can be used in conjunction with an ftp:// URL. This file manage controller can be seen in Figure 6.11.

```
class FtpController extends Controller
{
    public function View($gsid)
    {
        $this->_ValidateServerPermissions($gsid);
        $dir = '';
        if(isset($_GET['path']))
        {
            $dir = $_GET['path'];
        }
        $gameserver = GameServer::GetModelById($gsid);


        $link = "ftp://{$gameserver->GetFtpUser()}:{$gameserver->GetFtpPass()}@{$gameserver->GetIp()}:21/{$dir}";

        $content = file_get_contents($link);

        $this->Display('default/ftp/edit', array(
            'cur_path' => $dir,
            'content'   =>  $content
        ));
    }

    public function Dir($gsid)
    {
        $this->_ValidateServerPermissions($gsid);
        $dir = '';
        if(isset($_GET['path']))
        {
            $dir = $_GET['path'];
            if(substr($dir, strlen($dir) -1 ) != '/')
            {
                $dir .= '/';
            }
        }
        $gameserver = GameServer::GetModelById($gsid);


        $link = "ftp://{$gameserver->GetFtpUser()}:{$gameserver->GetFtpPass()}@{$gameserver->GetIp()}:21/{$dir}";

        $items = array();

        if ($handle = opendir($link))
        {
            while (false !== ($entry = readdir($handle)))
            {
                //popilate the object array for the view
            }

            $this->Display('default/ftp/dir_list', array(
                'items' => $items,
                'cur_path' => $dir
            ));
            closedir($handle);
        }
    }

}
```

Figure 6.11 File Manager Code - PHP

7 Testing

Testing is a crucial component to success of any software project. A variety of testing can be carried out in order to ensure that a system is robust, and that any issues are clearly documented.

7.1 User Testing

Throughout the projects implementation phase, several users were asked to use the system and test it as it was being developed. This is a great way of testing and falls in line with the incremental build methodology chosen earlier on. Most of the issues identified by the users were fixed and these included things like validation error, and user interface glitches. This also gave a chance for users to give feedback and for this feedback to be incorporated. The final product has also been user tested and minor defects with the user interface in respect to the responsive design have been found.

7.2 System Testing

System testing will allow thorough testing of the entire system. A test plan has been produced and filled in at appendix 10.5. The testing has been broken into layers, first requirements which this will allow rigorous confirmation whether or not requirements have been met, then in to success and failure cases for all the relevant use cases.

Several tests were failed and below they will be looked at in closer detail.

7.2.1 The File Manager Bugs

The file manager was a hot spot for issues, however none of these presented major security risks. The bugs found in the file manager did however present usability issues for example in test cases #79 and #80 the file manager allows the creation of a file which does not exist in the system via the edit functionality, this is not an intended feature and is possible due to the lack of validation in this class, also due to the lack of validation it is possible to force the system to open files which should not be edited via a text editor such as binary files. To fix this issue validation should be enforced on the system to ensure that access to the files is permitted, and that the files are safe to display in a website.

Next test cases #87-90 show a state representation bug with the system, to fix, the file manager may have be recoded from the ground up being powered by json rather then being rendered on the server. This is a consequence of bad design for this module.

7.2.2 Extended Server View

Test cases #106-109 document some issues with the extended server view, the system correctly detects the invalid permissions but a mistake in the program causes

the system to throw an exception. Trivial things like this stress the importance of testing.

7.2.3 Update FTP Password Bug

In test case #323 there is a bug, which causes the users, manage section to show an exception while a server is due for a password update. This problem happens as the database only stores the most current password, and as there is a delay between the Queue synchronizing the password. This is not a major issue but is something that should be addressed by capturing the error and providing a user-friendly message.

7.2.4 Critical Bug: Reinstall with invalid Id

Whilst testing the system a critical bug was found that would cause the software to delete the operating system. The bug has been hotfixes as it poses a serious security and stability risk to the system, once again this bug highlights the full importance of thorough testing.

An administrator could attempt to reinstall a server, which is no longer on the system. This bug stems from a lower level bug with the data access object, the object will assume that a query is successful even if no rows are returned and will populate the model with blank data. When the console then processes the queue it loads the model using the DAO, but the model did not actually exist therefore the queue continued to process with a blank model, as shown earlier the system cleans up the game servers directory before reinstalling. In this case the game servers director was blank and thus the entire root was deleted.

The other thing to note about this issue is that the commands were being run as the root user, has this not been the case the bug would not have caused so much damage to the operating system. For future revisions of the system this should be addressed and all commands should be run under the appropriate users to avoid this happening again.

7.3 Unit Testing

Regression testing is an important aspect of creating new features on the system where files from existing features are modified. There are unit-testing framework for most languages available, these include PHPUnit for PHP and QUnit for JavaScript. Unit tests were not used in this project but should be a future consideration as the project scales larger.

8 Evaluation

The following sections will discuss the project critically from all aspects aiming to highlight what went well but also comment on what could have gone better.

8.1 Aims

As stated in the introduction the aim of this project is to product a functional starting point for a game server control panel. The project quickly became larger than first imagined and before embarking the project the ambition had to be downsized with respect to time available to complete the project. That said with carful Analysis of the problem and user input the scope become well defined. It is safe to say that the initial aim of the project has been, the system functions mostly as expected without any major hiccups and has been well grounded as a foundation for expansion.

8.2 Objectives

The objects are perhaps the most important area for evaluations as it defines the learning outcomes and personal goals underlying the project, the objectives stated earlier in the project and the extent, which they were achieved, has been documented in the following sections.

8.2.1 To develop a deep understanding of Web Technologies

The project in many ways helped to develop a deep understanding of web technologies, en example of this is the PHP framework, which has been created for the system, has opened the gate for issues that are very deeply embedded into the web to be explored, and it looks closely at the model view controller pattern. Some of the challenges presented by producing the system, required solutions, which were outside the box, and required a deep understanding of the technologies.

8.2.2 To gain experience in the full software life cycle

The project has enabled the author to obtain experience with the full software lifecycle, this begins with the planning and analysis stage where stake holders were identified and requirements have been defined, information gathered in this stage was then used to break down the problem into a manageable design stage, in the design stage all areas of the system were carefully considered and UML models were used to present ideas. The design phase was then used as a reference for implementation, in which technology decisions were made and problems were tackled that had not been foreseen before. Finally the system was tested and deployed live, testing identified outstanding issues that are important to be aware of and show some of the flaws that may be present with the system.

8.2.3 To explore new, interesting and emerging web technologies

As mentioned earlier there were many obstacles to tackle in the system and the use of new technologies like HTML5 and CSS3 helped overcome these, in the implementation section the specific use of these technologies have been documented, one this that has not been discussed though is the limitation that this objective creates, the system may be cross-browser compatibility issues especially with older browsers and may need to have some redundancy build in to overcome this.

8.2.4 To explore ideas surrounding the problem space and providing adequate solutions

This objective encapsulates the previous three, all problems which have been documented in the implementation phase have been attempted to be solved, and however some solutions feel less elegant than other like the state synchronization via header which may point to an overall design issue. That said all problems have been solved to a certain degree and the final product functions well.

8.3 Methodology

The methodology chosen for this project was incremental build, and has been essential for the project's success. Through this model, project stages were refined and updated to reflect changes, the final product went through a series of user feedback to become what it is today, this would not have been possible using the waterfall model alone.

The stages of the waterfall model are still clearly present throughout the report, which include Analysis, Design, Implementation and Testing. There have been several occasions where users have influences changes in the system, such as extra requirements like the file browser. This is where incremental build allows for changes to be made.

8.4 Requirements

All essential and desirable functional requirements have been met to a lesser or greater degree, the system performs mostly well and as mostly expected however as pointed out in the testing phase there are some underlying issues that should be looked into.

R3 perhaps has many issues which were highlighted in the testing phase, for this module does not conform to the rest of the framework and this is why most of the issues with this happen, it also has a lack of validation thus the best option going forward with this module it to rewrite it and revisit the design for this.

R15 presented a major bug to the system, the bug was addressed but the underlying causes have not been. Ultimately the bug exists because the framework did not properly introduce the DAO pattern, and checks for empty results sets were not made. This issue would not have been a problem if using another framework as they

have been tested thoroughly, ultimately this was one of the risks that were taken when writing one from the ground up.

The non-functional requirements were also achieved to a similar degree, the system scales to mobile with a suitable UI using responsive design techniques mentioned in the implementation section, which satisfies NF3, there are still small issues with certain screen sizes that should be addressed in the future.

NF2 was also achieved successfully however did have some unforeseen effects that were documented in testing that would make caused some usability issues.

Overall the requirements were tackled to a good degree.

8.5 Existing Systems

In the background research section existing systems were reviewed and feature sets were listed. Looking back on this almost all existing systems are further developed with more features then the solution that has been made here. However it is important to note that these control panels have been in development for many years and had to start somewhere.

As a starting point the system lines up well into the market, it has potential driven by the focus on the client end user interface. The administration section however needs to be reviewed to be more scalable, things like pagination need to be added and a friendlier UI should be developed. The administration section also has bad screen scalability. More time spent on targeting the smaller screen sizes for a better mobile experience is needed. However, administrators did say earlier that mobile was not critical for them, rather for their clients.

The system uses a new technique for offloading work into the background, and is one of the only panels, which keep the user section synchronized with ajax.

8.6 Future Prospects

Many of the future requirements have been stated in sections 3.3.2. These form a basis to future extension.

After having reviewed the projects, the author was slightly dis-satisfied with the framework produced, it achieved the goal it set out to do which was to provide flexibility and give deeper insight into PHP frameworks, however much time was spent refining the framework rather then producing a better end product.

In hindsight, a proven framework should have been used, and more focus could have been spent on the problem space. Going forward the code could be refactored into other generally available frameworks such as CodeIngiter.

However before approaching new features, the existing flaws should be investigated and fixed at the root to stop un-welcome bugs appearing later in the development cycle.

8.7 Overall Report

The problem tackled was extremely large, and documentation may have failed to cover some aspects. The reports thoroughly Analysis the problem space, designs a solution and provides this solution which is tested thoroughly.

There was a lack of talk about the underlying SQL used in these projects, and this was simply due to time constraints. The other section that needed more elaboration was implementation of the administration section.

Overall the report does a good job at covering most of the topic initiated by the problem statement.

8.8 Conclusion

The project set out to create a functional game server control panel, to document its creation and to learn skills along the way. This is something that it has clearly achieved. The process did not come without its problems but the solution. Many objectives were also achieved, a deeper insight into web technologies was obtained and experience of the software lifecycle was gained.

9 References

Alam, S., (2012). Waterfall Model Advantages and Disadvantages. [online] Available from: <http://www.buzzle.com/articles/waterfall-model-advantages-and-disadvantages.html> [Accessed 21st Apr 2013].

Byers, J., (2013). Mobile Responsive Design 101. [online] Available from: <http://www.copyblogger.com/mobile-responsive-design-101/> [Accessed 24th Apr 2013].

Bewick, C., (2010). HTML5 Custom Data Attributes (data-*). [online] Available from: <http://html5doctor.com/html5-custom-data-attributes/> [Accessed 21st Apr 2013].

Cashmore, P., (2012). Mashable - Why 2013 Is the Year of Responsive Web Design. [online] Available from: < http://mashable.com/2012/12/11/responsive-web-design/> [Accessed 24th Apr 2013].

G, A., (2010). Spiral Model Advantages and Disadvantages. [online] Available from: <http://www.buzzle.com/articles/spiral-model-advantages-and-disadvantages.html> [Accessed 21st Apr 2013].

Monnox, A., (2005). Rapid J2EE Development: An Adaptive Foundation for Enterprise Applications. Prentice Hall.

NASA. (2004). NASA Software Safety Guidebook. [online] Available from: <http://www.hq.nasa.gov/office/codeq/doctree/871913.pdf>. pp.56-57. [Accessed 15th Apr 2013].

Pressman, R., (2009). Software Engineering: A Practitioner's Approach. McGraw-Hill Higher Edudation. pp. 41-42.

Recordon, D., (2010). Facebook - Using HTML5 Today. [online] Available from: <https://www.facebook.com/note.php?note_id=438532093919>. [Accessed 24th Apr 2013].

Rouse, M., (2007). Spiral Mode. [online] Available from: <http://searchsoftwarequality.techtarget.com/definition/spiral-model> [Accessed 22nd Apr 2013].

UXMovement. (2011). Why Rounded Corners are Easier on the Eyes. [online] Available from: <http://uxmovement.com/thinking/why-rounded-corners-are-easier-on-the-eyes/> [Accessed 21st Apr 2013].

Satalkar, B., (2011). Comparison Between Waterfall Model and Spiral Model. [online] Available from: <http://www.buzzle.com/articles/comparison-between-waterfall-model-and-spiral-model.html> [Accessed 21st Apr 2013].

Vo, T., (2007). Software development process. [online] Available from: <http://cnx.org/content/m14619/latest/>. [Accessed 22nd Apr 2013]

W3C. (1998). Cool URIs don’t change. [online] Available from: <http://www.w3.org/Provider/Style/URI> [Accessed 23rd Apr 2013].

W3C. (2012). HTML5. [online] Available from: <http://www.w3.org/TR/html5/browsers.html#history> [Accessed 23rd Apr 2013].

Wrensoft. (2012). Search Benchmarks. [online] Available from: <http://www.wrensoft.com/zoom/benchmarks.html> [Accessed 23rd Apr 2013].

10 Appendices

10.1 Interviews

Administrator 1

1. Are you already running a game server management control panel for your business?
   Yes I am running tc admin.

2. Which feature would you say is essential and [control panel here]'s biggest selling point?
   I love the way it can install mods for game servers however the most important thing is that my clients can stop/start/restart server.

3. If you had something bad to say about [control panel here] what would it be?
   the user interface is cluttered.

4. Is there anything you wish [control panel here] could do, that it currently cannot?
   not really off the top of my head.

5. Have you ever managed a set of game servers without a control panel (eg. Command line)?
   yes.

6. What was the most repetitive task?
   Installing the servers from scratch.

7. Was it ever difficult to remember any of the steps to installing a game server?
   yes I always had to use websites like SRCDS.

8. Did you ever lose track of anything in particular (eg. server passwords)?
   generally everything from IP's, to passwords and even clients.

Administrator 2

1. Are you already running a game server management control panel for your business?
   I currently use GameCP .

2. Which feature would you say is essential and [control panel here]'s biggest selling point?
   Essential is just the ability to create new servers easily and then control these on the web.

3. If you had something bad to say about [control panel here] what would it be?
   Its very buggy and looks horrible.

4. Is there anything you wish [control panel here] could do, that it currently cannot?
   Mobile support would be great for the clients.

5. Have you ever managed a set of game servers without a control panel (eg. Command line)?
   no, I have always managed this via the control panel that was setup at the company.

Client 1

1. Would you consider purchasing a game server that did not offer a control panel?
   Not really, I just want the server to work with minimal effort .

2. How would you describe your understanding of FTP and how to use it?
   um im not really sure what ftp is.

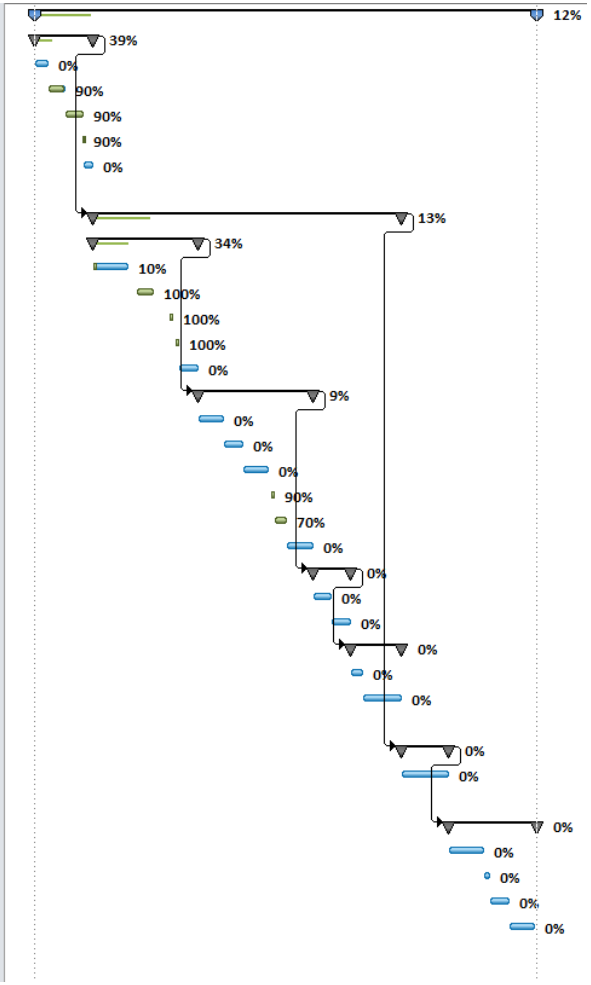
3. Do you usually install game mods on your server, and if so how often?
   yes I did, but this could be done through the control panel.

4. What is the most common setting that you need to change on a particular game server, and how is this setting changed?
   the main server settings such as the server's name and message of the day ect..

5. What is the bare minimum control that you need over a game server?
   I need to be able to stop and start the server, and I also need to be able to change the settings.

10.2 Provision Project Plan

0			GameServerControlPanel	12%	39.38 days	Tue 02/10/12	Wed 19/12/12	
1			**Preliminary Plan**	**39%**	**4.88 days**	**Tue 02/10/12**	**Wed 10/10/12**	
2			Plan and write up Requirements	0%	6 hrs	Tue 02/10/12	Wed 03/10/12	Diogo Moura
3			Create Ghantt Chart	90%	4 hrs	Wed 03/10/12	Sat 06/10/12	Diogo Moura
4			Create ERD	90%	4 hrs	Sat 06/10/12	Tue 09/10/12	Diogo Moura
5			Setup Version Control	90%	0.5 hrs	Tue 09/10/12	Tue 09/10/12	Diogo Moura
6			Low Fidelity Prototype for Views	0%	5 hrs	Tue 09/10/12	Wed 10/10/12	Diogo Moura
7								
8			**Back End Implementiation**	**13%**	**22 days**	**Wed 10/10/12**	**Wed 28/11/12**	**1**
9			**Core Framework**	**34%**	**7.25 days**	**Wed 10/10/12**	**Sat 27/10/12**	
10			SSH Comminucation Class	10%	8 hrs	Wed 10/10/12	Tue 16/10/12	Diogo Moura
11	✓		View Rendering Class	100%	3 hrs	Thu 18/10/12	Sat 20/10/12	Diogo Moura
12	✓		Class Autoloader	100%	2 hrs	Tue 23/10/12	Tue 23/10/12	Diogo Moura
13	✓		Configuration Class	100%	1 hr	Wed 24/10/12	Wed 24/10/12	Diogo Moura
14			MySQL Wrapper	0%	6 hrs	Wed 24/10/12	Sat 27/10/12	Diogo Moura
15			**Models**	**9%**	**8.75 days**	**Sat 27/10/12**	**Wed 14/11/12**	**9**
16			User Model	0%	8 hrs	Sat 27/10/12	Wed 31/10/12	Diogo Moura
17			Dedicated Server Model	0%	8 hrs	Wed 31/10/12	Sat 03/11/12	Diogo Moura
18			Game Server Model	0%	6 hrs	Sat 03/11/12	Wed 07/11/12	Diogo Moura
19			Permission Flags Model	90%	2 hrs	Wed 07/11/12	Thu 08/11/12	Diogo Moura
20			Permissions Model	70%	2 hrs	Thu 08/11/12	Sat 10/11/12	Diogo Moura
21			Game Server Template Model	0%	8 hrs	Sat 10/11/12	Wed 14/11/12	Diogo Moura
22			**Data Access Objects**	**0%**	**2 days**	**Wed 14/11/12**	**Tue 20/11/12**	**15**
23			Game Server DAO	0%	4 hrs	Wed 14/11/12	Sat 17/11/12	Diogo Moura
24			User Data DAO	0%	4 hrs	Sat 17/11/12	Tue 20/11/12	Diogo Moura
25			**Background Worker**	**0%**	**4 days**	**Tue 20/11/12**	**Wed 28/11/12**	**22**
26			Server Side Queueing Handler	0%	8 hrs	Tue 20/11/12	Thu 22/11/12	Diogo Moura
27			Internal Queue API Class	0%	8 hrs	Thu 22/11/12	Wed 28/11/12	Diogo Moura
28								
29			**Developer REST Api Implementation**	**0%**	**4 days**	**Wed 28/11/12**	**Wed 05/12/12**	**8**
30			full implementation of rest api	0%	16 hrs	Wed 28/11/12	Wed 05/12/12	Diogo Moura
31								
32			**Front End Implementation(+Controllers)**	**0%**	**8.5 days**	**Wed 05/12/12**	**Wed 19/12/12**	**29**
33			Global Header & Footer	0%	8 hrs	Wed 05/12/12	Tue 11/12/12	Diogo Moura
34			Login Page	0%	2 hrs	Tue 11/12/12	Wed 12/12/12	Diogo Moura
35			User Dashboard	0%	12 hrs	Wed 12/12/12	Sat 15/12/12	Diogo Moura
36			Administrator Dashboard	0%	8 hrs	Sat 15/12/12	Wed 19/12/12	Diogo Moura
37			Game Server File Manager	0%	4 hrs	Wed 19/12/12	Wed 19/12/12	Diogo Moura



10.3 Use Case Descriptions

10.3.1 Create Game Server

Description

This use case describes how the administrator requests a game server to be installed.

Actors

- Dedicated Server
- Administrator

Pre-conditions

1. The system must be operational.
2. There must be a dedicated server online with free space and the game files installed.

Trigger

The administrator has requested to create a game server.

Basic Flow

1. The system shall present dedicated servers.
2. The system shall present a list of games.
3. The system shall present a list of clients.
4. The administrator shall pick a dedicated server from the list.
5. The system shall provide a default IP for the server.
6. The administrator shall select a game from the list.
7. The system will populate the game default fields.
8. The administrator will select a client from the list.
9. The administrator will submit the request.
10. The system will dele gate the instructions to the dedicated server.
11. The ftp user will be created.
12. The details are added to the database.
13. The use case ends successfully.

Exceptions

Server could not be installed step 9.

1. Alert the administrator that the server could not be installed.
2. Go back to step 1.

Post-Conditions

Successful

The administrator has successfully queued a game server installation. The queue will be processed and a confirmation will be given once the process is complete.

10.3.2 Delete Game Server

Description

This use case describes how the administrator requests a game server to be removed from the system.

Actors

- Dedicated Server
- Administrator

Pre-conditions

1. The system must be operational.
2. There must be a game server installed and on the system.

Trigger

The administrator has requested to delete a game server.

Basic Flow

1. The system will ask for confirmation of the request.
2. The system will delegate the instructions to the dedicated server.
3. The details are removed into the database.
4. The use case ends successfully.

Alternative Flow

Server is online at step 2.

1. Stop the server (Use Case: Stop Game Server)
2. Return to step 2.

Exceptions

Administrator rejects confirmation step 1.

1. The administrator rejects the conformation.
2. Use case ends unsuccessfully.

Post-Conditions

Successful

The administrator has successfully queued a game server to be deleted. The queue will be processed and a confirmation will be given once the process is complete.

10.3.3 Update Game Server

Description

This use case describes how the administrator can update the details for a game server.

Actors

- Dedicated Server
- Administrator

Pre-conditions

1. The system must be operational.
2. There must be a game server installed and on the system.

Trigger

The administrator has requested to modify a game server.

Basic Flow

1. The system will present the current settings.
2. The administrator will modify the required settings.
3. The system will save these changes to the database
4. The use case ends successfully.

Alternative Flow

Administrator specifies a new FTP password at step 2.

1. The system will delegate the password update request to the dedicated server.
2. Return to step 2.

Post-Conditions

Successful

The administrator has successfully updated the settings for a game server and the changes have been saved.

10.3.4 Browse Game Servers

Description

This use case describes how the user can browse game servers they own OR the administrators can browse game servers on the system.

Actors

- Dedicated Server
- Administrator
- User

Pre-conditions

1. The system must be operational.
2. There must be a game server installed and on the system.

Trigger

The user wants to view a game server list OR the administrator wants to manage the servers on the system.

Basic Flow

1. The system will display a list of appropriate game servers.
2. The use case ends successfully.

Post-Conditions

Successful

The system has displayed a list of game servers to the user or administrator.

10.3.5 Stop Game Server

Description

This use case describes how a user or administrator is able to request for a server to be stopped.

Actors

- Dedicated Server
- Administrator
- User

Pre-conditions

1. The system must be operational.
2. There must be a game server installed and on the system.
3. The game server selected must be online.

Trigger

The administrator or user has pressed a button to stop the game server.

Basic Flow

1. The user or administrator presses the stop button for the server.
2. The system sets the game server status to pending.
3. The system asks the dedicated server to stop the game server.
4. The game server stops.
5. The dedicated server triggers Use case Update Server Status.
6. The use case ends successfully.

Exceptions

The server does not stop at step 4.

1. The system will notice the failure after a timeout and update the status.
2. Use case ends unsuccessfully.

Post-Conditions

Successful

The administrator or a user has successfully stopped a game server.

10.3.6 Start Game Server

Description

This use case describes how a user or administrator is able to request for a server to be started.

Actors

- Dedicated Server
- Administrator
- User

Pre-conditions

1. The system must be operational.
2. There must be a game server installed and on the system.
3. The game server selected must be offline.

Trigger

The administrator or user has pressed a button to start the game server.

Basic Flow

1. The user or administrator presses the start button for the game server.
2. The system sets the game server status to pending.
3. The system asks the dedicated server to start the game server.
4. The game server start.
5. The dedicated server triggers Use case Update Server Status.
6. The use case ends successfully.

Exceptions

The server does not stop at step 4.

1. The system will notice the failure after a timeout and update the status.
2. Use case ends unsuccessfully.

Post-Conditions

Successful

The administrator or a user has successfully started a game server.

10.3.7 Restart Game Server

Description

This use case describes how a user or administrator is able to request for a server to be restarted.

Actors

- Dedicated Server
- Administrator
- User

Pre-conditions

1. The system must be operational.
2. There must be a game server installed and on the system.
3. The game server selected must be online.

Trigger

The administrator or user has pressed a button to start the game server.

Basic Flow

1. The user or administrator presses the re start button for the game server.
2. The system sets the game server status to pending.
3. Use case Stop Game Server is invoked.
4. Use case Start Game Server is invoked.
5. The game server status is updated.
6. The use case ends successfully.

Post-Conditions

Successful

The administrator or a user has successfully re-started a game server.

10.3.8 Reinstall Game Server

Description

This use case describes how a user or administrator is able to request for a server to be restarted.

Actors

- Dedicated Server
- Administrator
- User

Pre-conditions

1. The system must be operational.
2. There must be a game server installed and on the system.
3. The game server selected must be either online or offline.

Trigger

The administrator or user has pressed a button to reinstall the game server.

Basic Flow

1. The user or administrator presses the reinstall button for the game server.
2. The system sets the game server status to pending.
3. Use case Stop Game Server is invoked.
4. The system delegates the commands to the dedicated server.
5. The dedicated server deletes the current files for the game server.
6. The dedicated server installs the game server.
7. The game server status is updated.
8. The use case ends successfully.

Exceptions

Failure at step 5 or 6.

1. Alert the administrator of the problem.
2. Return to step 1.

Post-Conditions

Successful

The administrator or a user has successfully re-installed a game server.

10.3.9 Update Server Status

Description

This use case describes how the dedicated server updates the status on the system.

Actors

- Dedicated Server

Pre-conditions

1. The system must be operational.

Trigger

An operation such as stop, start or restart has been performed successfully on the dedicated server OR a game server has finished installing.

Basic Flow

1. The server relays feedback to the system that an event has happened.
2. The system updates the status in the database.
3. The system will notify the user of this change.
4. The use case ends successfully.

Post-Conditions

Successful

The system has notified the user of the change.

10.3.10 Add User

Description

This use case describes how the administrator is able to add a new user to the system.

Actors

- Administrator

Pre-conditions

1. The system must be operational.

Trigger

The administrator has requested to create a user.

Basic Flow

1. The system will provide a form with the required fields, such as username password and first/last names.
2. The administrator submits the form.
3. The system inserts the details into the database.
4. The use case ends successfully.

Exceptions

Username is taken at step 3.

1. The system alerts the administrator of the problem.
2. Return to step 1.

Post-Conditions

Successful

The system has added the new user to the database.

10.3.11 Modify User

Description

This use case describes how the administrator is able to modify a user on the system.

Actors

- Administrator

Pre-conditions

1. The system must be operational.
2. There must be at least one user on the system.

Trigger

The administrator has requested to modify a user.

Basic Flow

1. The system displays a form with the details required such as username, password, and email, first and last name prefilled with the current details.
2. The administrator makes changes and submits the form.
3. The system saves these changes to the database.
4. The use case ends successfully.

Exceptions

Username is taken at step 2.

1. The system alerts the administrator of the problem.
2. Return to step 1.

Post-Conditions

Successful

The system has updated the user's details.

10.3.12 Delete User

Description

This use case describes how the administrator is able to delete a user from the system.

Actors

- Administrator

Pre-conditions

1. The system must be operational.
2. There must be at least one user on the system.

Trigger

The administrator has requested to delete a user.

Basic Flow

1. The administrator requests for the user to be deleted
2. The system removes the user from the database
3. The use case ends successfully.

Post-Conditions

Successful

The system has removed the user's details from the database.

10.3.13 Add Game

Description

This use case describes how the administrator is able to add a new game to the system.

Actors

- Administrator

Pre-conditions

1. The system must be operational.

Trigger

The administrator has requested to add a new game.

Basic Flow

1. The system will provide a form with the required fields, such as game name and required files.
2. The administrator submits the form.
3. The system inserts the details into the database.
4. The use case ends successfully.

Post-Conditions

Successful

The system has added the new game to the database.

10.3.14 Modify Game

Description

This use case describes how the administrator is able to modify a game on the system.

Actors

- Administrator

Pre-conditions

1. The system must be operational.
2. There must be at least one game on the system.

Trigger

The administrator has requested to modify a game.

Basic Flow

1. The system displays a form with the details required such as game name and required files prefilled with the existing details.
2. The administrator makes changes and submits the form.
3. The system saves these changes to the database.
4. The use case ends successfully.

Post-Conditions

Successful

The system has updated the games details.

10.3.15 Delete Game

Description

This use case describes how the administrator is able to delete a game from the system.

Actors

- Administrator

Pre-conditions

1. The system must be operational.
2. There must be at least one game on the system.

Trigger

The administrator has requested to delete a game.

Basic Flow

1. The administrator requests for the user to be deleted.
2. The system removes the game from the database.
3. The use case ends successfully.

Post-Conditions

Successful

The system has removed the game from the database.

10.3.16 Add Dedicated Server

Description

This use case describes how the administrator is able to add a new dedicated server to the system.

Actors

- Administrator

Pre-conditions

1. The system must be operational.

Trigger

The administrator has requested to add a dedicated server.

Basic Flow

1. The system will provide a form with the required fields, such as IP Address.
2. The administrator submits the form.
3. The system inserts the details into the database.
4. The use case ends successfully.

Post-Conditions

Successful

The system has added the new dedicated server to the database.

10.3.17 Modify Dedicated Server

Description

This use case describes how the administrator is able to modify a dedicated server on the system.

Actors

- Administrator

Pre-conditions

1. The system must be operational.
2. There must be at least one dedicated server on the system.

Trigger

The administrator has requested to modify a dedicated server.

Basic Flow

1. The system displays a form with the details required such as IP Address for the dedicated server prefilled with the existing details.
2. The administrator makes changes and submits the form.
3. The system saves these changes to the database.
4. The use case ends successfully.

Post-Conditions

Successful

The system has updated the dedicated server details.

10.3.18 Delete Dedicated

Description

This use case describes how the administrator is able to delete a dedicated server from the system.

Actors

- Administrator

Pre-conditions

1. The system must be operational.
2. There must be at least one dedicated server on the system.

Trigger

The administrator has requested to delete a dedicated server.

Basic Flow

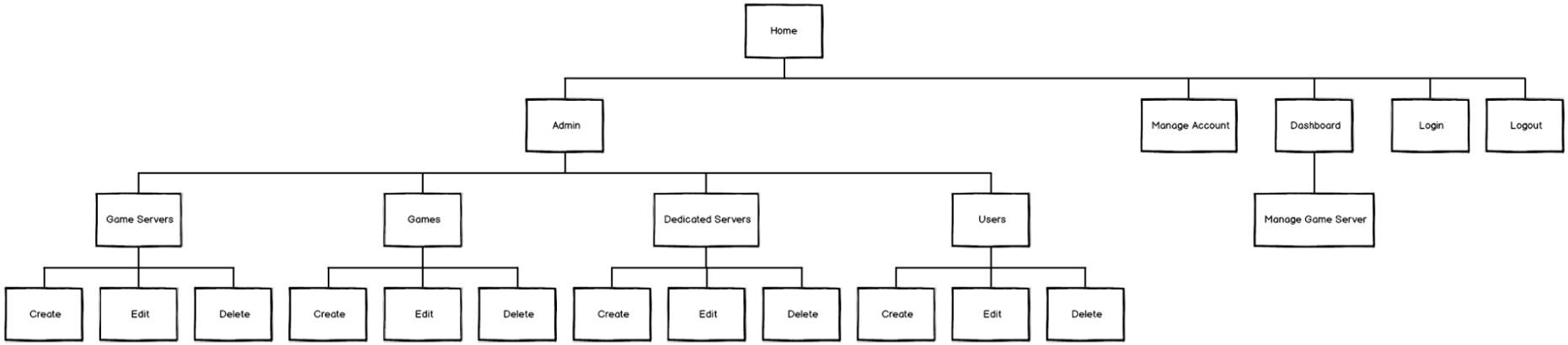
1. The administrator requests for the dedicated server to be deleted.
2. The system removes the game from the database.
3. The use case ends successfully.

Post-Conditions

Successful

The system has removed the dedicated server from the database.

10.4 Sitemap



10.5 System Testing

#	REQ	Test	Expected Steps/Results	Pass	Actual Results/Notes
1	***R1. Clients must be able to stop/start/restart game servers they own***			✔	
2		**Start Successful**		✔	
3			Pressing start causes ajax request to server.	✔	
4			GameServerController adds start command to queue	✔	
5			Game Server Status is changed to pending	✔	
6			Stop/Start/Restart buttons are hidden in the UI.	✔	
7			The queue is processed via the cron job	✔	
8			The screen is created on the server and has a process inside	✔	
9			The status is updated to online on the website	✔	
10			A notification of the server coming online is shown	✔	

11	The server has started with the correct ip/port	✔	
12	Restart / Stop buttons are shown in the UI	✔	
13	**Stop Successful**	✔	
14	Pressing stop causes ajax request to server.	✔	
15	GameServerController adds stop command to queue	✔	
16	Game Server Status is changed to pending	✔	
17	The queue is processed via the cron job	✔	
18	The screen(s) is/are terminated on the server	✔	
19	A notification of the server going offline is shown	✔	
20	The status is updated to offline on the website	✔	
21	start button is shown in the UI	✔	
22	**Restart Successful**	✔	
23	Pressing restart causes ajax request to server.	✔	
24	GameServerController adds restart command to queue	✔	
25	Game Server Status is changed to pending	✔	
26	The queue is processed via the cron job	✔	
27	The screen(s) is/are terminated on the server	✔	
28	The screen is created on the server and has a process inside	✔	
29	The status is updated to online on the website	✔	
30	A notification of the server coming online is shown	✔	
31	The server has started with the correct ip/port	✔	
32	Restart / Stop buttons are shown in the UI	✔	
33	**Start Unsucessfull - Insufficient Permissions**	✔	
34	System detects user does not have permissions, shows warning.	✔	
35	**Stop Unsucessfull - Insufficient Permissions**	✔	
36	System detects user does not have permissions, shows warning.	✔	
37	**Restart Unsucessfull - Insufficient Permissions**	✔	

38	System detects user does not have permissions, shows warning.	✔	
39	**Start Unsucessfull - Not Logged In**	✔	UI shows server to pending. Does not actually happen
40	System detects user is not logged in, shows warning.	✔	
41	**Stop Unsucessfull - Not Logged In**	✔	UI shows server to pending. Does not actually happen
42	System detects user is not logged in, shows warning.	✔	
43	**Restart Unsucessfull - Not Logged In**	✔	UI shows server to pending. Does not actually happen
44	System detects user is not logged in, shows warning.	✔	
45	**Start Unsucessfull - Pending/Online**	✔	
46	System detects user is not logged in, shows warning.	✔	
47	**Stop Unsucessfull - Pending/Offline**	✔	
48	System detects user is not logged in, shows warning.	✔	
49	**Restart Unsucessfull - Pending/Ofline**	✔	
50	System detects user is not logged in, shows warning.	✔	
51	***R2. Clients may be able to reinstall a game server they own***	✔	
52	**Reinstall Successful**	✔	
53	Pressing restart causes ajax request to server.	✔	
54	Restart request is added to the queue	✔	
55	Server status is set to pending.	✔	
56	Stop, Start, Restart buttons are hidden from the user.	✔	
57	Queue is processed.	✔	
58	Server status set to installing.	✔	
59	User files are deleted.	✔	
60	New game files are unzipped into user directory.	✔	
61	File permissions are set correctly.	✔	
62	Server status is set to offline	✔	
63	Start button is shown to the user.	✔	
64	**Reinstall Unsuccessfull - Does Not Have Permissions**	✔	

65	System detects user does not have permissions, shows warning.	✔	
66	**Reinstall Unsuccessfull - Not Logged In**	✔	UI shows server to pending. Does not actually happen
67	System detects user is not logged in, shows warning.	✔	
68	***R3. Clients shall be provided with a baisc file manager…***	✔	
69	**List Directory Sucessful**	✔	
70	Correct list of files/folders displayed.	✔	
71	**Open Config File Sucessful**	✔	
72	File contents sucessfully displayed to client.	✔	
73	**Save Changed To Config File Sucessful**	✔	
74	File contents sucessfully saved.	✔	
75	**List Directory Unsucessful - Invalid Directory Dosn't Exist**	✔	
76	Error is shown.	✔	Error not user friendly.
77	**Open File Unsucessful - Invalid File Doesn’t Exist**	✔	
78	Error is shown.	✔	Error not user friendly.
79	**Save Changes Unsucessful - New File**	✘	New File is Created
80	Error is shown.	✘	New File is Created
81	**Open File Unsucessful - Invalid File Type (binnary)**	✘	Garabge shown in text box
82	Error is shown.	✘	
83	**Browse Directory Unsucessul - Invalid Permissions**	✔	Error not user friendly.
84	System detects user does not have permissions, shows warning.	✔	
85	**Open File Unsucessful - Invalid Permissions**	✔	Error not user friendly.
86	System detects user is not logged in, shows warning.	✔	
87	**Browse Directory Unsucessul - Not Logged In**	✘	Login Page Overlays Server Details
88	System detects user is not logged in, shows warning.	✘	
89	**Open File Unsucessul - Not Logged In**	✘	Login Page Overlays Server Details
90	System detects user is not logged in, shows warning.	✘	
91	***R4. Clients shall be able to view a list of servers they own***	✔	

92	**Game Server Listing Successful**	✔	
93	List of game servers complete	✔	
94	List of servers accurate with status'	✔	
95	**Game Server Listing Unsuccessful - Not Logged In**	✔	
96	User is redirected to login page	✔	
97	***R5. Clients shall be able to view extended information ……***	✔	
98	**Extended infromation view Sucessful**	✔	
99	Create game type present	✔	
100	Correct ip/port present	✔	
101	Correct number of slots present	✔	
102	Correct FTP ip present	✔	Note: ftp ip based on dedicated server main ip.
103	Correct Ftp user present	✔	
104	Correct FTP Password present	✔	
105	File browser present with correct details.	✔	
106	**Extended infromation view Unsucessfull - Invalid Permissions**	✘	
107	Show warning to user	✘	program crahses due to missing method.
108	**Extended infromation view Unsucessfull - Invalid Game Server**	✘	
109	Show warning to user	✘	view displays with no data
106	**Extended infromation view Unsucessfull - Not Logged In**	✔	
107	Redirect to Login Page	✔	
112	***R6. Clients must be able to update their password***	✔	
113	**Password Change Successful**	✔	
114	password with correct minimum length & confirmation	✔	
115	**Password Change Unsuccessful - Invalid user input**	✔	
116	password w/o confirmation - invalid password warning	✔	
117	password not matching confirmation - invalid password warning	✔	
118	confirmation w/o password - invalid password warning	✘	password not update, however invalid input not

			detected
119	password too sort - invalid password warning.	✔	
120	blank password - password not updated.	✔	
121	***R7. Administrators shall be able to stop/start/restart game servers…***	✔	
122	**Start Successful**	✔	
123	GameServerController adds start command to queue	✔	
124	Game Server Status is changed to pending	✔	
125	Stop/Start/Restart buttons are hidden in the UI.	✔	
126	The server process is started.	✔	
127	The status is updated to online on the website	✔	
128	A notification of the server coming online is shown	✔	
129	The server has started with the correct ip/port	✔	
130	Restart / Stop buttons are shown in the UI	✔	note: requires refresh if admin does not own server
131	**Stop Successful**	✔	
132	GameServerController adds stop command to queue	✔	
133	Game Server Status is changed to pending	✔	
134	The screen(s) is/are terminated on the server	✔	
135	A notification of the server going offline is shown	✔	
136	The status is updated to offline on the website	✔	note: requires refresh if admin does not own server
137	start button is shown in the UI	✔	note: requires refresh if admin does not own server
138	**Restart Successful**	✔	
139	GameServerController adds restart command to queue	✔	
140	Game Server Status is changed to pending	✔	
141	The screen(s) is/are terminated on the server	✔	
142	The screen is created on the server and has a process inside	✔	
143	The status is updated to online on the website	✔	
144	A notification of the server coming online is shown	✔	

145	The server has started with the correct ip/port	✔	note: requires refresh if admin does not own server
146	Restart / Stop buttons are shown in the UI	✔	note: requires refresh if admin does not own server
147	**Start Unsucessful - Invalid Game Server ID**	✔	
148	Invalid server popup shown	_	invalid permissions popup shown instead.
149	**Stop Unsucessful - Invalid Game Server ID**	✔	
150	Invalid server popup shown	_	invalid permissions popup shown instead.
151	**Restart Unsucessful - Invalid Game Server ID**	✔	
152	Invalid server popup shown	_	invalid permissions popup shown instead.
153	***R8. Administrators shall be able to manage users….***	✔	
154	**Create/Edit New User Sucessful**	✔	
155	Valid unique username	✔	
156	Valid email	✔	
157	Valid Password	✔	
158	Valid First Name	✔	
159	Valid Last Name	✔	
160	**List users successful**	✔	
161	users of the system are accuratly listed.	✔	
162	**Delete user successful**	✔	
163	user is removed from the system.	✔	
164	**Create/Edit New User Unsucessful - Invalid Data**	✔	
165	Duplicate Username - Show Error	✔	
166	Invalid Email - Show Error	✔	
167	Invalid First Name - Show Error	✔	
168	Invalid Last Name - Show Error	✔	
169	Short password - Show Error	✔	
170	Fields missing - Show Error	✔	
171	**Delete User Unsucessful - invalid id**	✔	

172	Show error / don't delete	✔	note: no error shown but no user deleted.
173	**Edit User Unsucessful - Invalid id**	✔	
174	Show error / Redirect to new user	✔	
175	**List users Unsuccessful - not logged in**	✔	
176	redirect user to login	✔	
177	**Edit Users Unsuccessful - not logged in**	✔	
178	redirect user to login	✔	
179	**Create Users Unsuccessful - not logged in**	✔	
180	redirect user to login	✔	
181	**Delete Users Unsuccessful - not logged in**	✔	
182	redirect user to login	✔	
177	**Edit Users Unsuccessful - Not an admin**	✔	
178	Redirect to dashboard	✔	
177	**Create Users Unsuccessful - Not an admin**	✔	
178	Redirect to dashboard	✔	
177	**Delete Users Unsuccessful - Not an admin**	✔	
178	Redirect to dashboard	✔	
177	**List Users Unsuccessful - Not an admin**	✔	
178	Redirect to dashboard	✔	
191	***R9. Administrators shall be able to manage game servers on the system***	✔	
192	**Create/Edit New Game Server Sucessful**	✔	
193	Valid nickname	✔	
194	Valid deidcated server chosen	✔	
195	Valid Template Chosen	✔	
196	Valid Owner Chosen	✔	
197	Valid IP provided	✔	
198	Valid Port Provided	✔	

199	Valid slot count provided	✔	
200	Gameserver is created with pending status.	✔	
201	Queue is processed.	✔	
202	Game Server set with installing status.	✔	
203	new user created on the dedicated sever.	✔	
204	FTP password correctly generated and set.	✔	
205	game extracted and installed.	✔	
206	Server status is updated to offline.	✔	
207	**List Game Servers successful**	✔	
208	Game servers are accuratly listed on the system.	✔	
209	**Delete Game Server Successful**	✔	
210	Gameserver status is set to pending.	✔	
211	Instances of the gameserver are stopped.	✔	
212	Game Server directory is removed.	✔	
213	FTP User is deleted from the Dedicated server.	✔	
214	Game Server is deleted from the system.	✔	
215	**Create/Edit New Game Server Unsucessful - Invalid Data**	✔	
216	Nickname blank - Show Error	✔	
217	Port Not Valid - Show Error	✔	
218	Ip Not a valid ip addresss - Show Error	✔	
219	Invalid template specified - Show Error	✔	
220	Invalid deciated server specified - Show Error	✔	
221	Invalid owner specified - Show Error	✔	
222	Invalid player slots specified - Show Error	✔	
223	Duplicate IP/Port Combo detected.	✔	
224	**Delete Game Server Unsucessful - invalid id**	✔	
225	Show error / don’t delete	✔	redirected to listing, server not deleted

226	**Edit Game Server Unsucessful - Invalid id**	✔	
227	Show error / Redirect to new Game server	✔	
228	**List Game Server Unsuccessful - not logged in**	✔	
229	redirect user to login	✔	
230	**Edit Game Server Unsuccessful - not logged in**	✔	
231	redirect user to login	✔	
232	**Create Game Server Unsuccessful - not logged in**	✔	
233	redirect user to login	✔	
234	**Delete Game Server Unsuccessful - not logged in**	✔	
235	redirect user to login	✔	
236	**Edit Game Server Unsuccessful - Not an admin**	✔	
237	Redirect to dashboard	✔	
238	**Create Game Server Unsuccessful - Not an admin**	✔	
239	Redirect to dashboard	✔	
240	**Delete Game Server Unsuccessful - Not an admin**	✔	
241	Redirect to dashboard	✔	
242	**List Game Server Unsuccessful - Not an admin**	✔	
243	Redirect to dashboard	✔	
	R10. Administrators shall be able to manage Dedicated Servers…	✔	
245	**Create/Edit New Deidcated Server Sucessful**	✔	
246	Valid unique Main IP	✔	
247	Valid Nickname	✔	
248	Valid SSH IP	✔	
249	Valid Port	✔	
250	Valid SSH Username	✔	
251	Valid SSH Password	✔	
252	Valid Game Files Directory	✔	

253	**List Dedicated Servers successful**	✔	
254	Dedicated Servers on the system are accuratly listed.	✔	
255	**Delete Dedicated Servers successful**	✔	
256	Dedicated Server is removed from the system.	✔	
257	**Create/Edit New Dedicated Servers Unsucessful - Invalid Data**	✔	
258	Duplicate Main IP - Show Error	✔	
259	Invalid IP - Show Error	✔	
260	Invalid Port - Show Error	✔	
261	**Delete Dedicated Server Unsucessful - invalid id**	✔	
262	Show error / don't delete	✔	note: no error shown but no dedicated server deleted.
263	**Edit Dedicated Server Unsucessful - Invalid id**	✔	
264	Show error / Redirect to new user	✔	
265	**List Dedicated Server Unsuccessful - not logged in**	✔	
266	redirect user to login	✔	
267	**Edit Decicated Server Unsuccessful - not logged in**	✔	
268	redirect user to login	✔	
269	**Create Dedicated Server Unsuccessful - not logged in**	✔	
270	redirect user to login	✔	
271	**Delete Dedicated Server Unsuccessful - not logged in**	✔	
272	redirect user to login	✔	
273	**Edit Dedicated Server Unsuccessful - Not an admin**	✔	
274	Redirect to dashboard	✔	
275	**Create Dedicated Server Unsuccessful - Not an admin**	✔	
276	Redirect to dashboard	✔	
277	**Delete Dedicated Server Unsuccessful - Not an admin**	✔	
278	Redirect to dashboard	✔	
279	**List Dedicated Server Unsuccessful - Not an admin**	✔	

280	Redirect to dashboard	✔	
281	***R11. Administrators shall be able to manage Game Templates.***	✔	
282	**Create/Edit New Game Template Sucessful**	✔	
283	Valid Name	✔	
284	Valid Min Slots	✔	
285	Valid Max Slots	✔	
286	Valid Default Slots	✔	
287	Valid Default Port	✔	
288	Valid Default CmdLine	✔	
289	Valid Game File	✔	
290	**List Game Templates successful**	✔	
291	Game Templates on the system are accuratly listed.	✔	
292	**Delete Game Template successful**	✔	
293	Game Template is removed from the system.	✔	
294	**Create/Edit New Game Template Unsucessful - Invalid Data**	✔	
295	Minimum Slots > Maxmim Slots - show error	✔	
296	Default slots < Min Slots - show error	✔	
297	Default slots > Max Slots - show error	✔	
298	Min Slots not integer - Show Error	✔	
299	Max Slots not Integer - Show Error	✔	
300	Default Slots Not Integer - Show Error	✔	
301	Name is blank - Show Error	✔	
302	**Delete Game Template Unsucessful - invalid id**	✔	
303	Show error / don’t delete	✔	note: no error shown but no game template deleted.
304	**Edit Game Template Unsucessful - Invalid id**	✔	
305	Show error / Redirect to new user	✔	
306	**List Game Template Unsuccessful - not logged in**	✔	

307	redirect user to login	✔	
308	**Edit Game Template Unsuccessful - not logged in**	✔	
309	redirect user to login	✔	
310	**Create Game Template Unsuccessful - not logged in**	✔	
311	redirect user to login	✔	
312	**Delete Game Template Unsuccessful - not logged in**	✔	
313	redirect user to login	✔	
314	**Edit Game Template Unsuccessful - Not an admin**	✔	
315	Redirect to dashboard	✔	
316	**Create Game Template Unsuccessful - Not an admin**	✔	
317	Redirect to dashboard	✔	
318	**Delete Game Template Unsuccessful - Not an admin**	✔	
319	Redirect to dashboard	✔	
320	**List Game Template Unsuccessful - Not an admin**	✔	
321	Redirect to dashboard	✔	
322	***R12. System shall notify clients when their server status changes.***	✔	
323	**Server Offline Notification**	✔	
324	**Server Online Notification**	✔	
325	**Server Restarting Notification**	✔	
326	**Server Installing Notification**	✔	
327	***R13. system will automatically generate ftp credentials***	✔	
328	**Create Server Test**	✔	
322	***R14. Administrators may be able to update ftp credentials for the server***	✔	
323	**Update Password Success**	✔	Note: Ftp Browser is bugged untill queue is processed
331	***R15. Administrators shall be able to reinstall game servers***	✔	
332	**Reinstall Successful**	✔	
333	Pressing restart causes ajax request to server.	✔	

334	Restart request is added to the queue	✔	
335	Server status is set to pending.	✔	
336	Stop, Start, Restart buttons are hidden from the user.	✔	
337	Queue is processed.	✔	
338	Server status set to installing.	✔	
339	User files are deleted.	✔	
340	New game files are unzipped into user directory.	✔	
341	File permissions are set correctly.	✔	
342	Server status is set to offline	✔	note: requires refresh if admin does not own server
343	Start button is shown to the user.	✔	note: requires refresh if admin does not own server
344	**Reinstall Unsuccessful - invalid id**	✘	
345	Error is shown to the administrator.	✘	***CRITICAL BUG: DELETES OPERATING SYSTEM. [FIXED].***
346	***R16. Administrators shall be able to browse files on clients server.***	✔	
347	**List Directory Sucessful**	✔	
348	Correct list of files/folders displayed.	✔	
349	**Open Config File Sucessful**	✔	
350	File contents sucessfully displayed to client.	✔	
351	**Save Changed To Config File Sucessful**	✔	
352	File contents sucessfully saved.	✔	
353	**List Directory Unsucessful - Invalid Directory Dosn't Exist**	✔	
354	Error is shown.	✔	Error not user friendly.
355	**Open File Unsucessful - Invalid File Doesn’t Exist**	✔	
356	Error is shown.	✔	Error not user friendly.
357	**Save Changes Unsucessful - New File**	✘	New File is Created
358	Error is shown.	✘	New File is Created