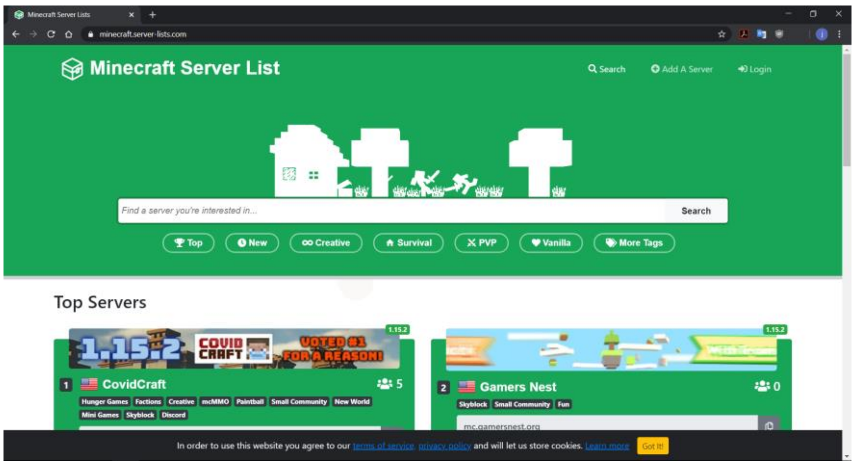


In order to use this website you agree to our terms of service, privacy policy and will let us store cookies. Learn more

GAME SERVER WEBSITE

IPT Task 3 - Project

ABOUT

A website designed to showcase various Game Servers so that players can join them quickly and easily.

By Jack Donaldson

Table of Contents

Understanding the Problem

Problem Statement

The aim of this project is to produce a solution for a Client that allows Game Server owners to advertise their servers to players in a quick and easy manner. It also needs to be appealing and compete with existing solutions, in turn becoming financially sustainable.

Preliminary Investigation

Justification of Research Methods

As part of my initial preliminary investigation, I have chosen to conduct research in the form of surveys and interviews. I chose to interview my client as they are the main key stakeholder and will give a good foundation for narrowing down the areas of further research, as such their responses would be the most valuable and need to be detailed. Unfortunately despite the detailed responses interviews allow for, they are particularly time-consuming meaning that I couldn't conduct many of them and as such have just decided to conduct them with a few key stakeholders, mainly the client, as well as some Game Server owners to better understands their needs.

After my main interview was conducted I decided to create and send a survey to potential users of the solution about what kind of features or capabilities they would want/expect in a service like this. The results of the interview helped direct what kind of questions to ask the potential users in this survey, for example, I focused a significant amount of questions on what format would be best suited for the solution, such as an app or a website as they were the two main preferred options illustrated in the interviews.

Unlike interviews, surveys are a lot quicker to produce and receive results from, they also can easily target a large number of people, and in this case, I was able to get over 110 responses to my survey. However, unlike interviews, surveys are often limited to simple questions and cannot provide very detailed data, hence quantitative data is really the only type you can receive from surveys, in turn limiting the variety of questions I could ask.

Interview

Client

As part of the long interview with the client, I have included the transcript for a few key questions within which will help narrow down our focus for the solution.

In terms of the actual solution is there any preference to what type of digital solution you would like?

Most of our competitors are using a main website, which has proven to be quite popular. Other considerations we would like you to look into would be a mobile app as a significant amount of users are now trying to find this information quickly using a mobile device. More recently, online communities or forums are also becoming popular which may be another avenue to look into.

Are there any specific games you would like us to support? If so what are they?

Well ideally, the solution would be as generic or malleable as possible in order for it to be used across a wide variety of games in the future. For now, the most popular game with this type of service would be Minecraft but we would like to see the solution work for other games. At the moment our competitors don't really have a diverse range of supported games and often target one, we plan to target as many as possible to help build our brand.

Those other games you refer to, do you have any examples?

Sure, Rust, Terraria, Ark: Survival Evolved, Unturned, Conan Exiles & even CS:GO surprisingly.

As a start-up, would you prefer a larger solution with more ongoing maintenance costs or a smaller more cheaper one?

We do understand that there will be inevitable ongoing costs for running the provided solution, ideally, we would prefer as little ongoing costs as possible. However, we are prepared to spend a bit more early on if it means a more popular product in the long term as that can become a good selling point, particularly against our competitors. That said, we would prefer more ongoing technical costs if it means we don't have to employ as many people to oversee the maintenance in the long run as they can be expensive, particularly in Australia.

This interview helped elaborate on what the client was looking for in a solution, identifying the different types of solutions they would prefer, mainly an app or website. The questions were designed to mainly clarify the expectations of the client, in this case, what ongoing costs they were prepared to accommodate as well as any games they wanted to be supported. In this case, they divulged how they wanted it to work for as many games as possible in order to differentiate from their competitors who often only worked with single games. It became apparent that they were prepared to spend initially in order to build a better product then their competitors as competition in this area was quite fierce.

Game Server Owners

In addition to the main interview with the client, I decided to interview some Game Server owners to get an understanding of what they would like in a solution of this kind as well as what they have found works effectively.

In terms of the actual solution is there any preference to what type of digital solution you would like?

Most of our competitors are using a main website, which has proven to be quite popular. Other considerations we would like you to look into would be a mobile app as a significant amount of users are now trying to find this information quickly using a mobile device. More recently, online communities or forums are also becoming popular which may be another avenue to look into.

Are there any specific games you would like us to support? If so what are they?

Well ideally, the solution would be as generic or malleable as possible in order for it to be used across a wide variety of games in the future. For now, the most popular game with this type of service would be Minecraft but we would like to see the solution work for other games. At the moment our competitors don't really have a diverse range of supported games and often target one, we plan to target as many as possible to help build our brand.

Those other games you refer to, do you have any examples?

Sure, Rust, Terraria, Ark: Survival Evolved,
Unturned, Conan Exiles & even CS:GO surprisingly.

As a start-up, would you prefer a larger solution with more ongoing maintenance costs or a smaller more cheaper one?

We do understand that there will be inevitable ongoing costs for running the provided solution, ideally, we would prefer as little ongoing costs as possible. However, we are prepared to spend a bit more early on if it means a more popular

PRODUCT IN THE LONG TERM AS THAT CAN BECOME A GOOD SELLING POINT, PARTICULARLY AGAINST OUR COMPETITORS. THAT SAID, WE WOULD PREFER MORE ONGOING TECHNICAL COSTS IF IT MEANS WE DON'T HAVE TO EMPLOY AS MANY PEOPLE TO OVERSEE THE MAINTENANCE IN THE LONG RUN AS THEY CAN BE EXPENSIVE, PARTICULARLY IN AUSTRALIA.

This interview helped elaborate on what the client was looking for in a solution, identifying the different types of solutions they would prefer, mainly an app or website. The questions were designed to mainly clarify the expectations of the client, in this case, what ongoing costs they were prepared to accommodate as well as any games they wanted to be supported. In this case, they divulged how they wanted it to work for as many games as possible in order to differentiate from their competitors who often only worked with single games. It became apparent that they were prepared to spend initially in order to build a better product then their competitors as competition in this area was quite fierce.

Survey

After conducting the interview, as elaborated above I decided to survey Gamers, particularly ones who had used a similar service before in order to try and understand what features they want/expected as well as to get a gist of some other general expectations. This provides a good foundation for what the 'customers' are actually expecting from this product as well to reaffirm that the assumptions of the client, as illustrated through the interview are accurate. As seen below, this interview was relatively short in order to encourage people to fill it out, the initial questions were mainly for just ensuring that the only results which were examined were those of our target market. This meant that the latter questions regarding features were only answered by people who knew what they were talking about.

After the initial confirmation questions, I then tried to understand peoples preferences in regards to format, essentially suggesting the options of a website, app and game forum which came from the clients suggestions when interviewed. Furthermore, the popularity of features was obtained, here fairly common features were listed such as 'Server Voting' to confirm whether the users of a solution like this actually needed/requested these features despite it essentially becoming 'standard' to have voting on these types of sites. Generally speaking, the responses from this allowed me to learn where to prioritise my time in terms of development as lower priority features could become time consuming and wasteful.

Furthermore, the question regarding user's habits with these types of solutions was placed in order to determine how to prioritise search engine optimisation (SEO) in our feature set. Essentially, users were asked if they stayed with these type of solutions through bookmarks or if they consistently just googled and picked the first result. This is important as often sites that are unique and with a higher perceived value will be bookmarked, meaning that if an audience is built up through the knowledge of a brand then a drop in ranking on search engines such as Google or Yahoo wouldn't matter as much.

Next, Question 6, regarding user interface design is meant to analyse what type of sites users are looking for. Combined with competitor research and analysis there often seems to be a clear trend for solutions that are 'too detailed' and end up sacrificing user interface design and space in order to cram as much information on the screen as possible. As such, I was trying to determine if that trend was because of user interaction or if it was just because there wasn't really that many well designed competing sites forcing them to innovate. This can be seen with this (https://minecraft-mp.com/) site which is extremely detailed but still highly successful. However, other server list sites (notably not game-related ones) such as this one (https://top.gg/) use a much simpler design and still work well. Hence, determining if a much more visually appealing site would be viable and liked by Gamers is important. Finally, any extra opinions were welcome but it was rather difficult to analyse this question effectively due to the highly individualised responses.

Game Server Project Preliminary Survey

Hi Everyone, would you mind taking this short survey which is focused on our determining what you would like to see in our new Game Server List project!

* Required

1. Have you used a Game Server website or app before?

   Mark only one oval.

   - Yes
   - No

2. How regularly do you play multiplayer games?

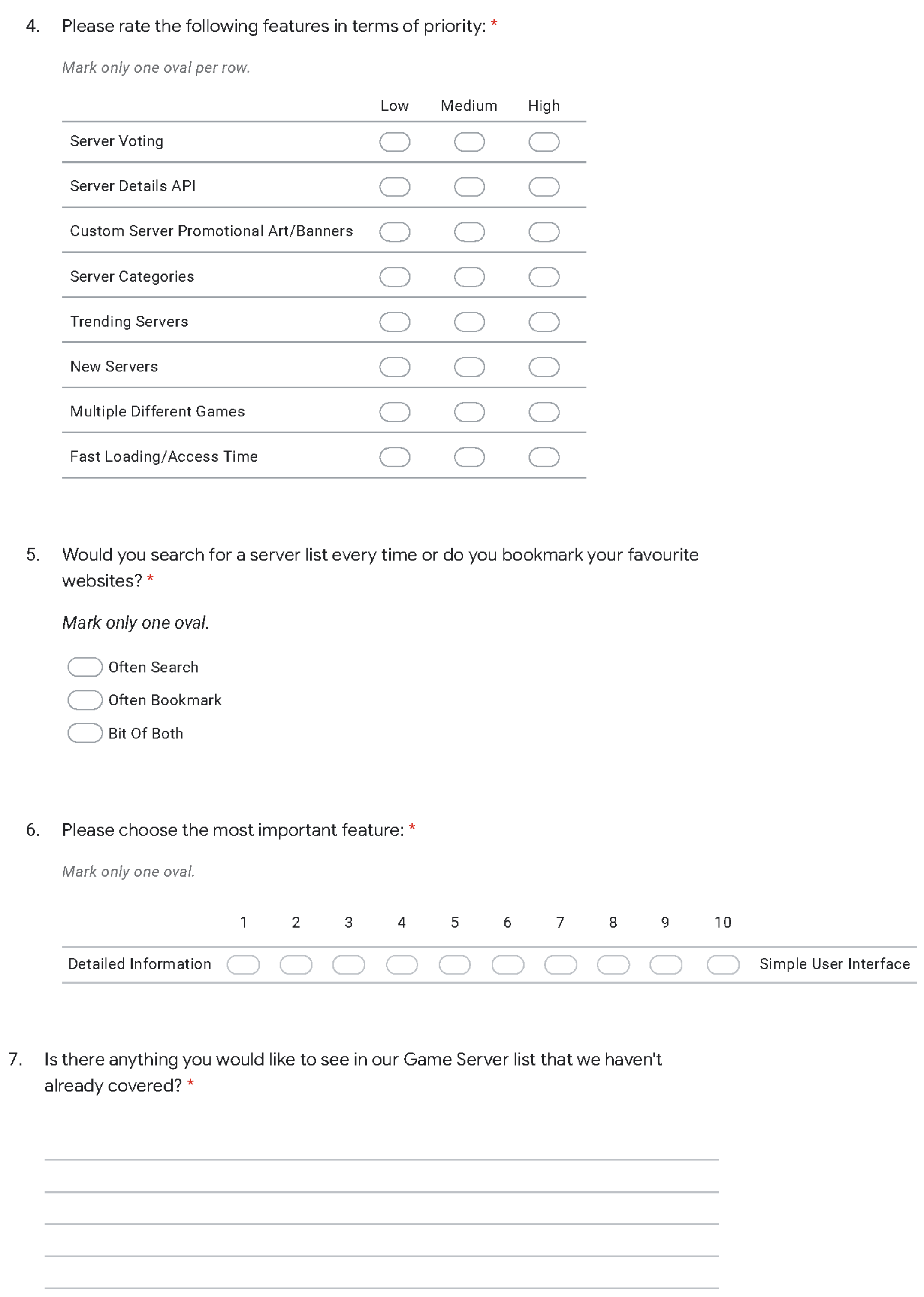
   Mark only one oval.

   - Every Day
   - At least once a week
   - Once a month
   - A few times a year
   - Almost never

3. Which of the following would you prefer to use to find a server? *

   Mark only one oval.

   - Website
   - Mobile App
   - Discord/Game Forum
   - Other: ____________________



4. Please rate the following features in terms of priority: *

Mark only one oval per row.

	Low	Medium	High
Server Voting	◯	◯	◯
Server Details API	◯	◯	◯
Custom Server Promotional Art/Banners	◯	◯	◯
Server Categories	◯	◯	◯
Trending Servers	◯	◯	◯
New Servers	◯	◯	◯
Multiple Different Games	◯	◯	◯
Fast Loading/Access Time	◯	◯	◯

5. Would you search for a server list every time or do you bookmark your favourite websites? *

Mark only one oval.

◯ Often Search
◯ Often Bookmark
◯ Bit Of Both

6. Please choose the most important feature: *

Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Detailed Information	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Simple User Interface

7. Is there anything you would like to see in our Game Server list that we haven't already covered? *

Conclusion

Constraints

Some of the main constraints I have identified for the project from my preliminary research which need to be considered are:

- Time available and deadline for the project

- Ensuring that the client is happy with the end product
- Making sure the project is financially viable
- The budget available
- IT/Technical Skills required to complete the project

Criteria for Success

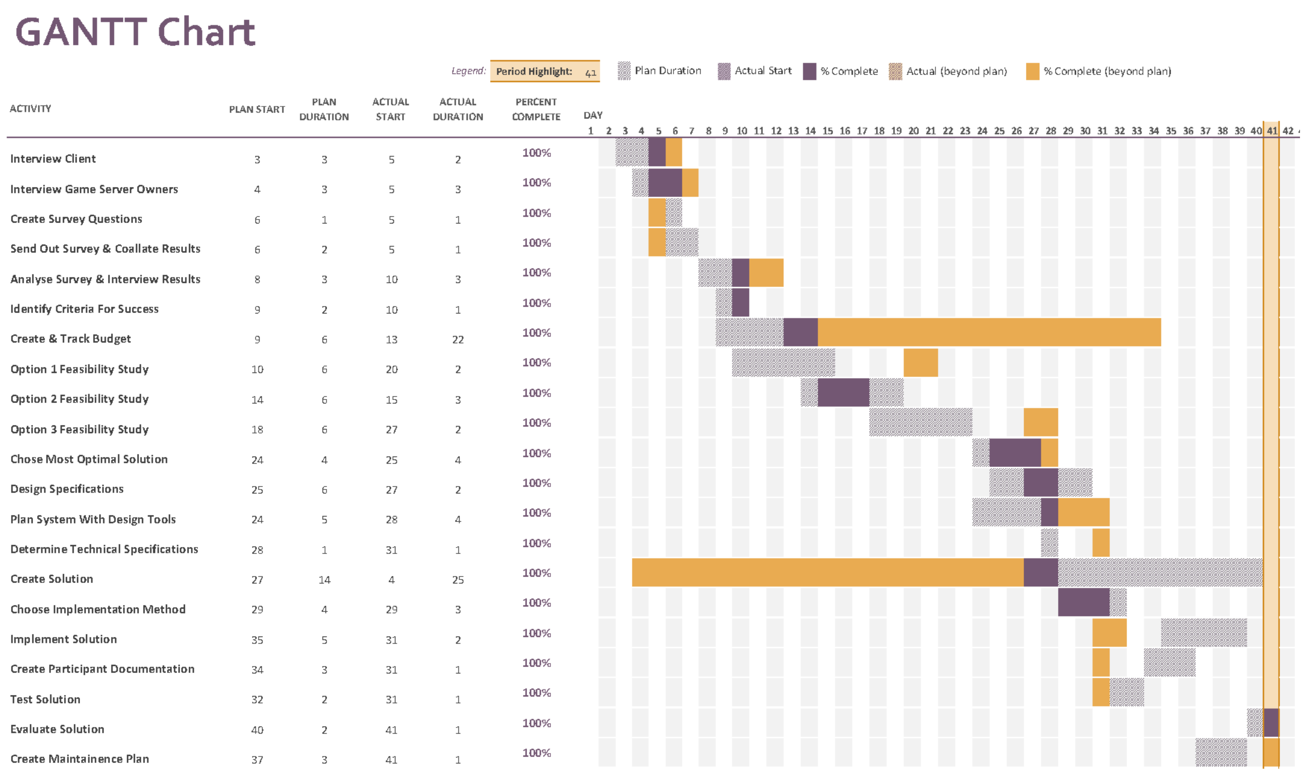
After analysing the responses from both the interviews and surveys I compiled a list of key criteria that the project should meet in order to evaluate its success. These are categorised by their priority meaning that I will focus on the high priority tasks first as they are more important.

Criteria	Priority
Needs to be easily accessible on Mobile devices	**High**
Rank high and optimise for search results	**High**
Allow server owners to upload custom promotional images/art.	**High**
Have a dedicated administrator section	**High**
Allow voting for favourite servers	**High**
Accessed quickly and fast	**High**
Within the budget of $500	**Medium**
Works on Old devices	**Medium**
Requires little maintenance	**Medium**
Inclusive and accessible to screen readers and has a minimal CLS (Cumulative Layout Shift)	**Low**
Runs efficiently, requiring less server resources	**Low**
Allow server owners to have analytics	**Low**

Overall most of the priority criteria relate to features such as ranking high in search results or allowing server owners to upload custom art. This can be deemed as a crucial part of the project as both the interviews and surveys confirmed that this was important to the end-users and without it the project wouldn't be successful. However, not everything can be a high priority and as such the ability to work on old devices and to remain within budget is fairly flexible, meaning it is not required but still would be good to have. This is mainly because the client didn't mind too much on budget and neither do I nor will most of the people using the website be on older devices as often tech enthusiasts are using the latest technology. In regards to low priority features things such as optimisation and analytics/tracking aren't required in order for the project to be a success however if time permits they are still good things to consider but ultimately aren't pivotal to the success of the project.

Project Plan

GANTT Chart



GANTT Chart

Legend: Period Highlight: 41 ▨ Plan Duration ▨ Actual Start ■ % Complete ▨ Actual (beyond plan) ■ % Complete (beyond plan)

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Interview Client	3	3	5	2	100%
Interview Game Server Owners	4	3	5	3	100%
Create Survey Questions	6	1	5	1	100%
Send Out Survey & Coallate Results	6	2	5	1	100%
Analyse Survey & Interview Results	8	3	10	3	100%
Identify Criteria For Success	9	2	10	1	100%
Create & Track Budget	9	6	13	22	100%
Option 1 Feasibility Study	10	6	20	2	100%
Option 2 Feasibility Study	14	6	15	3	100%
Option 3 Feasibility Study	18	6	27	2	100%
Chose Most Optimal Solution	24	4	25	4	100%
Design Specifications	25	6	27	2	100%
Plan System With Design Tools	24	5	28	4	100%
Determine Technical Specifications	28	1	31	1	100%
Create Solution	27	14	4	25	100%
Choose Implementation Method	29	4	29	3	100%
Implement Solution	35	5	31	2	100%
Create Participant Documentation	34	3	31	1	100%
Test Solution	32	2	31	1	100%
Evaluate Solution	40	2	41	1	100%
Create Maintainence Plan	37	3	41	1	100%

Finance Plan

Although I am willing to spend up to around $500 as part of the project I am still trying to keep costs as low as possible in order to maximise profit. As part of this, I have chosen competitive pricing, particularly in regards to Server Hosting.

ITEM	NOTES	PURCHASE COST	ACTUAL COST TO ME

DOMAIN NAME	Used A Coupon For 40% off ☺	$19.99	$11.99
AMAZON EC2 VIRTUAL SERVER HOSTING (OHIO)	Allowed a 'FREE' tier for first 12 months as a student.	$0.0116/Hour	$0
AMAZON RDS RELATIONAL DATABASE SERVER HOSTING (OHIO)	Allowed a 'FREE' tier for first 12 months as a student.	$0.017/Hour	$0
AMAZON ELASTICSEARCH SERVER HOSTING (OHIO)	Allowed a 'FREE' tier for first 12 months as a student.	$0.018/Hour	$0
AMAZON DATA TRANSFER FROM OHIO TO NORTH VIRGINIA	Required in order to send data from server host to email server. Usually only less than a GB each month.	$0.01 per GB	$0.01
AMAZON SIMPLE EMAIL SERVICE (NORTH VIRGINIA)	Free 62,000 emails per month. Shouldn't need more than this.	$0.10 for every 1000 emails over 62,000	$0
REPL IT HACKER PLAN	Free With Github Student Education Pack. Needed to use Source Control in Repl.it	$5/Month	$0
LAPTOP/COMPUTER	Laptop was not purchased just for this project and cheaper ones would be available and suffice.	$1,100	$0
INTERNET PLAN – TELSTRA NBN UNLIMITED	Already had an internet plan for additional interests, hence not	$90/Month	$0

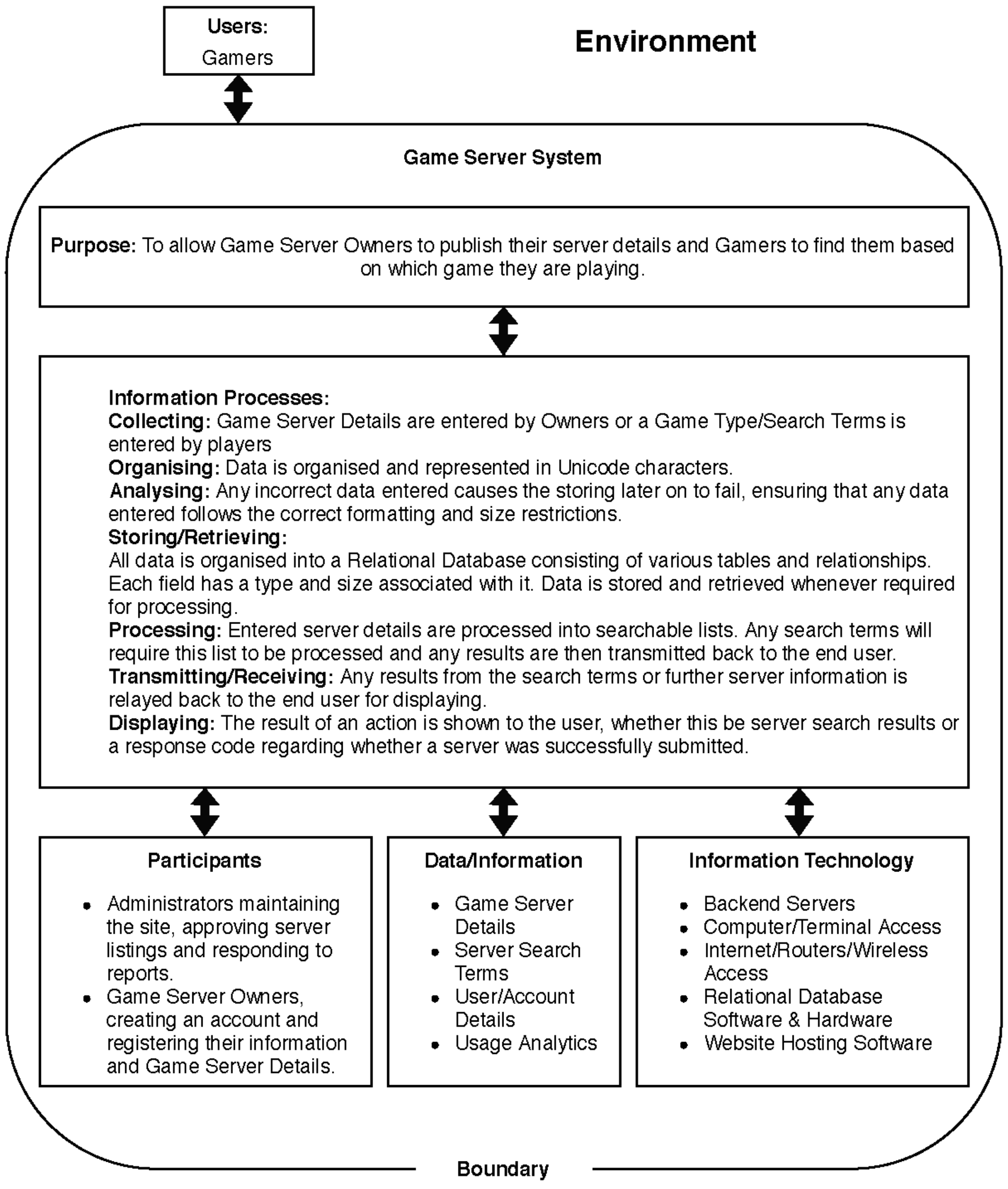
	associating this cost directly to the project.
	Total: $12.00

As illustrated above, Amazon has a rather generous 'Free tier' on their web hosting platform that lasts for a year. This means the majority of the services required will be free for the first year of operation, however in regards to industry pricing, once the free tier is up the pricing is very competitive. In this case, the pricing given is their 'On-demand' pricing which is what will be used initially. Once the free tier is up and payments start been made for the servers and their power Amazon will allow you to reduce the cost of your servers by locking in a plan for either 1 or 3 years and 'reserving' your servers. This will provide a discount of 40% on the prices provided above and as such if the solution is a success it will be very cheap and efficient to run.

As an additional note, post evaluation Amazon has announced support for purchasing email servers in their Ohio region. Once the application has been accepted this will allow our emails to be sent from Ohio, in turn meaning that no data will be needed to be transferred to North Virginia, therefore the cost will be reduced. However, their opening date is past the submission of this project, hence it still being required in this project.

Project Journal

Summary



Purpose: To allow Game Server Owners to publish their server details and Gamers to find them based on which game they are playing.

Information Processes:
Collecting: Game Server Details are entered by Owners or a Game Type/Search Terms is entered by players
Organising: Data is organised and represented in Unicode characters.
Analysing: Any incorrect data entered causes the storing later on to fail, ensuring that any data entered follows the correct formatting and size restrictions.
Storing/Retrieving:
All data is organised into a Relational Database consisting of various tables and relationships. Each field has a type and size associated with it. Data is stored and retrieved whenever required for processing.
Processing: Entered server details are processed into searchable lists. Any search terms will require this list to be processed and any results are then transmitted back to the end user.
Transmitting/Receiving: Any results from the search terms or further server information is relayed back to the end user for displaying.
Displaying: The result of an action is shown to the user, whether this be server search results or a response code regarding whether a server was successfully submitted.

Participants

- Administrators maintaining the site, approving server listings and responding to reports.
- Game Server Owners, creating an account and registering their information and Game Server Details.

Data/Information

- Game Server Details
- Server Search Terms
- User/Account Details
- Usage Analytics

Information Technology

- Backend Servers
- Computer/Terminal Access
- Internet/Routers/Wireless Access
- Relational Database Software & Hardware
- Website Hosting Software

As illustrated above on the Information System in Context Diagram, this system generally has one main type of user which is Gamers who are looking for a server. This they then interact with this system whose goal is to gather all server information and return relevant results to them in order to allow them to find a suitable server that they may enjoy. As with any information system, the information processes are involved, essentially information such as search terminology is collected from the user and processed through various mechanisms, generally with the aid of a database in order to find appropriate results. However, in order for these processes to work it requires participants such as site administrators who approve server listings, monitor and maintain both the website and database. It also needs Game Server Owners to register their server details with the site in order for it to become searchable. Additionally, the data/information used throughout is required whether that be the raw server details, usage analytics or the search terminology entered by the user. Linking this together is the aid of Information Technology, whether that be the servers hosting and processing the database or website or even the internet's arrays of routers, switches and hubs transmitting packets between the individual's devices or their computers. These components are all required to work together to achieve one goal in order to allow this Game Server Information System to work properly.

Making Decisions

Analysis Report

Mobile App

Overview

This solution involves designing a Mobile App that users can download in order to browse and search for their favourite games servers. This would have to be suitable for both Android and IPhone users in order to maximise the effective target market and doing so would mean that any advertising revenue would potentially get cut by 30% due to app store policies. However, the mobile app market is forever expanding with majority of the world's population owning a phone, giving them easier access than ever. This type of solution also hasn't been adopted yet by any competitor Game Server Listing sites so it could be a very open market to break into, particularly as a fledgling product.

Feasibility Study

Technical

This would require both an Android and an Apple IPhone to develop on and test the app repeatedly. This is one of the challenges required when building an app as it has to be 'cross-platform' this means that although Android and Apple both use different languages when programming apps you often have to have an 'intermediary' language, for example like Xamarin Forms. In this scenario the layouts of the apps can be created from one language (Xamarin) but any functionality has to be split between the platforms and programmed in two different languages as each company has their own syntax and rules for accessing the phones. This means ultimately that I would have to learn three different programming languages to even attempt this solution which requires almost double the amount of work to just support Android and Apple products.

This is echoed in today's society as majority of solo app developers choose one platform and stick to it, in this case ruining market potential.

Operational

Economical

This solution would require the purchasing of an IPhone as well as an Android Phone without a cracked screen. This alone would cost well over $1000 which is significantly higher than our preferred budget as outlined in the finance plan and criteria for success. Although I am flexible this is still significantly higher than I would like, considering the need to also purchase recurring Apple developer licenses in order to publish the app as well as a Google Play license needed to publish. Alongside this in order to export an Apple app you need an apple product such as a MacBook Air with the other alternative been to pay a subscription for a cloud based compiling product. In conjunction the creation of an app would mean it has to abide by Google & Apple's terms of service, generally meaning they take a slice of revenue usually around 30% which is a significant margin, particularly for a start-up.

Schedule

As I haven't had much experience with mobile apps or related products before this may take significantly longer than expected. Alongside this, it has been recommended by the client to try and get the project finished as quickly as possible. As such, the proposed solution may struggle to meet the deadline and has a significantly high potential to go over the deadline as the chance for bugs is high enough without considering I'm a novice in regards to app development.

Advantages

The main advantages of this solution is that it is a growing market with big potential and very limited competition in this niche. As such, the rise of mobile gaming has become a considerable factor in how popular this type of app could become. This type of app could potentially also allow for the ability of an offline list, allowing people to view it when they have poor or limited internet connectivity, although generally if they are playing multiplayer games, internet connectivity isn't really an issue.

Disadvantages

Some of the main disadvantages of this solution involve having to purchase some significant hardware in order to start even developing and compiling the app which would blow the budget out of proportions. This also requires learning three new languages which will take a lot of time and potentially cause the project to go over budget which is the last thing the client wants.

Website

Overview

Feasibility Study

Technical

Operational

Economical

Schedule

Advantages

Disadvantages

Discord/Game Forum Server

Overview

Feasibility Study

Technical

Operational

Economical

Schedule

Advantages

Disadvantages

Chosen Solution

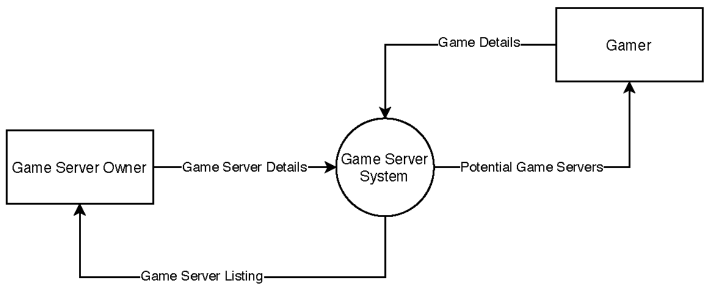
Design Specifications

Provide details of what you require to solve the problem – hardware, software, input, output processing. Add table with requirements + reasons for use

Designing Solutions

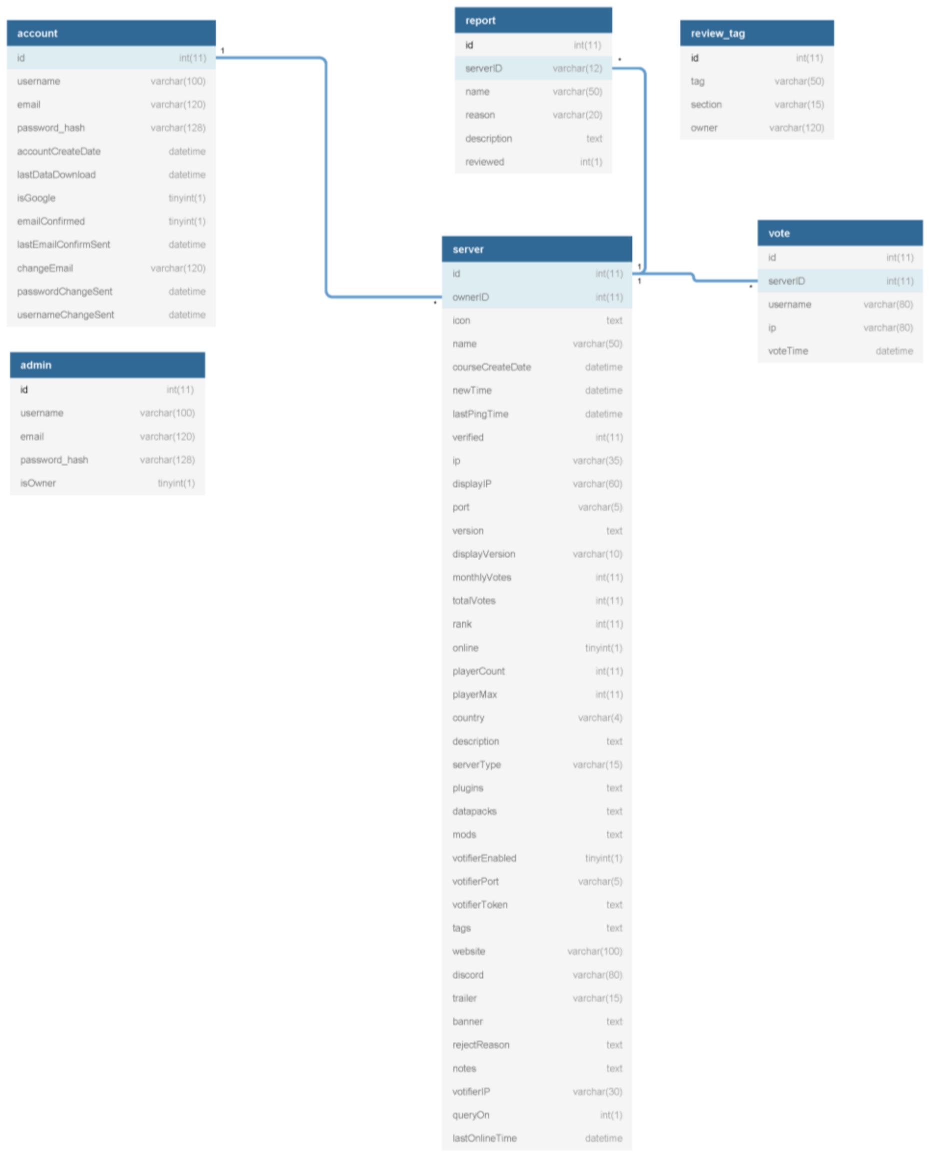
Design Tools

Context Diagram



Data Flow Diagrams

Database Schematic Diagram



account	
id	int(11)
username	varchar(100)
email	varchar(120)
password_hash	varchar(128)
accountCreateDate	datetime
lastDataDownload	datetime
isGoogle	tinyint(1)
emailConfirmed	tinyint(1)
lastEmailConfirmSent	datetime
changeEmail	varchar(120)
passwordChangeSent	datetime
usernameChangeSent	datetime

admin	
id	int(11)
username	varchar(100)
email	varchar(120)
password_hash	varchar(128)
isOwner	tinyint(1)

report	
id	int(11)
serverID	varchar(12)
name	varchar(50)
reason	varchar(20)
description	text
reviewed	int(1)

review_tag	
id	int(11)
tag	varchar(50)
section	varchar(15)
owner	varchar(120)

vote	
id	int(11)
serverID	int(11)
username	varchar(80)
ip	varchar(80)
voteTime	datetime

server	
id	int(11)
ownerID	int(11)
icon	text
name	varchar(50)
courseCreateDate	datetime
newTime	datetime
lastPingTime	datetime
verified	int(11)
ip	varchar(35)
displayIP	varchar(60)
port	varchar(5)
version	text
displayVersion	varchar(10)
monthlyVotes	int(11)
totalVotes	int(11)
rank	int(11)
online	tinyint(1)
playerCount	int(11)
playerMax	int(11)
country	varchar(4)
description	text
serverType	varchar(15)
plugins	text
datapacks	text
mods	text
votifierEnabled	tinyint(1)
votifierPort	varchar(5)
votifierToken	text
tags	text
website	varchar(100)
discord	varchar(80)
trailer	varchar(15)
banner	text
rejectReason	text
notes	text
votifierIP	varchar(30)
queryOn	int(1)
lastOnlineTime	datetime

Data Dictionary

account

Column	Type	Null	Default	Comments
id *(Primary)*	int(11)	No		A unique ID for each account.
username	varchar(100)	No		A unique username for each account that users can login with.
email	varchar(120)	No		The email address linked to a users account.
password_hash	varchar(128)	Yes	*NULL*	The hashed/encrypted version of a users password.
accountCreateDate	datetime	No		The date an account was created.
lastDataDownload	datetime	Yes	*NULL*	The last time a user downloaded their data.
isGoogle	tinyint(1)	No		Whether the account is linked to a Google Account.
emailConfirmed	tinyint(1)	No		Whether the account has it's email confirmed.
lastEmailConfirmSent	datetime	No		The last time an account email confirmation was sent.
changeEmail	varchar(120)	No		The new email address a user is trying to change to.
passwordChangeSent	datetime	No	CURRENT_TIMESTAMP	The last time a password change email was sent.
usernameChangeSent	datetime	No	CURRENT_TIMESTAMP	The last time a username change email was sent.

admin

Column	Type	Null	Default	Comments
id *(Primary)*	int(11)	No		A unique ID for each administrator account.
username	varchar(100)	No		The username for the admin account. Used to login.
email	varchar(120)	No		The email associated with the administrator account.
password_hash	varchar(128)	Yes	*NULL*	The hashed/encrypted password for this account that is used to log in.
isOwner	tinyint(1)	No		Whether this administrator account has access to create other administrators.

report

Column	Type	Null	Default	Comments
id *(Primary)*	int(11)	No		A unique ID for each report.
serverID	varchar(12)	No		The server ID associated with this report. (Foreign Key)
name	varchar(50)	No		The name associated with the user that submitted the report.
reason	varchar(20)	No		The reason the user is submitting a report.
description	text	No		The description of the report.
reviewed	int(1)	No	0	Whether this report has been reviewed by an administrator.

review_tag

Column	Type	Null	Default	Comments
id *(Primary)*	int(11)	No		The unique ID of the review tag.
tag	varchar(50)	No		The name of the tag.
section	varchar(15)	No		The category/section the tag is associated with, e.g Plugins or Mods
owner	varchar(120)	No		The username of the person who requrested this tag to be added.

server

Column	Type	Null	Default	Comments
id *(Primary)*	int(11)	No		The unique ID of this server.
ownerID	int(11)	Yes	*NULL*	The unique ID of the account which submitted this server. (Foreign Key)
icon	text	Yes	*NULL*	The icon associated with this server.
name	varchar(50)	No		The name of this server.
courseCreateDate	datetime	No		The date this server was initially registered.
newTime	datetime	Yes	*NULL*	The date this server was first verified. Used to order the 'New Servers' category.
lastPingTime	datetime	Yes	*NULL*	The last time this server was online. Used to de-register servers that haven't been online for more then a week straight.
verified	int(11)	No		Whether this server has been verified by an administrator.
ip	varchar(35)	No		The IP used to connect to this server.
displayIP	varchar(60)	Yes	*NULL*	The IP that is shown on the website. Usually combined with the port if differing from the default 25565.
port	varchar(5)	No		The port of the server.
version	text	Yes	*NULL*	The API version of the server.
displayVersion	varchar(10)	Yes	*NULL*	The game version of the server, verified by an admin.
monthlyVotes	int(11)	No		The amount of votes this server has recieved this month.
totalVotes	int(11)	No		The total amount of votes this server has received since submitted on the platform.
rank	int(11)	No		The rank of this server, sorted by monthly votes.
online	tinyint(1)	No		Whether this server is online.
playerCount	int(11)	No		The amount of players on the server last time it was pinged.
playerMax	int(11)	No		The maximum amount of players on the server last time it was pinged.
country	varchar(4)	No		The country the server is hosted in.
description	text	No		The description of the server, provided by the owner and verified/modified by an administrator.

serverType	varchar(15)	No		The type of server, e.g Vanilla, Modded, Spigot etc Provided by server owner.
plugins	text	Yes	*NULL*	The list of plugins running on the server.
datapacks	text	Yes	*NULL*	The list of datapacks running on the server.
mods	text	Yes	*NULL*	The list of mods running on the server.
votifierEnabled	tinyint(1)	No		Whether this server has the votifier plugin enabled and details provided by the owner.
votifierPort	varchar(5)	No		The port used to connect to the votifier plugin on the server.
votifierToken	text	Yes	*NULL*	The token used to verify a vote with the votifier plugin on the server.
tags	text	Yes	*NULL*	The list of tags associated with the server, decided by the owner.
website	varchar(100)	Yes	*NULL*	The website associated with the server.
discord	varchar(80)	Yes	*NULL*	The discord associated with the server.
trailer	varchar(15)	Yes	*NULL*	A youtube link to a trailer showcasing the server.
banner	text	Yes	*NULL*	A banner showcasing the features of the server. Displayed in search results.
rejectReason	text	Yes	*NULL*	The reason why a server submission was rejected by an administrator.
notes	text	No		Any notes on the server. Only accessible to administrators for logging history.
votifierIP	varchar(35)	No		The IP used to connect to the Votifier Plugin running on the server. Sometimes different to the server IP.
queryOn	int(1)	No	0	Whether the server has query mode enabled allowing more detailed pinging by our API.
lastOnlineTime	datetime	No	CURRENT_TIMESTAMP	The last time the server was online.

vote

Column	Type	Null	Default	Comments
id *(Primary)*	int(11)	No		The unique ID of this vote.
serverID	int(11)	Yes	*NULL*	The server this vote is registered to.
username	varchar(80)	No		The minecraft username of the user who voted.
ip	varchar(80)	No		The IP of the person who voted.
voteTime	datetime	No		The time this vote was submitted.

Other options include Storyboards, Data dictionaries + decision trees & tables (Not really needed but would be good)

Technical Specifications

Hardware

The chosen solution will require a significant amount of hardware. However, in this case, some of that hardware particularly the servers hosting the website will be rented out as it is much more economic and can scale very easily with the help of companies such as Amazon Web Services. To begin with, computers are required in order to program the website using various software. In this case, I have used my personal Desktop Computer alongside my School laptop as well as my personal laptop. Each of these comes equipped with a fairly powerful processor (Intel i5 or above) in order to ensure code execution time is minimised as well as a fast NVME SSD allowing quick file browsing and access. The faster code runs, the faster I can program and in turn complete the project. In this case, my testing sandbox is located on an external server array hosted by https://Repl.It. The server which actually runs the demo application runs with 2 GB of RAM as well as 2 vCPU's giving plenty of room for my quick testing and running any changes. However in order to access both this server and the test website I require fast internet access. Unfortunately, my current situation means that I am unable to get a decent internet connection as it currently is very slow and cuts out making it increasingly more difficult to program on the demo server at home. The infrastructure we have is NBN Fixed Wireless however, my school has a much faster internet connection making it easier to program there. As such I have often stayed back and worked on the project using the faster internet on my laptop, in turn increasing efficiency thanks to much lower latency.

Furthermore, the production servers the website run on are hosted in Ohio, USA. The reason for this is it is a fairly central location in terms of the world whilst having a reasonable ping to European countries and one to the outer Asian regions. This is particularly important considering website latency is one of the major factors that drive users to click away from a site before it even loads. Another advantage of this location is it is very central to my user base, i.e. most of them are from either America or Europe, particularly the United Kingdom. Currently, the website runs on a virtualised hosting server, meaning that it is 'virtually' split up with software and its resources are allocated to different users. In this case, users pay for the number of resources you will need which is much more beneficial when compared to a shared hosting alternative offered by many cheaper website hosts. Shared hosting allows websites unlimited access to the machine and usually is advertised with high CPU counts and RAM but in reality, it just takes one application to completely throttle the resources of the whole server to take it down, hence the more secured, slightly more expensive version of virtualised hosting is the better option in this scenario. The actual resources allocated to the server at the moment is 2 vCPUs and 2GB of RAM which thanks to optimisations in my code should be able to run around 50,000 concurrent users according to stress tests I have performed (more detail later on). The reason for this particular resource usage is mainly due to it being the highest amount covered under Amazon's 12 month 'Free Tier'. However, the website code isn't the only thing requiring a virtualised server, both the database and the elastic search engine (responsible for indexing and processing search results) run on separate virtualised servers. These all have a similar configuration with slightly more ram for the database and slightly less CPU power in the elastic search configuration, mainly due to it not being as needed. These severs all interact using a Local Area Network and 'virtualised' connections, meaning that it is very simple and quick for them to interact with each other. In this case, each of the virtualised servers is stored in a different building to increase data redundancy but is physically connected via Fibre Optic cables. This allows high amounts of data between the virtualised servers to be transferred, peaking at over 2tb/s (The NVME storage often limits before this but multiple concurrent connections can reach a max of 2tb/s). This network of routers and switches connecting the buildings as illustrated earlier creates a 'Local Area Network' for the hardware thanks to virtualisation software, even though it is technically a Wider Area Network or WAN.

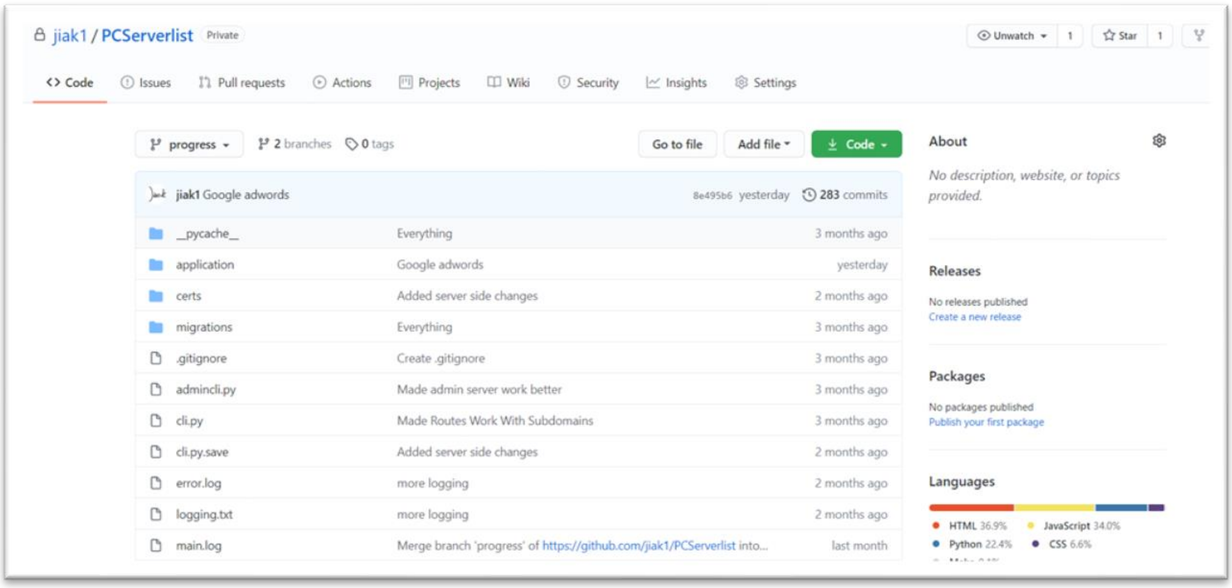
Finally, Amazon provides a dedicated Mail Server which the website uses to send and receive emails. This is run in a similar fashion on a virtualised machine, however, in this case, Amazon doesn't currently offer mail servers in Ohio where the other instances are located. As such, these servers are located in North Virginia and have to communicate through the internet, luckily Amazon has a private internet line running the majority of the way, connecting the two data centres allowing internet/data transfer costs between the instances to be lower than standard Telco rates.

Software

The website uses a significant amount of software to run but in terms of the most important the Ubuntu Server distribution would come at the top. This is the operating system that all of the virtual machines run as it is free, efficient and very small meaning it takes up a very small amount of disk space. In regards to specifics, Amazon uses containerisation technologies such as Docker in order to create these ‘virtualised instances’ on the servers as such all of my code is run inside a ‘container’ that Docker creates. These containers states than can be simply saved, stopped and started at any time allowing fast changes to my server, whenever I need via a simple online control panel. Inside my main Ubuntu container, I have a variety of software running, with the website being controlled by Python using a library called Flask. This library, however, needs other software to run efficiently in a production environment. In the case of the demo server running the code using a standard Python interpreter works fine, however, this would run on only one thread, meaning that the code has to run synchronously, i.e. one after the other. This means if the code has to wait, i.e. if someone is uploading a file than any other people trying to access the website would have to wait until that file is uploaded and the code can continue running. This naturally is not appropriate for production and is why I used GUnicorn as a WSGI server. A WSGI (Web Server Gateway Interface) translates incoming HTTP (Web Based) Packets and forwards them to the appropriate section of Python code. In this process, it also manages any services of Python and creates multiple threads. This means that multiple workers or ‘copies’ of the python program are run each time a request is received hence stopping the freezing problem illustrated earlier. However, software like GUnicorn needs to be configured appropriately to ensure that there are always threads available on the CPU and it doesn’t end up trying to use more than it has available, in turn crashing the whole OS. However, GUnicorn isn’t the only software used to receive web requests. When a packet first is directed to the server it gets received by a program called Apache. This program allows me to define rules and firewall protection as well as defining where each request goes depending on the location and header URL attached. For example, if a URL contains https://minecraft.server-lists.com/static I know that it just wants to return the file on the hard drive and as such it is much more efficient to use Apache to return the file rather than bothering to load up a valuable Python service worker with GUnicorn. In other cases I can specify specific subdomains, for example, any request to admin.server-lists.com will go to a separate GUnicorn instance that is connected to my Admin Flask Python program with a similar strategy employed for the Minecraft section. This means that certain domains can be enabled and disabled easily without affecting each other, as long as the server is still running effectively.

In regards to syncing the files across to the server, I use a program called Git. This is a tool that allows you to track any file changes in a directory or project and download just those. It also allows any easy rollbacks if mistakes are made. In this case, I store all of my website code and files on a free service called

Github in a private repository. This service is extremely useful as you can see all of your files online and edit any changes or revert to any previous commits (commits are just changes to files).



jiak1 Google adwords 8e495b6 yesterday 283 commits

pycache	Everything	3 months ago
application	Google adwords	yesterday
certs	Added server side changes	2 months ago
migrations	Everything	3 months ago
.gitignore	Create .gitignore	3 months ago
admincli.py	Made admin server work better	3 months ago
cli.py	Made Routes Work With Subdomains	3 months ago
cli.py.save	Added server side changes	2 months ago
error.log	more logging	2 months ago
logging.txt	more logging	2 months ago
main.log	Merge branch 'progress' of https://github.com/jiak1/PCServerlist into...	last month

About

No description, website, or topics provided.

Releases

No releases published
Create a new release

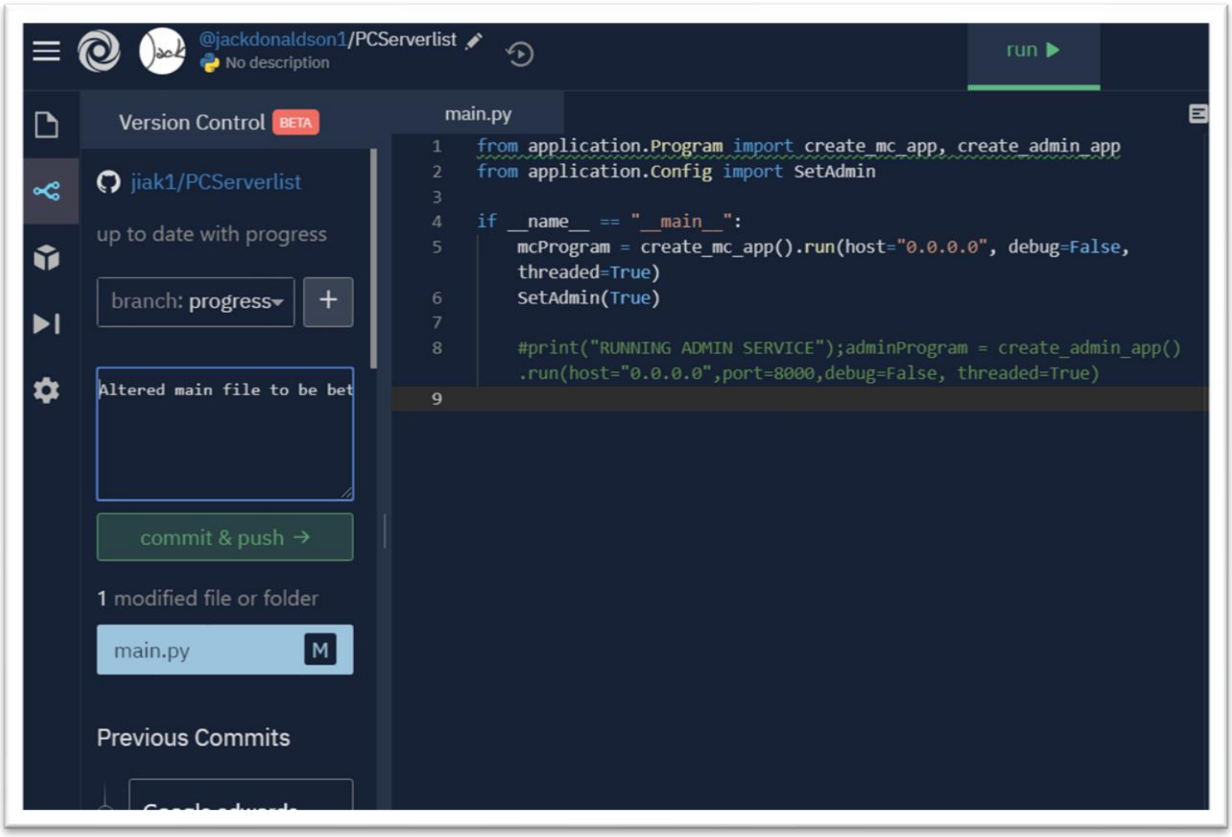
Packages

No packages published
Publish your first package

Languages

● HTML 36.9% ● JavaScript 34.0%
● Python 22.4% ● CSS 6.6%

Pushing (Sending files) to Github is relatively simple as well, it is all done through the command line with a few simple steps (Once git is installed and setup naturally). Just type **git push** and then **git commit –a** and you will be prompted to enter a message with your changes and it will then upload them instantly. Once a change has been pushed I simply remote access into the server using a program called Putty which gives me access to the command line of the server as long as I enter my password correctly. All I have to do is then simply type **sudo git pull** and any changes that it has detected will then be downloaded. Then I just have to restart the GUnicorn service works by typing **sudo systemctl restart adminprogram** or **mcprogram** and it will restart the respective sections of the website, usually with almost 0 delay. However, in order to push my changes to the Github repository from my demo server, I use repl.it's new feature available to subscribers of its hacker plan. As repl.it is an online code editor and runner it will automatically track any changes to files I have made. All I have to do is then simply select the source control tab, type in a commit message and press the commit button. If I want to undo any of my changes I can simply select any commits in my history and rollback to them straight away.



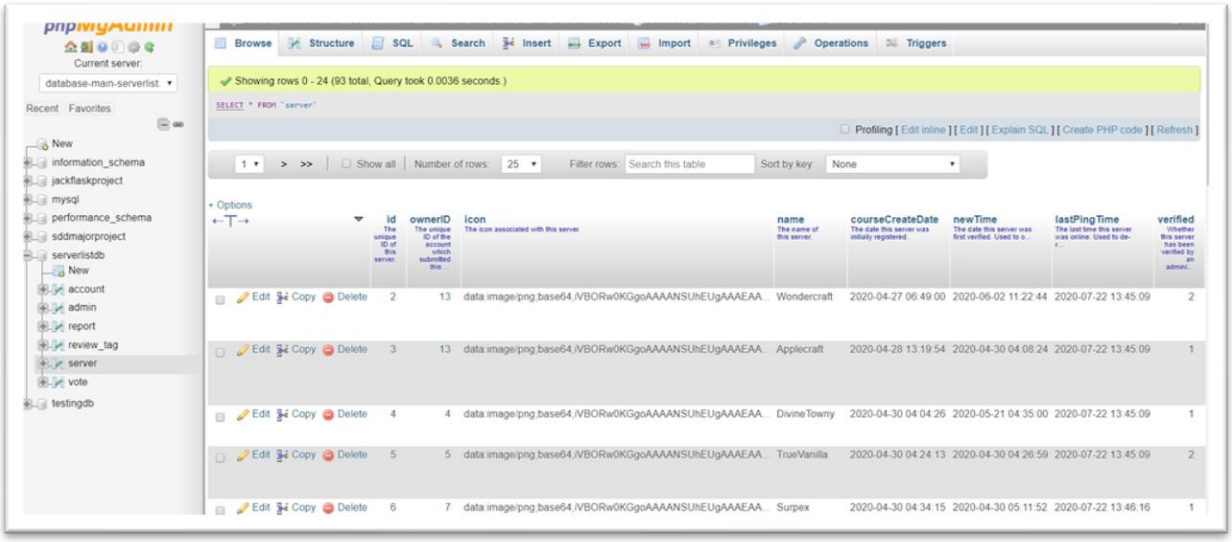
```
from application.Program import create_mc_app, create_admin_app
from application.Config import SetAdmin

if __name__ == "__main__":
    mcProgram = create_mc_app().run(host="0.0.0.0", debug=False,
    threaded=True)
    SetAdmin(True)

    #print("RUNNING ADMIN SERVICE");adminProgram = create_admin_app()
    .run(host="0.0.0.0",port=8000,debug=False, threaded=True)

```

Amazon mainly manages my Relational Database for me meaning I don't have to do too much except run queries on it. However, I prefer to have a way of visualising my database and the default MySQL software that Amazon runs doesn't allow for this, as such I have installed PHPMyAdmin which is a web tool that allows you to visualise a MySQL Database. In this case, I installed the software on the same server that the website is running on allowing the login for PHPMyAdmin to be available at https://db.server-lists.com/phpmyadmin. Once logged in I am then able to visualise all of the data in my database easily and edit it if need be. This also helps when debugging as my demo database is available here as well.



Current server: database-main-serverlist

Recent Favorites

- New
- information_schema
- jackflaskproject
- mysql
- performance_schema
- sddmajorproject
- serverlistdb
  - New
  - account
  - admin
  - report
  - review_tag
  - server
  - vote
- testingdb

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Triggers

✓ Showing rows 0 - 24 (93 total, Query took 0.0036 seconds.)

```
SELECT * FROM `server`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 > >> | Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

+ Options

	id The unique ID of this server.	ownerID The unique ID of the account which submitted this ...	icon The icon associated with this server.	name The name of this server.	courseCreateDate The date this server was initially registered.	newTime The date this server was first verified. Used to o...	lastPingTime The last time this server was online. Used to de-r...	verified Whether this server has been verified by an admini...
Edit Copy Delete	2	13	data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAEAA...	Wondercraft	2020-04-27 06:49:00	2020-06-02 11:22:44	2020-07-22 13:45:09	2
Edit Copy Delete	3	13	data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAEAA...	Applecraft	2020-04-28 13:19:54	2020-04-30 04:08:24	2020-07-22 13:45:09	1
Edit Copy Delete	4	4	data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAEAA...	DivineTowny	2020-04-30 04:04:26	2020-05-21 04:35:00	2020-07-22 13:45:09	1
Edit Copy Delete	5	5	data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAEAA...	TrueVanilla	2020-04-30 04:24:13	2020-04-30 04:26:59	2020-07-22 13:45:09	2
Edit Copy Delete	6	7	data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAEAA...	Surpex	2020-04-30 04:34:15	2020-04-30 05:11:52	2020-07-22 13:46:16	1

Implementation

Conversion Method

Justification

In regards to this project, there are two main conversion methods that could be potentially appropriate, direct and parallel. The main advantages of direct are that it is instantaneous and cheap as well as quick which allows this project to be completed quite quickly and hopefully earlier than expected. However, this has the disadvantage that if any bugs were to appear then I wouldn't have any time to fix them, potentially making a bad first impression on some viewers. In this case, I have decided to go with a more slightly phased approach. This means that I release individual features over time to see how the public reacts to them. This also means that initially there shouldn't be as big a chance for bugs as not all of the site is released. This is particularly well suited to this type of site as it doesn't have much initial traffic and it can perform its function with just a few features initially until I get a chance to add more later on. Often a disadvantage of a phased approach is having to allow both systems to work together but in this case, there isn't an old system to have to worry about as it is a brand new product. In regards to the other two implementation methods, a Parallel approach doesn't necessarily work very well here as there was no previous system so a parallel approach essentially isn't available as it would just become a direct cutover. In regards to a pilot conversion, it is technically feasible to have a skilled group try to test and break the system but in this case, I feel I didn't necessarily have the time to run this detailed of a conversion as the client specifically wanted it to be ready as soon as possible. So although there could be a few more bugs at the beginning because it is brand new and likely to have little traffic, to begin with, it shouldn't necessarily be as big of a problem.

Results

Overall, choosing to implement the website in phases worked rather well, initially I released a pretty barebones version which had a few bugs but eventually there were fewer bugs each time until I had fully released everything that was planned. The main stages I released was adding categories, implementing Votifier votes, image optimisation + reformatting and finally the main SEO change where I added all the relevant tags and descriptions. By the final few stages, I had gotten used to the process of deploying changes and finding any bugs on the demonstration version before they were deployed meaning that the upgrading was generally seamless. Overall the choice of a Phased approach did take more time than a direct one but it was well worth it as my users and client appreciate the seeing how quickly the website developed.

Participant Training

a. *[Provide details of what, who and how you are going to train participants]*

b. *[Provide details of any documentation you will provide participants. Actual documents should be in the Appendix]*

Initial Testing

The main areas of testing that I have conducted on my website are Stress Testing, using both Live and Fake Test Data as well as Range and Acceptance Testing on any forms or data entry points that users can enter into. Alongside this peer testing was also conducted where a Cyber Security expert analysed my code and attempted to break the website. This variety of testing methods will help ensure that when the project launches it shouldn't break and in most case scenarios it should be bug free, in turn increasing the overall user experience and making the site more enjoyable to use.

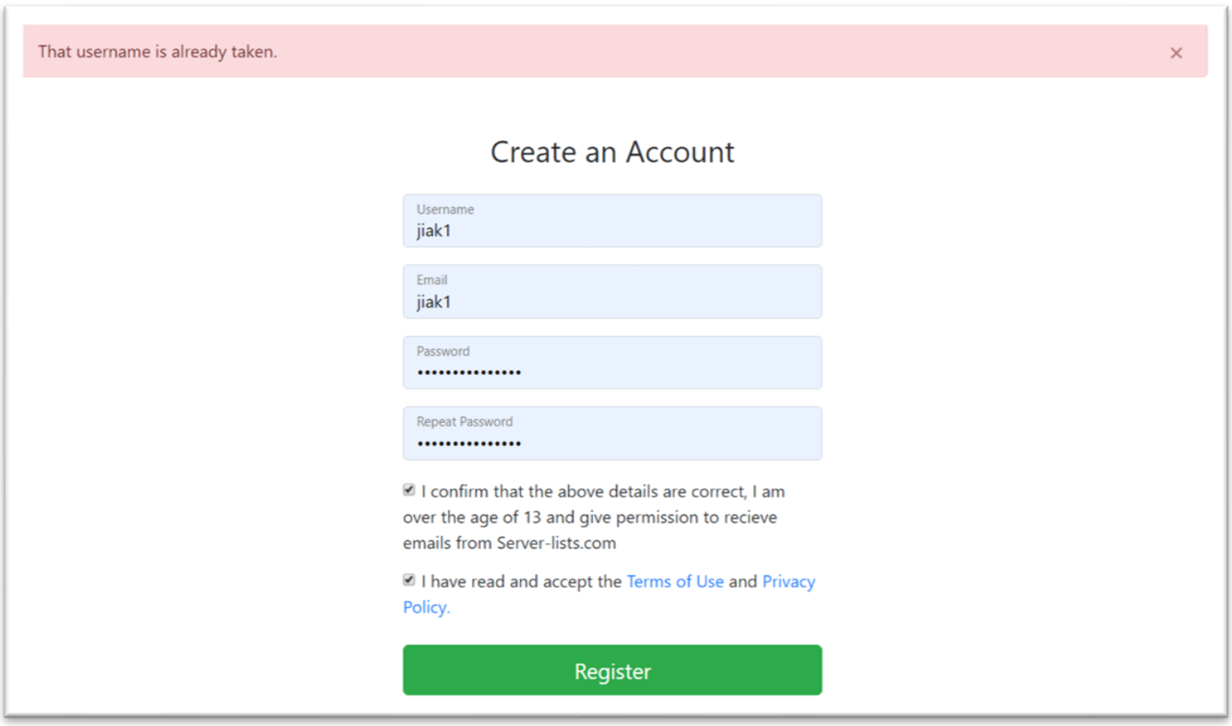
Testing, Evaluation & Maintenance

Testing

The main areas of testing that I have conducted on my website are Stress Testing, using both Live and Fake Test Data as well as Range and Acceptance Testing on any forms or data entry points that users can enter into. Alongside this peer testing was also conducted where a Cyber Security expert analysed my code and attempted to break the website. This variety of testing methods will help ensure that when the project launches it shouldn't break and in most case scenarios it should be bug free, in turn increasing the overall user experience and making the site more enjoyable to use.

To begin with I decided to Stress test my website. One main way that websites can get taken down is with DDOS attacks which are of malicious intent and where a bunch of nodes try to flood a web server with lots of fake packets until it doesn't have enough processing power to handle them all and ends up freezing legitimate users from the site. As a way of mitigating this I secured my website behind Cloudflares free DDOS protection plan. This is a wonderful service that has extremely high security standards and invests in mitigating malicious attacks. The way that this works is rather then routing traffic directly to my web server it first goes through Cloudflares servers. As more than 40% of the webs traffic flows through Cloudflares servers they have developed sophisticated technologies for detecting and outright blocking DDOS attacks with the largest attack ever totalling 1.3 TB/s. This kind of technology helps mitigate any purposeful DDOS attacks against my server but the possibility of having too many real users trying to access the website and crashing it still exists. To mitigate this I have decided to use an inbuilt tool developed by the creators of Flask which when run locally on the server hosting the machine will attempt to see how many requests it can handle before crashing. In my case the website with its current resource allocation will be able to handle about 50,000 concurrent requests every second before experiencing degradation. For reference a concurrent request is sent each time a page is loaded and as users are often doing this every second the theoretical maximum amount of users is more likely around 200,000. As such with the help of Amazon's automatic scaling libraries I have programmed it such that when the CPU or RAM usage of the server reaches higher than 95% for more than 2 minutes it will increase its resource for the next hour. If the traffic slows down it will then release those resources, in turn saving money and becoming more efficient. Hence the theoretical maximum amount of concurrent users on the site could be in the millions relatively easily as long as the server is able to continuously scale up its resources.

Additionally, I ensured that any data entered into forms particularly in regards to adding/setting account or server details was validated properly and acceptable. To do this I used a mix of fake test data that was designed to test edge cases as well as live data entered by other users who may have different ideas to break the system. Examples of test cases that needed to be checked was when creating an account, here the usernames all have to be unique, particularly as they are a primary key in the database. This means that if a username is already used it needs to display an error message as seen below.



That username is already taken.

Create an Account

Username: jiak1

Email: jiak1

Password: •••••••••••••••

Repeat Password: •••••••••••••••

☑ I confirm that the above details are correct, I am over the age of 13 and give permission to recieve emails from Server-lists.com

☑ I have read and accept the Terms of Use and Privacy Policy.

Other examples include ensuring passwords entered are strong or even in scenarios such as setting a Servers port, for this particular field it needs to be entirely numerical as well as greater than 1000 and less than 10000. To combat this there is both client and server side validation, having both is extremely beneficial. Majority of the time client/browser side validation will catch a form before it is submitted meaning the server doesn't even have to both processing the request as it is never sent. However, in the odd case that client side validation doesn't work or someone has a malicious browser there is also the exact same verification server side that will in this case prevent the form from submitting and updating the database. Naturally, during testing there were a few bugs found from this type of testing for example Votifier details didn't get checked because they were optional and not required. These kind of cases are very difficult to pick up without using live test data. Overall the conjunction of live and pre-generated test data worked well in finding bugs, as illustrated above they were mainly in regards to input validation not correctly checking types or sizes properly.

Finally as a precaution I asked a cyber-security friend to go through and review my code. This was quite beneficial because often another point of view from a programmer can show bugs or issues that you wouldn't have thought of. In this case issues appeared when trying to insert raw html into either the reporting section or a server description. This is referred to as an injection attack and means that when it is displayed the HTML will actually run, meaning that anyone can insert their own HTML onto the website and in this case it would often appear in the administrator section. Alongside this he was also able to do a similar trick when emails were sent were it would inject HTML into the body. Overall he didn't find too many issues and the peer check was rather successful in finding those final bugs.

In conclusion the combination of stress testing, form validation checking using live and fake test data as well as getting a peer to review the code was worth it. I do believe that particularly going through these options in this order is best, particularly leaving the peer review to last in order to find the most difficult of bugs rather than ones that would be picked up in the other tests.

Evaluation

[Does the new information system satisfy the purpose/requirements]

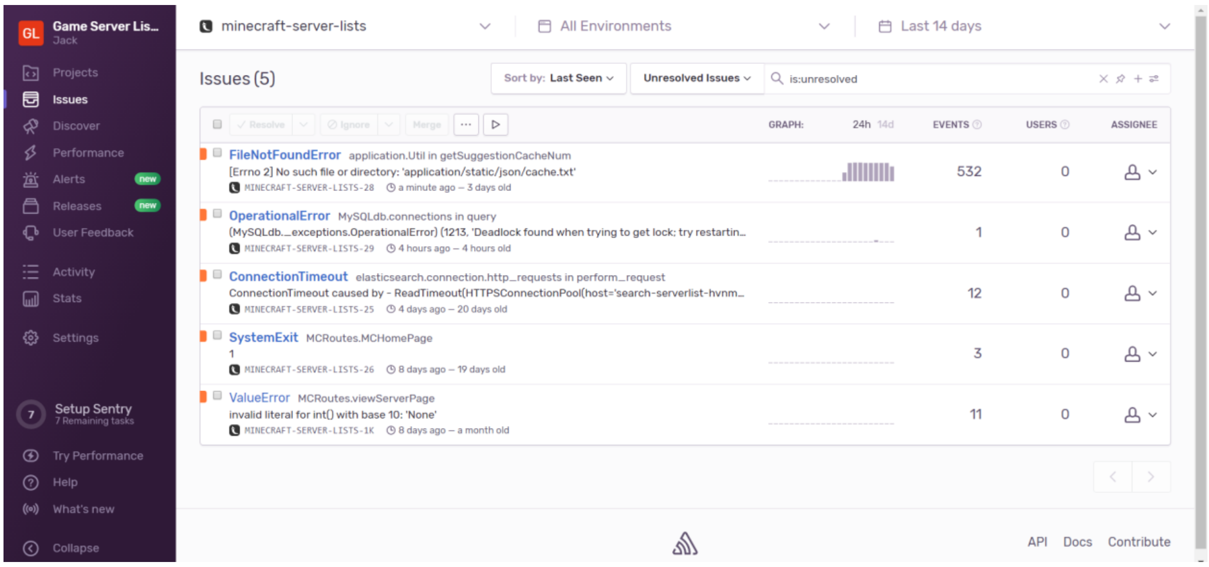
a. [Provide details of what technique you will use to collect this information – survey, interview, observations and measurements]
b. [Summarise the results – actual questions etc should be in the appendix]

Maintenance

Overall the system is very easy to change and a significant amount of it can be done with no coding experience at all. However currently there is some simply steps in the user documentation detailing how to change text on any of the pages if they need minor modifications. Otherwise a simple guide on how to log into the administrator section of the website at https://admin.server-lists.com should suffice in most of the daily ongoing tasks such as reviewing reports, server submissions or tag requests. Most of the systems in this panel are pretty self-explanatory and simple to use as such most users with the ability to access a computer, the internet and the ability to remember their login details should be capable of reviewing the aforementioned features.

In regards to any adjustments of the code, the demo server is always available on repl.it with instructions in the user manual on how to push changes to the server quickly and easily. As such the ability to preview any changes to the code without pushing to the live version is extremely helpful and should make maintaining fairly easy. In conjunction, the easy access of a visual database tool at https://db.server-lists.com makes it easy to view database records and adjust them if need be without programming at all.

If any errors were to occur whilst the server is running then the program Sentry will pick up on it.



Issues (5)

is:unresolved

	GRAPH: 24h 14d	EVENTS	USERS	ASSIGNEE
FileNotFoundError application.Util in getSuggestionCacheNum [Errno 2] No such file or directory: 'application/static/json/cache.txt' MINECRAFT-SERVER-LISTS-28 a minute ago – 3 days old		532	0	
OperationalError MySQLdb.connections in query (MySQLdb._exceptions.OperationalError) (1213, 'Deadlock found when trying to get lock; try restartin... MINECRAFT-SERVER-LISTS-29 4 hours ago – 4 hours old		1	0	
ConnectionTimeout elasticsearch.connection.http_requests in perform_request ConnectionTimeout caused by - ReadTimeout(HTTPSConnectionPool(host='search-serverlist-hvnm... MINECRAFT-SERVER-LISTS-25 4 days ago – 20 days old		12	0	
SystemExit MCRoutes.MCHomePage 1 MINECRAFT-SERVER-LISTS-26 8 days ago – 19 days old		3	0	
ValueError MCRoutes.viewServerPage invalid literal for int() with base 10: 'None' MINECRAFT-SERVER-LISTS-1K 8 days ago – a month old		11	0	

This program has a free tier that will allow up to 1000 errors/month and give detailed stack traces and logs of everything that happened around the time of the error. This is extremely helpful, particularly if you aren't technically minded because it can easily be forwarded to a programmer for fixing rather than just saying "the site is broken" a log is already created and ready to send off/view.

Overall, every effort has been made to ensure that this system is easy to maintain and update with a significant amount of tools and resources to help and problems can usually be fixed by reviewing the User Documentation provided.