MYORPG

Design Document

Team 24
Advisor: Mohammed Selim
Nadine Quibell: Design Lead
Henry Williams: Security Manager
Clayton Surfus: Server Developer
Jonathan Morris: Meeting Scribe

Executive Summary

Development Standards & Practices Used
MYORPG is developed using the Agile software development cycle.

Summary of Requirements
- Players can connect to the MYORPG website through a web browser
- Players can log in or create new accounts
- Gameplay allows players to explore the game world using the WASD keys
- Players can interact live with each other
- Players may chat with each other or the world as a whole using a semi-persistent chat
- Players may attack enemies using equipable weapons
- Players can collect and use items using a persistent inventory
- Players can upload and, with moderator approval, use their own player sprites and weaponry, given the use of an in-game item for creation
- Moderators can approve and suspend player-uploaded items, mute players, and request the banning of a player
- Admins can approve or deny a ban request and delete player-uploaded items, as well as use moderator powers
- Players can create and combine weapons using in-game materials and the Weapons Forge location

Applicable Courses from Iowa State University Curriculum
COMS 309, COMS 319, COMS 363

New Skills/Knowledge acquired that was not taught in courses
- Much of our node.js knowledge was learned for this project or gained independently during other project classes (where it was not required)
- Similarly, much of our knowledge of socketing used in this project was gained independently
- Where/how to host a website and database

Table of Contents

List of figures/tables/symbols/definitions

[1]*MMORPG* - Massively Multiplayer Online Role-Playing Game. A game genre in which players participate in an adventure together on a large scale.
[2]https://mmos.com/news/black-desert-announces-10000-costume-design-contest
[3]*Minimum Viable Product (or MVP)* - A game development term describing a version of the game which implements all key functionality but isn't necessarily fully developed.

1 Introduction

1.1 Acknowledgement

MYORPG has had significant time and effort put into it by all of its members.

1.2 Problem and Project Statement

Problem Statement: This project idea is driven by the lack of online role playing games that integrate their players into gameplay, specifically, allow for user-created content, short of modifications to existing games.

Solution Approach: MYORPG is a browser-based multiplayer online role playing game, also known as an MMORPG[1], which is built around the central concept of user-made content, allowing the player to contribute to the game in the form of their own avatars, weapons, items, monsters, and more, using their own art and stat distributions. Using components collectible in the game, users can submit custom content for moderators or administrators to approve and then be used in the game proper, a unique experience that has gone unutilized in the gaming world so far. The output of this project is a fun, playable multiplayer online role playing game.

1.3 Operational Environment

The MYORPG environment involves multiple users with a web based client connecting to a server. The users are able to access the web page as an HTML file as with any webpage. The web page displays what the users need to play the game, as well as connect to multiplayer. The server itself receives connections from multiple players, facilitates multiplayer, and stores data via a database. While the project was tested on a local network with direct IP connections and use of HTML and javascript files stored on a computer, the end product is accessing the HTML through an online webpage and the server is being connected over the internet. The webpage is optimized for desktop computers and the game's interface requires a mouse and keyboard. There will not be any special runtimes required. The server itself can be a normal computer that runs the server with Node.

1.4 Requirements

As a video game MYORPG contains several key components to make the game playable and a manageable experience. MYORPG contains an account system accessible by all via a signup up and login page that can be accessed repeatedly. Using the login connects the user to the server and is attached to a multiplayer instance. The game itself consolidates several elements in one page. There will be a main game window from which graphics are displayed. There is a chat window which allows a user to chat with others in the same instance of the game. There is an inventory window for managing ingame items and tying into a crafting and shop menu which allows a user to create, purchase, or sell existing items. The game world is controlled via the keyboard, and includes a detailed overworld with grass and buildings, along with dungeons with a more constrained environment. Finally there is also a combat system. Fully implemented combat will effectively be another menu that replaces that of the game world's when prompted. The game supports upwards of eighty players in one instance, multiple user cases, uploading

and rendering of graphics while maintaining an acceptable performance on all connected machines (30 frames per second) and is controllable by users with desktop computers.

1.4.1 Engineering Constraints

Our project requires a server that meets the following specifications in order to run our game smoothly and as intended by the client. Our server currently runs on a single CPU core. The storage requirement can be flexible as most of the content can be stored using the AWS S3 service. The memory requirement can be increased when more data is being stored in the server's memory.

Server Specifications		
CPU Cores	**Memory**	**Storage**
1 vCPU	1GB	40GB SSD

1.4.2 Non-functional Requirements

Non-Functional Requirement	**Outcome**
Frame Rate	A steady 30 FPS is maintained while playing the game.
Security	User information is kept safe. Unintended actions are not possible in the game.
Communication	Multiplayer interaction has acceptable uptime(>=99%) and response time (<1 second)
Zones	Communication and actions that happen in zones are kept to their specific zone.
Compatibility	The game works on web browsers. Each client is identical to other clients.

1.5 Intended Users and Uses

The RPG has three main users, players, moderators, and admins. Players can login, register, play the game, and use items to upload new weapons made with custom images. They have access to the screens associated with these activities. Moderators have all the same privileges that players would. In addition they are able to approve the custom images that players upload, mute players use the chat if they act inappropriately, and request a ban for players that routinely

break the rules. These would be implemented via additional screens and buttons added to the chat window. Admins have all of the privileges of both players and moderators. In addition to this they would be able to ban people and view requested bans through another window. They would have the most privileges out of all user cases as a result. The intended users of the game are anyone with a computer and an interest in playing video games. Moderators and admins are community volunteers who would spend extra time to ensure that the game's community remains fun for all.

1.6 Assumptions and Limitations

Assumptions:

- Our end product is accessed through an internet browser worldwide, where the website is permitted
- Users in the sub-category of 'players' are be able to submit items, dungeons, or other custom content for users in the sub-category of 'mods' or 'admins' to be added to the game at a later date.
- Up to 32 players are be able to interact on one world at once, although more than 32 will be able to play the game at once (just not interact directly)

Limitations:

- Users require an internet browser and internet connection to make use of our end product.
- User need an internet connection to access the product website
- Project schedule determined which features are be available in the initial release of the game

1.7 Expected End Project and Deliverables

A minimum-viable product version of MYORPG will be delivered for beta testing over the summer, in May 2020, with the end of S E 491. This version of the game will be a bare-bones but playable version of the game, allowing players to create accounts, log in, interact, and fight monsters. Item upload will be implemented and regulated, but without in-game balancing.

The end product of this project is MYORPG, a browser-based multiplayer RPG. The product will be available for use free of charge. It will require an internet connection and internet browser to access. Access to the product will also be controlled on a black-list basis by administrators. The game will be completed in December of 2020, with the end of S E 492.

2 Specifications and Analysis

2.1 Proposed Approach

When creating our approach to MYORPG's development, we split the design work into three main facets: the UI, the game code, and the database.

To encourage user participation, we chose a simple, 2D style for MYORPG's UI, where animation could be programmed through code and any uploads would only need to be a single image. We pulled inspiration from early 2000's flash games for the art style-- games that were often created by a single person or a small team-- so that the main game content and uploaded content wouldn't look jarring side-by-side. This also gave the game a nostalgic feel, something we continued to lean into when designing further UI throughout its development.

In terms of the code, MYORPG is developed with a client-side frontend and a web server backend. The focus of the frontend is to display the game and perform any user-centric logic, such as movement and interacting with game objects. The server performs any communication needed to keep all the players up-to-date with each other, including storing basic user information needed by new players joining the room and updating the location of each map's monsters. The frontend and server are written in JavaScript, with the server using node.js, and these two interact with sockets, making use of socket rooms to separate players based on which area of the game they are currently in. Finally, the webpages for the frontend are written in HTML and CSS.

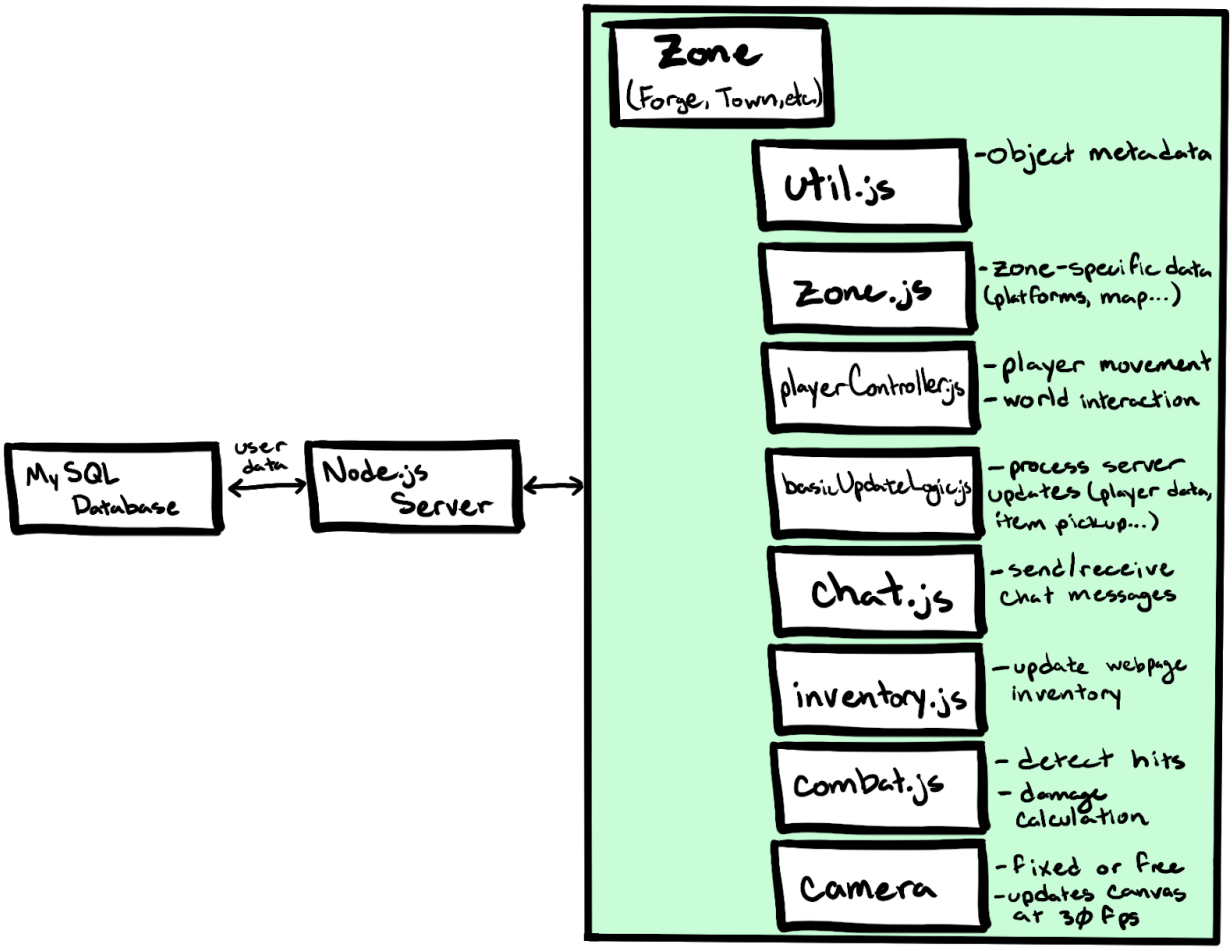
The database is a relational MySQL database, designed to allow flexibility in MYORPG's structure. Relations between items and users and n-to-n, and templates for things such as monsters are stored here, from which instances can be spawned in the server. Since this is the game's persistent storage, anything needing saved between player logins is saved to the database.

We have identified several node libraries to make use of in our project; primarily express and socket.io. The libraries allow us to rapidly develop and prototype certain features of the product, such as multiplayer capability, with ease. The libraries we chose to work with are heavily documented and are the industry standard when working with node.js.

2.2 Design Analysis

MYORPG lent itself very well to a modular design structure for its software architecture. The basic functionalities for zones (sections of the frontend) are split into small, mostly-independent scripts that are then referenced by each zone's main html page. The scripts basicUpdateLogic.js, chat.js, combat.js, cookies.js, inventory.js, and playerController.js, are all independent of each other, provide functionality for their named parts, and are included in every zone. A camera, either freeCameraController.js for larger maps or fixedCameraController.js for small maps, is then attached, and are completely interchangeable with each other. Each zone

also has a named js file that provides some basic, zone-specific information that is then referenced by the zone modules, and util.js is also included, containing basic Javascript object information used by the modules.



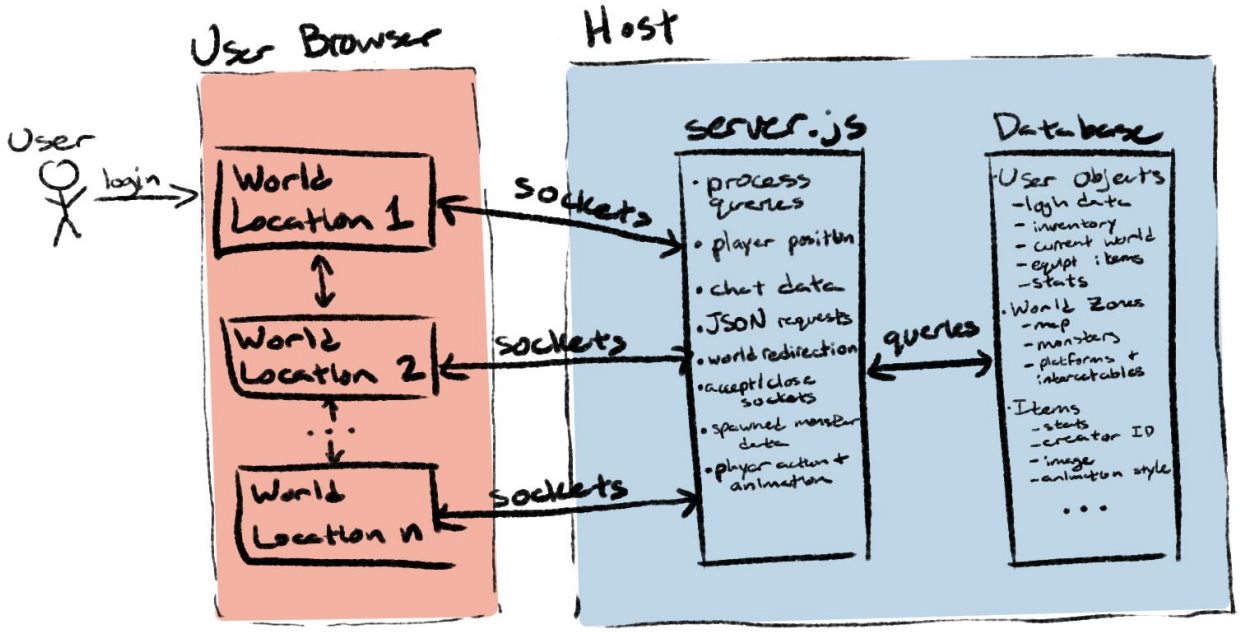
While the node.js server is confined to one file, the data structures on it are separated by zone as the frontend is, and all socket functionality is non-zone-specific.

2.3 Development Process

Our project followed an agile development process. Everyone was assigned a task for the current sprint, a 2 week period, and moved it through our task board until the task was completed. During weekly meetings, group members shared their current progress, and, at the start of the next sprint, were assigned new tasks based on our progress and the project schedule. Once a month, the team planned to meet with the project advisor to share status updates on the current project. This keeps everyone involved with the project informed of where

it is currently at and where it will be heading. Any project blocks that the team comes across will be maneuvered around early. By following the agile development process we helped guide the project to success.

2.4 Conceptual Sketch



There are three main modules of MYORPG: the website, the node.js server server.js, and the mySQL database.

The website, which is what the user connects to and loads via a web browser, consists of a main menu page for login, a number of administrative web pages such as the account page, and a series of world zone pages for each of the game areas the player can explore. The game pages consist of generally the same CSS block and in-game menu blocks, with the map and zone-specific data changing. The zone locations and their functionality is designed to be highly modular.

The server.js file is the node.js server that runs the game from our remote host. It calls basic functionality from module scripts, such as logging in and uploading an object. The main functions of server.js are to connect to the players in various world locations and host their sockets, updating player data to the users at 30 FPS, and to interact with the mySQL database using queries.

The database is the static storage for MYORPG, storing user, item, monster, zone, etc. data. It interacts with server.js using query responses.

3 Statement of Work

3.1 Previous Work and Literature

MYORPG is loosely based on old-school MMORPG games, such as RuneScape, WoW, Guild Wars, Everquest… The basic mechanics of those games and the aesthetic of those early days of online gaming are the foundations of MYORPG. Creating a fantasy fighter, equipping ridiculous weapons, fighting monsters, and dungeon-crawling are not new concepts to the MMORPG genre and are being re-used in this game to incite the nostalgia those things bring with them.

New to the MMORPG genre and much of gaming, however, is MYORPG's concept of adding player-created content to the game as a core feature. Player engagement like this is a step up from forums and the occasional user-create content contests many of these games have [2]. With this, MYORPG hopes to foster a unique, creative community among the MMORPG community.

3.2 Technology Considerations

AWS is scalable enough that serverside burdens are not much of a consideration right now. Client-side, the game should run at 30 frames per second, and is only designed in mind for personal computers rather than for mobile devices. The controls will keep in mind that the user has a full keyboard and a mouse or trackpad available.

3.3 Task Decomposition

Fundamentally, there are three major tasks to decompose for our project: Player Accounts, Multiplayer Functionality, and User Interface. Between the three of them, there are a number of smaller tasks that show up in their decomposition. These tasks are building the player inventory, incorporating submitted cosmetics, the combat system, dungeons, and the separation between different player zones. One primary dependency underlying all of this is the multiplayer server itself. Without a multiplayer server, players will have no way of getting game data for their inventories, cosmetics, combat, dungeons, or even what zone they are in.

3.4 Possible Risks and Risk Management

Right now the main risks would be instability or downtime. AWS is robust enough to have little downtime. However if the client performs poorly or has major technical flaws the game will need to be left online until these issues are addressed. With continuous integration even the flawed version will be kept online, which will require evaluating whether to keep the game online even in the face of major security flaws. To manage this and keep downtime at a low rate only upon major vulnerabilities being discovered should the game go down. Otherwise even in the face of poor performance changes can be made while the game itself stays up.

3.5 Project Proposed Milestones and Evaluation Criteria

Each of the milestones below represent a part of the game system. The system will interact with each other to provide an overall good gaming experience. Only once the client approves of the work done will the milestone be considered completed.

Milestone	**Description/Satisfiability**
Accounts	Users are able to enter their information such as username, email, and password to create an account. They may use the information entered to sign into the game.
Multiplayer	Players can interact with other players and monsters currently connected to the game in real time. Each interaction by a player or monster shows up on each client's browser.
User Interface	Players are able to view and use interfaces to gain knowledge about their player such as viewing items currently in their inventory.
Zones	Zones split up the players into different areas of the game. Players are only able to see actions of the other players in their zone.
Items	Items enhance the player's character in the game. Player's are able to store items in their inventory.
Combat System	Players can fight monsters around the game. The system can use the player's stats and the monsters stats to calculate the damage done and resulting effects.
Dungeons	The game will create dungeons instanced to a player. Each dungeon will generate their own monsters, items, and more. The dungeon can only be used by the player(s) assigned to the dungeon.

3.6 Project Tracking Procedures

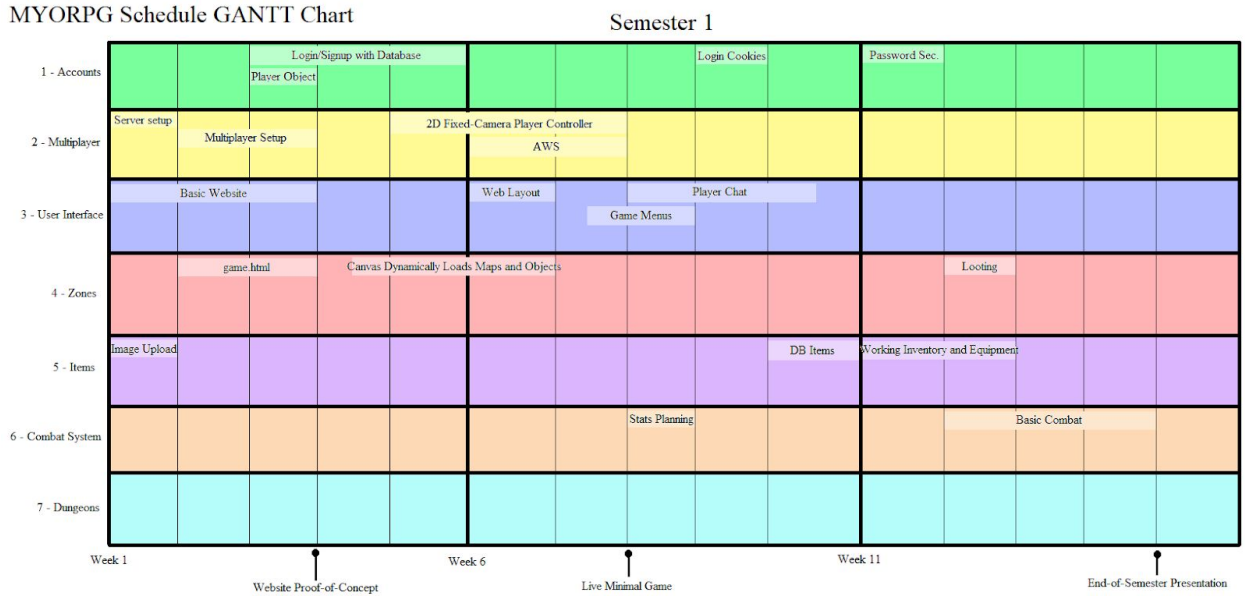
Trello will be used to store a backlog of tasks, current task progress, and completed tasks. Each team member will be assigned a task and update the task card as they make progress. Once the work for the task has been completed, the card should be marked as done.

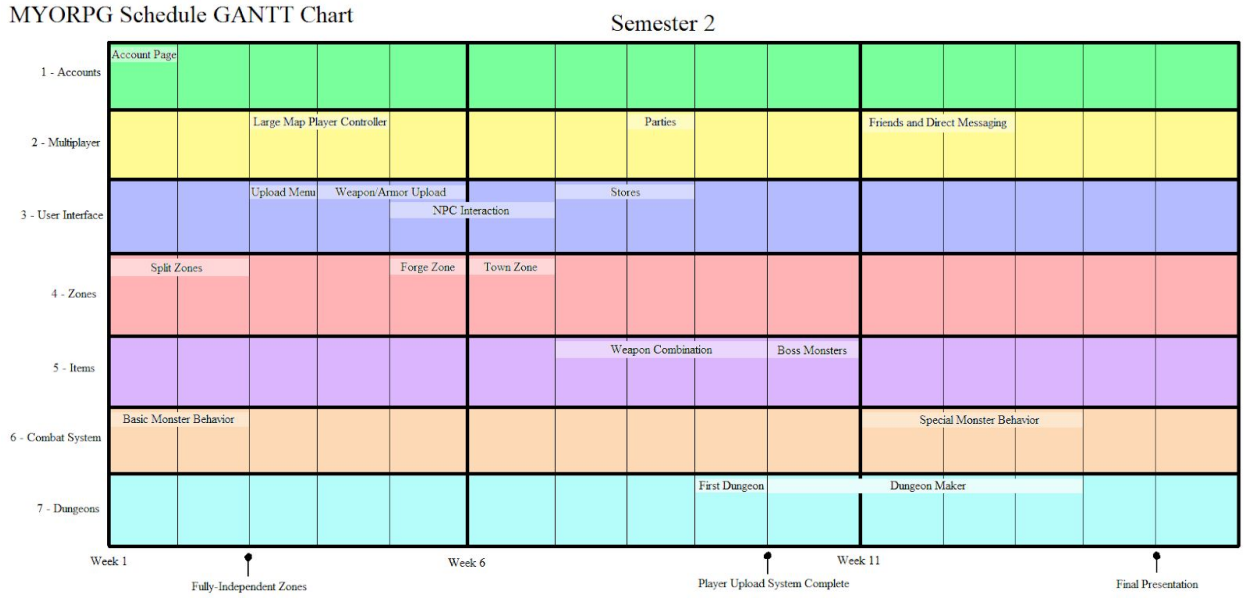
3.7 Expected Results and Validation

Each end product is expected to work alongside the rest of the project and depending on the product, independently. This is validated by running the result with the rest of the master branch on the AWS server and seeing if it works. When this is verified and any changes that need to be made are made, the result is marked as done.

4 Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline





The first semester consisted of completing a minimum viable product for the game that showcases all it can do, if not all to its fullest extent. Players can make accounts, join the game, chat, fight monsters, loot items, and, of course, upload images.

While much of semester one was spent building the game up from the ground and making it a workable product, semester two was focused on adding functionality and really making MYORPG into what it was meant to be. The ability to make custom uploads into usable items, new zones, dungeons, and the forge were the main focus in the second semester. The efforts were more focused in the first weeks on making zones more plug-in-able, after this, the team optimized scheduling by being able to work on these separate pieces concurrently without worrying about them affecting other pieces. The last two weeks of semester two were reserved for schedule changes, testing, and finishing touches. The second semester schedule maintained a similar momentum as semester 1 one as we held each other accountable, and allowed for creative freedom in how it's made, by continuing our twice-weekly team meetings.

4.2 Feasibility Assessment

Our goal was to develop a MMORPG that involves customization allowing for the player to upload their own images to create their own items and characters. Players are able to take the items and use them to fight monsters in the game. Developing the game for the browser allows users to quickly create an account and jump right into the game. This makes the game easily shareable via a link. No downloads will be required by the user in order to play the game. The timeframe of the project is two semesters. The first semester focused on prototyping and building the foundation while the second semester took the foundation and built on it. Getting the challenging tasks out of the way in the first semester allowed us more time to come up with solutions to challenging tasks.

Below are the anticipated challenges that were thought to occur during the development of the project.

Potential Project Challenges	
Challenge	**Description**
Concurrent Player Count	The game needs to handle the max amount of players per server with no noticeable drop in gameplay experience.
Custom Items and Characters	Players are able to upload custom images for items. Additional support will be needed to make sure the items fit the game and don't degrade the experience of the game such as framerate drops or connection issues.
Combat System	Developing the combat system involves coming up with a well thought out experience for the player. The system needs to take the player's and monsters stats into consideration when calculating damage done. Additionally, it will display what is happening on the

	backend to the currently in combat player and other players in the current zone.
Monster Artificial Intelligence	Developing the intelligence for monsters will be a large undertaking as it will need to interact with the players and combat system. Monsters need to know where it can move, what it can attack, and more. Different monster types will have different intelligence as not all monsters move the same way. Additionally, the monsters will need to interact with the combat system as their main purpose is to attack players.

4.3 Personnel Effort Requirements

Below are the projected efforts of tasks that were thought to take a considerable amount of development time. Requiring multiple developers to work through the problem in order to find a solution.

Task	**Personnel**	**Projected Effort**
Database Table Creation	Backend Developer	The table will need to be designed carefully but overall is quick to implement.
User Interface Creation	Game Developer	Depends on the complexity of the component needing the interface.
Sending Data Over Socket	Game Developer Backend Developer	Quick to add data that needs to be sent over a socket.
Creating Unit Test	Game Developer Backend Developer	Depends on the complexity of the component needing the test.
Backend Feature	Backend Developer	Depending on the complexity of the task. The feature may be a small feature that could be implemented quickly. While the feature could be large and requiring weeks of work to be completed.
Canvas Game Feature	Game Developer	Depending on the complexity

		of the task. The feature may be a small feature that could be implemented quickly. While the feature could be large and requiring weeks of work to be completed.

4.4 Other Resource Requirements

Due to the solely software nature of MYORPG, no additional outside resources were needed for constructing the project.

4.5 Financial Requirements

No financial resources were required in the making of MYORPG, as the team produced everything in-home using free software. In order to be hosted to a worldwide audience, however, a web host was needed to host the server and database for the game. MYORPG used Amazon Web Services to host the production server. Additionally, MYORPG used free credits for AWS. In the future, AWS will cost money to be used. In the table below are the costs and services to be utilized. AWS charged hourly based on type and size of instance being used. Our service used an EC2 instance to run our code, S3 for storing custom images uploaded by the users, and RDS for hosting our database. In the future, if MYORPG needed to be scaled, the price would be increased by deploying more instances.

AWS Instance	**Cost (Monthly)**
EC2 Micro (Server)	$0.0116 * 750 Hours = $8.70 [1]
S3 (Storage)	$0.023 * X GB ($0.023 * 50 GB = $1.15) [2]
RDS (Databases)	$0.017 * 750 Hours = $12.75 [3]
Total	$22.60

5 Testing and Implementation

5.1 Interface Specifications

There are no notable hardware/software interfaces involved in our project.

5.2 Hardware and Software

The MySQL database, Jasmine on Node.js, and an internet browser are being used to test the effectiveness of our product.

5.3 Functional Testing

Unit testing is used to test critical parts of our application. As features get added, tests are needed to keep the older set of features working and running as expected. Our project utilizes a javascript test runner called Jasmine. This framework allows us to do both client side and server side unit testing this way. The runner can run in a web browser or via command line. Additionally, the runner can be configured to run tests in a random order. Running tests in random order can help catch uncommon failures. Our routes such as /signin and /signup contain automated unit tests to send POST requests to the route in order to determine if the route is working as intended. Developers can run the automated tests by using the command “npm run tests”.

Functional Requirement	**Test Case**
Register	A user can create an account
Sign In	A user can sign in to the game.
Move	A user can move their character in the game.
Items	Items are spawned into the game. Players are able to pick up items and the items show up in their inventory.
Collision	Players, platforms, and monsters have accurate collision detection. They will not go through each other.

Ultimately, the client will try out the changes once a feature has been implemented. If the developed feature meets their satisfaction then the feature is completed. Furthermore, our application is continuously deployed on AWS from the master branch of our git repository.

5.4 Non-Functional Testing

Keeping performance, security, and compatibility in mind during development is key. Developers test for performance, security, and compatibility during development of their feature. If the requirements are impacted harshly, the feature will need to be reworked.

Non-Functional Requirement	**Outcome**
Frame Rate	A steady 30 FPS is maintained while playing the game.
Security	User information is kept safe. Unintended actions are not possible in the game.
Communication	Players are able to communicate with each other.
Zones	Communication and actions that happen in zones are kept to their specific zone.
Compatibility	The game works on web browsers. Each client is identical to other clients.

5.5 Process

As each branch was made to implement specific features, a trello card was moved from the backlog to being in progress. This was verified to ensure that we were following the agile methodology and our workflow was running smoothly. After commits were pushed to the gitlab page, we could then check the final result via CI/CD on our AWS server. After features were handled by this automated system they manually verified to ensure that everything was working properly. If not, then the feature could receive more work and the process repeated until the AWS server showed that everything was working as intended.

5.6 Results

In the future, our tests will ideally run when a branch is merged with the master branch. This will check to make sure the merged code does not break our functional features. This will be updated second semester to provide the results of our testing.

5.7 Libraries and Code

We used several node libraries in our project. The libraries allowed us to rapidly develop and prototype certain features of the product such as multiplayer with ease. This allowed us to focus on the task at hand instead of spending time reinventing the wheel. Several of the libraries such as socket.io are heavily documented and are the industry standard when working with node.js.

- Socket.IO is a library used for sockets. The library is offered for use on the server-side and client-side. This library simplifies the use of sockets throughout our application.
- Bcrypt is a library to handle hashing text such as passwords. Additionally, the library supports comparing hashes in order to verify the created hash matches the stored hash.
- Express is the web framework that runs the website for our project. The web framework handles routing such as the homepage, sign up page, sign in page, and more. Additionally, it offers more packages to handle user login sessions and POST/GET request parsing.
- Express-Handlebars is a view engine package made specifically to work with express. The view engine utilizes server-side page rendering allowing for us to insert server-side data pulled from the database or user session such as username, items, and more.
- Multer is a package made to handle multipart form-data specifically image uploading. This library allows us to verify MIME types and image size with ease. Additionally, it uploads to the correct directory as specified.
- SendGrid is a third-party service that provides us a mail server to use for delivering our password reset links into our user's mailboxes. We use their sendgrid/mail package with Node to communicate with their service by using an API key provided by them to us.

5.8 Database Structure

Our database is composed of multiple tables to keep track of specific data within our game. We used primary and foreign keys to reference data between tables. Thus, we can modify data in the item table and make changes to other tables referencing that key.

User Table								
varchar (KEY)	varchar	varchar	varchar	int	int	varchar	varchar	int
username	graphic	password	email	handx	handy	weapon	armor	power_level

Items Table				
varchar (KEY)	varchar	int	int	bool
iname	graphic	width	height	approved

Weapons Table				
varchar (KEY)	int	int	int	int

wname	watk	matk	hiltx	hilty

Armor Table		
varchar (KEY)	int	int
aname	wdef	mdef

Inventory Table		
varchar (KEY)	varchar	int
iname	username	num

ResetTokens Table		
varchar (KEY)	varchar	timestamp
token	user	created

Monsters Table									
varchar (KEY)	varchar	int	int	int	int	int	int	int	bool
monsterName	sprite	monsterHeight	monsterWidth	str	dex	inte	wis	vit	approved

Platforms Table			
varchar (KEY)	varchar	int	int
zoneName	graphic	xpos	ypos

Zones Table	
varchar (KEY)	varchar
zoneName	map

Uploads Table	
varchar (KEY)	varchar

filename	username

6 Closing Material

6.1 Conclusion

During the first semester of senior design, the MYORPG development team got decently ahead of the original schedule. Midway through the semester, a modified schedule (appendix 4.1.a) was put into place and a schedule for next semester (appendix 4.1.b) was created. Secure player login, the first game zone, multiplayer functionality including a player controller and chat, live web and database hosting, inventory, equipment, and minimal monsters are the major goals that have been completed so far.

During the second semester, our focus was turned from implementing functions of the game to readying it for future updating and preparing the deliverable: a minimum viable product[3] (MVP) for MYORPG. Primarily, the server was refactored to gracefully allow for connections from multiple game rooms using the socket room functionality. The game UI was redesigned with more reliable object collision, a functioning map-scrolling camera was added, weapons were fully implemented into the game, and monsters were re-added with functional spawning and player interaction. MYORPG's website was further fleshed out with many quality-of-life functions, including an account page, an upload and upload moderation page, and many security features, including end-to-end encryption and a password reset. Additionally, the basic functions of the weapons forge were implemented, allowing players to combine existing weapons into a single, more powerful weapon, a key feature for MYORPG.

Due to the shortened second semester, among other things, caused by COVID-19 measures, our team dropped dungeon implementation in favor of rounding out the existing functions of the game and delivering a clean MVP.

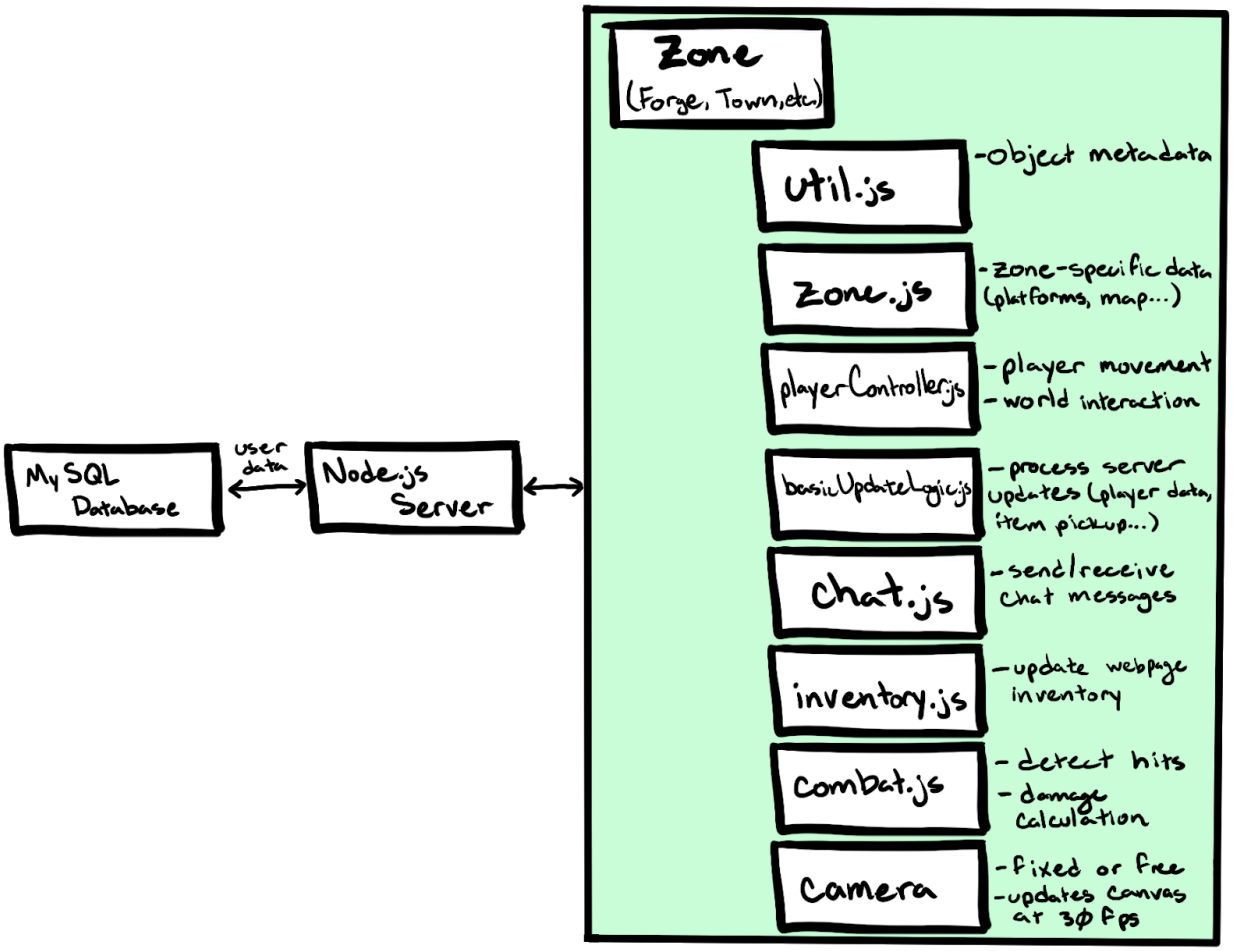
6.2 References

"Economics 2: EC2," *Amazon*. [Online]. Available: https://aws.amazon.com/ec2/pricing/on-demand/. [Accessed: 26-Apr-2020].

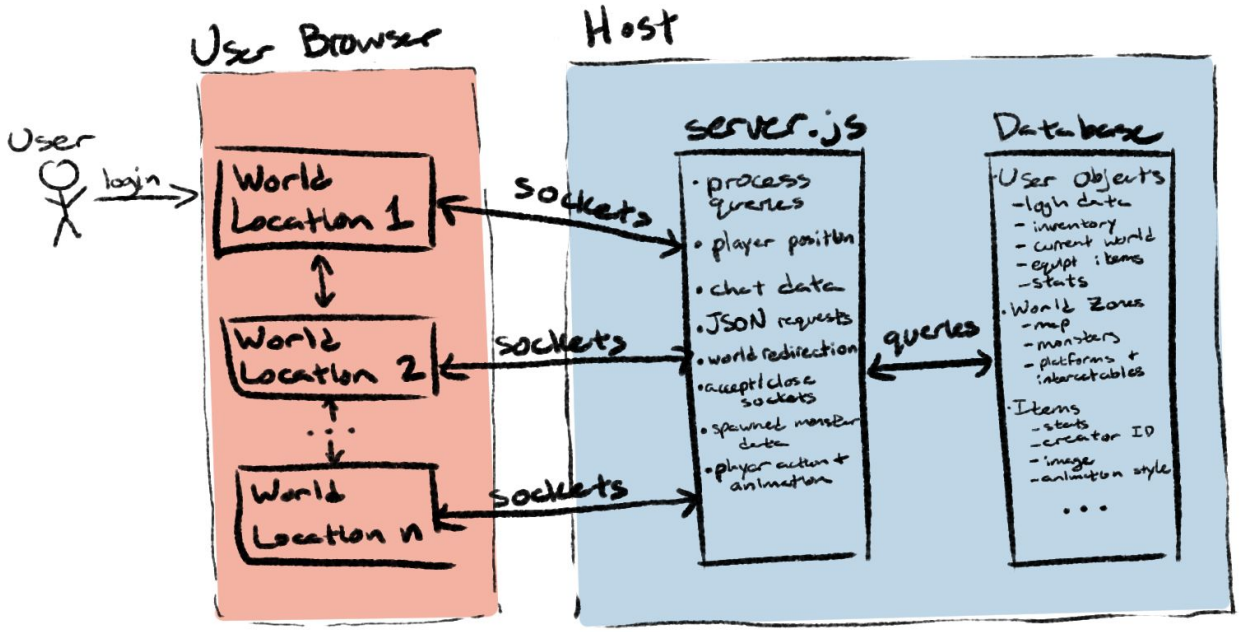
"Amazon S3 pricing," *Amazon*. [Online]. Available: https://aws.amazon.com/s3/pricing/. [Accessed: 26-Apr-2020].

"Amazon RDS for MySQL Pricing," *Amazon*. [Online]. Available: https://aws.amazon.com/rds/mysql/pricing/. [Accessed: 26-Apr-2020].

6.3 Appendices

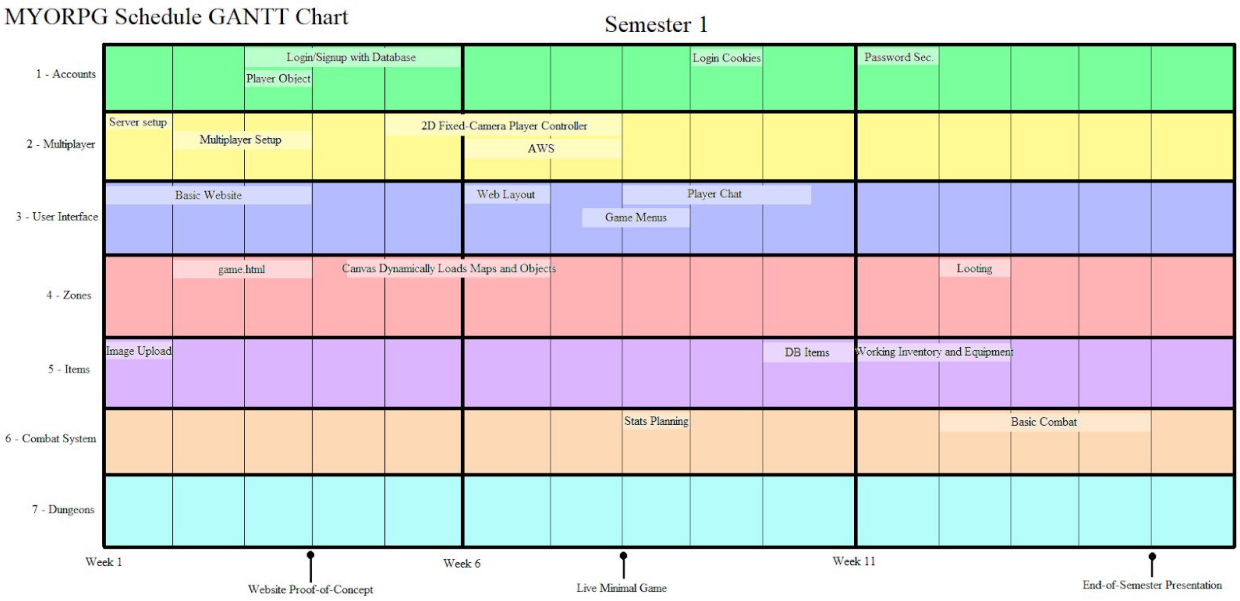

Zone
(Forge, Town, etc.)
util.js
-object metadata
zone.js
-zone-specific data (platforms, map...)
playerController.js
-player movement
-world interaction
basicUpdateLogic.js
-process server updates (player data, item pickup...)
Chat.js
-send/receive chat messages
inventory.js
-update webpage inventory
combat.js
-detect hits
-damage calculation
Camera
-fixed or free
-updates canvas at 30 fps
MySQL Database
user data
Node.js Server

2.2.a

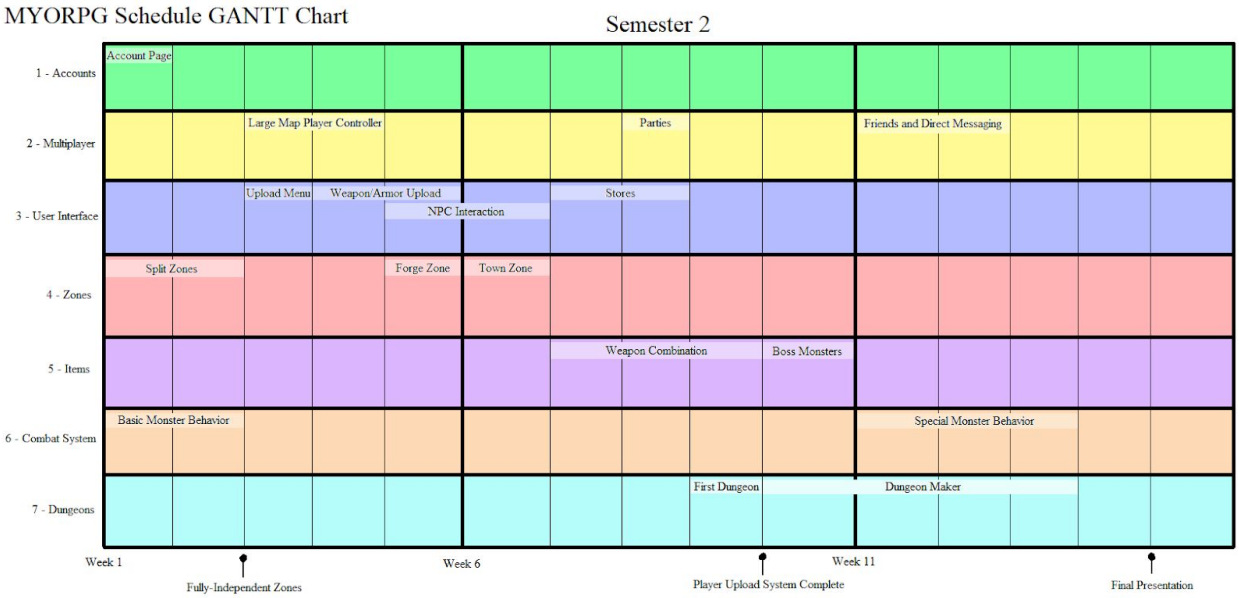

User Browser
Host
User
login
World Location 1
World Location 2
World Location n
sockets
sockets
sockets
server.js
• process queries
• player position
• chat data
• JSON requests
• world redirection
• accept/close sockets
• spawned monster data
• player action + animation
queries
Database
User Objects
-login data
-inventory
-current world
-equipt items
-stats
World Zones
-map
-monsters
-platforms + interactables
Items
-stats
-creator ID
-image
-animation style

2.4.a

GANTT charts for semesters 1 and 2 schedules (from section 4.1):



4.1.a



4.1.b

6.3.1 Operations Manual

MYORPG is managed through AWS. The game is playable with many standard web game controls, with additional instruction for moderation.

6.3.1.1 Server Management

1. Create an EC2 instance via AWS.

2. Create a RDS MySQL server via AWS.
3. Import the queries from “usefulQueries.sql” to create the tables with the correct schema.
4. SSH into the server and Install NodeJS.
5. Clone the repository on the server.
6. Navigate inside the myorpg folder and run “npm install”
7. Navigate to database.js and change the production credentials to match the RDS credentials.
8. Sign up for a SendGrid account and set up the account with your preferred domain name. Additionally, navigate to /routes/user.js and swap out the API key to your API key.
9. Install PM2 which is used to automatically restart the server if it crashes.
10. Run the server by using “pm2 start server.js -- production”.
11. Navigate to the server ip, the homepage should be displayed.

6.3.1.2 Administrative Operation

Content Approval

- Moderators can approve or deny user uploaded content. Navigate to your user profile by clicking the link at the bottom of the game page. Click the 'Approve Content' link, and you will see any monsters, items, or other user uploaded content that is pending approval. Click 'approve' next to an item to approve it, or click 'deny' next to an item to deny it.

User Promotion/Demotion

- Administrators can promote players or moderators to a higher status or demote them to a lower status. Click the 'User Promotion/Demotion' link, and you will see a list of all current users who you have permissions to promote or demote. Click 'promote' next to a user to promote them from Player to Moderator or Moderator to Admin. Click 'demote' next to a user to demote them from a Moderator to a player.

6.3.1.3 User Manual

Sign-Up:

- Navigate to the homepage at http://myorpg.com/ and click 'sign up'. Enter a username, email, and password, and then enter your password again. Click the button labeled 'Sign Up'.

Sign-In:

- Navigate to the homepage at http://myorpg.com/ and click 'sign in'. Enter your username and password in the boxes labeled as such, and then click 'Log In'.
- If you forgot your password, click the 'Forgot Your Password?' link. Enter your username and click 'Reset Password'. This sends an email sent to the email address you signed up with containing a link with further instructions.

Player Character Interaction:

- Press W to jump, A to move left, D to move right.
- Press E to interact with NPCs
- Press F to pick up items.

- Click to attack w/equipped weapon

Chat Usage

- To type and send text via the chat, click into the text box and type. While typing in the chat box, your player character will not receive movement input.
- To send a message into the chat, click the 'send' button or hit the 'enter' key.

Submitting New Content

- From the main game page, click the 'Profile' link at the bottom. From there, click 'Upload Content', and then fill in the boxes what type of content you are uploading. Once done, click 'Upload x', for whatever type you are creating.