Shooty Arena

AQA Computer Science NEA

Contents

Analysis

In this section I will find a problem and investigate it to find find how a new solution might work.

The Problem

There are currently no major arena shooter games without any major problems. This is the problem I am attempting to solve by create a new arena shooter game called Shooty Arena.

Investigating The Competition

In this section I will be investigating the pros and cons of other major arena shooters so I can build up an idea of what I should and should not include in Shooty Arena.

Game	Pros	Cons
Apotheon Arena	Unique style and mechanics.	Connection issues, not cross platform.
Brain / Out	Interesting mechanics, building.	Pay to win, very low time to kill, not cross platform.
Duck Game	High skill cap, great multiplayer.	Latency Issues, not worth the money, not cross platform.
Tee Worlds	Custom maps, many servers.	Complicated level editor, hacking is an issue, high latency.
Warside	Great visuals.	Not available anymore, servers offline, not cross platform.

The Pros

From this investigation I can see that there are many different factors that lead to a good arena shooter game. Some of these include having great multiplayer support as well as having great visuals and mechanics.

The Cons

From this investigation I can see that the biggest and most frequent con about this style of game is that many of them are not cross platform as well as the fact that many of them have server related issues for example very high latency.

The End User

The end user for this game is someone who enjoys to play computer games in their free time. For this reason I will interview a gamer so I can take their feedback on board and use it to create a better solution to the problem stated.

Do you play arena shooter games?

Yes.

How often do you play arena shooter games?

I play them often, usually on weekends.

What do you like about arena shooter games?

I enjoy the action packed gameplay, interesting mechanics and the competitive nature of it.

What do you dislike about arena shooter games?

There's nothing much I don't like about arena shooter games however I don't like waiting, arena shooter games should be action packed. I also don't like the pay to win aspects in certain arena shooter games. I also don't like the negative impact the game servers have on the environment.

What would a new shooter arena game have to have in order for you to play it?

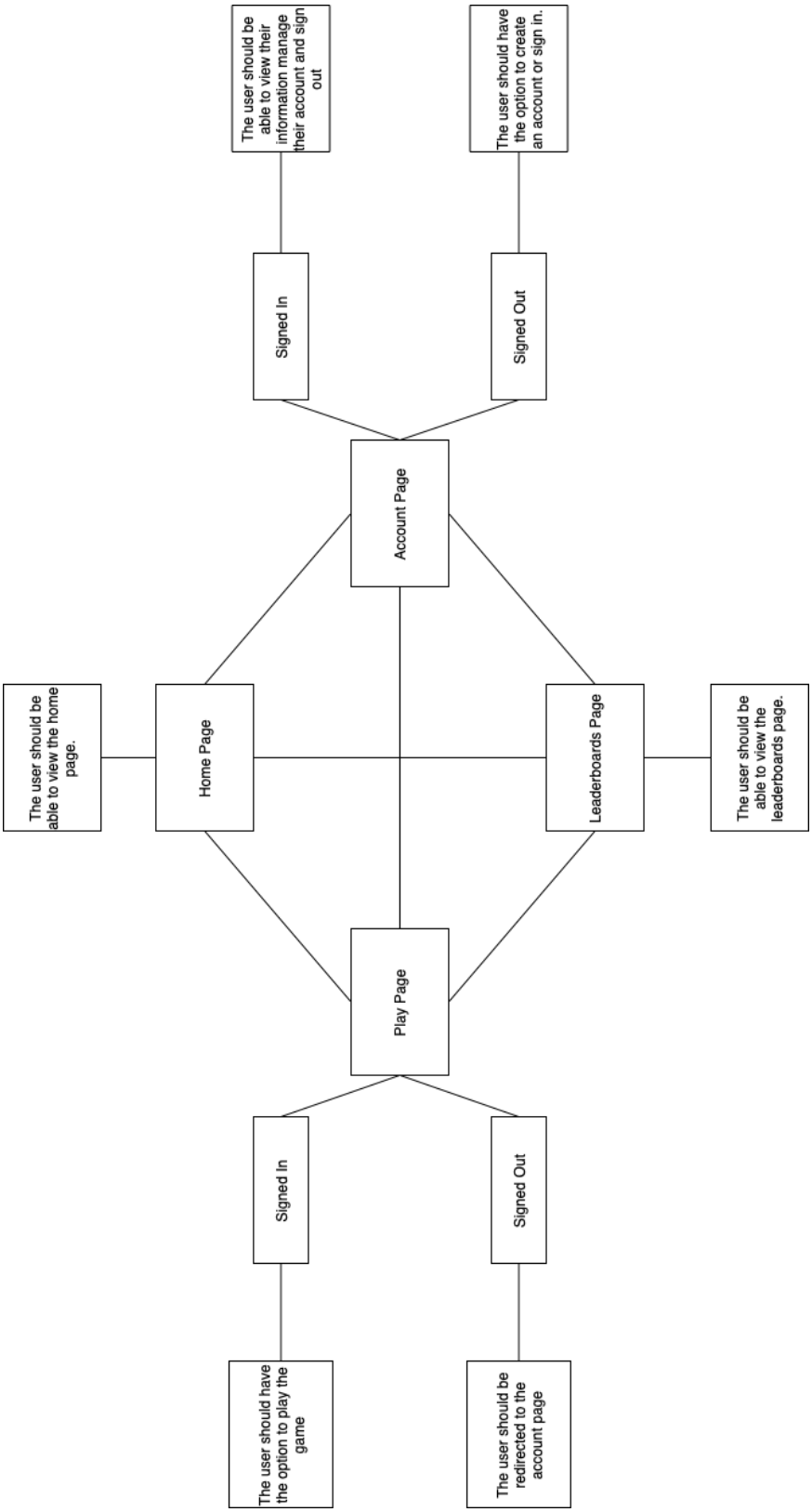
It would have to have a futuristic design, be action packed and have no pay to win aspects.

The New System

Using the data collected in the previous sections I can say that for Shooty Arena to be successful it must be a fun, competitive, cross platform arena shooter with great mechanics and visuals without any server related issues. It must also be secure so that certain malicious users or bots cannot ruin the experience for legitimate users. For this reason Shooty Arena will have a home, account, leaderboards and play page. The home page will just be for information, the account page will be so the user can manage their account, the leaderboards page will be so that users can compete with each other and the play page will be so they can actually play the game.

How Will The New System Work

The diagram below shows the different pages on the website and the basic functionality of each of them.



The Objectives

- Create a template that all pages are based off and make sure that the design is responsive so that all features are accessible at all screen sizes.
  - This template should include a header that contains a navigation bar so that the user can navigate between all of the different pages.
  - This template should also include a footer that contains link to the contact page as well as legal pages.
- Create a home page where all features are accessible at all resolutions.
  - This page should explain what the game is and how it works.
- Create a visually appealing play page where all features are accessible at all resolutions.
  - If the user is not signed in when they visit this page they should be redirected to the account page.
  - There should be an option to join a game.
  - The user should be able to disconnect from the game they are participating in.
  - They should be an area where the game is. This area should become full screen when it is clicked on.
- Create a leaderboards page where all features are accessible at all resolutions.
  - Display the ELO required to reach each rank.
  - Display every player ordered by rank with pagination.
  - Allow the user to order players.
  - Allow the user to click on a player to view more detailed information.
    - Display the users username and rank.
    - Display the users performance and graphs.
- Create an account page where all features are accessible at all resolutions.
  - If the user is not singed in they should be given the option to create an account, login, or reset their password. If the decide to create an account the should be sent a validation email with a link they must visit to activate their account before they can start playing.
  - If the user is signed in they should be able to view their information, username, rank, performance and graphs. They should also have the option of being able to change their password.
- Create the game servers.
  - These should allow the player to connect and play the game. Once the user has finished and has decided to leave the game the information on the leaderboards should be update to display that users new information.
- Make sure that the game is secure.
- Host the required servers so that the game is accessible from anywhere with an internet connection. Ideally use servers powered by renewable energy.

Design

In this section I will research and produce a working, efficient solution to the problem described in the analysis.

The Method

The way I have chosen to solve the problem described in the analysis is to have the game web based. The main reason I have chosen the method is because it means that the game is cross platform and this was a major complaint with other arena shooter games. This method also gives other benefits such as making it easier to implement secure data transfer between servers and making so that the user does not have to download the game. This means that there is no waiting and the user can just play the game which my end user told me is an advantage. To implement a system like this I have chose to create a web application which is run on a web server and a game server which can be run on many other servers. I have chosen a system like this as it means that the solution is highly scaleable.

The Web Application

A web application normally consists of a front end and a back end. The front end consists of static files that are interpreted, run and displayed by the clients web browser whereas the back end consists of an application that is running on a server that receives and sends data. To create Shooty Arena I will need both a font end and back end application as I need to display information to the user who also has to be able to run the game but I also have to be able to interact with a database at perform other tasks which cannot securely be done on the front end. I know I need a font end and a backend but there are multiple ways to get them to communicate with each other.

For example you could have a client side rendered application where an empty HTML file is sent to the client which then uses Javascript to display content. This type of application fetches data from the backend using an API and handles everything itself. An advantage of this type of application is that it offers a great user experience as the website never as to be refreshed however it can lead to slow loading times, poor performance and poor search engine optimisation. On the other hand you could have a server side application where the server handles everything and HTML is rendered on the server before it is sent to the user. This type of application has great search engine optimisation and fast load times but it can lead to a poor user experience.

I want Shooty Arena to offer a great user experience while also being very fast and having great search engine optimisation so I am going to use a hybrid solution. This is where the HTML is rendered on the server and send to the client at which point client side rendering takes over. The best way I found to do this was to use NextJs with NodeJs as it seems to be the most mature and well implemented. It is also production ready. So for the web server I am going to use Javascript and NextJs.

The Game Server

The game server for Shooty Arena must be able to use sockets to communicate in real time with the client. I found an easy and performant way to do this was by using a library called socket-io. Socket-io has great support for Javascript and Python which is why I used Python for the game server. Python also has great web libraries such as flask that will make it much faster and easier to implement a game server.

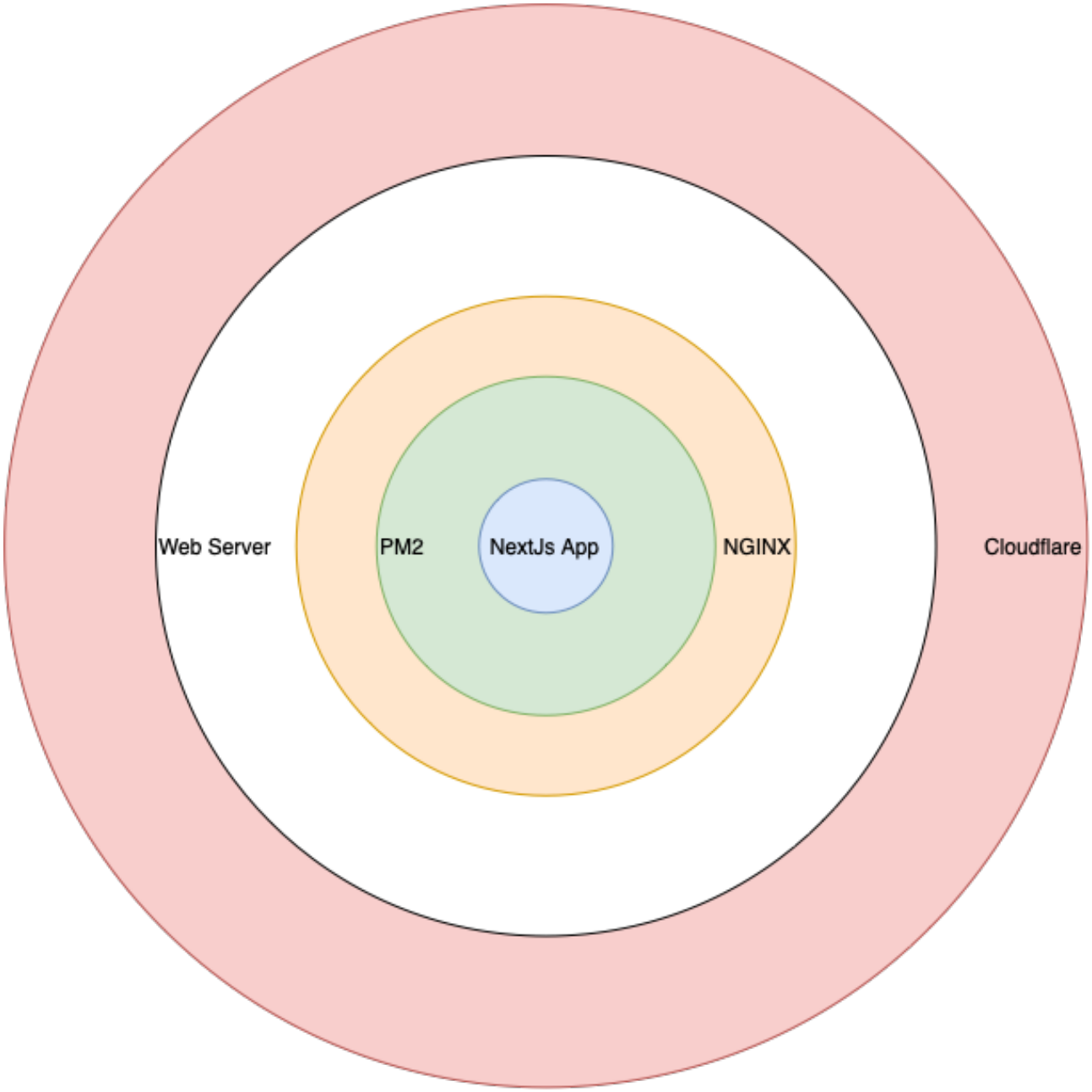
Database

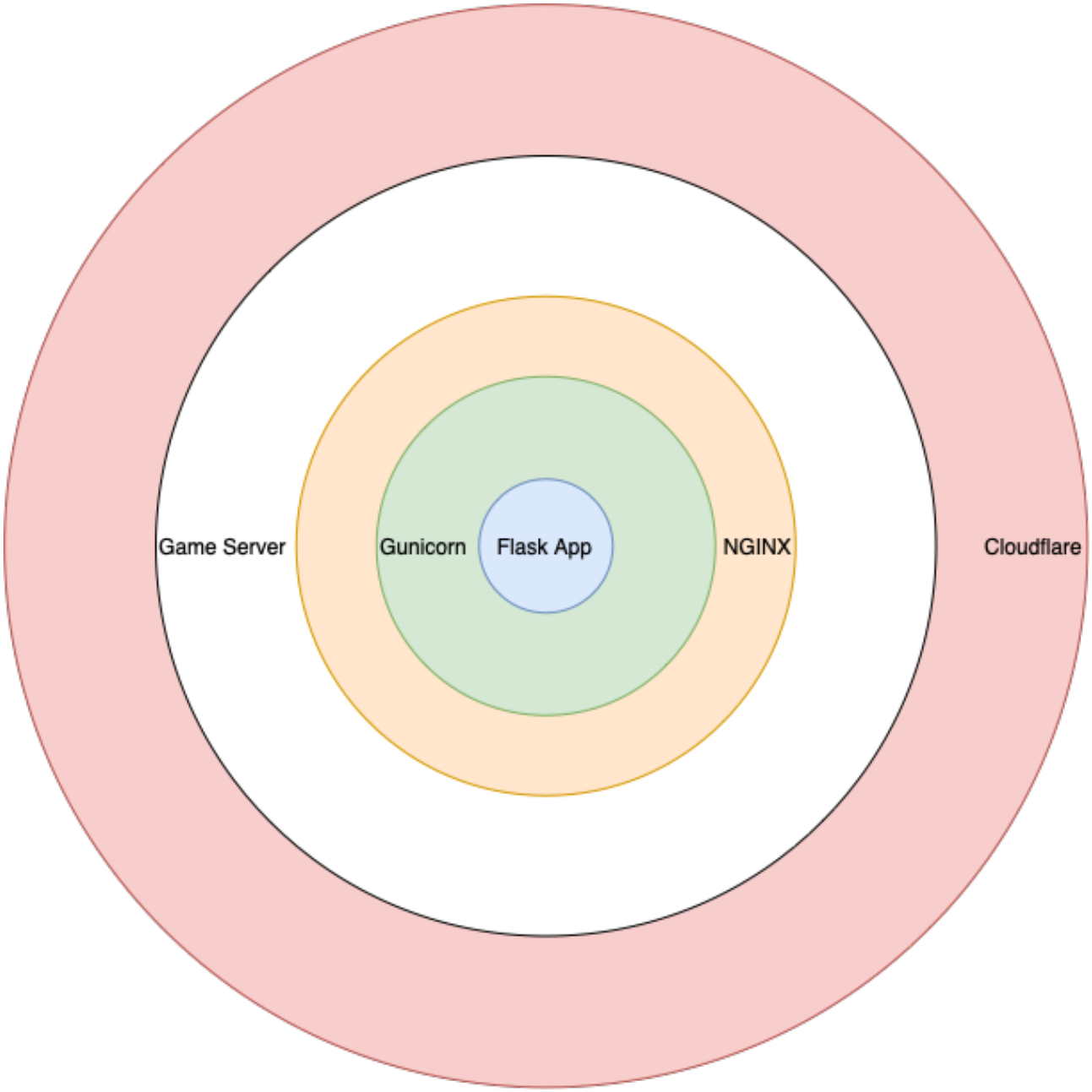
For the web application and game server to be able to operate they need to be able to communicate with a database. Here I have the option of using a relational database such as MySQL or a document-based database-based database such as MongoDB or even a graph based database. I could even use a tool like Redis to speed each one up. I however have chosen to use MySQL to power Shooty Arena due to its security, reliability and ease of use.

Deployment

To deploy the web server and the game server I will use a Linux distribution such as Ubuntu due to the fact that they are secure, reliable and and cost effective. To host the web application I will use NGINX as a reverse proxy to the NextJs application that will be run using PM2. I will also setup the MySQL server to be on the same machine. I will also deploy the game server on a Linux machine and I will also use NGINX on this machine however I will use Gunicorn to run the web application rather than PM2. All traffic from both servers will also be routed through Cloudflare for extra security and performance as well SSL certificates.

How The Servers Look

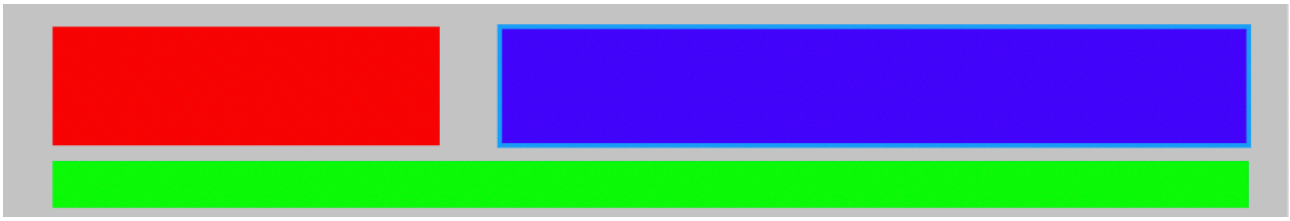



Game Server
Gunicorn
Flask App
NGINX
Cloudflare

User Interface

The header should at the top of every page. The grey rectangle is the entire header. The red rectangle is the title that can be clicked to return to the home page and the blue rectangle should be the navigation links to to the other pages. The title should have a constant effect whereas the links should only have a visual effect when you hover over them. On small screens the blue rectangle should wrap around so its under their red rectangle. Then each of them can get wider.

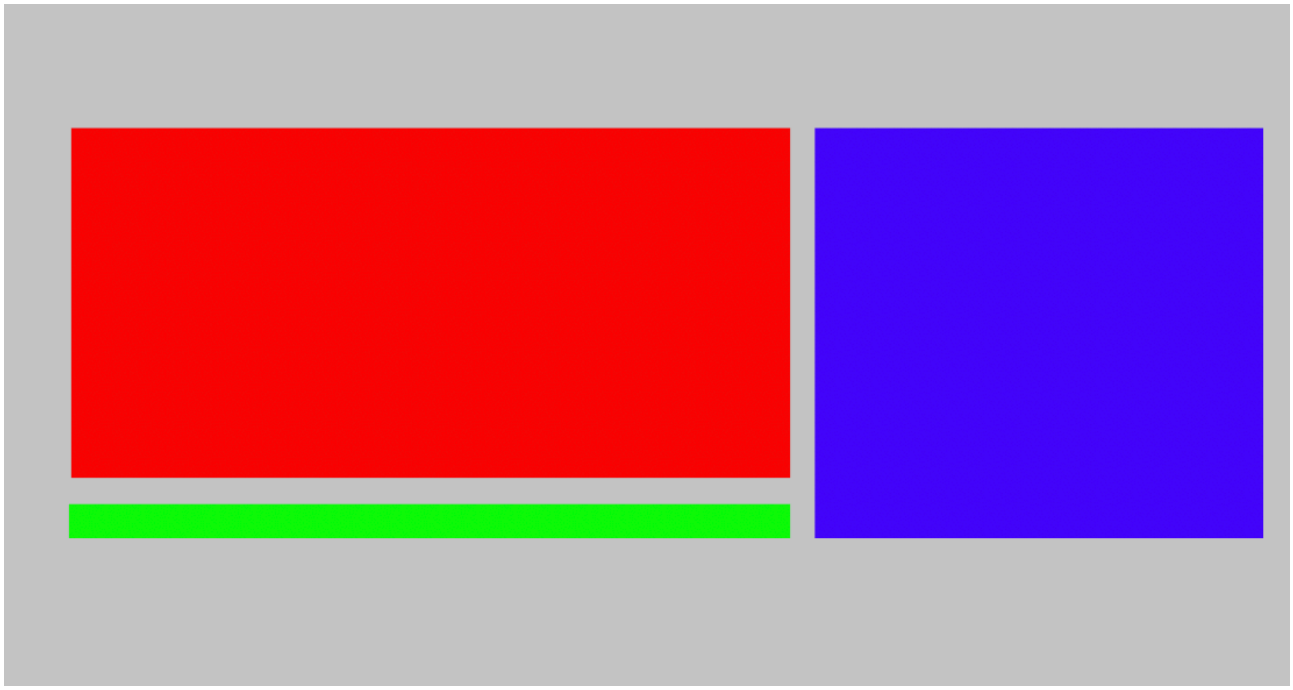
The footer should be at the bottom of every page. They red rectangle should be replaced with text mentioning about how the servers are run on renewable energy and it should have a constant visual effect. The blue rectangle should be replaced with other navigation links which also have a visual effect when you hover over them and the green rectangle should be replaced with a copyright message.



The title should be under the header on each page. The red rectangle should be replaced with the name of the page and the text should have an effect like the text in the red rectangle in the footer.

On the home and play pages the grey rectangle should contain the content for that page. On the home page it will be some text talking about the game and on the play page it will be filled up with the game window. This should be displayed under the title.

On the leaderboards page this should be displayed under the title. The blue rectangle should be replaced with a table of the different ranks and the ELO required to reach those ranks and the red rectangle should be replaced with a table of up to five users in order of the rank (highest to lowest). The green rectangle should be replaced with buttons that can be used to navigate between different pages of users.



Technical Solution

In this section I will create a solution for the problem described the the analysis.

Project Structure

This project is split into two main folders. One for the game server and the other for the web server. Below is the directory tree for the project containing all of the files I have created. The blank directories contain files that were auto-generated by other programs so I will not include the contents of them in my code. The same thing applies to package-lock.json. I will also not include the content of any file that is not text based.

```
game
| venv
| __init__.py
| generateTileMap.py
web
| .next
| authentication
| | cookie
| | | cookie.js
| | jsonwebtoken
| | | jsonwebtoken.js
| | authentication.js
| components
| | footer
| | | copyrightCard
| | | | copyrightCard.jsx
| | | | copyrightCard.module.css
| | | linkCard
| | | | linkCard.jsx
| | | | linkCard.module.css
| | | renewableCard
| | | | renewableCard.jsx
| | | | renewableCard.module.css
| | | footer.jsx
| | | footer.module.css
| | header
| | | navigationCard
| | | | navigationCard.jsx
| | | | navigationCard.module.css
| | | titleCard
| | | | titleCard.jsx
| | | | titleCard.module.css
| | | header.jsx
| | | header.module.css
| | title
| | | title.jsx
| | | title.module.css
| cryptography
| | encrypt
| | | encrypt.js
| | hash
| | | hash.js
| | cryptography.js
| database
| | database.js
| mail
| | send
```

```
| | | send.js
| | validate
| | | validate.js
| | mail.js
| node_modules
| pages
| | activate
| | | [activate].js
| | api
| | | getStats
| | | | [getStats].js
| | | createAccount.js
| | | registerServer.js
| | | serverIp.js
| | | signIn.js
| | | signOut.js
| | | updateStats.js
| | leaderboards
| | | [page].js
| | _app.js
| | 404.js
| | account.js
| | index.js
| | play.js
| public
| | backgrounds
| | | background.png
| | | backgroundLight.png
| | fonts
| | | NeonSans.ttf
| | players
| | | gunLeft.png
| | | gunRight.png
| | | left.png
| | | right.png
| | sounds
| | | die.mp3
| | | jump.mp3
| | | land.mp3
| | | shoot.mp3
| | tiles
| | | corner.png
| | | flat.png
| | favicon.ico
| styles
| | 404.module.css
| | account.module.css
| | fonts.css
| | index.module.css
| | leaderboards.module.css
| | play.module.css
| | reset.css
| next.config.js
| package-lock.json
| package.json
```

__init__.py

The purpose of this file is to run the game server.

```python
# Imports
from engineio.payload import Payload
from AesEverywhere import aes256
import flask_socketio
import threading
import requests
import random
import pygame
import flask
import time
import math

# Config
Payload.max_decode_packets = 500
webserver_address = "http://localhost:3000"
cryptography_password = 'password'
clock = pygame.time.Clock()
```

This code shows the start of a complex client server model.

```
tile_map = [[8, 80, 80, 160, 160], [1, 160, 80, 240, 160], [1, 240, 80, 320, 160
], [1, 320, 80, 400, 160], [1, 400, 80, 480, 160], [1, 480, 80, 560, 160], [1, 56
0, 80, 640, 160], [1, 640, 80, 720, 160], [1, 720, 80, 800, 160], [1, 800, 80, 88
0, 160], [1, 880, 80, 960, 160], [1, 960, 80, 1040, 160], [1, 1040, 80, 1120, 160
], [1, 1120, 80, 1200, 160], [1, 1200, 80, 1280, 160], [1, 1280, 80, 1360, 160],
[1, 1360, 80, 1440, 160], [1, 1440, 80, 1520, 160], [1, 1520, 80, 1600, 160], [1,
1600, 80, 1680, 160], [1, 1680, 80, 1760, 160], [1, 1760, 80, 1840, 160], [1, 184
0, 80, 1920, 160], [1, 1920, 80, 2000, 160], [1, 2000, 80, 2080, 160], [2, 2080,
80, 2160, 160], [7, 80, 160, 160, 240], [3, 2080, 160, 2160, 240], [7, 80, 240, 1
60, 320], [3, 2080, 240, 2160, 320], [7, 80, 320, 160, 400], [3, 2080, 320, 2160,
400], [7, 80, 400, 160, 480], [3, 2080, 400, 2160, 480], [7, 80, 480, 160, 560],
[5, 160, 480, 240, 560], [5, 240, 480, 320, 560], [5, 320, 480, 400, 560], [5, 40
0, 480, 480, 560], [5, 480, 480, 560, 560], [3, 2080, 480, 2160, 560], [7, 80, 56
0, 160, 640], [5, 880, 560, 960, 640], [5, 1520, 560, 1600, 640], [5, 1600, 560,
1680, 640], [5, 1680, 560, 1760, 640], [5, 1760, 560, 1840, 640], [3, 2080, 560,
2160, 640], [7, 80, 640, 160, 720], [3, 2080, 640, 2160, 720], [7, 80, 720, 160,
800], [5, 800, 720, 880, 800], [5, 1200, 720, 1280, 800], [3, 2080, 720, 2160, 80
0], [7, 80, 800, 160, 880], [3, 2080, 800, 2160, 880], [7, 80, 880, 160, 960], [3
, 560, 880, 640, 960], [5, 1040, 880, 1120, 960], [5, 1360, 880, 1440, 960], [3,
2080, 880, 2160, 960], [7, 80, 960, 160, 1040], [3, 560, 960, 640, 1040], [5, 128
0, 960, 1360, 1040], [5, 1440, 960, 1520, 1040], [5, 1520, 960, 1600, 1040], [5,
1600, 960, 1680, 1040], [5, 1680, 960, 1760, 1040], [3, 2080, 960, 2160, 1040], [
7, 80, 1040, 160, 1120], [5, 480, 1040, 560, 1120], [5, 1760, 1040, 1840, 1120],
[3, 2080, 1040, 2160, 1120], [7, 80, 1120, 160, 1200], [5, 1120, 1120, 1200, 1200
], [3, 2080, 1120, 2160, 1200], [6, 80, 1200, 160, 1280], [5, 160, 1200, 240, 128
0], [5, 240, 1200, 320, 1280], [5, 320, 1200, 400, 1280], [5, 400, 1200, 480, 128
0], [5, 480, 1200, 560, 1280], [5, 560, 1200, 640, 1280], [5, 640, 1200, 720, 128
0], [5, 720, 1200, 800, 1280], [5, 800, 1200, 880, 1280], [5, 880, 1200, 960, 128
0], [5, 960, 1200, 1040, 1280], [5, 1040, 1200, 1120, 1280], [5, 1120, 1200, 1200
, 1280], [5, 1200, 1200, 1280, 1280], [5, 1280, 1200, 1360, 1280], [5, 1360, 1200
, 1440, 1280], [5, 1440, 1200, 1520, 1280], [5, 1520, 1200, 1600, 1280], [5, 1600
, 1200, 1680, 1280], [5, 1680, 1200, 1760, 1280], [5, 1760, 1200, 1840, 1280], [5
, 1840, 1200, 1920, 1280], [5, 1920, 1200, 2000, 1280], [5, 2000, 1200, 2080, 128
0], [4, 2080, 1200, 2160, 1280]]
users = 0
user_positions = [0, 0, 0, 0, 0, 0]
tiles = []
bullets = []
health = []
directions = []

```

This code shows the use of two dimensional arrays.

```python
# Cryptography
class Cryptography ():
    # Encrypt
    @staticmethod
    def encrypt ( data, password ):
        return aes256.encrypt(data, password)
    # Decrypt
    @staticmethod
    def decrypt ( data, password ):
        return aes256.decrypt(data, password)

# Register Server
requests.post(webserver_address + "/api/registerServer", data = Cryptography.encrypt(f"{requests.get('https://api.ipify.org').text},6", cryptography_password))

```

This code shows the use of a candidate written class and shows more of the complex client server model.

```python
# Get Bullet Vector
def getBulletVector ( playerX, playerY, clickX, clickY ):
    return [ clickX - playerX, clickY - playerY ]

# Get Bullet Normalisation Value
def getBulletNormalisationValue ( distanceX, distanceY ):
    return 1 / math.sqrt((distanceX ** 2) + (distanceY ** 2))

# Get Normalised Bullet Vector
def getNormalisedBulletVector ( bulletVector, bulletNormalisationValue ):
    return [ bulletVector[0] * bulletNormalisationValue, bulletVector[1] * bulletNormalisationValue ]

# Creates Tiles
for tile in tile_map:
    tiles.append(pygame.Rect(tile[1], tile[2],80, 80))

```

This code shows the use of mathematical calculations and shows more two dimensional arrays and classes.

```python
# User
class User ():
    # Constructor
    def __init__ ( self, socketId, user, gameRank, elo, kills, deaths, player ):
        self.socketId = socketId
        self.user = user
        self.gameRank = gameRank
        self.elo = elo
        self.kills = kills
        self.deaths = deaths
        self.new_kills = 0;
        self.new_deaths = 0;
        self.angle = 0;
        self.inputs = []
        self.player = player

```

This code shows some more object oriented programming.

```python
# Player
class Player(object):
    # Constructor
    def __init__ ( self ):
        self.reset(random.randint(160, 1760), 160)
        self.game_over = False
        self.image_index = 0
```

This code shows even more object oriented programming.

```python
# Update
 def update( self, inputs, tiles, dt ):
    # Move
    dx = 0
    dy = 0
    # Jump
    self.landed = False
    if self.jl == True:
        self.jl = False
    if inputs[0] and self.jumped == False and self.in_air == False:
        self.vel_y = -22
        self.jumped = True
        self.jl = True
        self.touchfloor = False
    elif inputs[0] == False:
        self.jumped = False
    # Left
    if inputs[1]:
        dx -= 7 * dt
        self.direction = "left"
    # Right
    if inputs[3]:
        dx += 7 *dt
        self.direction = "right"
    # Gravity
    self.vel_y += 1 * dt
    if self.vel_y > 10:
        self.vel_y = 10 *dt
    dy += self.vel_y
    # Collisions
    self.in_air = True
    for tile in tiles:
        # X and Y
        if tile.colliderect(self.rect.x + dx, self.rect.y, self.width, self.height):
            if dx >= 0:
                dx = tile.left - self.rect.right
            elif dx < 0:
                dx = tile.right - self.rect.left
    # Direction
    if dx < 0:
        self.dir = -1
    elif dx > 0:
        self.dir = 1
    # Move
    self.rect.x += dx
    for tile in tiles:
        # Vertical Collisions
        if tile.colliderect(self.rect.x, self.rect.y + dy, self.width, self.height):
            if self.vel_y >= 0:
                dy = tile.top - self.rect.bottom
                self.in_air = False
                if self.touchfloor == False:
                    self.touchfloor = True
                    self.landed = True
            elif self.vel_y < 0:
                dy = tile.bottom - self.rect.top
            self.vel_y = 0
    # Update Cords
    self.ht = dy
    self.rect.y += dy
    # Return Cords
    return self.rect.x, self.rect.y
```

This code shows more object oriented programming and mathematical calculations.

```python
# Reset
def reset(self, x,y):
    self.width = 80
    self.height = 80
    self.rect = pygame.Rect(x,y,80,80)
    self.direction = 0
    self.vel_y = 0
    self.jumped = False
    self.in_air = True
    self.health = 8
    self.dir = 1
    self.touchfloor = False
    self.jl = False
    self.landed = False
    self.ht = 0

```

This code also shows more object oriented programming.

```python
# App
app = flask.Flask(__name__)
socketio = flask_socketio.SocketIO(app, cors_allowed_origins = webserver_address)

# Socketio Authentication
@socketio.on("authentication")
def authentication(code):
    code = str(Cryptography.decrypt(code.split("ShootyArenaGame=")[1], cryptography_password))[2:-1]
    code = code.split(",")
    for user in users:
        if user.user == code[0]:
            socketio.emit("disc", to = flask.request.sid)
            return
    users.append(User(flask.request.sid, code[0], code[1], int(code[2]), int(code[3]), int(code[4]), Player()))
    socketio.emit("map", tile_map)
    socketio.emit("player", code[0], to = flask.request.sid)

# Move
@socketio.on("move")
def move(movement):
    for user in users:
        if flask.request.sid == user.socketId:
            user.inputs = movement

# MMove
@socketio.on("mmove")
def mmove(angle):
    for user in users:
        if flask.request.sid == user.socketId:
            try:
                user.angle = int(angle["angle"])
            except:
                user.angle = 0

```

This code shows some more of the complex client server model as mell as some more interactions with two dimensional arrays.

```python
# Click
@socketio.on("click")
def cli(inputs):
    for user in users:
        if flask.request.sid == user.socketId:
            vec = [inputs["dx"], inputs["dy"]]
            nvec = getBulletNormalisationValue(vec[0], vec[1])
            fvec = getNormalisedBulletVector(vec, nvec)
            inputs["px"] += fvec[0] * 72
            inputs["py"] += fvec[1] * 72
            hit = False
            for tile in tile_map:
                if (inputs["px"] + 5 > tile[1] and inputs["px"] + 5 < tile[3]) and (inputs["py"] + 5 > tile[2] and inputs["py"] + 5 < tile[4]):
                    hit = True
            if hit == False:
                bullets.append([user.user, inputs["px"], inputs["py"], fvec[0], fvec[1]])

```

This code shows more of the client server model and includes more mathematical calculations.

```python
# Game Loop
last_time = time.time()
def game (last_time):
    while True:
        dt = time.time() - last_time
        dt *= 60
        last_time = time.time()
        health = []
        # Bullets
        for bulletnum, bullet in enumerate(bullets):
            bullet[1] += bullet[3] * 12 * dt
            bullet[2] += bullet[4] * 12 * dt
            for tile in tile_map:
                if (bullet[1] + 5 > tile[1] and bullet[1] + 5 < tile[3]) and (bullet[2] + 5 > tile[2] and bullet[2] + 5 < tile[4]):
                    del bullets[bulletnum]
            # Users
            for user in users:
                if (bullet[1] + 5 > user.player.rect.x and bullet[1] +5 < user.player.rect.x + 80) and (bullet[2] +5 > user.player.rect.y and bullet[2] +5 < user.player.rect.y + 80):
                    if bullet[0] != user.user:
                        del bullets[bulletnum]
                        user.player.health -= 1
                        if user.player.health == 0:
                            user.new_deaths += 1
                            socketio.emit("die", to = user.socketId)
                            user.player.reset(random.randint(160, 1760), 160)
                            for user2 in users:
                                if bullet[0] == user2.user:
                                    user2.new_kills += 1
        directions = []
        angles = []
        # Users
        for usernum, user in enumerate(users):
            if len(user.inputs) != 0:
                user_positions[usernum] = [user.user, user.player.update(user.inputs, tiles, dt)]
                if user.player.jl == True:
                    socketio.emit("ju", to = user.socketId)
                if user.player.landed == True:
                    socketio.emit("la", to = user.socketId)
                directions.append([user.user, user.player.dir])
                health.append([user.user, user.player.health])
                angles.append([user.user, user.angle])
        socketio.emit("users", user_positions)
        socketio.emit("directions", directions)
        socketio.emit("health", health)
        socketio.emit("bullets", bullets)
        socketio.emit("ang", angles)
        clock.tick(60)

```

This code also shows more of the client server model and it also includes the use of two dimensional arrays and also more mathematical calculations.

```python
# Run Game Loop
thread = threading.Thread(target=game, args = (last_time,))
thread.start()

```

This code shows multithreading.

```python
# Factorial
def factorial ( number ):
    if number == 0:
        return 1
    else:
        return number * factorial(number - 1)

# Calculate Elo Change
def calculateEloChange ( kills, deaths ):
    sc = kills - deaths
    if sc >= 2:
        return factorial(sc) // factorial(sc - 2)
    else:
        return ( kills - deaths ) * 10

```

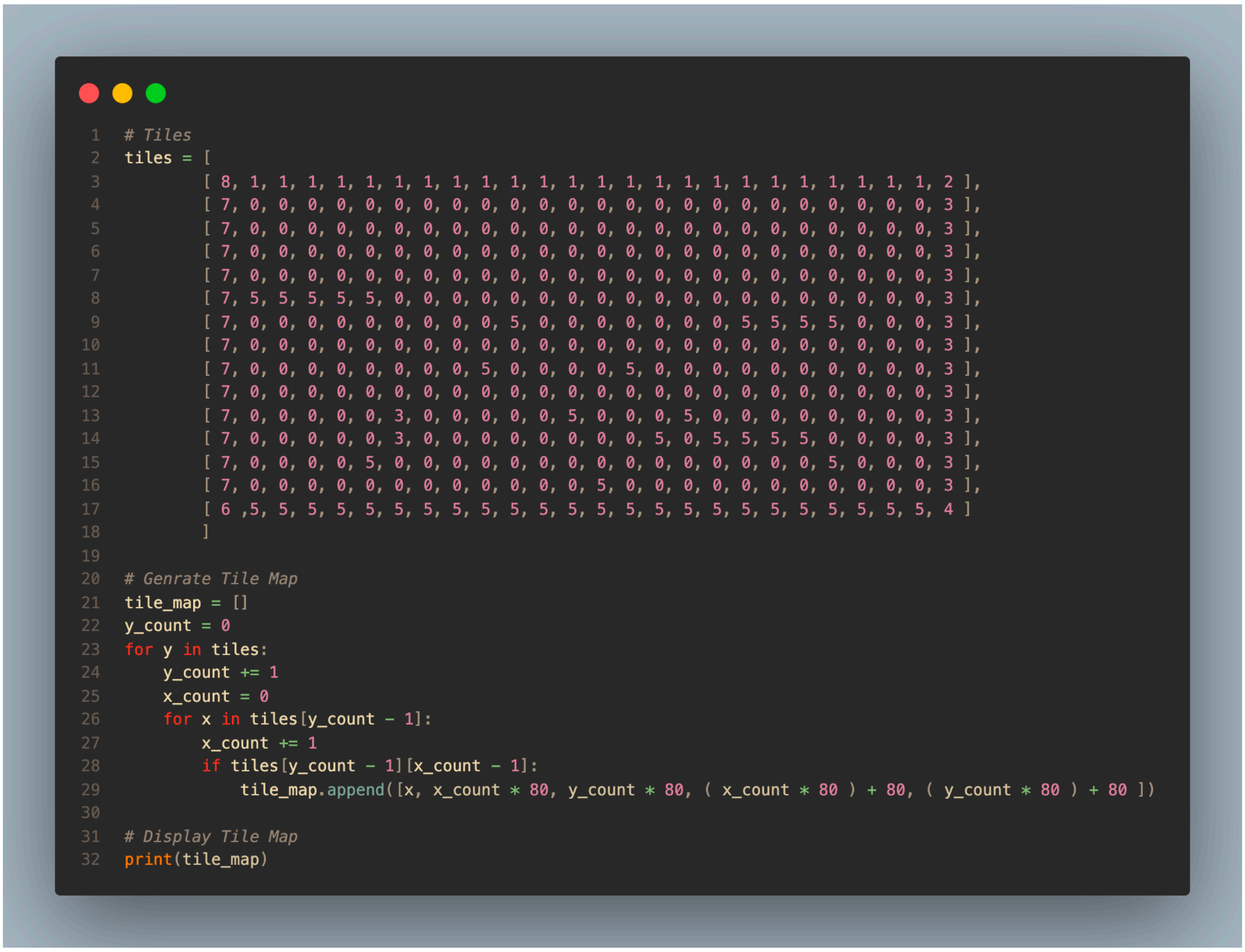
This code shows the use of recursion.

```python

# Calculate Rank
def calculateRank ( elo ):
    if elo < 0:
        return "Unranked"
    elif elo < 1000:
        return "Shitty Shooty"
    elif elo < 2000:
        return "Cutey Shooty"
    elif elo < 3000:
        return "Beauty Shooty"
    else:
        return "Snooty Shooty"

# Dissconnect
@socketio.on("disconnect")
def disconnect ():
    for usernum, user in enumerate(users):
        if flask.request.sid == user.socketId:
            data = aes256.encrypt(f"{user.user},{user.elo+calculateEloChange(user.new_kills, user.new_deaths)},{calculateRank(user.elo + calculateEloChange(user.new_kills, user.new_deaths))},{user.kills + user.new_kills},{user.deaths + user.new_deaths}" , cryptography_password)
            requests.post(f"{webserver_address}/api/updateStats", data = data)
            del users[usernum]
            user_positions[usernum] = 0
            print("dis")

# Run Socketio
if __name__ == "__main__":
    socketio.run(app, debug = True)
```

This code shows the use of more mathematical calculations and more of the client server model.

generateTileMap.py

The purpose of this file is to generate a tile map which then can be used by __init__.py to generate the game world.



```
# Tiles
tiles = [
        [ 8, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2 ],
        [ 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3 ],
        [ 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3 ],
        [ 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3 ],
        [ 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3 ],
        [ 7, 5, 5, 5, 5, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3 ],
        [ 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 5, 5, 5, 5, 0, 0, 0, 3 ],
        [ 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3 ],
        [ 7, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3 ],
        [ 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3 ],
        [ 7, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 5, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 3 ],
        [ 7, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 5, 5, 5, 5, 0, 0, 0, 0, 3 ],
        [ 7, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 3 ],
        [ 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3 ],
        [ 6 ,5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4 ]
        ]

# Genrate Tile Map
tile_map = []
y_count = 0
for y in tiles:
    y_count += 1
    x_count = 0
    for x in tiles[y_count - 1]:
        x_count += 1
        if tiles[y_count - 1][x_count - 1]:
            tile_map.append([x, x_count * 80, y_count * 80, ( x_count * 80 ) + 80, ( y_count * 80 ) + 80 ])

# Display Tile Map
print(tile_map)
```

This code shows the use of two dimensional arrays and more mathematical calculations.

cookie.js

The purpose of this file is to export the functions required for the web server to be able to set and get cookies.

```javascript
// Strict Mode
"use strict";

// Imports
import cookies from "cookies";

// Set
function set ( req, res, name, data, httpOnly ) {
    new cookies(req, res).set(name, data, {
        secure: false,
        httpOnly: httpOnly,
        sameSite: true
    });
};

// Get
function get ( req, res, name ) {
    return new cookies(req, res).get(name);
};

// Exports
module.exports = {
    set,
    get
};
```

This code shows more of the client server model.

jsonwebtoken.js

The purpose of this file to export the functions required to allow the web server to be able to create and read json web tokens.

```
// Strict Mode
"use strict";

// Imports
import jsonwebtoken from "jsonwebtoken";

// Set
function set ( data, key ) {
    return JSON.stringify(jsonwebtoken.sign({ data: data }, key, { expiresIn: "30m" }));
};

// Get
function get ( data, key ) {
    return new Promise ((resolve) => {
        jsonwebtoken.verify(JSON.parse(data), key, (error, data) => {
            if (error) resolve(false);
            resolve(data);
        });
    });
};

// Exports
module.exports = {
    set,
    get
};
```

This file shows more of the client server model and shows one part of the server and client communicate.

authentication.js

The purpose of this file is to export the functions necessary to authenticate the user.

```javascript
// Strict Mode
"use strict";

// Imports
import cookie from "./cookie/cookie.js";
import jsonwebtoken from "./jsonwebtoken/jsonwebtoken.js";

// Key
const JsonwebtokenKey = "DdQ7xxgZnCpZazJxJ3Mkb0ZrciDjmobG";

// Set User
async function setUser ( req, res, data ) {
    cookie.set(req, res, "key", jsonwebtoken.set(data, JsonwebtokenKey), true);
};

// Get User
async function getUser ( req, res ) {
    return new Promise (async (resolve) => {
        const userCookie = cookie.get(req, res, "ShootyArenaUser");
        if (userCookie === undefined) {
            setUser(req, res, "");
            resolve("");
        } else {
            const user = await jsonwebtoken.get(userCookie, JsonwebtokenKey);
            if (user === false || user.data === "") {
                setUser(req, res, "");
                resolve("");
            } else {
                setUser(req, res, user.data);
                resolve(user.data);
            };
        };
    });
};

// Set Game
async function setGame ( req, res, data ) {
    cookie.set(req, res, "ShootyArenaGame", data, false);
};

// Exports
module.exports = {
    setUser,
    getUser,
    setGame
};
```

This file shows more about the client server model and how data is transmitted.

copyrightCard.jsx

The purpose of this file is to export a react component that is displayed on the web app.

```
// Strict Mode
"use strict";

// Styles
import styles from "./copyrightCard.module.css";

// Copyright Card
export default function CopyrightCard () {
    return (
        <div className = { styles.copyrightCard }>
            <h6>Copyright - Shooty Arena</h6>
        </div>
    );
};
```

copyrightCard.module.css

The purpose of this file is to contain the css used be the copyrightCard react component.

```css
/* Copyright Card */
.copyrightCard {
    width: 100vw;
    padding-bottom: 20px;
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: #1a1a1a;
}

/* Copyright Card H6 */
.copyrightCard h6 {
    font-family: NeonSans;
    font-size: 10px;
    color: #f700ff;
    text-shadow: 0 0 10px #f700ff, 0 0 20px #f700ff, 0 0 30px #f700ff, 0 0 40px #f700ff;
}
```

linkCard.jsx

The purpose of this file is to export a react component that is displayed on the site.

```
// Strict Mode
"use strict";

// Styles
import styles from "./linkCard.module.css";

// Link Card
export default function LinkCard ({ title, links }) {
    return (
        <div className = { styles.linkCard }>
            <h5>{ title }</h5>
            {
                links.map(link => <h5><a href = { link[1] }>{ link[0] }</a></h5>)
            }
        </div>
    );
};
```

linkCard.module.css

The purpose of this file is to contain the css used by the linkCard component.

```css
/* Link Card */
.linkCard {
    width: 200px;
    height: 100px;
    padding: 30px;
    background-color: #1a1a1a;
}

/* Link Card H5 */
.linkCard h5 {
    width: 100%;
    padding-top: 12px;
    font-family: NeonSans;
    font-size: 12px;
    color: #ff9d00;
    text-shadow: 0 0 10px #ff9d00, 0 0 20px #ff9d00, 0 0 30px #ff9d00, 0 0 40px #ff9d00;
    transition: text-shadow 0.2s linear;
}

/* Link Card H5 Hover */
.linkCard h5:hover {
    text-shadow: 0 0 10px #ff9d00, 0 0 20px #ff9d00;
}
/* Link Card H5 First Child */
.linkCard h5:first-child {
    padding-top: 0;
    font-size: 18px;
    color: #8c00ff;
    text-shadow: 0 0 10px #8c00ff, 0 0 20px #8c00ff, 0 0 30px #8c00ff, 0 0 40px #8c00ff;
}

/* Link Card A */
.linkCard a {
    color: #ff9d00;
}

/* Small Screens */
@media screen and (max-width: 779px) {
    /* Legal Card H5 */
    .linkCard h5 {
        text-align: center;
    }
}
```

renewableCard.jsx

The purpose of this file is to export a react component which is displayed on the site.

```jsx
// Strict Mode
"use strict";

// Styles
import styles from "./renewableCard.module.css";

// Renewable Card
export default function RenewableCard () {
    return (
        <div className = { styles.renewableCard }>
            <h4>Powered by renewable energy.</h4>
        </div>
    );
};
```

renewableCard.module.css

The purpose of this file is to contain the css used by the renewableCard component.

```css
/* Renewable Card Text Flicker */
@keyframes renewableCardTextFlicker {
    30%, 32%, 34%, 80% {
        text-shadow: 0 0 10px #00ff2a, 0 0 20px #00ff2a, 0 0 30px #00ff2a, 0 0 40px #00ff2a;
    }
    0%, 31%, 33%, 35%, 81% {
        text-shadow: 0 0 10px #00ff2a, 0 0 20px #00ff2a;
    }
}

/* Renewable Card */
.renewableCard {
    width: 200px;
    padding: 30px;
    background-color: #1a1a1a;
}

/* Renewable Card H4 */
.renewableCard h4 {
    width: 100%;
    height: 100%;
    text-align: center;
    font-family: NeonSans;
    font-size: 30px;
    color: #00ff2a;
    text-shadow: 0 0 10px #00ff2a, 0 0 20px #00ff2a, 0 0 30px #00ff2a, 0 0 40px #00ff2a;
    animation: renewableCardTextFlicker 6s infinite;
}
```

footer.jsx

The purpose of this file is to export a react component which is displayed on the site.

```jsx
// Strict Mode
"use strict";

// Styles
import styles from "./footer.module.css";

// Components
import CopyrightCard from "./copyrightCard/copyrightCard.jsx";
import LinkCard from "./linkCard/linkCard";
import RenewableCard from "./renewableCard/renewableCard.jsx";

// Footer
export default function Footer () {
    return (
        <div className = { styles.footer }>
            <div>
                <RenewableCard/>
                <LinkCard title = "Legal" links = { [["Privacy Policy", "./privacypolicy"], ["Terms and Conditions", "./termsandconditions"], ["Accessibility Statement", "./accessibilitystatement"]] }/>
                <LinkCard title = "Other" links = { [["Contact", "./contact"]] }/>
            </div>
            <CopyrightCard/>
        </div>
    );
};
```

footer.module.css

The purpose of this file is to export a react component that is displayed on the site.

```css
/* Footer */
.footer {
    width: 100vw;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: flex-start;
    background-color: #1a1a1a;
}

/* Footer Div First Child */
.footer div:first-child {
    display: flex;
    flex-wrap: wrap;
}

/* Small Screens */
@media screen and (max-width: 779px) {
    /* Footer */
    .footer {
        align-items: center;
    }
    /* Footer Div First Child */
    .footer div:first-child {
        width: 260px;
    }
}
```

navigationCard.jsx

The purpose of this file is to export a react component that is displayed on the site.

```
// Strict Mode
"use strict";

// Styles
import styles from "./navigationCard.module.css";

// Navigation Card
export default function NavigationCard () {
    return (
        <div className = { styles.navigationCard }>
            <h5><a href = "/play">Play</a></h5>
            <h5><a href = "/leaderboards/1">Leaderboards</a></h5>
            <h5><a href = "/account">Account</a></h5>
        </div>
    );
};
```

navigationCard.module.css

The purpose of this file is to contain the css used by the navigationCard component.

```css
/* Navigation Card */
.navigationCard {
    width: 40%;
    padding: 30px;
    display: flex;
    justify-content: flex-start;
    align-items: center;
    flex-wrap: wrap;
    background-color: #1a1a1a;
}

/* Navigation Card H5 A */
.navigationCard h5 a {
    font-family: NeonSans;
    font-size: 20px;
    color: #ff9d00;
    text-shadow: 0 0 10px #ff9d00, 0 0 20px #ff9d00, 0 0 30px #ff9d00, 0 0 40px #ff9d00;
    transition: text-shadow 0.2s linear;
}

/* Navigation Card H5 A Hover */
.navigationCard h5 a:hover {
    text-shadow: 0 0 10px #ff9d00, 0 0 20px #ff9d00;
    transition: text-shadow 0.2s linear;
}

/* Small Screens */
@media screen and (max-width: 1027px) {
    /* Navigation Card */
    .navigationCard {
        width: 200px;
        padding-top: 0;
        justify-content: center;
    }
}
```

titleCard.jsx

The purpose of this file is to export a react component that is displayed on the site.

```
// Strict Mode
"use strict";

// Styles
import styles from "./titleCard.module.css";

// Title Card
export default function TitleCard () {
    return (
        <div className = { styles.titleCard }>
            <h4><a href = "/">Shooty</a></h4>
            <h4><a href = "/">Arena</a></h4>
        </div>
    );
};
```

titleCard.module.css

The purpose of this file is to contain the css used by the titleCard component.

```css
/* Title Card Shooty Text Flicker */
@keyframes titleCardShootyTextFlicker {
    37%, 39%, 41%, 43% {
        text-shadow: 0 0 10px #0062ff, 0 0 20px #0062ff, 0 0 30px #0062ff, 0 0 40px #0062ff;
    }
    35%, 38%, 40%, 42% {
        text-shadow: 0 0 10px #0062ff, 0 0 20px #0062ff;
    }
}

/* Title Card Arena Text Flicker */
@keyframes titleCardArenaTextFlicker {
    38%, 40% {
        text-shadow: 0 0 10px #ff3300, 0 0 20px #ff3300, 0 0 30px #ff3300, 0 0 40px #ff3300;
    }
    35%, 39% {
        text-shadow: 0 0 10px #ff3300, 0 0 20px #ff3300;
    }
}

/* Title Card */
.titleCard {
    width: 40%;
    padding: 30px;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    background-color: #1a1a1a;
}

/* Title Card H4 First Child */
.titleCard h4:first-child {
    text-shadow: 0 0 10px #0062ff, 0 0 20px #0062ff, 0 0 30px #0062ff, 0 0 40px #0062ff;
    animation: titleCardShootyTextFlicker 4s infinite;
}

/* Title Card h4 Last Child */
.titleCard h4:last-child {
    text-shadow: 0 0 10px #ff3300, 0 0 20px #ff3300, 0 0 30px #ff3300, 0 0 40px #ff3300;
    animation: titleCardArenaTextFlicker 6s infinite;
}

/* Title Card H4 First Child A */
.titleCard h4:first-child a {
    font-family: NeonSans;
    font-size: 50px;
    color: #0062ff;
}

/* Title Card H4 Last Child A */
.titleCard h4:last-child a {
    font-family: NeonSans;
    font-size: 50px;
    color: #ff3300;
}
```

header.jsx

The purpose of this file is to export a react component that is displayed on the site.

```
// Strict Mode
"use strict";

// Styles
import styles from "./header.module.css";

// Components
import NavigationCard from "./navigationCard/navigationCard.jsx";
import TitleCard from "./titleCard/titleCard.jsx";

// Header
export default function Header () {
    return (
        <div className = { styles.header }>
            <TitleCard/>
            <NavigationCard/>
        </div>
    );
};
```

header.module.css

The purpose of this file is to contain the css used by the header component.

```
/* Header */
.header {
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: #1a1a1a;
    flex-wrap: wrap;
}

/* Header A */
.header a {
    font-family: neon;
    text-decoration: none;
    padding: 20px;
}

/* Small Screens */
@media screen and (max-width: 1027px) {
    /* Header */
    .header {
        flex-direction: column;
    }
}
```

title.jsx

The purpose of this file is to export a react component that is displayed on the site.

```jsx
// Strict Mode
"use strict";

// Styles
import styles from "./title.module.css";

// Title
export default function Title ( props ) {
    return (
        <div className = { styles.title }>
            <h1>{ props.title }</h1>
        </div>
    );
};
```

title.module.css

The purpose of this file is to contain the css used by the title component.

```css
/* Title Text Flicker */
@keyframes titleTextFlicker {
    44%, 48%, 50% {
        text-shadow: 0 0 10px #00ff2a, 0 0 20px #00ff2a, 0 0 30px #00ff2a, 0 0 40px #00ff2a;
    }
    42%, 46%, 49% {
        text-shadow: 0 0 10px #00ff2a, 0 0 20px #00ff2a;
    }
}

/* Title */
.title {
    padding: 30px;
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: #1a1a1a;
}

/* Title H1 */
.title h1 {
    font-family: NeonSans;
    font-size: 70px;
    color: #00ff2a;
    text-shadow: 0 0 10px #00ff2a, 0 0 20px #00ff2a, 0 0 30px #00ff2a, 0 0 40px #00ff2a;
    animation: titleTextFlicker 4s infinite;
}
```

encrypt.js

The purpose of this file is to export the functions required for the web server to be able to encrypt and decrypt data.

```javascript
// Strict Mode
"use strict";

// Imports
import aesEverywhere from "aes-everywhere";

// Encrypt Key
const encryptKey = 'password';

// Encrypt
async function encrypt ( data ) {
    return aesEverywhere.encrypt(data, encryptKey);
};

// Decrypt
async function decrypt ( data ) {
    return aesEverywhere.decrypt(data, encryptKey);
};

// Exports
module.exports = {
    encrypt,
    decrypt
};
```

This code shows the use of encryption.

hash.js

The purpose of this file is to export the functions required for the web server to be able to hash data and compare the hashes of data.

```javascript
// Strict Mode
"use strict";

// Imports
import bcrypt from "bcrypt";

// Hash
async function hash ( data ) {
    return bcrypt.hash(data, await bcrypt.genSalt(8));
};

// Compare Hash
async function compareHash ( data, hash ) {
    return bcrypt.compare(data, hash);
};

// Exports
module.exports = {
    hash,
    compareHash
};
```

This code shows the use of hashing.

cryptography.js

The purpose of this file is to export the functions exported by encrypt.js hash.js.

```javascript
// Strict Mode
"use strict";

// Imports
import encrypt from "./encrypt/encrypt.js";
import hash from "./hash/hash.js";

// Exports
module.exports = {
    encrypt: encrypt.encrypt,
    decrypt: encrypt.decrypt,
    hash: hash.hash,
    compareHash: hash.compareHash
};
```

database.js

The purpose of this file is to export all of the functions required for that web server to be able to interact with the database.

```javascript
// Strict Mode
"use strict";

// Imports
import mysql from "mysql";

// Connection
const connection = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: 'password',
    database: "shootyarena"
});

// Connect
connection.connect(( error ) => {
    if (error) throw error;
});

// Create Tables
async function createTables () {
    connection.query("CREATE table users ( username VARCHAR(10) NOT NULL UNIQUE PRIMARY KEY, email VARCHAR(331) NOT NULL UNIQUE, passphrase VARCHAR(60) NOT NULL, active VARCHAR(60) NOT NULL, lastLoginTime INT NOT NULL );", ( error, _ ) => {
        if (error) throw error;
    });
    connection.query("CREATE TABLE stats ( username VARCHAR(10) NOT NULL UNIQUE PRIMARY KEY, FOREIGN KEY(username) REFERENCES users(username), gameRank VARCHAR(13) NOT NULL, elo INT NOT NULL, kills INT NOT NULL, deaths INT NOT NULL );", ( error, _ ) => {
        if (error) throw error;
    });
    connection.query("CREATE TABLE servers ( address VARCHAR(45) UNIQUE, playerLimit INT NOT NULL, playerCount INT NOT NULL );", ( error, _ ) => {
        if (error) throw error;
    });
    connection.query("CREATE TABLE codes ( username VARCHAR(10) NOT NULL UNIQUE PRIMARY KEY, FOREIGN KEY(username) REFERENCES users(username), code INT NOT NULL );", ( error, _ ) => {
        if (error) throw error;
    })
};

```

This code shows SQL embedded into the code. It also shows the creation of related tables.

```javascript
// Drop Tables
async function dropTables () {
    connection.query("DROP TABLE servers;", ( error, _ ) => {
        if (error) throw error;
    });
    connection.query("DROP TABLE stats;", ( error, _ ) => {
        if (error) throw error;
    });
    connection.query("DROP TABLE codes;", ( error, _ ) => {
        if (error) throw error;
    });
    connection.query("DROP TABLE users;", ( error, _ ) => {
        if (error) throw error;
    });
};

// Create User
async function createUser ( username, email, passphrase, active ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("INSERT INTO users ( username, email, passphrase, active, lastLoginTime ) VALUES ?;", [[[ username, email, passphrase, active, 0 ]]], ( error, _ ) => {
            if (error) {
                resolve(false);
            } else {
                resolve(true);
            };
        });
    });
};
```

This also shows SQL embedded in the code.

```javascript
// Create Stat
async function createStat ( username ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("INSERT INTO stats ( username, gameRank, elo, kills, deaths ) VALUES ?;", [[[ username, "Shitty Shooty", 800, 0, 0 ]]], ( error, _ ) => {
            if (error) {
                resolve(false);
            } else {
                resolve(true);
            };
        });
    });
};

// Create Server
async function createServer ( address, playerLimit ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("INSERT INTO servers ( address, playerLimit, playerCount ) VALUES ?;", [[[ address, playerLimit, 0 ]]], ( error, _ ) => {
            if (error) {
                resolve(false);
            } else {
                resolve(true);
            };
        });
    });
};

```

```javascript
// Create Code
async function createCode ( username, code ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("INSERT INTO codes ( username, code ) VALUES ?;", [[[ username, code ]]], ( error, _ ) => {
            if (error) {
                resolve(false);
            } else {
                resolve(true);
            }
        })
    })
}

// Delete Server
async function deleteServer ( address ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("DELETE FROM servers WHERE address = ?;", [[ address ]], ( error, _ ) => {
            if (error) {
                resolve(false);
            } else {
                resolve(true);
            };
        });
    });
};

```

This also shows SQL embedded in the code.
This also shows SQL embedded in the code.

```javascript
// Username Exists
async function usernameExists ( username ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("SELECT username FROM users WHERE username = ?;", [[ username ]], ( error, result ) => {
            if (error || result[0] !== undefined) {
                resolve(true);
            } else {
                resolve(false);
            };
        });
    });
};

// Email Exists
async function emailExists ( email ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("SELECT email FROM users WHERE email = ?;", [[ email ]], ( error, result ) => {
            if (error || result[0] !== undefined) {
                resolve(true);
            } else {
                resolve(false);
            };
        });
    });
};

```

```javascript
// Active Exists
async function activeExists ( active ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("SELECT active FROM users WHERE active = ?;", [[ active ]], ( error, result ) => {
            if (error || result[0] !== undefined) {
                resolve(true);
            } else {
                resolve(false);
            };
        });
    });
};

// Activate User
async function activateUser ( active ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("UPDATE users SET active = '' WHERE active = ?;", [[ active ]], ( error, result ) => {
            if (error || result.affectedRows === 0) {
                resolve(false);
            } else {
                resolve(true);
            };
        });
    });
};

```

This also shows SQL embedded in the code.
This also shows SQL embedded in the code.

```javascript
// Login User
async function loginUser ( email ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("SELECT passphrase FROM users WHERE email = ?;", [[ email ]], ( error, result ) => {
            if (error || result[0] === undefined) {
                resolve(false);
            } else {
                resolve(result[0]);
            };
        });
    });
};

// Get All Stats
async function getAllStats () {
    return new Promise (async ( resolve, _ ) => {
        connection.query("SELECT * FROM stats ORDER BY elo DESC;", ( error, result ) => {
            if (error) {
                resolve(false);
            } else {
                resolve(result);
            };
        });
    });
};

```

```javascript
// Get Stats
async function getStats ( username ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("SELECT gameRank, elo, kills, deaths FROM stats WHERE username = ?;", [[ username ]], ( error, result ) => {
            if (error || result[0] === undefined) {
                resolve(false);
            } else {
                resolve(result);
            };
        });
    });
};

// Update Game Rank
async function updateGameRank ( username, gameRank ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("UPDATE stats SET gameRank = ? WHERE username = ?;", [[ gameRank ], [ username ]], ( error, result ) => {
            if (error || result.affectedRows === 0) {
                resolve(false);
            } else {
                resolve(true);
            };
        });
    });
};

```

This also shows SQL embedded in the code.
This also shows SQL embedded in the code.

```javascript
// Update Elo
async function updateElo ( username, elo ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("UPDATE stats SET elo = ? WHERE username = ?;", [[ elo ], [ username ]], ( error, result ) => {
            if (error || result.affectedRows === 0) {
                resolve(false);
            } else {
                resolve(true);
            };
        });
    });
};

// Update Kills
async function updateKills ( username, kills ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("UPDATE stats SET kills = ? WHERE username = ?;", [[ kills ], [ username ]], ( error, result ) => {
            if (error || result.affectedRows === 0) {
                resolve(false);
            } else {
                resolve(true);
            };
        });
    });
};

```

```javascript
// Update Deaths
async function updateDeaths ( username, deaths ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("UPDATE stats SET deaths = ? WHERE username = ?;", [[ deaths ], [ username ]], ( error, result ) => {
            if (error || result.affectedRows === 0) {
                resolve(false);
            } else {
                resolve(true);
            };
        });
    });
};

// Get Free Server
async function getFreeServer () {
    return new Promise (async ( resolve, _ ) => {
        connection.query("SELECT address FROM servers WHERE playerCount < playerLimit ORDER BY  playerCount DESC;", ( error, result ) => {
            if (error || result[0] === undefined) {
                resolve(false);
            } else {
                resolve(result[0]);
            };
        });
    });
};

```

This also shows SQL embedded in the code.
This also shows SQL embedded in the code.

```javascript
// Set Player Count
async function setPlayerCount ( address, playerCount ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("UPDATE servers SET playerCount = ? WHERE address = ?;", [[ playerCount ], [ address ]], ( error, _ ) => {
            if (error || result.affectedRows === 0) {
                resolve(false);
            } else {
                resolve(true);
            };
        });
    });
};

// Get Active
async function getActive ( email ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("SELECT active FROM users WHERE email = ?;", [[email]], ( error, result ) => {
            if (error == null && result[0].active == "") {
                resolve(true);
            } else {
                resolve(result[0].active);
            }
        });
    });
};

```

```javascript
// Get Username
async function getUsername ( email ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("SELECT username FROM users WHERE email = ?;", [[email]], ( error, result ) => {
            if (error || result[0] === undefined) {
                resolve(false);
            } else {
                resolve(result[0].username)
            }
        })
    })
}

// Get Code
async function getCode ( username ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("SELECT code FROM codes WHERE username = ?;", [[ username ]], ( error, result ) => {
            if (error || result[0] === undefined) {
                resolve(false);
            } else {
                resolve(result[0].code);
            }
        })
    })
}

```

This also shows SQL embedded in the code.
This also shows SQL embedded in the code.

```javascript
// Set Code
async function setCode ( username, code ) {
    return new Promise (async ( resolve, _ ) => {
        connection.query("UPDATE codes SET code = ? WHERE username = ?", [[code], [username]], ( error, result ) => {
            if (error) {
                resolve(false);
            } else {
                resolve(true);
            }
        })
    })
}


// Exports
module.exports = {
    createTables,
    dropTables,
    createUser,
    createStat,
    createServer,
    deleteServer,
    usernameExists,
    emailExists,
    activeExists,
    activateUser,
    loginUser,
    getAllStats,
    getStats,
    updateGameRank,
    updateElo,
    updateKills,
    updateDeaths,
    getFreeServer,
    setPlayerCount,
    getActive,
    getUsername,
    createCode,
    getCode,
    setCode,

};

```

This also shows SQL embedded in the code.

send.js

The purpose of this file is to export the functions required be the web server to be able to send

```javascript
// Strict Mode
"use strict";

// Imports
import nodemailer from "nodemailer";

// Transport
const transport = nodemailer.createTransport({
    host: "smtp.gmail.com",
    port: 465,
    secure: true,
    auth: {
        user: "shootyarenamaster@gmail.com",
        pass: "password"
    }
});

// Send Account Activation Email
async function sendAccountActivationEmail ( to, link ) {
    return new Promise (async ( resolve, _ ) => {
        const mailOptions = {
            from: "Shooty Arena - shootyarenamaster@gmail.com",
            to: to,
            subject: "Shooty Arena Account Activation",
            text: "Welcome to Shooty Arena! To activate your account visit { link }".replace("{ link }", link)
        };
        transport.sendMail(mailOptions, ( err, _ ) => {
            if (err) resolve(false);
            resolve(true);
        });
    });
};

// Exports
module.exports = {
    sendAccountActivationEmail
};
```

emails.
This code shows more of the complex client server model.

validate.js

```javascript
// Strict Mode
"use strict";

// Imports
const deepEmailValidator =  require("deep-email-validator");

// Validate Email
async function validateEmail ( email ) {
    const check = await deepEmailValidator.validate(email);
    if (check.validators.regex.valid && check.validators.mx.valid) return true;
    return false;
};

// Exports
module.exports = {
    validateEmail
};
```

The purpose of this file is to export the functions required by the web server for it to be able to
This code shows the usage of regular expressions.

mail.js

```javascript
// Strict Mode
"use strict";

// Imports
import validate from "./validate/validate.js"
import send from "./send/send.js";

// Exports
module.exports = {
    validateEmail: validate.validateEmail,
    sendAccountActivationEmail: send.sendAccountActivationEmail
};
```

The purpose of this file is to export the functions exported by send.js and validate.js

[activate].js

The purpose of this file is to activate the users account and give the option to the user to sign in

```
// Strict Mode
"use strict";

// Imports
import React, { useEffect, useState } from "react";
import authentication from "../../authentication/authentication.js";
import database from "../../database/database.js";
import styles from "../../styles/account.module.css";
import Header from "../../components/header/header.jsx";
import Title from "../../components/title/title.jsx";
import Footer from "../../components/footer/footer.jsx";

// Get Server Side Props
export async function getServerSideProps ( ctx ) {
    try {
        if (await database.activateUser(ctx.query.activate)) {
            await authentication.getUser(ctx.req, ctx.res);
            return {
                props: {
                    message: "Account Activated"
                }
            };
        } else {
            await authentication.getUser(ctx.req, ctx.res);
            return {
                props: {
                    message: "Error Activating Account"
                }
            };
        };
    } catch {
        await authentication.getUser(ctx.req, ctx.res);
        return {
            props: {
                message: "Error"
            }
        };
    };
};

```

or create a new account.
This shows the use of the authentication and database modules I coded.

```
// Account
export default function Home( props ) {
    const [ signInError, setSignInError ] = useState("");
    const [ signUpError, setSignUpError ] = useState("");
    useEffect(() => {
        const host = "http://localhost:3000/";
        try {
            const signInForm = document.querySelector(".signInForm");
            signInForm.addEventListener("submit", async function ( event ) {
                try {
                    event.preventDefault();
                    const formData = new FormData(signInForm).entries();
                    const request = await fetch(host + "api/signIn", {
                        method: "POST",
                        headers: { "Content-Type": "application/json" },
                        body: JSON.stringify(Object.fromEntries(formData))
                    });
                    const result = await request.json();
                    if (result.response.success == true && result.response.activated == true) {
                        window.location = host + "account";
                    } else if (result.response.success == true && result.response.activated == false) {
                        setSignInError("Account Not Activated");
                    } else {
                        setSignInError("Bad Login");
                    };
                } catch {
                    setSignInError("Error");
                };
            }, []);
        } catch {};
        try {
            const signUpForm = document.querySelector(".signUpForm");
            signUpForm.addEventListener("submit", async function ( event ) {
                try {
                    event.preventDefault();
                    const formData = new FormData(signUpForm).entries();
                    const response = await fetch(host + "api/createAccount", {
                        method: "POST",
                        headers: { "Content-Type": "application/json" },
                        body: JSON.stringify(Object.fromEntries(formData))
                    });
                    const result = await response.json();
                    if (result.response.error != undefined) {
                        setSignUpError(result.response.error);
                    } else {
                        setSignUpError("Check Your Email To Activate Your Account");
                    }
                } catch {
                    setSignUpError("Error");
                };
            });
        } catch {};
        try {
            const signOutForm = document.querySelector(".signOutForm");
            signOutForm.addEventListener("submit", async function ( event ) {
                try {
                    event.preventDefault();
                    const formData = new FormData(signOutForm).entries()
                    const response = await fetch(host + "api/signOut", {
                        method: "POST",
                        headers: { "Content-Type": "application/json" },
                        body: JSON.stringify(Object.fromEntries(formData))
                    });
                    await response.json();
                    window.location = host + "account";
                } catch {
                    window.location = host + "account";
                };
            }, []);
        } catch {};
    });
    return (
        <div>
            <Header/>
            <Title title = "Account"/>
            <Title title = { props.message }/>
            {
                props.loggedIn ?
                    <div className = { styles.account }>
                        <form className = "signOutForm">
                            <button type = "submit">Sign Out</button>
                        </form>
                    </div>
                : (
                    <div className = { styles.account }>
                        <div>
                            <h2>Sign In</h2>
                            <form className = "signInForm">
                                <input type = "email" placeholder = "Email" name = "Email"/>
                                <input type = "password" placeholder = "Password" name = "Password"/>
                                <button type = "submit">Submit</button>
                            </form>
                            {
                                signInError ?
                                    <h3>{ signInError }</h3>
                                : (
                                    <h3> </h3>
                                )
                            }
                        </div>
                        <div>
                            <h2>Sign Up</h2>
                            <form className = "signUpForm">
                                <input type = "text" placeholder = "Username" name = "Username"/>
                                <input type = "email" placeholder = "Email" name = "Email"/>
                                <input type = "password" placeholder = "Password" name = "Password"/>
                                <input type = "password" placeholder = "Confirm Password" name = "Confirm Password"/>
                                <button type = "submit">Submit</button>
                            </form>
                            {
                                signUpError ?
                                    <h3>{ signUpError }</h3>
                                : (
                                    <h3> </h3>
                                )
                            }
                        </div>
                    </div>
                )
            }
            <Footer/>
        </div>
    );
};
```

This shows more of the complex client server model.

[getStats].js

```javascript
// Strict Mode
"use strict";

// Imports
import database from "../../../database/database.js";

// Handler
export default async function handler ( req, res ) {
    try {
        res.send(await database.getStats(req.query.getStats)[0]);
    } catch {
        res.send(500);
    };
};
```

The purpose of this file is to return all of the users statistics.
This shows more of the complex client server model and the use of an API I have created.

createAccount.js

The purpose of this file is to create an account for the user.

```javascript
// Strict Mode
"use strict";

// Imports
import cryptography from "../../cryptography/cryptography.js";
import database from "../../database/database.js";
import mail from "../../mail/mail.js";

// Handler
export default async function handler ( req, res ) {
    try {
        console.log(req.body.Username.match(/^\w+$/) == null)
        const host = "http://localhost:3000/";
        const response = {
            success: false,
            error: ""
        };
        req.body.Username = req.body.Username.toUpperCase();
        if (req.body.Username.length < 1) {
            response.error = "Enter A Username";
            res.send({
                response
            });
        } else if (req.body.Username.length > 10) {
            response.error = "Username Must Be 10 Characters Or Less";
            res.send({
                response
            });
        } else if (req.body.Username.match(/^\w+$/) == null) {
            response.error = "Username Must Only Contain Alphanumberic Characters";
            res.send({
                response
            });
        } else if (await database.usernameExists(req.body.Username)) {
            response.error = "Username Already In Use";
            res.send({
                response
            });
        } else if (await mail.validateEmail(req.body.Email) == false) {
            response.error = "Email Is Not Valid";
            res.send({
                response
            });
        } else if (await database.emailExists(req.body.Email)) {
            response.error = "Email Already In Use";
            res.send({
                response
            });
        } else if (req.body.Password.length < 4) {
            response.error = "Password Must Be At Least 4 Characters Long";
            res.send({
                response
            });
        } else if (await database.createUser(req.body.Username, req.body.Email, await cryptography.hash(req.body.Password), req.body.Username)) {
            response.success = true;
            await mail.sendAccountActivationEmail(req.body.Email, host + "activate/" + req.body.Username);
            await database.createStat(req.body.Username);
            res.send({
                response
            });
        } else {
            response.error = "Error Creating User";
            res.send({
                response
            });
        };
    } catch {
        response.success = false;
        response.error = "Error";
        res.send({
            response
        });
    };
};
```

This file shows more of the complex client server model as well as hashing and the use of SQL.

registerServer.js

```javascript
// Strict Mode
"use strict";

// Imports
import database from "../../database/database.js";

// Handler
export default async function handler ( req, res ) {
    try {
        const aes256 = require("aes-everywhere");
        const decryptedData = aes256.decrypt(req.body, 'passphrase = encrypt("Shooty Arena")').split(",");
        if (decryptedData.length != 2) {
            res.send(500);
        } else {
            await database.createServer(decryptedData[0], decryptedData[1]);
            res.send(200);
        };
    } catch {
        res.send(500);
    };
};
```

The purpose of this file is to register game game server in the database when it goes online. This shows more of the complex client server model and the use of an API I have created.

serverIp.js

The purpose of this file is to send the client the address of a free game server.

```
// Strict Mode
"use strict";

// Imports
import database from "../../database/database.js";

// Handler
export default async function handler ( req, res ) {
    try {
        res.send(await database.getFreeServer());
    } catch {
        res.send(500);
    };
};
```

Again this file shows the complex client server model of Shooty Arena and is also shows the use of an API I have created.

signIn.js

The purpose of this file is to sign the user in.

```javascript
// Strict Mode
"use strict";

// Imports
import authentication from "../../authentication/authentication.js";
import cryptography from "../../cryptography/cryptography.js";
import database from "../../database/database.js";
import mail from "../../mail/mail.js";

// Handler
export default async function handler ( req, res ) {
    try {
        const response = {
            success: false,
            activated: false
        };
        const password = await database.loginUser(req.body.Email);
        if (password == false) {
            res.send({
                response
            });
        } else {
            const active = await database.getActive(req.body.Email);
            if (await cryptography.compareHash(req.body.Password, password.passphrase)) {
                if (active === true) {
                    const user = await database.getUsername(req.body.Email);
                    await authentication.setUser(req, res, user);
                    response.success = true;
                    response.activated = true;
                    res.send({
                        response
                    });
                } else {
                    await mail.sendAccountActivationEmail(req.body.Email, "http://localhost:3000/" + "activate/" + await database.getActive(req.body.Email));
                    response.success = true;
                    res.send({
                        response
                    });
                };
            } else {
                res.send({
                    response
                });
            };
        };
    } catch {
        res.send(500);
    };
};
```

The file shows the use of hashing, an API I have created and more of the complex client server model in Shooty Arena.

signOut.js

The purpose of this file is to sign the user out. This file shows the use of the authentication package I have coded and the use of an API I have created.

```
// Strict Mode
"use strict";

// Imports
import authentication from "../../authentication/authentication.js";

// Handler
export default async function handler ( req, res ) {
    try {
        await authentication.setUser(req, res, "");
        res.send(200);
    } catch {
        res.send(500);
    };
};
```

updateStats.js

```javascript
// Strict Mode
"use strict";

// Imports
import database from "../../database/database.js";

// Handler
export default async function handler ( req, res ) {
    try {
        const aes256 = require("aes-everywhere");
        const data = aes256.decrypt(req.body , 'password');
        const decryptedData = data.split(",");
        if (decryptedData.length != 5) {
            res.send(500);
        } else {
            await database.updateElo(decryptedData[0], decryptedData[1]);
            await database.updateGameRank(decryptedData[0], decryptedData[2]);
            await database.updateKills(decryptedData[0], decryptedData[3]);
            await database.updateDeaths(decryptedData[0], decryptedData[4]);
            res.send(200);
        };
    } catch {
        res.send(500);
    };
};
```

The purpose of this file is to update the stats of the user after they leave the game server. This file shows the use of the database module I coded as well as more of the complex client server model

[page].js

The purpose of this file is to display the leaderboards to the user.

```
// Strict Mode
"use strict";

// Imports
import authentication from "../../authentication/authentication.js";
import database from "../../database/database.js";
import styles from "../../styles/leaderboards.module.css";
import Header from "../../components/header/header.jsx";
import Title from "../../components/title/title.jsx";
import Footer from "../../components/footer/footer.jsx";

// Get Server Side Props
export async function getServerSideProps ( ctx ) {
    await authentication.getUser(ctx.req, ctx.res);
    const stats = await database.getAllStats();
    const pageStats = [];
    let previousPage = false;
    let nextPage = false;
    if (ctx.query.page != 1) {
        previousPage = Number(ctx.query.page) - 1;
    };
    for (let stat = ((ctx.query.page - 1) * 4); stat < stats.length; stat++) {
        if (stat <= ((ctx.query.page - 1) * 4) + 3) {
            pageStats.push(stats[stat]);
        } else {
            nextPage = Number(ctx.query.page) + 1;
        };
    };
    if (pageStats.length == 0) {
        return {
            redirect: {
                permanent: false,
                destination: "/leaderboards/1"
            }
        };
    };
    return {
        props: {
            pageStats: JSON.stringify(pageStats),
            previousPage: previousPage,
            nextPage: nextPage
        }
    };
};

```

This shows more of the client server model and the use of some modules I have created.

```
// Leaderboard
export default function Leaderboards ( props ) {
    return (
        <div>
            <Header/>
            <Title title = "Leaderboards"/>
            <div className = { styles.leaderboards }>
                <div>
                    <table>
                        <tr>
                            <th>
                                Username
                            </th>
                            <th>
                                Game Rank
                            </th>
                            <th>
                                Elo
                            </th>
                            <th>
                                Kills
                            </th>
                            <th>
                                Deaths
                            </th>
                        </tr>
                        {
                            JSON.parse(props.pageStats).map(stats => {
                                return (
                                    <tr>
                                        <td>
                                            { stats.username }
                                        </td>
                                        <td>
                                            { stats.gameRank }
                                        </td>
                                        <td>
                                            { stats.elo }
                                        </td>
                                        <td>
                                            { stats.kills }
                                        </td>
                                        <td>
                                            { stats.deaths }
                                        </td>
                                    </tr>
                                );
                            })
                        }
                    </table>
                    <table>
                        <tr>
                            <th>
                                Rank
                            </th>
                            <th>
                                Elo
                            </th>
                        </tr>
                        <tr>
                            <td>
                                Shitty Shooty
                            </td>
                            <td>
                                0
                            </td>
                        </tr>
                        <tr>
                            <td>
                                Cutie Shooty
                            </td>
                            <td>
                                1000
                            </td>
                        </tr>
                        <tr>
                            <td>
                                Beauty Shooty
                            </td>
                            <td>
                                2000
                            </td>
                        </tr>
                        <tr>
                            <td>
                                Snooty Shooty
                            </td>
                            <td>
                                3000
                            </td>
                        </tr>
                    </table>
                </div>
                {
                    props.previousPage != false && props.nextPage != false &&
                        <div>
                            <a href = { "/leaderboards/?".replace("?", props.previousPage) }>Previous Page</a>
                            <a href = { "/leaderboards/?".replace("?", props.nextPage) }>Next Page</a>
                        </div>
                }
                {
                    props.previousPage != false && props.nextPage == false &&
                        <div>
                            <a href = { "/leaderboards/?".replace("?", props.previousPage) } style = {{"text-align": "center"}}>Previous Page</a>
                        </div>
                }
                {
                    props.nextPage != false && props.previousPage == false &&
                        <div>
                            <a href = { "/leaderboards/?".replace("?", props.nextPage) } style = {{"text-align": "center"}}>Next Page</a>
                        </div>
                }
            </div>
            <Footer/>
        </div>
    );
};
```

This shows the uses of one dimensional arrays.

_app.js

```
// Strict Mode
"use strict";

// Styles
import "../styles/reset.css"
import "../styles/fonts.css"

// My App
export default function MyApp ( { Component, pageProps } ) {
    return (
        <Component { ...pageProps }/>
    );
};
```

The purpose of this file is to export the root react component.
This shows the use of objects.

404.js

The purpose of this file is to display the 404 page when a page is not found.

```
// Strict Mode
"use strict";

// Imports
import styles from "../styles/404.module.css";
import Header from "../components/header/header.jsx";
import Title from "../components/title/title.jsx";
import Footer from "../components/footer/footer.jsx";

// Error 404
export default function Error404 () {
    return (
        <div>
            <Header/>
            <Title title = "Error"/>
            <div className = { styles.error404 }>
                <h2>404</h2>
            </div>
            <Footer/>
        </div>
    );
};
```

This file also shows the use of objects.

account.js

The purpose of this file is to show the user the account page.

```
// Strict Mode
"use strict";

// Imports
import React, { useEffect, useState } from "react";
import authentication from "../authentication/authentication.js";
import styles from "../styles/account.module.css";
import Header from "../components/header/header.jsx";
import Title from "../components/title/title.jsx";
import Footer from "../components/footer/footer.jsx";

// Get Server Side Props
export async function getServerSideProps ( ctx ) {
    const user = await authentication.getUser(ctx.req, ctx.res);
    if (user == "") {
        return {
            props: {
                loggedIn: false
            }
        };
    } else {
        return {
            props: {
                loggedIn: true
            }
        };
    };
};

```

This shows the use of some of the components I have made.

```
// Account
export default function Home( props ) {
    const [ signInError, setSignInError ] = useState("");
    const [ signUpError, setSignUpError ] = useState("");
    useEffect(() => {
        const host = "http://localhost:3000/";
        try {
            const signInForm = document.querySelector(".signInForm");
            signInForm.addEventListener("submit", async function ( event ) {
                try {
                    event.preventDefault();
                    const formData = new FormData(signInForm).entries();
                    const request = await fetch(host + "api/signIn", {
                        method: "POST",
                        headers: { "Content-Type": "application/json" },
                        body: JSON.stringify(Object.fromEntries(formData))
                    });
                    const result = await request.json();
                    if (result.response.success == true && result.response.activated == true) {
                        window.location = host + "account";
                    } else if (result.response.success == true && result.response.activated == false) {
                        setSignInError("Account Not Activated");
                    } else {
                        setSignInError("Bad Login");
                    };
                } catch (error) {
                    console.log(error)
                    setSignInError("Error");
                };
            }, []);
        } catch {};
        try {
            const signUpForm = document.querySelector(".signUpForm");
            signUpForm.addEventListener("submit", async function ( event ) {
                try {
                    event.preventDefault();
                    const formData = new FormData(signUpForm).entries();
                    const response = await fetch(host + "api/createAccount", {
                        method: "POST",
                        headers: { "Content-Type": "application/json" },
                        body: JSON.stringify(Object.fromEntries(formData))
                    });
                    const result = await response.json();
                    if (result.response.error != "") {
                        setSignUpError(result.response.error);
                    } else {
                        setSignUpError("Check Your Email To Activate Your Account");
                    }
                } catch (error) {
                    setSignUpError("Error");
                    console.log(error)
                };
            });
        } catch {};
        try {
            const signOutForm = document.querySelector(".signOutForm");
            signOutForm.addEventListener("submit", async function ( event ) {
                try {
                    event.preventDefault();
                    const formData = new FormData(signOutForm).entries()
                    const response = await fetch(host + "api/signOut", {
                        method: "POST",
                        headers: { "Content-Type": "application/json" },
                        body: JSON.stringify(Object.fromEntries(formData))
                    });
                    await response.json();
                    window.location = host + "account";
                } catch {
                    window.location = host + "account";
                };
            }, []);
        } catch {};
    });
    return (
        <div>
            <Header/>
            <Title title = "Account"/>
            {
                props.loggedIn ?
                    <div className = { styles.account }>
                        <form className = "signOutForm">
                            <button type = "submit">Sign Out</button>
                        </form>
                    </div>
                : (
                    <div className = { styles.account }>
                        <div>
                            <h2>Sign In</h2>
                            <form className = "signInForm">
                                <input type = "email" placeholder = "Email" name = "Email"/>
                                <input type = "password" placeholder = "Password" name = "Password"/>
                                <button type = "submit">Submit</button>
                            </form>
                            {
                                signInError ?
                                    <h3>{ signInError }</h3>
                                : (
                                    <h3> </h3>
                                )
                            }
                        </div>
                        <div>
                            <h2>Sign Up</h2>
                            <form className = "signUpForm">
                                <input type = "text" placeholder = "Username" name = "Username"/>
                                <input type = "email" placeholder = "Email" name = "Email"/>
                                <input type = "password" placeholder = "Password" name = "Password"/>
                                <input type = "password" placeholder = "Confirm Password" name = "Confirm Password"/>
                                <button type = "submit">Submit</button>
                            </form>
                            {
                                signUpError ?
                                    <h3>{ signUpError }</h3>
                                : (
                                    <h3> </h3>
                                )
                            }
                        </div>
                    </div>
                )
            }
            <Footer/>
        </div>
    );
};
```

This shows more of the complex client server model of Shooty Arena.

index.js

The purpose of this file is to show the user the home page.
This file also shows the use of the authentication module I have created.

```
// Strict Mode
"use strict";

// Imports
import React, { useEffect, useState, useRef } from "react";
import authentication from "../authentication/authentication.js";
import database from "../database/database.js";
import styles from "../styles/play.module.css";
import Header from "../components/header/header.jsx";
import Title from "../components/title/title.jsx";
import Footer from "../components/footer/footer.jsx";
import io from "socket.io-client";

// Get Server Side Props
export async function getServerSideProps ( ctx ) {
    let user = ""
    try {
        user = await authentication.getUser(ctx.req, ctx.res)
    } catch (e) {
        user = ""
    }
    if (user==="") {
        return {
            redirect: {
                destination: '/account',
                permanent: false,
            },
        }
    } else {
        var AES256  = require('aes-everywhere');
        const stats = await database.getStats(user)
        await authentication.setGame(ctx.req, ctx.res, AES256.encrypt(`${user},${stats[0].gameRank},${stats[0].elo},${stats[0].kills},${stats[0].deaths}`, 'passphrase = encrypt("Shooty Arena")'))
        return {
            props: {
                loggedIn: true
            }
        }
    }
}

```

play.js

The purpose of this file is to display the game window to the user and allow them to play the game.

```
// Strict Mode
"use strict";

// Imports
import authentication from "../authentication/authentication.js";
import styles from "../styles/index.module.css";
import Header from "../components/header/header.jsx";
import Title from "../components/title/title.jsx";
import Footer from "../components/footer/footer.jsx";

// Get Server Side Props
export async function getServerSideProps ( ctx ) {
    await authentication.getUser(ctx.req, ctx.res);
    return {
        props: {}
    };
};

// Index
export default function Index () {
    return (
        <div>
            <Header/>
            <Title title = "Home"/>
            <div className = { styles.index }>
                <h2>Shooty Arena is an online multiplayer platformer shooter! Sign in or create an account to get started.</h2>
                <h3>Currently in alpha. V0.1.0</h3>
            </div>
            <Footer/>
        </div>
    );
};
```

```
// Play
export default function Play () {
    // Refs
    const canvasRef = useRef(null);
    let socket = null;
    let menu = true;
    // Use Effect
    useEffect(() => {

        // Load Images
        const image_flat = new Image();
        image_flat.src = "/tiles/flat.png";
        const image_corner = new Image();
        image_corner.src = "/tiles/corner.png";
        const image_background = new Image();
        image_background.src = "/backgrounds/background.png";
        const image_background_light = new Image();
        image_background_light.src = "/backgrounds/backgroundLight.png";
        const image_player_right = new Image();
        image_player_right.src = "/players/right.png";
        const image_player_left = new Image();
        image_player_left.src = "/players/left.png"
        const image_gun_right = new Image();
        image_gun_right.src = "/players/gunRight.png";
        const image_gun_left = new Image();
        image_gun_left.src = "/players/gunLeft.png";

        // Sounds
        const shotSound1 = new Audio("/sounds/shoot.mp3");
        const shotSound2 = new Audio("/sounds/shoot.mp3");
        const shotSound3 = new Audio("/sounds/shoot.mp3");
        const shotSound4 = new Audio("/sounds/shoot.mp3");
        const shotSound5 = new Audio("/sounds/shoot.mp3");
        const shotSound6 = new Audio("/sounds/shoot.mp3");
        const shotSound7 = new Audio("/sounds/shoot.mp3");
        const shotSound8 = new Audio("/sounds/shoot.mp3");
        const shotSound9 = new Audio("/sounds/shoot.mp3");
        const shotSound10 = new Audio("/sounds/shoot.mp3");
        const shotSound11 = new Audio("/sounds/shoot.mp3");
        const shotSound12 = new Audio("/sounds/shoot.mp3");
        const shotSound13 = new Audio("/sounds/shoot.mp3");
        const shotSound14 = new Audio("/sounds/shoot.mp3");
        const shotSound15 = new Audio("/sounds/shoot.mp3");
        const shotSound16 = new Audio("/sounds/shoot.mp3");

        // Jump Sounds
        const jumpSound1 = new Audio("/sounds/jump.mp3");
        const jumpSound2 = new Audio("/sounds/jump.mp3");
        const jumpSound3 = new Audio("/sounds/jump.mp3");
        const jumpSound4 = new Audio("/sounds/jump.mp3");

        // Jump Sounds
        const landSound1 = new Audio("/sounds/land.mp3");
        const landSound2 = new Audio("/sounds/land.mp3");
        const landSound3 = new Audio("/sounds/land.mp3");
        const landSound4 = new Audio("/sounds/land.mp3");
        const landSound5 = new Audio("/sounds/land.mp3");
        const landSound6 = new Audio("/sounds/land.mp3");
        const landSound7 = new Audio("/sounds/land.mp3");
        const landSound8 = new Audio("/sounds/land.mp3");

        // Die Sound
        const dieSound = new Audio("/sounds/die.mp3");

```

```javascript
// Button
class Button {
    constructor() {
        this.startx = 0;
        this.starty = 0;
        this.x = 400;
        this.y = 300;
    };
    click(canvas, event) {
        const rect = canvas.getBoundingClientRect();
        const x = event.clientX - rect.left;
        const y = event.clientY - rect.top;
        if (x > this.startx && x < this.x) {
            if (y > this.starty && y < this.y) {
                return true;
            };
        };
        return false;
    };
};

// Fullscreen
function openFullscreen ( c ) {
    if (c.requestFullscreen) {
        c.requestFullscreen();
    } else if (c.webkitRequestFullscreen) {
            c.webkitRequestFullscreen();
    } else if (c.msRequestFullscreen) {
        c.msRequestFullscreen();
    };
};

```

This shows more of the complex client server model of Shooty Arena.

This show even more classes I have coded.

```javascript
// Canvas
const canvas = canvasRef.current;
const ctx = canvas.getContext("2d");
ctx.canvas.width = 1920;
ctx.canvas.height = 1080;

// Draw Images
function drawImage( ctx, image, x, y, w, h, degrees ){
    ctx.save();
    ctx.translate(x+w/2, y+h/2);
    ctx.rotate(degrees*Math.PI/180.0);
    ctx.translate(-x-w/2, -y-h/2);
    ctx.drawImage(image, x, y, w, h);
    ctx.restore();
};

function drawImage2( ctx, image, x, y, w, h, degrees ){
    ctx.save();
    ctx.translate(x, y + 20);
    ctx.rotate(degrees*Math.PI/180.0);
    ctx.translate(-x, -y - 20);
    ctx.drawImage(image, x, y, w, h);
    ctx.restore();
};

function drawImage3( ctx, image, x, y, w, h, degrees ){
    ctx.save();
    ctx.translate(x + w, y + 20);
    ctx.rotate(degrees*Math.PI/180.0);
    ctx.translate(-x - w, -y - 20);
    ctx.drawImage(image, x, y, w, h);
    ctx.restore();
};

```

```javascript
// Data
const playButton = new Button();
let map = [];
let player;
let players = [];
let bullets = [];
let scroll = [0, 0]
let health = [];
let directions = [];
let angles = [];
let lastClick = [0,0];
let shotnum = 1;
let jc = 1;
let lc = 1;

// Draw Timeout
setInterval(() => {

    // Wipe Screen
    ctx.fillStyle = "#1c1c1c";
    ctx.fillRect(0, 0, 1920 , 1080);

    // Scroll
    let bakc1 = 160 - scroll[0];
    let bakc2 = 160 - scroll[1];
    let b1 = scroll[0];
    let b2 = scroll[1];

    // Backround
    ctx.drawImage(image_background, bakc1, bakc2);

    // Menu
    if (menu == true) {
        ctx.fillStyle = "#FF0000";
        ctx.fillRect(playButton.startx, playButton.starty, playButton.x, playButton.y);

```

This shows the use of mathematical calculations.

This shows the use of more two dimensional arrays.

```javascript
    // Game
    } else {
        // Map
        for (let i = 0; i < map.length; i++) {
            ctx.fillStyle = "#FF0000";
            if (map[i][0] == 1) {
                ctx.drawImage(image_flat, parseInt(map[i][1] - scroll[0], 10), parseInt(map[i][2] - scroll[1], 10));
            } else if (map[i][0] == 2) {
                ctx.drawImage(image_corner, parseInt(map[i][1] - scroll[0], 10), parseInt(map[i][2] - scroll[1], 10));
            } else if (map[i][0] == 3) {
                drawImage(ctx, image_flat, parseInt(map[i][1] - scroll[0], 10), parseInt(map[i][2] - scroll[1], 10), 80, 80, 90);
            } else if (map[i][0] == 4) {
                drawImage(ctx, image_corner, parseInt(map[i][1] - scroll[0], 10), parseInt(map[i][2] - scroll[1], 10), 80, 80, 90);
            } else if (map[i][0] == 5) {
                drawImage(ctx, image_flat, parseInt(map[i][1] - scroll[0], 10), parseInt(map[i][2] - scroll[1], 10), 80, 80, 180);
            } else if (map[i][0] == 6) {
                drawImage(ctx, image_corner, parseInt(map[i][1] - scroll[0], 10), parseInt(map[i][2] - scroll[1], 10), 80, 80, 180);
            }else if (map[i][0] == 7) {
                drawImage(ctx, image_flat, parseInt(map[i][1] - scroll[0], 10), parseInt(map[i][2] - scroll[1], 10), 80, 80, 270);
            } else if (map[i][0] == 8) {
                drawImage(ctx, image_corner, parseInt(map[i][1] - scroll[0], 10), parseInt(map[i][2] - scroll[1], 10), 80, 80, 270);
            };
        };

        // Bullets
        for (let i = 0; i < bullets.length; i++ ) {
            ctx.fillStyle = "FF0000";
            ctx.fillRect(bullets[i][1] - scroll[0], bullets[i][2] - scroll[1], 10, 10);
        };

        // Players
        for (let i = 0; i < players.length; i++) {
            ctx.fillStyle = "00FF00";
            for (let y = 0; y < directions.length; y++) {
                if (players[i][0] == directions[y][0]) {
                    if (directions[y][1] < 0) {
                        ctx.drawImage(image_player_left, players[i][1][0] - b1, players[i][1][1] - b2, 80, 80);
                        for (let an = 0; an < angles.length; an ++) {
                            if (angles[an][0] == players[i][0]) {
                                drawImage3(ctx, image_gun_left, players[i][1][0] - b1 - 45, players[i][1][1] - b2 + 28, 80, 32, 180 + angles[an][1]);
                            };
                        };
                    } else {
                        ctx.drawImage(image_player_right, players[i][1][0] - b1, players[i][1][1] - b2, 80, 80);
                        for (let an = 0; an < angles.length; an ++) {
                            if (angles[an][0] == players[i][0]) {
                                drawImage2(ctx, image_gun_right, players[i][1][0] - b1 + 35, players[i][1][1] - b2 + 28, 80, 32, angles[an][1]);
                            };
                        };
                    };
                };
            };
            // Scroll
            if (players[i] != 0) {
                if (players[i][0] == player) {
                    scroll[0] += (players[i][1][0] - scroll[0] - 900)/20;
                    scroll[1] += (players[i][1][1] - scroll[1] - 500)/20;
                }
            }
            // Health
            for (let x = 0; x < health.length; x ++) {
                if (players[i][0] == health[x][0]) {
                    ctx.fillStyle = "FF0000";
                    ctx.fillRect(players[i][1][0] - b1, players[i][1][1] - b2 - 40, 80, 10);
                    ctx.fillStyle = "00FF00";
                    ctx.fillRect(players[i][1][0] - b1, players[i][1][1] - b2 - 40, health[x][1] * 10, 10);
                };
            };
        };

        // Light
        ctx.drawImage(image_background_light, bakc1, bakc2)

        // Mouse Move
        if (socket != null) {
            let px, py;
            for (let i = 0; i < players.length; i++) {
                if (players[i][0] == player) {
                    px = players[i][1][0] + 40 - scroll[0];
                    py = players[i][1][1] + 40 - scroll[1];
                };
            };
            let fx = lastClick[0] - px;
            let fy = lastClick[1] - py;
            let angle = Math.atan(fy/fx);
            let deg = (angle * 180) / Math.PI;
            if (lastClick[0] < px) deg += 180;
            if (deg == null || deg == undefined) deg = 0;
            socket.emit("mmove", {
                angle: deg
            });
        };
    };
}, 20);
```

```
// Inputs
let inputs = [false, false, false, false, [false, 0, 0]];

// Click
function  clickLocation( canvas, evt ) {
    var rect = canvas.getBoundingClientRect();
    let scaleX = canvas.width / rect.width;
    let scaleY = canvas.height / rect.height;
    return {
        x: (evt.clientX - rect.left) * scaleX,
        y: (evt.clientY - rect.top) * scaleY
    }
}

```

This show the use of more two dimensional arrays and mathematical calculations.

This also shows the uses of more two dimensional arrays and mathematical calculations.

```
// Clicks
canvas.addEventListener("mousedown", (event) => {
    openFullscreen(canvas)
    // No
    if (socket == null) {
        if (playButton.click(canvas, event)) {
            shotSound1.load()
            shotSound2.load()
            shotSound3.load()
            shotSound4.load()
            shotSound5.load()
            shotSound6.load()
            shotSound7.load()
            shotSound8.load()
            shotSound9.load()
            shotSound10.load()
            shotSound11.load()
            shotSound12.load()
            shotSound13.load()
            shotSound14.load()
            shotSound15.load()
            shotSound16.load()
            jumpSound1.load()
            jumpSound2.load()
            jumpSound3.load()
            jumpSound4.load()
            landSound1.load()
            landSound2.load()
            landSound3.load()
            landSound4.load()
            landSound5.load()
            landSound6.load()
            landSound7.load()
            landSound8.load()
            dieSound.load()
```

This show the use of event listeners.

```javascript
// Fetch Ip
fetch("http://localhost:3000/api/serverIp").then(res => {return res.json()}).then(data => {

    // Game Server
    socket = io( "http://localhost" + ":5000/")
    menu = false;

    // Mouse Move
    canvas.addEventListener("mousemove", e => {
        let { x,y } = clickLocation(canvas, e);
        lastClick = [x,y];
        let px, py;
        for (let i = 0; i < players.length; i++) {
            if (players[i][0] == player) {
                px = players[i][1][0] + 40 - scroll[0]
                py = players[i][1][1] + 40 - scroll[1]
            };
        };
        let fx = x - px;
        let fy = y - py;
        let angle = Math.atan(fy/fx);
        let deg = (angle * 180) / Math.PI;
        if (x < px) deg += 180;
        if (deg == null || deg == undefined) deg = 0;
        socket.emit("mmove", {
            angle: deg
        });
    });

    // Connect
    socket.on("connect", () => {
        socket.emit("authentication", document.cookie);
    });

    // Map
    socket.on("map", (dat) => {
        map = dat;
    });
```

```javascript
// Users
socket.on("users", (dat) => {
    console.log("rec")
    players = dat;
});

// Player
socket.on("player", (dat) => {
    player = dat;
});
```

```javascript
// Player
socket.on("player", (dat) => {
    player = dat;
});

// Bullets
socket.on("bullets", (dat) => {
    bullets = dat;
});

// Health
socket.on("health", (data) => {
    health = data;
});

// Directions
socket.on("directions", (data) => {
    directions = data;
});

// Disc
socket.on("disc", () => {
    window.location.reload();
})

// Ang
socket.on("ang", (data) => {
    angles = data;
});

// Die
socket.on("die", () => {
    dieSound.volume = 0.5;
    dieSound.play();
});

// Ju
socket.on("ju", () => {
    // Jump Sound
    jumpSound1.volume = 0.025;
    jumpSound2.volume = 0.025;
    jumpSound3.volume = 0.025;
    jumpSound4.volume = 0.025;
    if (jc == 5) jc = 1;
    if (jc == 1) jumpSound1.play();
    if (jc == 2) jumpSound2.play();
    if (jc == 3) jumpSound3.play();
    if (jc == 4) jumpSound4.play();
    jc += 1;
});

// La
socket.on("la", () => {
    // Volume
    landSound1.volume = 0.2
    landSound2.volume = 0.2
    landSound3.volume = 0.2
    landSound4.volume = 0.2
    landSound5.volume = 0.2
    landSound6.volume = 0.2
    landSound7.volume = 0.2
    landSound8.volume = 0.2
    if (lc == 1) landSound1.play();
    if (lc == 2) landSound2.play();
    if (lc == 3) landSound3.play();
    if (lc == 4) landSound4.play();
    if (lc == 5) landSound5.play();
    if (lc == 6) landSound6.play();
    if (lc == 7) landSound7.play();
    if (lc == 8) landSound8.play();
    if (lc == 9) lc = 1;
    lc += 1;
});
```

This shows even more of the complex client server model of Shooty Arena and some more of the mathematical calculations it uses.
This shows more of the client server model.

```
                    // Inputs
                    setInterval(()=>{
                        socket.emit("move", inputs)
                    }, 2);
                });
            }
        } else {
            for (let i = 0; i < players.length; i++) {
                if (players[i] != 0) {
                    if (players[i][0] == player) {
                        let { x, y } = clickLocation(canvas, event);
                        let playerX = players[i][1][0] + 40 - scroll[0];
                        let playerY = players[i][1][1] + 40 - scroll[1];
                        // Shot Sounds
                        if (shotnum == 17) shotnum = 1;
                        if (shotnum == 1) shotSound1.play();
                        if (shotnum == 2) shotSound2.play();
                        if (shotnum == 3) shotSound3.play();
                        if (shotnum == 4) shotSound4.play();
                        if (shotnum == 5) shotSound5.play();
                        if (shotnum == 6) shotSound6.play();
                        if (shotnum == 7) shotSound7.play();
                        if (shotnum == 8) shotSound8.play();
                        if (shotnum == 9) shotSound9.play();
                        if (shotnum == 10) shotSound10.play();
                        if (shotnum == 11) shotSound11.play();
                        if (shotnum == 12) shotSound12.play();
                        if (shotnum == 13) shotSound13.play();
                        if (shotnum == 14) shotSound14.play();
                        if (shotnum == 15) shotSound15.play();
                        if (shotnum == 16) shotSound16.play();
                        shotnum += 1;
                        // Click
                        socket.emit("click", {
                            clickX: x,
                            clickY: y,
                            dx: x - playerX,
                            dy: y - playerY,
                            px: players[i][1][0] + 35,
                            py: players[i][1][1] + 35
                        });
                    };
                };
            };
        };
});
```

This shows more of the client server model.

```javascript
    // Keyboard
    document.addEventListener("keydown", e => {
        if (socket != null) {
            if (e.key == "w") {
                inputs[0] = true;
            };
            if (e.key == "a") {
                inputs[1] = true;
            };
            if (e.key == "s") {
                inputs[2] = true;
            };
            if (e.key == "d") {
                inputs[3] = true;
            };
        };
    });
    document.addEventListener("keyup", e => {
        if (socket != null) {
            if (e.key == "w") {
                inputs[0] = false;
            };
            if (e.key == "a") {
                inputs[1] = false;
            };
            if (e.key == "s") {
                inputs[2] = false;
            };
            if (e.key == "d") {
                inputs[3] = false;
            };
        };
    });
}, []);
```

This shows more of the client server model and a few more mathematical calculations.

```
    // Page
    return (
        <div>
            <Header/>
            <Title title = "Play"/>
            <div className = { styles.play }>
                <canvas id="myCanvas" width="1920" height="1080" ref = {canvasRef}></canvas>
            </div>
            <Footer/>
        </div>
    );
};

```

This shows the use of more event listeners.

This shows the HTML in the play component.

404.module.css

The purpose of this file is to contain the css used the the 404 page.

```
/* Error404 */
.error404 {
    background-color: #1a1a1a;
}

/* Error404 H2 */
.error404 h2 {
    text-align: center;
    font-family: NeonSans;
    font-size: 40vw;
    color: #8c00ff;
    text-shadow: 0 0 4vw #8c00ff, 0 0 6vw #8c00ff;
}
```

account.module.css

```css
/* Account */
.account {
    display: flex;
    justify-content: center;
    background-color: #1a1a1a;
}

/* Account Div */
.account div {
    width: 30%;
    padding: 50px;
    display: flex;
    flex-direction: column;
    text-align: center;
}

/* Account Div H2 */
.account div h2 {
    padding: 40px;
    font-family: NeonSans;
    font-size: 25px;
    color: #8c00ff;
    text-shadow: 0 0 10px #8c00ff, 0 0 20px #8c00ff, 0 0 30px #8c00ff, 0 0 40px #8c00ff;
}

/* Account Div H3 */
.account div h3 {
    padding: 10px;
    font-family: NeonSans;
    font-size: 20px;
    background-color: #1a1a1a;
    color: #f700ff;
    text-shadow: 0 0 10px #f700ff, 0 0 20px #f700ff, 0 0 30px #f700ff, 0 0 40px #f700ff;
}

/* Account Div Form */
.account div form {
    display: flex;
    flex-direction: column;
}

/* Account Div Form Input */
.account div form input {
    padding: 15px;
    margin: 15px;
    border: 1px solid #f700ff;
    border-radius: 5px;
    font-family: NeonSans;
    font-size: 20px;
    background-color: #1a1a1a;
    color: #f700ff;
    text-shadow: 0 0 10px #f700ff, 0 0 20px #f700ff, 0 0 30px #f700ff, 0 0 40px #f700ff;
}

/* Account Div Form Input Placeholder */
.account div form input::placeholder {
    font-family: NeonSans;
    color: #f700ff;
}

/* Account Div Form Button */
.account div form button {
    padding: 15px;
    margin: 15px;
    cursor: pointer;
    border: 1px solid #ff9d00;
    border-radius: 5px;
    font-family: NeonSans;
    font-size: 20px;
    background-color: #1a1a1a;
    color: #ff9d00;
    text-shadow: 0 0 10px #ff9d00, 0 0 20px #ff9d00, 0 0 30px #ff9d00, 0 0 40px #ff9d00;
    transition: text-shadow 0.2s linear;
}

/* Account Div Form Button Hover */
.account div form button:hover {
    text-shadow: 0 0 10px #ff9d00, 0 0 20px #ff9d00;
    transition: text-shadow 0.2s linear;
}

/* Account Form Button */
.account form button {
    padding: 15px;
    margin: 15px;
    margin-bottom: 30px;
    cursor: pointer;
    border: 1px solid #ff9d00;
    border-radius: 5px;
    font-family: NeonSans;
    font-size: 20px;
    background-color: #1a1a1a;
    color: #ff9d00;
    text-shadow: 0 0 10px #ff9d00, 0 0 20px #ff9d00, 0 0 30px #ff9d00, 0 0 40px #ff9d00;
    transition: text-shadow 0.2s linear;
}

/* Account Form Button Hover */
.account form button:hover {
    text-shadow: 0 0 10px #ff9d00, 0 0 20px #ff9d00;
    transition: text-shadow 0.2s linear;
}

/* Small Screens */
@media screen and (max-width: 1000px) {
    /* Account */
    .account {
        flex-direction: column;
        align-items: center;
    }
    /* Account Div */
    .account div {
        width: 80%;
        padding-top: 20px;
        padding-bottom: 20px;
    }
}
```

The purpose of this file is to contain the css used by the account page.

font.css

The purpose of this file is to set the font face for the different fonts used.

```
/* NeonSans Font Face */
@font-face {
    font-family: NeonSans;
    src: url(../public/fonts/NeonSans.ttf);
}
```

index.module.css

The purpose of this file is to contain the css used on the home page.

```css
/* Index */
.index {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}

/* Index H2 */
.index h2 {
    width: 80%;
    padding: 20px;
    text-align: center;
    font-family: NeonSans;
    font-size: 25px;
    color: #8c00ff;
    text-shadow: 0 0 10px #8c00ff, 0 0 20px #8c00ff, 0 0 30px #8c00ff, 0 0 40px #8c00ff;
}

/* Index H3 */
.index h3 {
    width: 80%;
    padding: 20px;
    padding-bottom: 40px;
    text-align: center;
    font-family: NeonSans;
    font-size: 15px;
    color: #f700ff;
    text-shadow: 0 0 10px #f700ff, 0 0 20px #f700ff, 0 0 30px #f700ff, 0 0 40px #f700ff;
}
```

leaderboards.module.css

```
/* Leaderboards */
.leaderboards {
    background-color: #1a1a1a;
}

/* Leaderboards Div First Child */
.leaderboards div:first-child {
    padding: 10px 40px;
    padding-top: 40px;
    display: flex;
    justify-content: space-around;
    align-items: flex-start;
}

/* Leaderboards Div First Child Table */
.leaderboards div:first-child table {
    border-spacing: 20px;
}

/* Leaderboards Div First Child Table First Child */
.leaderboards div:first-child table:first-child {
    width: 65%;
}

/* Leaderboards Div First Child Table Last Child */
.leaderboards div:first-child table:last-child {
    width: 25%;
}

/* Leaderboards Div First Child Table Tr */
.leaderboards div:first-child table tr {
    font-family: NeonSans;
}

/* Leaderboards Div First Child Table Tr Th */
.leaderboards div:first-child table tr th {
    text-align: start;
    font-size: 25px;
    color: #8c00ff;
    text-shadow: 0 0 10px #8c00ff, 0 0 20px #8c00ff, 0 0 30px #8c00ff, 0 0 40px #8c00ff;
}

/* Leaderboards Div First Child Table Tr Td */
.leaderboards div:first-child table tr td {
    font-size: 20px;
    color: #f700ff;
    text-shadow: 0 0 10px #f700ff, 0 0 20px #f700ff, 0 0 30px #f700ff, 0 0 40px #f700ff;
}

/* Leaderboards Div Last Child */
.leaderboards div:last-child {
    width: 100%;
    display: flex;
    justify-content: center;
    align-items: center;
}

/* Leaderboards Div Last Child A */
.leaderboards div:last-child a {
    width: 160px;
    padding: 30px;
    padding-bottom: 40px;
    font-family: NeonSans;
    font-size: 20px;
    color: #ff9d00;
    text-shadow: 0 0 10px #ff9d00, 0 0 20px #ff9d00, 0 0 30px #ff9d00, 0 0 40px #ff9d00;
    transition: text-shadow 0.2s linear;
}

/* Leaderboards Div Last Child A Hover */
.leaderboards div:last-child a:hover {
    text-shadow: 0 0 10px #ff9d00, 0 0 20px #ff9d00;
    transition: text-shadow 0.2s linear;
}

/* Small Screens */
@media screen and (max-width: 1111px) {
    /* Leaderboards Div First Child */
    .leaderboards div:first-child {
        flex-direction: column-reverse;
        justify-content: center;
        align-items: center;
    }
    /* Leaderboards Div First Child Table First Child */
    .leaderboards div:first-child table:first-child {
        padding-top: 50px;
        width: 100%;
    }
    /* Leaderboards Div First Child Table Last Child */
    .leaderboards div:first-child table:last-child {
        width: 100%;
    }
}
```

play.module.css

```
/* Play Canvas */
.play canvas {
    width: 100vw;
}
```

reset.css

The purpose of this file is to reset default css styles.

```
/* Body */
body {
    background-color: #1a1a1a;
}

/* All Elements */
* {
    padding: 0;
    margin: 0;
    border: 0;
    text-decoration: none;
}
```

next.config.css

The purpose of this file is to configure NextJs.

```
// Strict Mode
"use strict";

// Exports
module.exports = {
    reactStrictMode: true
};
```

This file shows the use of objects.

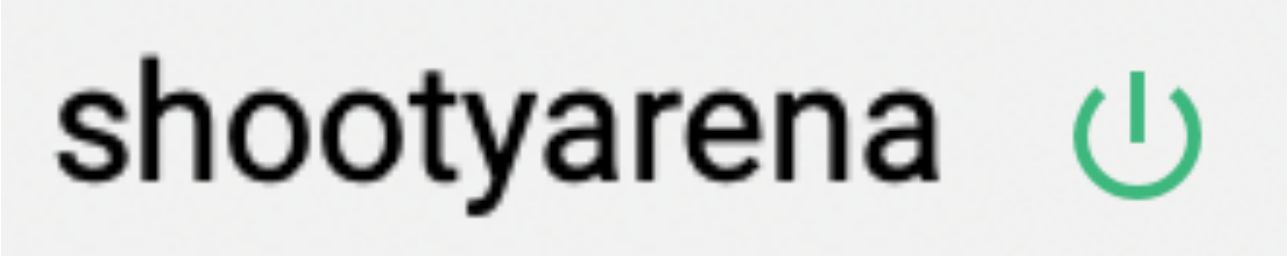
package.json

The purpose of this file is to keep track of dependencies and create executable scripts.

```
{
    "name": "web",
    "private": true,
    "scripts": {
        "dev": "next dev",
        "build": "next build",
        "start": "next start",
        "lint": "next lint"
    },
    "dependencies": {
        "aes-everywhere": "^1.0.0",
        "bcrypt": "^5.0.1",
        "config": "^3.3.7",
        "cookies": "^0.8.0",
        "deep-email-validator": "^0.1.21",
        "jsonwebtoken": "^8.5.1",
        "mysql": "^2.18.1",
        "next": "^12.0.10",
        "nodemailer": "^6.7.2",
        "react": "^17.0.2",
        "react-dom": "17.0.2",
        "socket.io-client": "^4.4.1"
    },
    "devDependencies": {
        "eslint": "8.6.0",
        "eslint-config-next": "12.0.7"
    }
}
```

Deployment

I have deployed this project on the domain shootyarena.com. The web server is running on the domain shootyarena.com and the game server is running on game-1.shootyarena.com. Both domains are using Cloudflare name servers, all traffic is proxied through Cloudflare and that all data sent to and received from from both servers are encrypted. Each server is running NGINX as a reverse proxy to either the NextJs application running through PM2 or the flask application running through Gunicorn.



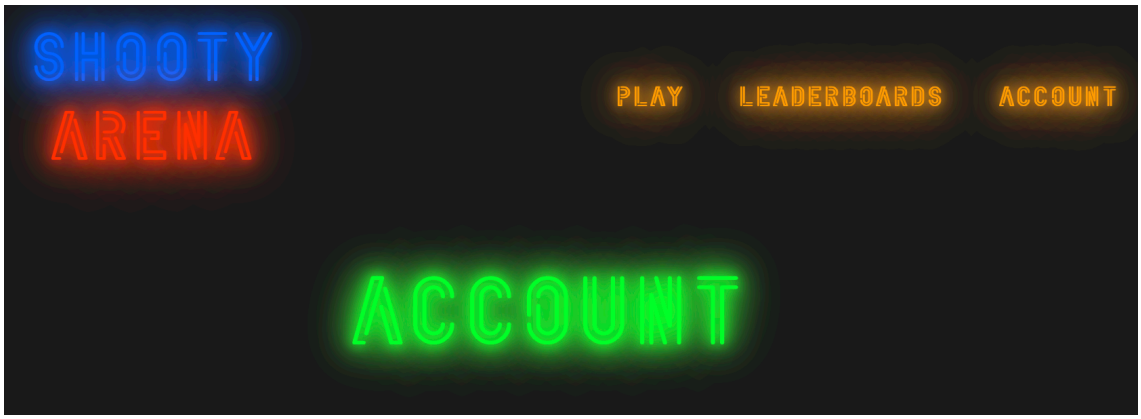
Type	Name
A	game-1
A	shootyarena.com

Testing

I tested application on Safari, Firefox, Microsoft Edge and Google Chrome. Here are the results.

Site Navigation

Test	User Input	Expected Output	Actual Output	Success
Click on the home button.	Click.	The user should be redirected to the home page.	The user is redirected to the home page.	Yes.
Click on the play button while logged out.	Click.	The user should be redirected to the account page.	The user is redirected to the account page.	Yes.
Click on the play button while logged in.	Click.	The user should be redirected to the play page.	The user is redirected to the play page.	Yes.
Click on the leaderboards button.	Click.	The user should be redirected to the leaderboards page.	The user is redirected to the leaderboards page.	Yes.
Click on the account button.	Click.	The user should be redirected to the account page.	The user is redirected to the account page.	Yes.
Visit a page that doesn't exist.	Click.	The user should be redirected to a 404 page.	The user is redirected to the 404 page.	Yes.



Account Management

Test	User Input	Expected Output	Actual Output	Success
Create an account with no invalid or duplicate data.	Keyboard and clicks.	The account is created and the user receives a message telling them to check their email for an activation link.	The account is created and the user receives a message telling them to check their email for an activation link.	Partially, It works but the second password field is not checked.
Create an account with invalid data.	Keyboard and clicks.	The account is not created and the user receives a message telling them which pieces of data is invalid.	The account is not created and the user receives a message telling them which pieces of data is invalid.	Yes.
Create an account with duplicated data.	Keyboard and clicks.	The account is not created and the user receives a message telling them which pieces of data already exist.	The account is not created and the user receives a message telling them which pieces of data already exist.	Yes.
The user clicks on the activation link sent to their email.	Click.	The user sees a message telling them their account is activated above the sign in and sign up forms.	The user sees a message telling them their account is activated above the sign in and sign up forms.	Yes.
The user clicks on the activation link again.	Click.	The user sees a message telling them there was an error activating the account above the sign in and sign up forms.	The user sees a message telling them there was an error activating the account above the sign in and sign up forms.	Yes.
The user visits and invalid activation link	Keyboard and clicks.	The user sees a message telling them there was an error activating the account above the sign in and sign up forms.	The user sees a message telling them there was an error activating the account above the sign in and sign up forms.	Yes.
The user clicks on the sign out button once they are signed in.	Click.	The user is signed out and the page is refreshed.	The user is signed out and the page is refreshed.	Yes.

Leaderboards

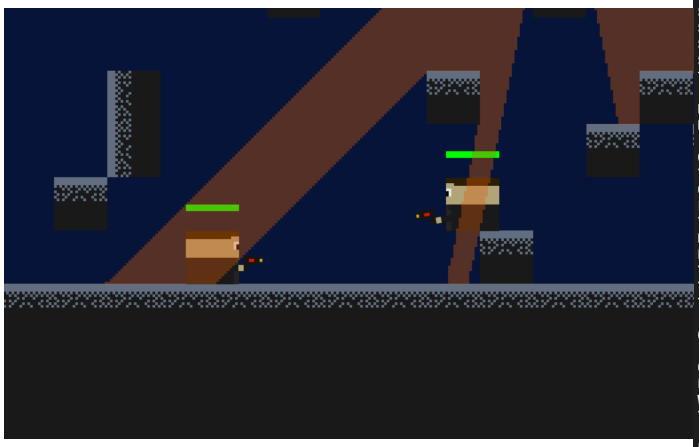
Test	User Input	Expected Output	Actual Output	Success
The user can view the top five users on the leaderboard.	None.	The user can view the top five users on the leaderboard along with the buttons to navigate through the leaderboards if more than five users are on it.	The user can view the top five users on the leaderboard along with the buttons to navigate through the leaderboards if more than five users are on it.	Yes.
The user can view other users on the leaderboard by navigating through it using the buttons that appear when their a enough users on the leaderboards.	Clicks.	The user can navigate through the leaderboards.	The user can navigate through the leaderboards.	Yes.
Updating scores.	None.	The users scores on the leaderboard are updated a short amount of time after leaving a game.	The users scores on the leaderboard are updated a short amount of time after leaving a game.	Partially, it works but there is an inconsistent amount of time from when the user leaves the game and when their score is updated.

Play

Test	User Input	Expected Output	Actual Output	Success
The user can maximise the game window by clicking on it.	Click.	The game window becomes maximised.	The game window becomes maximised.	Yes.
The user can connect to the game server and start playing by pressing on the play button.	Click.	The client is connected to the game server and can start playing.	The client is connected to the game server and can start playing.	Yes.
The user can move around.	WAD keys.	The player moves around the map.	The player moves around the map.	Yes.
The player can aim their gun.	Mouse movement.	The players arm rotates to point towards the users mouse.	The players arm rotates to point towards the users mouse.	Yes.
The player can shoot.	Click.	A bullet leaves the players gun.	A bullet leaves the players gun.	Yes.
The player can take damage.	None.	The players health bar moves down.	The players health bar moves down.	Partially, it works but only in Safari.
The player can die and respawn	None.	The player respawns at the top of the screen.	The player respawns at the top of the screen.	Yes.
The game plays the appropriate sounds.	WAD keys and click.	The appropriate sounds play.	The appropriate sounds play.	Yes.
The player can leave the game by visiting a different page.	Keyboard and clicks.	The player is removed from the game and the leaderboards are updated.	The player is removed from the game and the leaderboards are updated.	Partially, it works but the player is not always removed straight away.
Multiple players can play together.	Keyboard and clicks.	The players can see each other and each others bullets.	The players can see each other and each others bullets.	Yes.

Other

Test	User Input	Expected Output	Actual Output	Success
All of the data being sent to and from the site and game server are encrypted.	None.	Valid SSL certificates.	Valid SSL certificates.	Yes.
The web server can handle many requests at once.	None.	Unknown.	163.7 Requests per second.	Yes.
The game server can handle multiple people at once.	None.	Unknown.	Can handle more than one player at once.	Yes.



```
tylantyson@Computer ~ % ab -t 10 -n 10000 -c 100 https://shootyarena.com/
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking shootyarena.com (be patient)
Completed 1000 requests
Finished 1650 requests


Server Software:        cloudflare
Server Hostname:        shootyarena.com
Server Port:            443
SSL/TLS Protocol:       TLSv1.2,ECDHE-ECDSA-CHACHA20-POLY1305,256,256
Server Temp Key:        ECDH X25519 253 bits
TLS Server Name:        shootyarena.com

Document Path:          /
Document Length:        3270 bytes

Concurrency Level:      100
Time taken for tests:   10.080 seconds
Complete requests:      1650
Failed requests:        0
Total transferred:      7123664 bytes
HTML transferred:       5423838 bytes
Requests per second:    163.70 [#/sec] (mean)
Time per request:       610.881 [ms] (mean)
Time per request:       6.109 [ms] (mean, across all concurrent requests)
Transfer rate:          690.18 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       44  258 102.3    247    1225
Processing:    59  319 103.5    304    1024
Waiting:       59  293  74.0    297     602
Total:        193  577 152.3    563    1608

Percentage of the requests served within a certain time (ms)
  50%    563
  66%    602
  75%    617
  80%    646
  90%    779
  95%    840
  98%    923
  99%   1027
 100%   1608 (longest request)
tylantyson@Computer ~ %
```

Evaluation

I revisited by objectives to review how successful they were. I also spoke to my end user.

Objectives

Objective	Success	Details
Create a template that all pages are based off and make sure the design is responsive so all features ares accessible at all screen sizes.	Yes.	None.
This template should include a header that contains a navigation bar so that the user can navigate between all of the different pages.	Yes.	Yes.
This template should also include a footer that contains link to the contact page as well as the legal pages.	Partially.	The links in the footer lead to the 404 page.
Create a visually appealing home page where all features are accessible at all resolutions.	Yes.	None.
The home page should explain what the game is and how it works.	No.	I did not have time to implement this.
Create a visually appealing play page where all features are accessible at all resolutions.	Yes.	None.
If the user is not signed in when they visit this page they should be redirected to the account page.	Yes.	None.
On the play page there should be an option to join a game.	Yes.	None.
The user should be able to disconnect from the game they are participating in.	Yes.	None.
On the play page the game window becomes full screen once it is clicked on.	Yes.	None.
Create a leaderboards page where all features are accessible at all resolutions.	Yes.	None.
The leaderboards page should display the ELO required to reach each rank.	Yes.	None.
Display every player by rank with pagination.	Yes.	None.

Objective	Success	Details
On the leaderboards page allow the user to order users.	No.	I have not had time to implement this.
Allow the user to click on a player to view more detailed information.	No.	I have not had time to implement this.
Create an account page where all features are accessible at all resolutions.	Yes.	None.
If the user is not signed in they should be given the option to create an account, login, or reset their password. If they decide to create an account they should be sent a validation email with a link they must visit in order to activate their account before they can login.	Partially.	The user currently does not have the option to reset their password.
If the user is signed in they should be able to view their information, username, rank performance and graphs, They should also have the option to change their password.	Partially.	The user currently only has the option to sign out.
Create the game servers.	Yes.	None.
The servers should allow people to connect and play. Once the player leaves their updated information should be sent to the web server.	Yes.	None.
Host the required servers power them by renewable energy and make them accessible from anywhere.	Yes.	None.
Make sure the game is secure.	Yes.	None.

End User

What did you think about the game?

Overall I enjoyed it. It is visually appealing and has simple controls and nice sounds which all make the experience more immersive. I also liked the leaderboards page and how the players are ranked. It was also nice that it was very quick and easy to make an account and sign in.

Did the game meet your requirements?

Overall I would say the project was a success as it met most of my requirements and I enjoyed the experience.

What do you think can be improved?

I think the first thing to improve is the health bars. It would be great if it worked across all browsers. It would also great if the map was slightly larger and there were more game rooms.

Conclusion

I really enjoyed working on this project and I think that overall the project was a success. Although there are a number of things that can be improved most of my objectives and most of the end users requirements were met. The end user enjoyed the experience and thought that the result was great. If I had more time I would have completed the objectives that are not completely finished, I would make the map slightly larger and add more game rooms. I would also add features such as the ability to delete your account and reset your password. If I has even more time I would write the game server in C++ to improve performance, I would add the ability to equip skins and I would refine the game and website.