MIDDLE EAST TECHNICAL UNIVERSITY
NORTHERN CYPRUS CAMPUS

Computer Engineering Program

CNG 300

SUMMER PRACTICE REPORT

Name of Student : Mete KARASAKAL

ID Number : 2315380

Name of Company : Ata Bilişim

Project Title : CS:GO Server Management System

Date of Submission : 2 November 2020

ABSTRACT

This report includes information about the problems that I faced while doing my internship and how I handled these problems. During my internship I have done two jobs, first was Wordpress website development and maintenance of them, also used some PHP to implement Wordpress Plugins and meanwhile I developed my HTML & CSS skills. Second was a client's project called Ligazone which is currently inactive e-sport platform, they demanded a data transfer system between their main server and Counter Strike: Global Offensive servers which are powered by Get5 open-source plugin, for this purpose I used Python and as a framework I used Flask and built-in library called Socket. In conclusion, I gained knowledge of how a REST API works and how it should be designed, how Sockets work and how it should be designed in varying conditions, how to handle data transfer and how data structures should be used in a big project, how to develop a Wordpress site and use HTML & CSS properly. I built a middleware WebApp which is deployed on Apache HTTP server on a Windows OS computer and several Sockets inside middleware to transfer data between CS:GO servers and middleware. I used Sockets instead of WebApp to make the game servers transfer data in a lightweight way and to use remote console controls of the game's own console. At the moment, this system can work in the most optimized way, thanks to the structure of data I use to store and transfer, the system is capable of making players play 1v1 games currently but it can be easily configured to make 5v5 games playable because of its flexible design, it brings data to games' server and takes the statistics back to main server, if any problem happens it gives response codes to main server e.g. if a game' server is not available it posts a response code.

TABLE OF CONTENS

LIST OF FIGURES

1. ATA BİLİŞİM

Ata Bilişim is a private company and they usually do web development, system design and development, graphics design, and digital marketing. This company has 4 branches at TRNC, Japan, Turkey, and Qatar. I worked at the TRNC branch of it at Lefkoşa. I did web development, and system design and development but mostly have done system design and development for a customer, mostly on my own and with the demands of my advisor Kaan Pargan who was the CTO of Ata Bilişim and the firm called Ligazone. Apart from these I worked on web development with a variety of companies in TRNC. There were 2 other engineers working there too.

2. INTRODUCTION

I have done my internship at Ata Bilişim which is located in Lefkoşa TRNC. Because of the pandemia I decided to stay in TRNC so I made a research about the companies at Cyprus and found Ata Bilişim, there were not too much companies accepting 2nd year students in Cyprus so I made a research about Ata Bilişim and found out that was compatible with my internship requirements. It was a good experience for me, workers were kind and helpful to the interns, actually this is a small company but there were 6-8 interns and they usually worked on web projects, I was familiar with web technologies before because of that I asked for some other projects to learn new technologies, therefore they asked me to build a tournament system for CS:GO but meanwhile I worked on web projects with others too. To conclude I developed my HTML, CSS, PHP, Python skills with the help of my advisors' and other interns' knowledge.

3. PROBLEM STATEMENT

In my summer internship I have worked on one problem statement and assigned with web development projects.

For web development projects I didn't face any problems as I said before I am familiar with web technologies. In some cases I needed to use PHP to add non-significant features to some Wordpress Plugins.

My main objective was building a json data transfer system between the main website to CS:GO servers' computers and the reverse of it. I have done it by transferring data to middleware via an API on a Windows machine for development purposes. Middleware had the tasks of controlling a game servers' availability, getting the game statistics data and starting the game on the games' server, and posting the data back to the main website.

Firstly because of I didn't have any permission to see the console of games' servers console and log it I needed to make a way to get the data of if the server is available or not, second problem was starting the game on games' servers from the middleware because I didn't have the permission to write commands of the games' servers command line, other problem was getting the match IDs from games' servers I needed these IDs because I needed to store json statistic files for sake of file handling, another problem was happened because of API posts when request comes so I needed to post json files by order, last problem was getting the game data to middleware from games' server because of using a webapp on games' servers (using Apache HTTP server) would be a heavy duty on server and damage the game experience so I couldn't make 2nd api on it and I needed another option to transfer the data back. Finally I managed to handle these problems and built the system, now it is working smoothly without loss of data.

4. WORDPRESS

WordPress is an open-source website creation platform that is written in PHP and uses a MySQL database. I used this platform for most of the web page designing tasks [1].

4.1. HTML & CSS

HTML is the standard markup language for creating Web pages, I used it to add extras to Wordpress sites and developed standalone HTML 5 & CSS 3 sites too and edited some sites.

4.2. PHP - Hypertext Preprocessor

PHP is a popular general-purpose scripting language that is especially suited to web development [2]. I used basic PHP to implement some Wordpress Plugins in my internship. The version that I used was 7.3.24.

5. PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics [3]. I used Python in the API that I developed and Sockets that I used, since the Get5 server plugin was working on Python and It was easy to implement on this game server management system thanks to ease of writing and available libraries. I used Python 3.7.4 for this project.

5.1. Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions [4]. I wanted to use Flask because it needs less power than other deployed webapps; it has included what I needed and it doesn't make decisions for you as an example some other web frameworks may want you to use specific DBMS but if someone doesn’t want to use DB they don’t in Flask. It is extendable and simple such as you can build a webapp in an only single python file but at the same time you can extend it with external libraries and technologies you may choose. I used Flask 1.1.2 for this project.

5.2. Sockets

Sockets are basically programs that have 2 ends first is server-side second is client-side and while one is giving data other is taking data and and in the final data transfer happens between these two serially. These are usually used for serial

connecting data transfer systems like chatting applications. I used it for transferring json files between middleware and CS:GO servers.

5.3. JSON Data Format

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate[5]. I used JSON data since the plugin Get5 uses JSON data to start information of a game and return statistics of a game in JSON format.

5.4. Middleware

Middleware is a software which is also called “software glue” because it is making communication between softwares easier, and modulates the main software to have a lighter software. It can also be used as an authentication tool to complete a task. I used it while I was building my API because the main server also had other tasks to do and couldn’t handle the task that I did on my middleware.

6. APACHE HTTP SERVER PROJECT

Apache is a HTTP server project that helps to deploy web projects. I deployed the API on an Apache HTTP server. For more information check [6].

7. CS:GO

CS:GO stands for Counter Strike: Global Offensive which is a first person shooter game available on game platform Steam. Gamers have their own SteamIDs. There are different variations of these ID formats. These are steamID, steamID3, steamID64 I needed to use steamID format to give information about to games’ server that who will be playing.

7.1. Get5 Plugin

Get5 plugin is an open-source project that adds some features to game playing which I will not extend this topic because mainly I worked on how to start and configure game settings from a distant computer and get data. Another feature is that it can start a game with a given data. The reason why did I used this plugin is:

- It is open source.
- It has features that can be developed to run and maintain it from a distant computer, such as it can get json data directly from an API.

It was a manual plugin before I built the API system, before that someone had to get into the games console from the games' server then enter the game details but now It can be accessible from the main system of Ligazone. Get5 has its own API but there is a warning on it about it is not working properly and having bugs, it recommends writing your own API system so I did that.

8. MAIN PURPOSE AND DESIGN OF THE SYSTEM

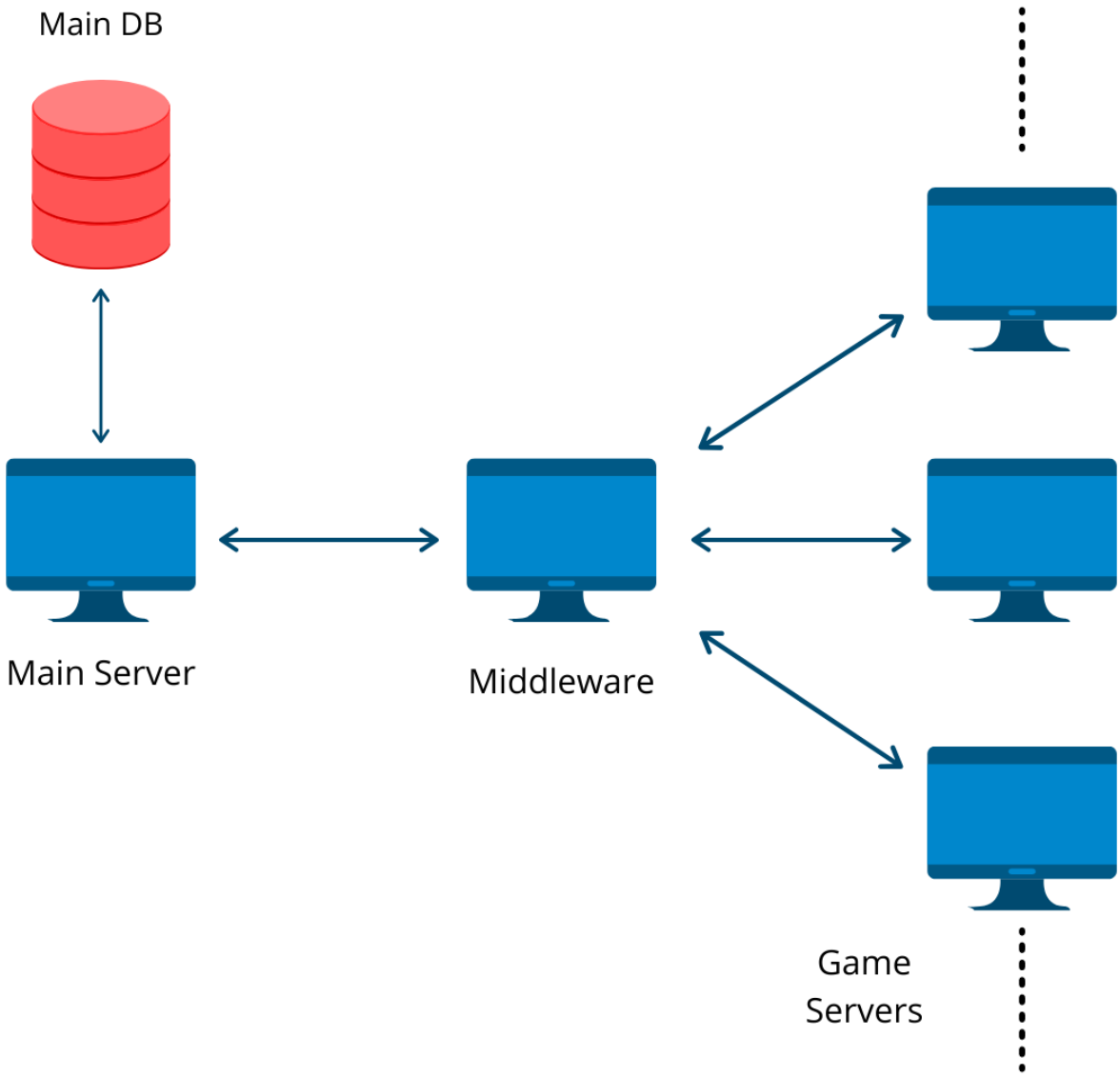

Figure 1: Relations of the system

Before I start, how did I solve the problems while I was developing the system. I need to explain how I managed to design the project. Firstly, I haven't worked on the main server but the main problem was basically delivering the data from the main server and giving back data to it to store the game data to the main DB. If you check Figure 1 you can see how relationships are made. I didn't have permission to see the main server and I had to deliver games to specific game servers and manage them from another software so I wanted to build a middleware first, to manage servers easily.

In conclusion, the main server was giving me the data of the game via its own api which I will be explaining the incoming data in the next section of this part. Because the main server was streaming the data via an api, I built an API on middleware and

received the data and posted another data back to it via my api, after that on game servers I built a client socket on game servers and I built a server socket on middleware which made all the data transfer from game servers to middleware.

8.1. Json Data Coming From Main Server

The incoming Json file includes game configurations and middleware functionality configurations. I will not show the file itself but only highlight the parts that are needed to understand how a game server works. Before I start to talk about how middleware works it is essential to know about these configurations. Incoming data is listed below:

- **Game IP:** This is the IP of the game server where the game will be played on.
- **Match ID:** This is a unique ID of a game.
- **In-Game Configs:** num_maps, players_per_team, min_players_to_ready, min_spectators_to_ready, skip_veto, veto_first, side_type, maplist, and team configs these are all related with gameplay of a game so these are not essential to know.

9. MIDDLEWARE

Middleware consists of three parts which are API, file manager and socket, and rcon which stands for remote console. It is a socket. Before I start talking about the parts of middleware, I will describe the features of the middleware respectively:

- Takes initial data from the main server and starts the game on the specified server.
- Checks if the server is available or not, means that it checks if the server is on/off and is there any game currently playing or not with an array which stores game ID and game IPs.
- Alter the statistic files and add game id and ip on it to log the data on the main DB.
- Posts statistics files to the main server.
- There is a queue that checks if files are received by the main server or not otherwise it won't pop from the queue. This does not let data get lost.
- There is another queue that checks if the rcon initialized the game or not.
- Gives response codes back to the main server about if a task is done or not.

To conclude, these are the main functions of the middleware I built, these were the most challenging tasks that I have done in my internship as well.

9.1. API

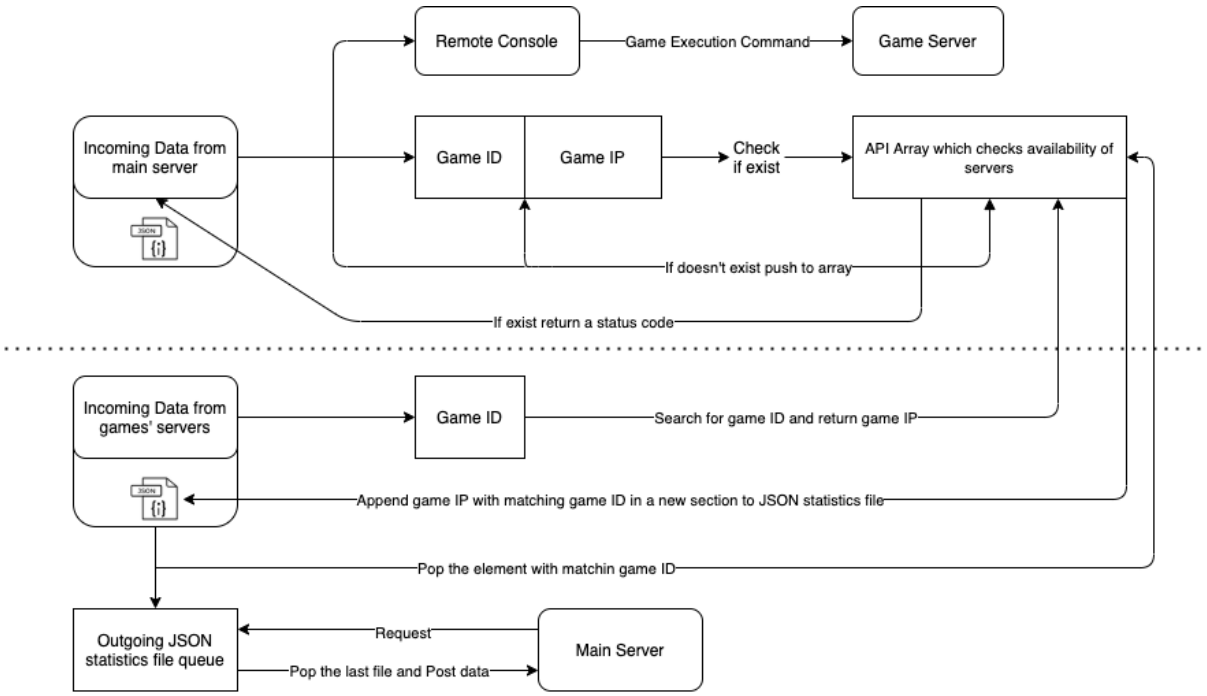

Figure 2: Whole API system workflow diagram

API handles 2 main operations in the upper part of Figure 2 it handles the incoming data, in lower part it handles outgoing data.

In the incoming data part it gets the json file of a games' data that I mentioned at part 8.1. API fetches the data of game ID and game IP and checks if it does exist in the array of unavailable servers array if it is in the unavailable servers array it returns a bad status code to the main server. This case is not possible because the main server also checks for game servers availability but in the case of it is not available, the main server gives another IP for the game and checks it as it is not an available server. This is only for precaution, it is not expected to give this return code. If IP does not exist then it is added to the unavailable server list until a data comes back to the middleware from the specific game server which I will be explaining in the next paragraph. After adding this data to the array It will also send a signal to rcon to start the game on a specified game server with the altered json file. API alters the json file because Get5 plugin has its own configuration file and game IP is added by the main server so I needed to remove the game IP part and post it to the game server.

In the outgoing data part after API gets the data from the game server then it finds the related game IP found by game ID on an unavailable server list and appends to the json file which came from the game server. I needed to add this property because the main server must know which server is available or not and it is not included in incoming data because Get5 doesn't have any feature of adding IP of a game server in a statistics file. Statistic file includes data about gameplay like how much kill did a person get in game or who won the game. I wanted to add the game IP property inside the json file in the middleware because in game server it would be a heavy duty on the particular server (opening a json file and writing on it). After adding the game IP to the returning json file middleware pops the server IP and game ID from the unavailable game server array and adds the game ID on outgoing json file queue, because of API works on request-get logic I wanted main server to request from server frequently and get the json files in the list whenever main server requests API posts a statistic file if there is no statistic files left in middleware main server gets a status code to give a cooldown to API. I did that to not get too many GET requests.

I deployed this API on Apache HTTP project server with some wsgi configuration which I will not explain, this was a simple task.

9.2. File Manager & Socket

If you check Figure 1 again there are multiple game servers related to the middleware so I needed to build a multiple ended socket to get the data from these servers without a loss of data. File manager is basically in the API. It works when a request comes. It is the part that I mentioned at the previous section I POST the JSON files via API but I think it is more related with the Socket because these are working together it can't be possible to POST a data before it comes to middleware. You can see the file managers' code on Figure 3. It checks if a file added and if a file added adds information to the file which includes the game IP and ID section and posts the file with declared status codes and removes the file on middleware.

```
def push_stat():
    file_path = os.getcwd() + "\\stat\\"
    file_list = os.listdir(file_path)
    try:
        for i in range(len(file_list)):
            if current_ids[i][0] in file_list:
                with open(file_path + file_list[i], 'r+') as statfile:
                    stat = json.load(statfile)
                    stat = {"stat": stat}
                    stat.update({"info": {"ip": current_ids[i][1], "matchid":current_ids[i]
[0].split('.')[0]}})
                    current_ids.remove(current_ids[i])
                os.remove(file_path + file_list[i])
            return stat, 200  # istatistik başarıyla yollandı
    except json.decoder.JSONDecodeError:
        return {"response": 506}, 506  # gönderilecek istatistik yok
    except IndexError:
        return {"response": 506}, 506  # gönderilecek istatistik yok
```

Figure 3: File Manager

The queue of statistic files that I mentioned is really a composition between file manager and socket.

In the middleware socket is working non stop because there can always be a connection happening. I have added an example of how it is working when a binding happens between client and server socket and when data comes socket changes the name of the file as match_id.json to search in the file manager part. You can see the example socket data transfer on Figure 4. I will add another example of how socket is working on the client side on a game server when I am giving information about game servers.

Figure 4: Example run of server socket

9.3. RCON - Remote Console

RCON is a socket for remote connection to an ingame console. In my case I found an open source code because there was no need to rewrite it. You can see the full RCON code from here [7].

```
rcon = srcds.SourceRcon(rcon_ip,27015,'test12345')
rcon.rcon('get5_loadmatch_url 178.20.224.155:5000/')
```

Figure 5: Code snippet of RCON

I used the RCON inside the API as you can see in Figure 5. I first built an RCON object which does the authorization and determine specifications (because there are more than one game server, rcon_ip is a variable depending on incoming data) on the console of the game (for test reason I used test inputs) and in the second line I executed the game on the games server. RCON works after the match's initial data comes.

In conclusion, this part may seem simple but I didn’t know anything about RCON protocol in the beginning so this became a research task for me. There is a big research part laying behind these 2 lines of Python code. I successfully implemented it inside the middleware.

10. GAME SERVER

Game servers consist of three parts. These are RCON server socket, file checker, and client socket to transfer data.

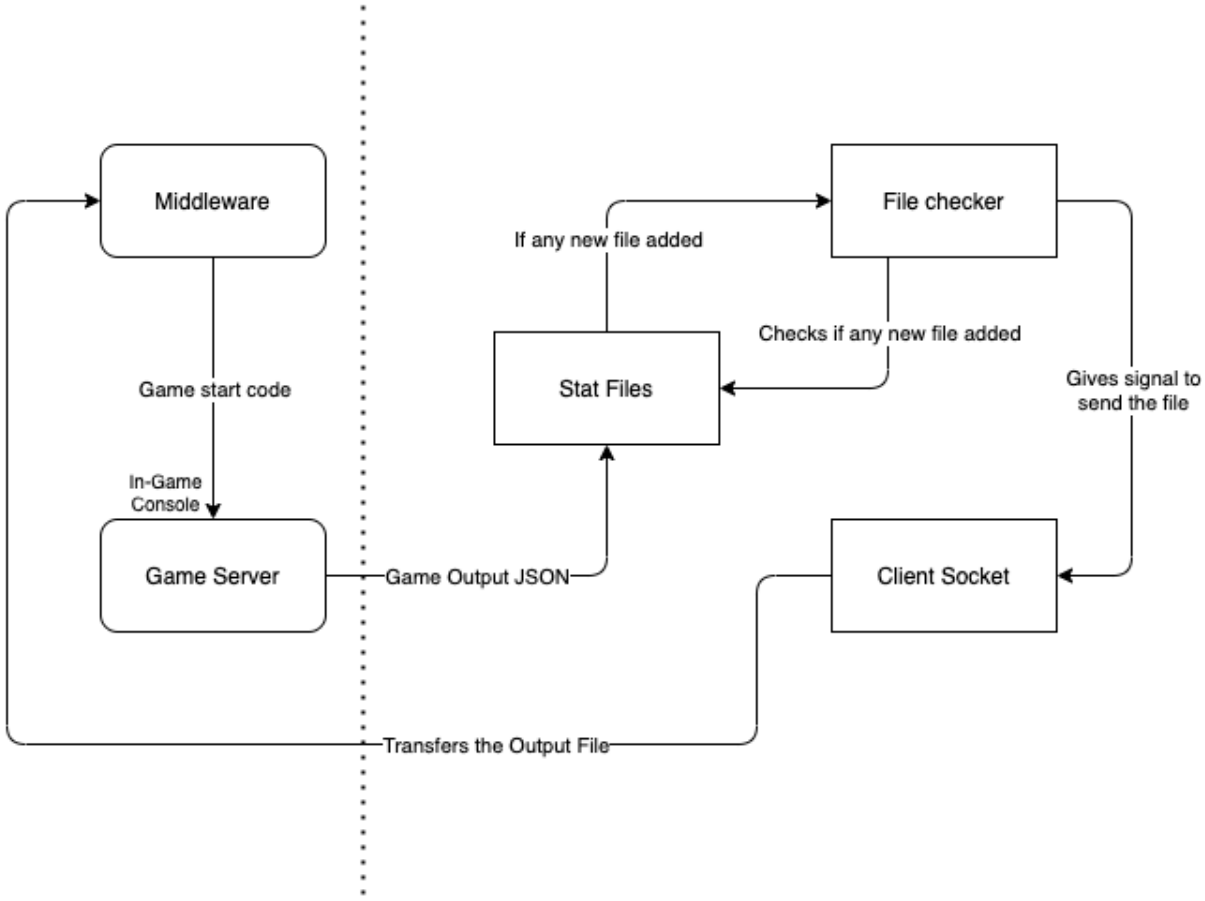

Figure 6: Game servers' system workflow diagram

If you check Figure 6 you can see two parts on the left hand side of the diagram is about initializing the game which is done by code executed at Figure 5, right hand side is returning the game statistics to middleware.

Game initializing starts with Figure 5 codes' second line which executes the code for game initialization on Get5 plugin with the specified match configuration which is POSTed by the API. RCON receiver is pre-included inside CS:GO's in-game server.

Before game starts a JSON statistic file initializes by Get5 file checker adds it to an array after game is played statistics are exported inside this initialized file to a specified destination folder which I have declared by modifying Get5 Plugin code, then file checker checks for the destination folder, after that file checker checks last modification time of initialized statistic folder if it is edited it means that game is over.

When the game is over the file checker gives a signal to the client socket which transfers the output file to middleware, and the file checker pops the data file from the queue and moves the statistics file inside a backup folder. There are more than one game can be played on a server for this reason there is a queue inside the file checker socket is working in nanoseconds but for the reason if it does not push the file successfully it has 3 seconds cooldown to push the file before the next item in the queue.

```
>>> start_sending('test.txt')
[+] Connecting to 127.0.0.1:5001
[+] Connected.
Sending test.txt: 0.00B [00:00, ?B/s]Sending test.txt: 0.00B [00:00, ?B/s]
>>>
```

Figure 7: Test run for client socket

You can see the example run of the client socket in Figure 7. In the meantime you can see what middleware' socket outputs on Figure 4.

To conclude, I had two choices to transfer the statistics data: these were via an API or Socket. I used socket because I didn't need to run Apache HTTP project server on it. Running Apache on a game server could be a heavy task and it could damage the gaming experience so I choose to use socket only at the end of the game.

11. WEB DEVELOPMENT

I did work on several projects but mostly edited web pages meanwhile I gained experience of using HTML, CSS, PHP, JavaScript, and WordPress.

I edited and developed approximately 15 webpages and developed 1 on my own. Most of the sites were using WordPress at the both back-end and front-end. I have done 2 SEO work by using Yoast SEO plugin. I worked on sites Keleş Matbaa, Adakent Üniversitesi, 3KDevelopment, Kamacıoğlu Yurt and so on. With other Interns I have developed 3 sites using HTML templates so it made me gain experience working with a group.

I have done my project on WordPress. When I was doing my own project I worked with the client myself and gained experience of human relationships. You can check the site here [8]. I used a car booking plugin and altered the PHP code of it to show default times on the main page.

In conclusion, it was also a good experience for me, working as a web developer. I learned the conditions of working as a web developer in Cyprus. After the internship I started to learn other web technologies and frameworks like Django.

12. CONCLUSION

During my internship I built a fully functional game tournament system in which I used some data structure systems and some data transfer methods and developed and edited websites.

I firstly have built a middleware which manages CS:GO game servers using an API, 2 sockets and 1 data manager and for games servers I have developed 1 socket and 1 file checker program. I designed, edited and developed websites in my second part of my internship.

The LigaZone company will be available soon and they will use the system that I built for CS:GO servers. The program that I built is a big contribution for that company because this system has extendable features such as they want 1v1 games to be played but in the future they say that they want 5v5 games to be played if they want to do it only thing should be changed is game initialization data. For the web development part I build modern sites and use plugins that can be used for long term.

To conclude, I especially used the knowledge of my data structure I gained at data structure course and the researching experience again I gained in METU. I gained knowledge about how to design a project before starting a project. How to work in a project with other engineering students. I learned how a restful API works and learned about network connections. Finally projects I built were all satisfactory for me and for the companies that I worked with. Especially for the system that I built let me gain too much experience of being in a work environment, research the technologies that I don’t know.

REFERENCES

1. https://ithemes.com/tutorials/what-is-wordpress/
2. https://www.php.net/
3. https://www.python.org/
4. https://en.wikipedia.org/wiki/Flask_(web_framework)
5. https://www.json.org/
6. https://httpd.apache.org/
7. https://github.com/frostschutz/SourceLib
8. http://ercanhavaalanirentacar.net/