Design Report

CS:GO Dashboard

April 22, 2022
University of Twente

Authors
Arjan Blankestijn (s2371073)
Mark Kok (s1730304)
Mark van Wijk (s2283166)
Valentijn Hol (s2295040)

Supervisor
Estefanía Talavera Martínez

Contents

Chapter 1

Introduction

1.1 Client

The project has been proposed by Esports Team Twente (ETT). ETT is the seventh student team associated to the University of Twente. They try to combine Esports with research to improve performance. For this reason, they compete in different types of games and several tournaments and competitions [1].

1.2 Platform

This project aims to support ETT with their Counter-Strike: Global Offensive (CS:GO) analysis. CS:GO is a popular first-person shooter, with millions of players playing the game daily [2].

It might not come as a big surprise then that, due to it being a popular competitively-focused game, there is also a big professional scene around the game. In fact, the CS:GO pro scene can be regarded as the second largest Esport-scene in the world [3] [4]. It is, however not known by all, and we will spend the remainder of this section on giving a high-level overview of the game.

There are several game modes within CS:GO, but the most popular one by far - as well as most used in competitive play - is 'competitive'. In this mode the Terrorists (T) try to make a bomb explode at one of the two dedicated bombsites in the map. The Counter-Terrorists (CT) aim to prevent this from happening. Each attempt of this is called a 'round', where a round is ended when either the opposing team has no more players, or the bomb explodes. The first to hit 16 rounds wins, with at most 30 rounds being played (ignoring overtime in case of ties) [5]. In the remainder of the report, we will always talk in the context of the 'competitive' mode when we are talking about CS:GO.

While this principle sounds simple, in higher levels the gameplay becomes

quite complex [6]. The game rules always remain the same, there is no possibility of getting a 'better' character, for example. It is purely on the players to outplay each other and become better within the bounds of the game rules.

Utility

In the game, there are several different types of so called 'utility'. Utility is not the main weapon in the game, but the can help players if it is used well. A piece of utility is something than can be thrown by the player, and upon landing (or after a set period of time) will actuate some effect upon either the map or other players.

Flashbang

First of all, there are the flashbangs. When a flashbang explodes, anyone looking at it, will have their screen turn white for a set period of time. This duration is based on how close they are to the flashbang and at which angle they were looking at it. During this period the flashed player is an easier target, since it is harder for them to fight back [7].

HE Grenade

The second piece of utility is the highly explosive (HE) grenade, or grenade for short. It can be thrown at an enemy position and explode there. Anyone close to the blast will receive damage, which might kill the enemy or at least soften them up to make it easier to kill them later [8].

Smoke

Next up, we have the smoke. When a smoke gets thrown it, contrary to the other utilities, does not go off after a set period of time. It instead 'pops' after it has become stationary. This means that smokes can travel further. When a smoke pops, a cloud of smoke gets released, making it impossible to see through. This can, for example, help to prevent incoming Counter-Terrorists from immediately seeing the Terrorists on the bombsite [9].

Molotov

The last kind of utility we are going to look at is the molotov. Technically, the molotov is only for the Terrorists and Counter-Terrorists have the Incendiary Grenade, but it performs exactly the same. After it's been thrown it will pop automatically if it hasn't reached the ground within a few seconds. If this happens, the Molotov will be wasted, since there is nothing to

burn. If it hits the ground before this time, it will burst into flames, which will hurt anything standing inside them. This can help to clear out certain positions [10].

Roles

One of the big differences between lower and higher level players, is that higher level players will work with assigned roles. Although there are no official in-game roles to speak of, they will make these decisions as a team to help players prioritise certain objectives. We will cover the roles on the Terrorist side since they are more interesting than the roles on the Counter-Terrorist side. These roles include: the ingame leader (IGL), sniper, support, lurker and entry fragger [11] [12]. It should however be mentioned that roles are not static. It might be that a certain player dies, while their role still needs to be played, someone might not feel like a certain role or someone might be in a better position to do it. It is also possible that certain roles get combined.

In-game Leader

The IGL is the brain of the team. They will decide where their team will be during the round and when and where they attack [13]. As a result the team can work together, making their gameplay more efficient. Since the IGL has to think way more actively about the game, for example where opponents might be and what they should do as a response, IGLs used to have less kills. For this reason, IGLs that do tend to get quite some kills, got the nickname 'fragging IGL'. Nowadays however, the meta of CS:GO has shifted a bit and fragging IGLs are becoming more frequest, however they are usually still not the players with the most kills [14].

Sniper

The sniper is the person who usually takes the sniper-weapon, called the AWP, when it is available to the team. Given that this gun can even kill an enemy with one hit in their body on a long distance, the sniper can take try to pick off enemies from long distances, or hold an extremely narrow angle, since there is a scope on their weapon. Given that a body is bigger than a head, snipers tend to aim more for the body than for the head, decreasing their headshot percentage [15] [16].

Entry Fragger

Next up, we have the entry fragger. The entry fragger is the first one to enter a bombsite in an attack. Since the Counter-Terrorists can be in a lot of positions on the bombsite, and the Terrorists can only come from a

few positions, the entry fragger always has a disadvantage. If they succeed, however, their flashy plays are what makes the crowd love to watch the game. It is possible, and in-fact likely, that they die during an attack. Their job is ensure that, even if they do die, they are close to a teammate who can immediately kill the person who killed them (also referred to as 'trading').

Support

The next role we will look at is the support. The support primarily supports the entry fragger. They do this by using their utility as best they can, to make way for the entry fragger and by trading them when they die. With this utility, they can blind the enemy, burn them out of certain positions, damage the opponents with grenades, or smoke off positions in the bombsite. [17].

Lurker

The last role we will look at is the lurker. As their teammate enter the bombsite to take it, the lurker will generally stay behind to catch enemies who are trying to reach the compromised bombsite. Since there are multiple positions on the map where lurkers can be, they are annoying for the Counter-Terrorists trying to help their teammates stuck in the bombsite. This means that the lurker can often catch people off-guard [18].

Economy

While each round gives a point for winning it, rounds are not completely independent. The reason for this is the economy. At the start of the round, players can buy guns, utility, armour and a defuse kit for the Counter-Terrorists. To do this, one needs to have money. The guns and utility can also be dropped to teammates, which can be useful if one player is richer than the other.

There are several ways to obtain money in the game. The first of which is killing enemies, the reward for this depends on the gun used to kill the enemy. This tends to be inversely proportional to how good the gun is. The better the gun, the lower the kill reward. As a consequence, it can at times be better to send people with lesser guns towards under-armed enemies, since they can receive more money that way.

The next way to obtain money is by using the bomb as a Terrorist. When the bomb is planted, the planter gets a little bit of money. After the round, every Terrorist gets a bonus as well. This means that in a round that cannot be won, it is still extremely useful to get the bomb down.

For the Counter-Terrorists on the other hand, a bonus gets awarded when they manage to defuse the bomb. It is, however, not advisable to let the bomb be planted only to be able to defuse it. This is due to the fact

that the reward for defusing the bomb is way smaller than the reward for the Terrorists receive for planting the bomb.

Lastly, there is always a bonus at the end of each round. When a team wins, they get a set amount of money at the end. For the losses however, the money is flexible since there is a loss bonus reward. This might sound weird, but if you lose more round, you will get more money. This helps to bring the losing side back in the game, but it is almost always smaller than the reward for winning a round. It should also be mentioned that a Terrorist only receives this reward if they died before the end of the timer, otherwise they get nothing [19].

The economy also gets influenced by players who survive the previous round. If they do, they still get all their equipment from the previous round. This means that if someone survives, they get the money for playing the round, but might not need to spend it, thus building up more money for when they need it in the future. It is also possible that someone does not have have much money, which influences their ability to purchase guns.

This leads to several different types of buys. The types are: eco, anti eco, half buy, force and full buy. They represent which guns are bought by a certain team. Since teams might have a worse buy than their opponent, teams do not have the same odds of winning each round [20].

This means that to have more chances in the next round, it might be better for a player to not attempt to win the current round since they will keep their gun. This phenomenon is called saving.

1.3 State of the Art

The current state of the art is centred around Leetify [21] and CS:GO stats [22]. Especially Leetify is head and shoulders above their other free services provided for CS:GO.

In Leetify, players get not only insight in how many kills they are getting, they can also compare their aim, which consists of headshot accuracy, accuracy, spray accuracy, counter-strafing, crosshair placement and time to damage, to players of their chosen skill-level. By doing this they get a good insight into which parts they are still behind the average player on this level. They can then train this if they wish to. The same holds true for utility, there is also a rich variety of statistics to analyze how well a certain player is using it compared to the average in their skill-level.

Where Leetify is centered around general aim and utility, CS:GO Stats is more focused on the performance with specific weapons. In CS:GO stats it is possible for players to see their accuracy on certain places on the body, split out between the weapons. This is useful since a player can get insight into how often they are hitting headshots with their pistols, where a headshot is usually the only useful hit, while with the AWP, it is usually best if the

player hits people in their chest or torso.

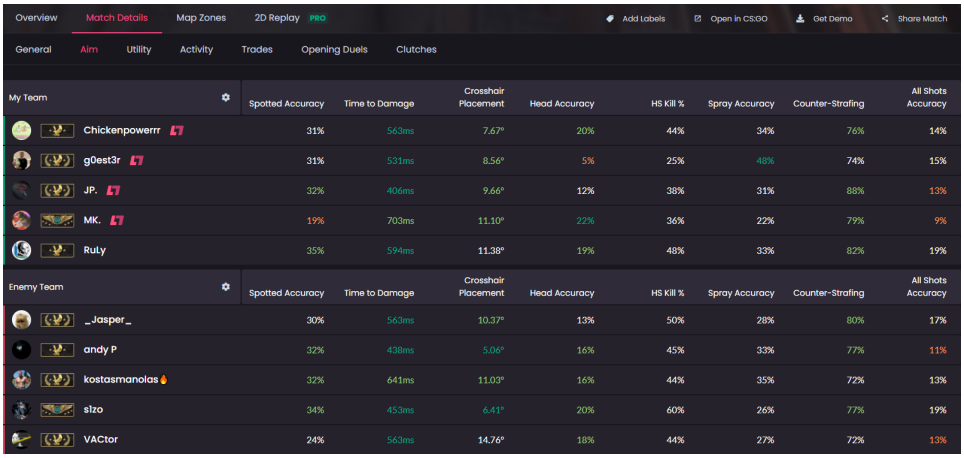

Figure 1.1: A Match in Leetify

These statistics however, are extremely centered around the performance of individual players and how they can improve. It does not give a lot of information on how a given team performs in a certain situations. For example, it can be nice to see for a team how they perform in certain economical situations, perform with a man down or in certain setups [A.1].

These websites also do not provide an open API, making it hard to use the information for research. While research is not the focus for most teams, it is for ETT, since they are also a research organization. It can for example be relevant how players react to stress or physical exercise [A.1] [A.5].

1.4 Vocabulary

- **Esports**: competition using video games
- **CS:GO**: an online first-person shooter. When referring to CS:GO we will always mean the competitive mode within CS:GO
- **ETT**: Esports Team Twente
- **IGL**: In-game Leader, the person who decides what a team is going to do
- **Sniper**: The main AWPer of a team
- **Entry Fragger**: The first Terrorist to enter a bombsite in an attack
- **Support**: The person supporting the Entry Fragger
- **Lurker**: The Terrorist staying behind during a push to cut of rotating Counter-Terrorists

- **Grenade**: Can be thrown to positions and will explode after a few seconds. It will damage to any person nearby the explosion
- **Flashbang**: Can be thrown and will pop after a few second. Anyone looking to it will be blind for a few seconds, depending on how close they were to it and how straight they were looking into it
- **Smoke**: Can be thrown and will pop after landing on the ground. It will make it impossible to see through it for a few seconds
- **Molotov**: Can be thrown and will pop after a few seconds, it is not on the ground by this time, it won't do anything, otherwise it will create flames on the ground for a few seconds, damaging anyone inside of them
- **Utility**: Grenades, Flashbangs, Smokes and Molotovs

Chapter 2

Problem Analysis

2.1 Stakeholders

- **Coaches**: The coaches of the teams that are competing in CS:GO.
- **Players**: There are several roles within a team [11] [12], since all of them have different needs, it makes sense to split them out.
  - **Ingame Leader**: The player who makes the important decisions within the round, for example on how a certain site should be approached.
  - **Entry Fragger**: The player on the terrorist side that takes first contact when entering a site.
  - **Support**: The player that supports their team members by baiting for them and throwing utility.
  - **Lurker**: The player who does not necessarily move together with the rest of the team, but moves to unexpected positions to catch opponents of guard.
  - **Sniper**: The player who uses the AWP.
  - **Anchor**: The player who stays on a site, unless it is confirmed that this site will not be the target of the Terrorist attack.
  - **Rotator**: Rotates to the site where more counter-terrorists are required.
- **Data Scientist**: Uses the data gathered by the platform and analyses it and/or combines it with their own data for their research.
- **Software Maintainer**: The person that has to maintain the developed software.

Chapter 3

Existing Product

3.1 Description

Before the start of the project, another group has already worked on the product in a previous design project. In this project, they created a website for both CS:GO and Rocket League analysis. This project version however, is not functional. While it is possible to login and the interface looks nice, no actual analysis of games is performed.

3.2 Frontend

While the frontend is functional with support for Rocket League and some placeholders in place for CS:GO, it uses the same page for displaying statistics for both games with very few things that could be changed per game. However, this kind of generalization does not work for the range of statistics we implement.

Because of this, we decided to separate the games in the frontend so that each game can have their own set of pages with game-specific information, with the current landing page instead acting as a portal to the implemented games. This will also make some of the information more organized as more games are added, as for example the teams for all games are currently grouped together.

3.3 Backend

In the backend, there is some structure, but really tightly couples the games by trying to generalize as much as possible. For example, results of matches are generalized to scores for each team. However, it is perfectly possible that new games get added in the future. These games do not necessarily need to work in the same way, meaning that the non-shared parts should be extracted again, which violates the Open/Closed Principle [23].

For this reason, it would be great if each game is its own part, which does not need to change if a new game is added. Also everything is grouped based on properties instead of on features. This means that for example all database related files are in one directory. In the future, this can mean that these directories get quite large, which makes it hard to search for specific things and add new futures.

The database models are also shared throughout the application. While this sounds better than having multiple models for the same thing, this also means that it will be harder to change the database provider in the future. If the database can get abstracted away, it does make it easier to swap it out later, if this ever becomes necessary.

3.4 Database

The used database is MongoDB. This sounds nice, since the developer does not need to think about the structure of the data since they can just post objects and MongoDB takes care of it, but it can become a burden in the future. There is no way in the application to write migrations should the models ever change and there is no model enforced in MongoDB. This brings a heavy burden on the developer to keep track of each possible mutation of an object that might be in the database, meaning that effectively there is still a model, but it needs to be taken care of while extracting it from the database. This can become a big mess when a model has been updated a few times.

Chapter 4

Mission Statement

4.1 Motivation

ETT wants to analyze how they are performing as a team in CS:GO and want to be able to use this information in research. To do this, there needs to be a platform where they can upload their games and they can be easily accessed later.

This is possible since CS:GO stores games in a so called DEM (short for demo) format. These files are created to be able to playback the match in the CS:GO demo viewer [24]. Since these files can be used to playback an entire match, these files contain a lot of information about the match. This information can be extracted and used for the system this project wants to create.

4.2 Type of System

Esports Team Twente already has a dashboard, in the form of a website (see Fig. 4.1), to display statistics. This dashboard is only a dummy implementation however, since it does only generate random data and does not extract the information from actual games. For this project, this dashboard will rewritten in such a way that the demo files get analyzed and the results are stored in a SQL database.

By storing the results in a SQL database, researchers can extract the information they need if they wish to, but there is also an API to simplify the process. The API is a standard HTTP API [25], which allows easy access for researchers, since all the relevant statistics for example can be requested with a simple HTTP request, which can be achieved using a lot of free publicly available tools and libraries in most programming languages [26] [27] [28].

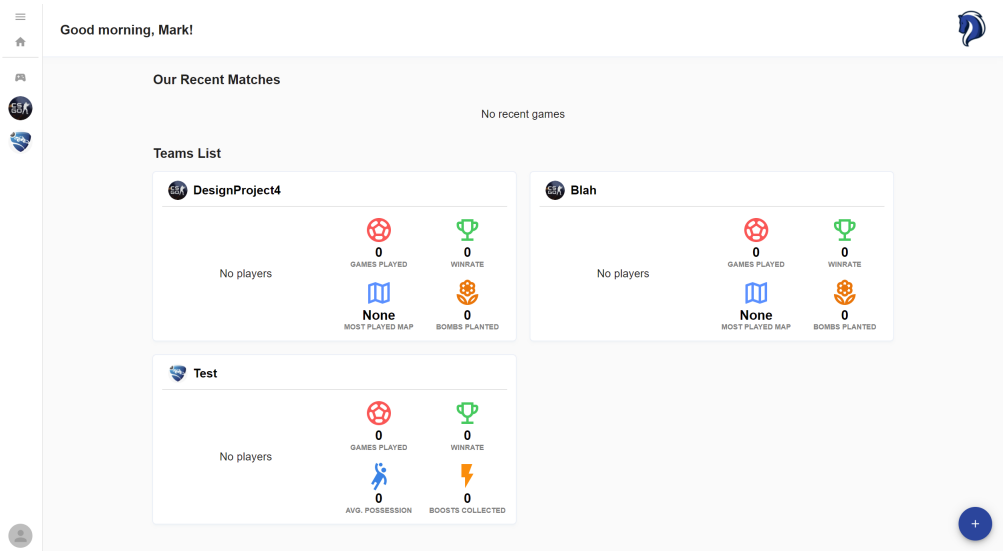

Figure 4.1: Pre-existing ETT dashboard

4.3 Project Goal

The goal of this project is to produce a system that is able to display statistics extracted from a CS:GO DEM file.

4.4 Exclusions

The data is only extracted from DEM files, which are not always available. When the teams are doing scrims to practice, they might only have screen recordings which cannot be analysed by the system.

4.5 Approach

Users will be able to upload DEM files to the dashboard. This file will then be uploaded to the backend, which will analyze the file and extract useful features.

These features are then stored into a SQL database, where they can later be accessed. It is then possible to easily access these features by using a HTTP API that groups all the information into easy to use categories.

Chapter 5

System-level Requirements

5.1 MoSCoW

To determine the further road map of the project, we will now visit the requirements. These requirements have been prioritized to determine where the focus on the project should go. This has been done using the MoSCoW (Must, Should, Could and Won't) model [29]. This model shows which parts must be completed in order to have a successful project and what can be done if there is time left.

Must

These requirements should be implemented in order to have a successful project. Without them, the project is incomplete.

- Show the results of recent matches
- Show an overview of all rounds in the back-end
- Display general statistics of a match (See 5.2 for specifics)
- Display general statistics of a single round (See 5.2 for specifics)
- Upload the CS:GO matches to the dashboard
- Refactor the existing back-end to be more modular

Should

These requirements should be in the project, but without them, the project is still complete. If they are not implemented it is not the end of the world, but it would be a huge loss to for the project.

- Be able to label rounds

- Be able to group labels under categories
- Be able to create categories/labels at will
- Evaluate players based on a specified role (Lurk, Entry, Support, see 5.2)
- Filter rounds based on the type of the round
- Filter rounds based on labels applied
- Display relevant statistics based on certain roles
- Display statistics of multiple rounds combined

Could

These requirements can be added to the project if there is time left after implementing the Musts and Shoulds. It would enrich the project, but it is not expected that all of them will be implemented.

- Be able to add comments to rounds and matches
- Be able to link VoDs to a match/round
- Show utility based statistics for flashes, grenades, and smokes (See 5.2)
- Show the change in economy related statistics over time (See 5.2)

Won't

These requirements would have been nice, but they are too much work to be added in the current project, so they definitely will not be implemented.

- Analyze screen recordings of matches

5.2 Statistics To Display

General

- Kills, indicates how many kills a player got, excluding kills of one's own teammates. Usually it is good to have a lot of kills, but some kills might be more important than other, since a kill with a sniper against a player with a low buy is usually way less impactful than killing a sniper with a low buy.

- Assists, indicates how many assists a player had, where one gets an assist when a player has done at least 41% of the damage on an enemy without killing them and there is no-one except for the killer with more damage dealt, and the enemy dies. A lot of assists could mean that someone is not great at finishing their kills.
- Deaths, how many times a player died, includes being killed by allies, fall damage and the bomb exploding. Some players might try to save a lot of their guns, which can be important for a sniper.
- Kill Death Ratio, simply $KD = \frac{Kills}{Deaths}$. A good Kill Death Ratio is frequently associated with a good game, but it does not tell the entire store. The kills might not be impactful or a player might try to hide instead of going for an attack which they should have joint.
- Average Damage per Round (ADR), the total damage done over the course of a match divided by the round tally. Having a high ADR, but few kills can indicate that a player is not finishing their kills, while a low one might indicate that a player is not hitting enough shots. AWPers might also have a low ADR since they do not deal a lot of damage outside of their kills.
- Headshot Percentage, the percentage of kills which were obtained with a headshot. A high headshot percentage might be desired when playing with an AK or a Desert Eagle, but a low one is desired when playing the AWP. The interpretation of this statistic is highly dependent on the weapons a player is using. It might also be better to aim for a body instead of the head when a player is already heavily damaged.

Role Specific

- Opening duels (Entry), how many opening duels were won by a player, where an opening duel is the first duel of a round with someone involved getting a kill. As the first one to enter a bombsite, getting kills to open the site is important.
- Clutches (2nd entry/lurk), where a clutch is registered when a player, as the last surviving player on their team, won the round. Having a lot of clutches as a team can indicate that the other players are dying too fast. Winning them is important, since it wins the round.
- Trades (2nd entry/lurk), where a trade is registered when a player kills the enemy who just killed one of the player's allies. This return kill must occur within 3 seconds of the allies' death. This is especially important for the 2nd entry since there is a good chance the entry will die. If the 2nd entry is able to trade their death, they can get more control over the site, while keeping the kills even on both sides.

- Potential smoke locations/heat maps (Overall/support), a heatmap of where smokes were thrown by a particular player. Being able to smoke the correct positions can help a team a lot while taking a bombsite, or delaying the other team from doing so.
- Flash Assists (Support), registered when a player throws a flash, and one of their allies subsequently kills an enemy who was blinded by that flash. Having a lot of flash assists is great, since it means that an easier fight is provided to the teammate.
- Enemies Flashed (Support), the amount of enemies that were flashed, even if briefly, by a player. This statistic might say something about if flashes are thrown at the right positions, but it might be misleading since they might also be used to clear out positions where a teammate might be. In this case they are still useful, but players who just throw a random flashbang if they know someone is there, but don't do anything with it might still be able to get a higher value.
- Flashtime (Support), the average amount of time spent 'flashed' (their screen was blank) for all Enemies Flashed by a player. The same logic applies to this statistic as to the enemies flashed, but it also says something about how effective the flash was on the enemies.
- Friends Flashed (Support), the amount of friendlies that were flashed by a player. Naturally, a player might want this value to be low, but lower is not necessarily better. A teammate can walk backwards to peek and wait for the flash, flashing them just a little bit, while the enemy is completely flashed. In this case the flash is still extremely useful, but the statistic might be a bit misleading.

Economy

- Money per round, the amount of money a team collectively has per round. This helps to see whether a team should have gone for a different buy type approach.
- Money per person, the amount of money each person has each round. This statistic can help to state whether a certain player should have dropped something to a teammate.
- Utility per person/round, the amount of utility (Flashes, Grenades, Molotovs, Smokes).
- Weapons per person/round, the amount of weapons owned by a person (Mainhand, and sidearm).

5.3 User Stories

As a coach I want...

- To be able to see how the team progresses
- The outcomes of the past matches
- To be able to see how my team performs with different states of the economy
- To be able to see how my team performs both sides (terrorist and counter-terrorist)
- To be able to see how my team performs at different stages of the game (pre-plant, post-plant)

As a player I want...

- To be able to analyze my FaceIt games
- To be able to analyze my matchmaking games
- To be able to analyze my matches on custom servers
- To be able to write comments on rounds, so that I can show the players what they should have done differently

As an ingame leader I want...

- To be able to analyze the statistics of rounds with the same type of strategy
- To be able to analyze the statistics of takes of the same bomb site

As an entry fragger I want...

- To be able to see my entry success with different weapons
- To be able to see my entry success at different bomb sites

As a support I want...

- To be able to see how long my enemies are blinded by my flashbangs
- To be able to see how much damage my HE grenades do
- To be able to see if there were enemies smoked of by my smokes
- To be able to see how many enemies were blinded by my flashbangs
- To be able to see how many teammates were blinded by my flashbangs

As a lurker I want...

- To be able to see how many people I killed with their knife out
- To be able to see how many people I killed on the bombsite opposite of where the bomb is planted

As a sniper I want...

- To be able to see how many kills I got with the AWP
- To be able to see how far I have to turn my mouse to get a kill
- To be able to see how many kills I get at different positions in the map

As an anchor I want...

- To be able to see how often my site was taken while I was at the other site
- To be able to see how often my site was taken after I died

As a rotator I want...

- To be able to see how often I was at the opposite site of where the bomb went down
- To be able to see how many kills I get on both bomb sites

5.4 Fulfilled Requirements Overview

Must Requirements	
Requirement	**Accomplished**
Show the results of recent matches	✓
Show an overview of all rounds in the back-end	✓
Display general statistics of a match	✓
Display general statistics of a single round	✓
Upload the CS:GO matches to the dashboard	✓
Refactor the existing back-end to be more modular	✓

Table 5.1: Must Requirements Overview

Should Requirements	
Requirement	**Accomplished**
Be able to label rounds	✓
Be able to group labels under categories	✓
Be able to create categories/labels at will	✓
Evaluate players based on a specified role	✗[1]
Filter rounds based on the type of the round	✓[2]
Filter rounds based on labels applied	✓
Display relevant statistics based on certain roles	✓
Display statistics of multiple rounds combined	✓

Table 5.2: Should Requirements Overview

Could Requirements	
Requirement	**Accomplished**
Be able to add comments to rounds and matches	✓
Be able to link VoDs to a match/round	✓
Show utility based statistics	✗[3]
Show the change in statistics over time	✗[4]

Table 5.3: Could Requirements Overview

[1] Many of the role statistics are implemented, but still need to be grouped per role
[2] Can filter based on map, date, and match id
[3] Advanced flash statistics exist, but grenades and smokes are missing
[4] Not implemented due to time constraints

Implemented Statistics

General Statistics	
Statistic	**Accomplished**
Kills	✓
Assists	✓
Deaths	✓
Kill death ratio	✓
Average damage per round	✓
Headshot percentage	✓

Table 5.4: General Statistics

Role Statistics	
Statistic	**Accomplished**
Opening duels (Entry)	✓
Trading	✓
Clutches	✗
Potential smoke locations/heat maps	✗[5]
Flash assists	✓
Enemies/Friends flashed	✓
Flashtime	✓

Table 5.5: Role Statistics

Economy Statistics	
Statistic	**Accomplished**
Money per round	✗
Enemy money per round	✗
Money per person	✗
Utility per person/round	✗
Weapons per person/round	✗

Table 5.6: Economy Statistics

[5]Out of scope due to complexity

Review

Whilst we completed all of our Must requirements, and almost all of our Should requirements, we didn't manage to meet every last one of them. The primary reason for this is the back-end refactor (part of the Must requirements), which was in essence re-doing all of the work of a previous project. This required new database, test, and application infrastructure, all of which took a sizable amount of time to get right. As a consequence we were unfortunately not able to meet all the Could requirements, which is why the Economy statistics are entirely unimplemented. Likewise for the advanced utility statistics and role grouping. There are however still a lot of statistics that are implemented which can be seen in Figure 5.1 and Figure 5.2 where the statistics are shown for a single CS:GO round and all rounds that satisfy the given filters respectively.

Thankfully, this was done in discussion with the client, who preferred having a more modular architecture such that they can further expand the existing base, and add new games, in the future.

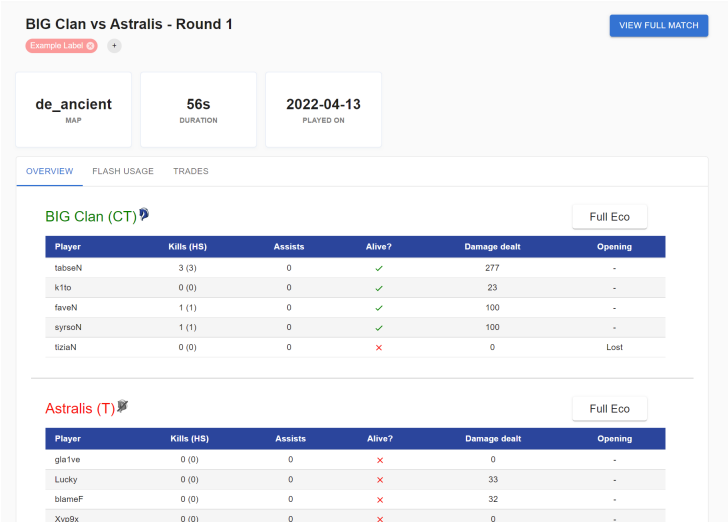

Figure 5.1: Overview of the statistics from a singe match. The user can select which statistics they want to see by changing from 'Overview' to the other tabs like 'Flash Usage'

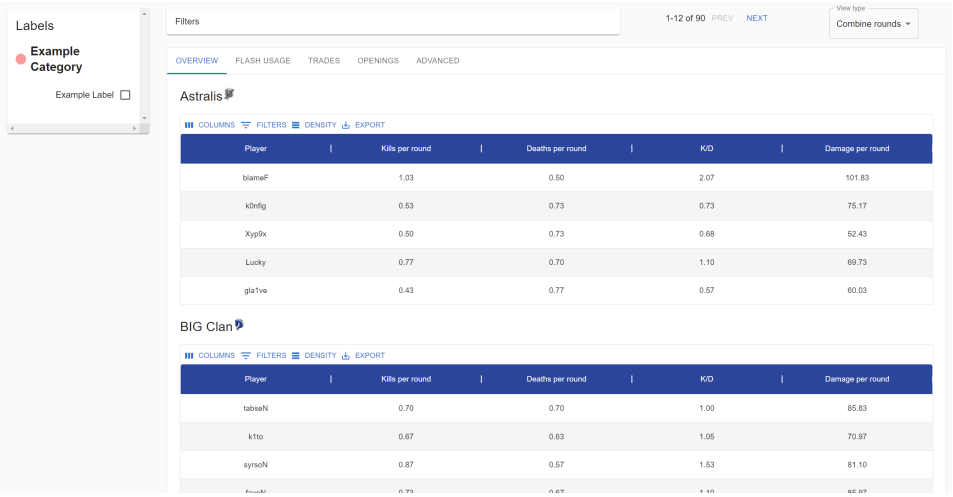

Figure 5.2: Overview of the statistics of the rounds that satisfy the given requirements, for example the map and the required labels. The user can select which statistics they want to see by changing from 'Overview' to the other tabs like 'Flash Usage'

Chapter 6

Tools

6.1 Frontend

The previous project was written in React[30] with Typescript, with MUI as the main user interface library. These are the most important libraries that have been used:

- **MUI**: MUI (formerly known as Material-UI) is a user interface library to streamline frontend visual design [31]
- **SWR**: SWR is a React library that acts as a wrapper around AJAX requests that checks constantly whether the requested data has been updated, and will automatically update the web page if anything has changed [32]
- **Formik**: Formik is a React library that allows for easier declaration and handling of forms, and validation of form data [33]
- **OpenApi Generator**: The backend hosts the OpenApi specification of its API. Using OpenApi Generator the frontend can generate the appropriate classes that interact with the API. Using this generation gives the benefit of using types thanks to TypeScript. Furthermore, it gives the benefit that whenever the backend changes its API, it is quite easy to modify the frontend accordingly. [34]

6.2 Backend

Since the previous project was written in Python and the clients asked to for the project to be written in Python, the backend is written in Python. To simplify development, there are also a few libraries that are used. The most important ones are listed:

- **FastAPI**: FastAPI is a small, but fast web framework, that allows us to write a HTTP API [35]
- **awpy**: awpy allows us to analyze CS:GO demos by extracting basic information, which can then be interpreted by the system [36]
- **SQLAlchemy**: SQLAlchemy is a SQL framework that allows for ORM models to be made, to abstract the database away [37]
- **Alembic**: Alembic ensures that changes in the schemas, based on the ORM models can be automatically detected and put into migrations. It will also automatically run them if needed [38]

Chapter 7

Architecture

7.1 Frontend

The previous team that started this project built the frontend with React and Typescript. We deemed this an appropriate choice and as such continued with this stack. The existing project tightly integrates all the games by having a shared dashboard between the games. Furthermore, most components where shared between the different games. This makes it difficult to customize pages for specific games or statistics. On top of that, it makes it difficult to add support for new games in the future. Therefore, we chose to refactor the frontend such that different games are completely separate, both in the user interface and in the code. This gives both us and potential future developers complete freedom on how to implement specific games into the dashboard.

The previous project organised the file structure mainly be type. For example, top-level directories included 'routes' and 'components'. We opted to refactor the file structure by feature, with top-level directories like 'csgo' and 'rocketleague'. Furthermore, the 'csgo' directory contains separate directories for each page, such as 'dashboardView' and 'labelView'. This way, all files and code for a certain feature are grouped together.

We have updated the Node.js packages used by the previous project. The version differences caused little issues. The only large change was Material-UI / MUI, which updated from major version 4 to version 5. We have used features from the latest version where possible, and updated some of the existing files. However, there are still parts of the project that currently use version 4 features as a full update would require much time for little benefit.

The frontend uses the backend extensively by making many API calls. It can often be difficult to keep the frontend and backend in sync with regards to the object models used for said API calls. The use of the OpenAPI Generator tool allows us to generate object models from the backend. By running this tool whenever the backend is updated, these models constantly reflect

the current backend. Because the frontend uses Typescript, the project will not compile if changes to the generated models break certain functionality. This makes it easy to keep the frontend in sync with the backend and minimize the chance of bugs occurring.

7.2 Backend

The existing project, was really focused on having common abstractions between multiple games. This forced for example matches to have teams and scores. While this kind of worked for the games they added dummy implementations for (CS:GO and Rocket League), we immediately started thinking about other games. After some talking to the client, it became clear that the old structure was clearly not the way to go, since there would be new games that could be added, which would constantly require a big refactor of all the data structures for each game. This violates the Open/Closed Principle [23].

Besides that, the architecture was really focused on functionality rather than feature. This means for example that there was a top-level directory called 'database' which contains the subdirectories 'schemas' and 'transactions'. While this does make it easy to find all database actions, it does make it harder to find all functionality related to for example players.

This is not a big problem while the project is still small, but as the project grows, this structure will start to crumble. It becomes harder to find for a new feature where all the requirements should be placed and if some features need other dependencies than most other requirements, new packages need to be added, further harming maintainability.

For this reason, we choose to go for a more modular approach. First of all, everything is now split on a game basis. This means that if a new game would need to be added, no changes are required to the other games. This means that the changes cannot hurt the other parts of the code, which is inline with the Open/Closed Principle.

Next, within each game, everything is split on a feature basis. This means that when a certain feature needs to be added, the other features can stay untouched, also inline with the Open/Closed Principle. This approach has helped the project already, by making sure that when a feature was ready, it would not all of the sudden break later in the project, saving precious development time.

The structure of the project now looks like this:

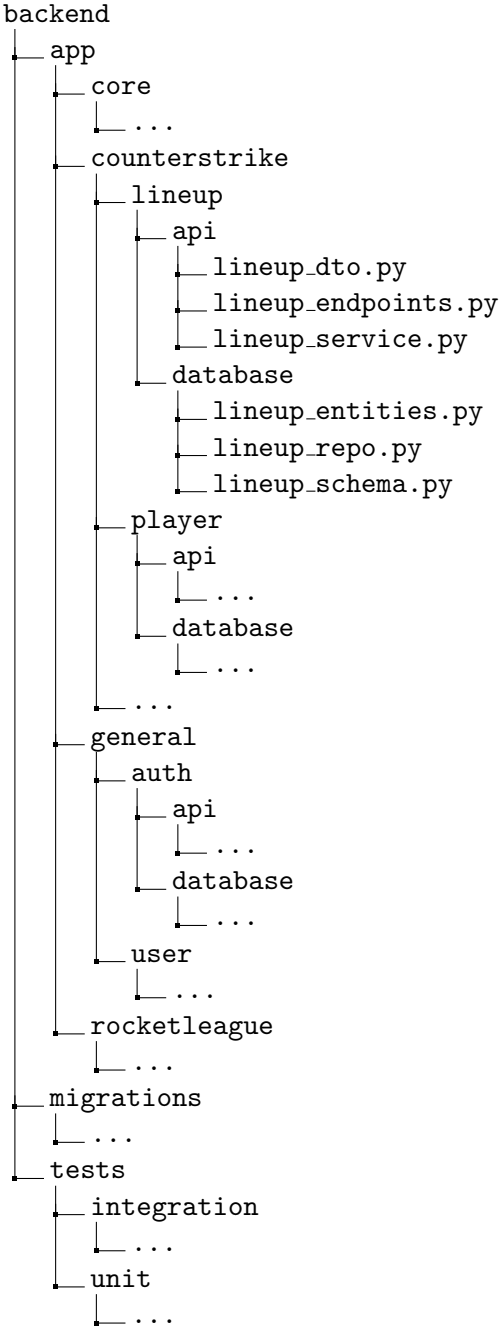

Figure 7.1: Project Structure

7.3 Database

The previous maintainers of the project used MongoDB, but did not do it in a very efficient or easy to use way. Since the main focus of this project is to create an environment which can be used by researchers to get data from CS:GO matches, it is very important for us to make the data accessible in an easy way.

While it is possible to enforce a schema in MongoDB [39], it is not as easy and common as in SQL. In SQL it is required to have a consistent schema, which ensures that everyone knows what to expect. Besides that, most students at the UT are more experienced with SQL than with MongoDB, which also makes it easier to maintain for future maintainers.

This is why PostgreSQL (Postgres) is used for this project. A nice feature of Postgres, it that it is possible to add schemas. This allows the project to have a general schema with objects shared across the application, but also allows each game to have its own schema. This makes it more maintainable to add more games in the future since only the schema for the new game will need to be edited. It also makes it clearer which tables are for which games.

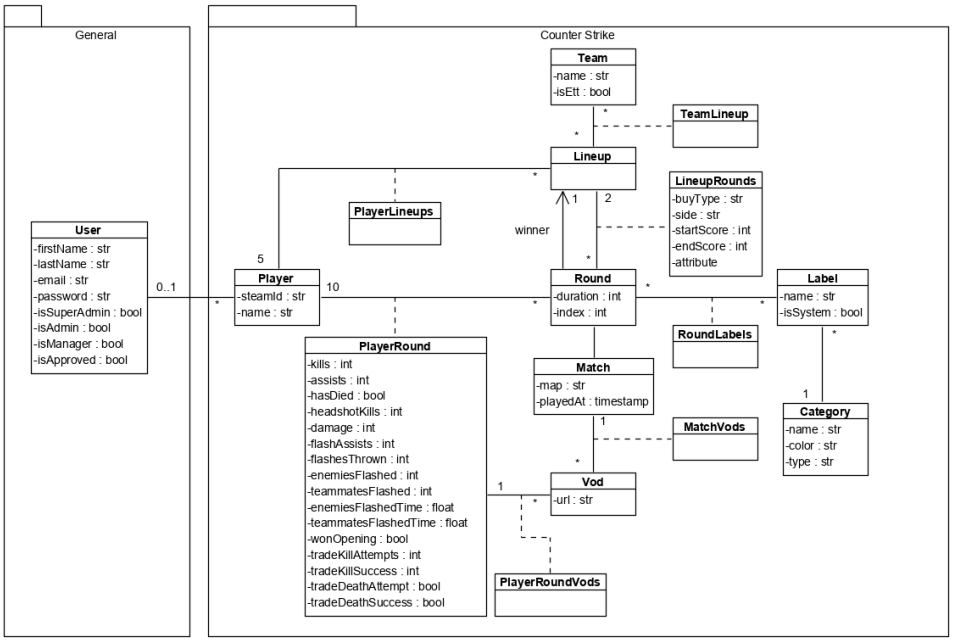

Figure 7.2: Database Diagram

In the current setup, most of the relevant statistics are stored in the PlayerRound table. These statistics show what a certain player did in a round. The reason for using a structure like this, is that now all statistics are based on individual players in individual rounds. This means that all

statistics can now be grouped by selecting certain rounds.

Traditionally, Counter Strike statistics are always shown as parts of matches, but for this project, we wanted to have more freedom in the statistics, such that the effect of certain changes, like strategies can be compared by grouping all these rounds together. This hopefully helps future researchers to filter on more specific data.

Since most players still like to think of statistics as game related, there is also a way to map all the rounds to matches. This makes sure that while there is more freedom in the grabbing of information, it is still possible to get the traditional statistics.

Chapter 8

System Testing

8.1 Frontend

Due to numerous factors, frontend testing can be quite difficult in any project. These reasons include the fact that websites can be visited on a number of different platforms, devices and browsers. Furthermore, in this project the primary task of the frontend is to provide an User Interface to display the content requested from the backend. There is not a lot of computation or processing done in the frontend. User Interfaces are inherently difficult to test because they depend on the content that it displays. Therefore, it is difficult to test a User Interface in isolation.

Integration testing

In order to have some basic testing in the frontend, the Cypress framework is used. This framework works by launching the website in a browser and simulating user input. Cypress is used in the frontend for integration tests. These integration test are mostly focused on the more basic features of the website. For example, logging in, registration, navigation to certain pages, and testing for permissions.

8.2 Backend

For the backend, both unit tests and integration tests are used. The unit tests can run way faster, making them ideal to quickly check if a refactoring did not cause obvious errors.

Unit Testing

Unit testing provides the developer with a quick way to verify whether small changes do not entirely break the code. This is achieved by testing a lot of very small isolated parts of the system.

Since the unit tests should not depend on each other they can easily run in parallel and since they should not need a web server or a database, they can run significantly faster then one could dream of ever achieving by starting up the application and trying to test whether the feature works. Besides that, it also immediately checks all the other parts of the code, which if for some unexpected reason break as a result of the change, should also result in failing tests, making it easier to find bugs.

Since the unit tests are meant for quick iterations, they should not take longer than five seconds to run, since that would discourage the developer from running them after refactoring small parts of the code. To ensure that they are always run before the code reaches production, the unit tests are also automatically run after a push or pull request to the repository.

Integration Testing

While unit testing gives quick feedback to the developer, making it ideal to verify whether refactoring the code does not break it immediately, it does not give a lot of information about how all the parts of the system work together. To test this, there are integration tests, which test how all the components work together.

The integration tests are focused on testing the two core parts of the project, namely API routes, and match analyses. The former ensures that the API is functional (and doesn't get broken during refactoring!) and that the database interactions are working correctly. The latter, in turn, ensures that the core goal of the project (CS:GO match analysis) is accomplished. It checks whether the expected statistics are extracted, *and* stored correctly.

While integration tests are great to test whether the system works, they are significantly slower than unit tests since they require the application to run, including a database and the web server. For this reason, they are great to test whether an entire feature works after implementing, but not for short refactors.

Because of the slower nature of the integration tests, they are usually not run by the developers of the project on their computer, but remotely. This is done by automatically running the tests on a pull request or a push to the Git repository with our CI pipeline. By doing this, the integration tests do not stall the computer of the developer, but they must always pass before the code can be merged into the main branch.

Chapter 9

Risk Analysis

Every project contains possible risks. In this section we will analyze possible risks and how to prevent them.

9.1 Scope creep

One of the risks that has already been identified at the start of the project is scope creep. It is vital that there is a well defined scope, including features that might be nice to have but not absolutely necessary. If we do not define the scope and all the requirements correctly beforehand, we run the risk of continuously adding new features. This could result in a finished product with lots of unfinished or badly implemented features. Instead, it is better to define all features beforehand, stick to that and implement all features really well. Of course, if time allows it should always be possible to add new features. But only if all core features have been implemented.

9.2 Incorrect time estimations

A major issue in every project is incorrect time estimations. Often, features require more time and effort to implement than estimated. In order to handle this, progress will be reviewed at every sprint review. If we are behind schedule action will be taken accordingly in consultation with the product owner. This could include scrapping the feature or reducing the size of the feature.

9.3 Unexpected member loss

It is possible that member(s) of the group have to stop with the module halfway through the project. This could be for all kinds of reasons, like sickness or other personal circumstances. If this happens, the scope of the project will have to be adjusted accordingly.

Chapter 10

Team Overview

10.1 Roles

Within the project there are a few separations of the roles within the team. The first split is between the frontend and backend. Both teams consist of two persons. The next major part of the project is the ethics report. This part is again done by two people, just like the last split part, which is the poster. All the other parts like the design report, presentations, interviews, meetings and peer-reviews have been worked on by all team members.

Arjan Blankestijn

During the project, Arjan has primarily focused on the frontend of the project, but when necessary, also did a little bit on the backend. They also worked on the poster for the project.

Mark Kok

From the start of the project, Mark Kok has worked on the frontend. At the end of the project they worked together with Arjan on the poster.

Mark van Wijk

During the project they worked on the back-end, as well as the ethics report, alongside Valentijn Hol

Valentijn Hol

During the project they worked on the back-end, as well as the ethics report, alongside Mark van Wijk.

10.2 Working Method

Since the team wanted to be flexible, an agile approach [40] was taken. With this, the team decided to have daily standups to ensure that everyone is always up-to-date on the state of the project and knows what the other members need to go on with their work. This also means that if there are any misconceptions, they can be caught early on, saving time later.

This approach also means that the project has been divided into sprints. With the project taking 10 weeks and sprints of 2 weeks, there are a total of 5 sprints, which are explained below. Each sprint is closed with a meeting with the client and the supervisor to ensure that they are aware of the progress made. For the client it also functions as a way to get feedback on the project to make sure that the project actually does what they want it to do.

Sprint 0

During this sprint, the project has primarily focused on finding a supervisor and gathering requirements. To gather the requirements, existing product were investigated [21] [41] [22] [42] and the client was interviewed, since we had to understand what they wanted that was not available in publicly available tools. During this time, we also checked the existing code base to determine the plan for the rest of the project.

Sprint 1

During Sprint 0 it became clear that the backend, contradictory to what the client thought, was non-functional. This meant that no actual data was analyzed yet, so this had to be implemented. There was already a structure for the backend, but it was not well-suited for the project that was requested by the client. For this reason, Sprint 1 has primarily focused on trying to rewrite the backend.

Sprint 2

At the start of the sprint, the focus was still very much on the rewrite of the backend, but there was also time to focus on the analysis of the data. This means that the pipeline could be used to handle incoming demo files from uploads and extract the basic statistics which could be stored in the database and returned via the API.

Sprint 3

Sprint 3 has been the sprint with the most new features to the project. With the backend now being suited to add new features rapidly, more advanced

statistics could be extracted from the games. This data could in its turn be accessed by the newly added API endpoints and the frontend that now supported the anticipated endpoints as well.

Sprint 4

This final sprint has primarily focused on wrapping up the project. This means that the poster and ethics report were written in this sprint, while also making the last touches to the program and finishing this design report.

Chapter 11

Prototypes

We'll briefly cover each iteration of our project's prototype, which was shown to the client at the end of every sprint. In total there were 4 prototypes, excluding the final version.

11.1 Prototype Zero

The very first prototype we made is not an actual working prototype, rather it is a couple of mockups of the most important pages, like the search page and match overview page. The mockups are made with Figma [43]. In these mockups we tried to design a rough layout of the page. The mockups where shown to the project owner who gave feedback on them. The mockups together with the feedback on them where used for the next prototype.

Figure 11.1: A mockup of the search page.

11.2 Prototype One

For the first real prototypes we rewrote the existing back-end to rely upon Postgresql, instead of MongoDB. We also restructured the remainder of the prior work to be more modular for future development. The initial demo analysis was also implemented.

On the frontend the project was restructured as well, where everything would now be centered around a 'portal' leading to each game's respective dashboard as seen in Fig. 11.2. The CS:GO dashboard received its first iteration of the search page, with some basic functionality and display implemented. Basic, unstyled, match and round statistics pages were also created.

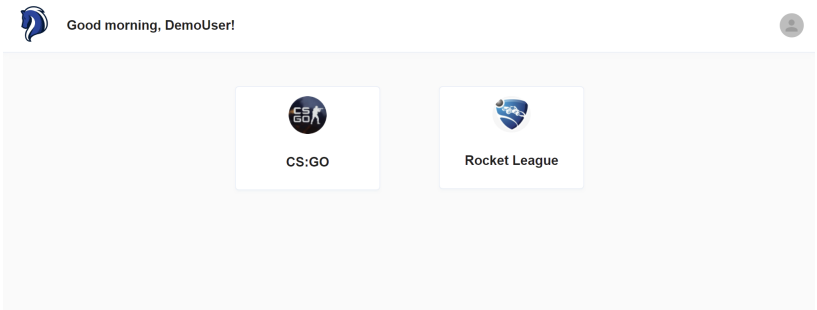

Figure 11.2: The landing page as a portal

11.3 Prototype Two

This prototype focused on fleshing out the remainder of the back-end, and polishing the upload/analysis feature. The feature was moved to be based on a pipeline architecture for easier composability. In addition, many API endpoints for teams/categories/labels/players were added.

For the frontend the focus was also on fleshing out the initial prototype versions of the pages from the last sprint. The search page (Fig. 11.3) was made fully functional and given a better look. The match and round statistics pages were further fleshed out and given a more uniform look with the rest of the application.

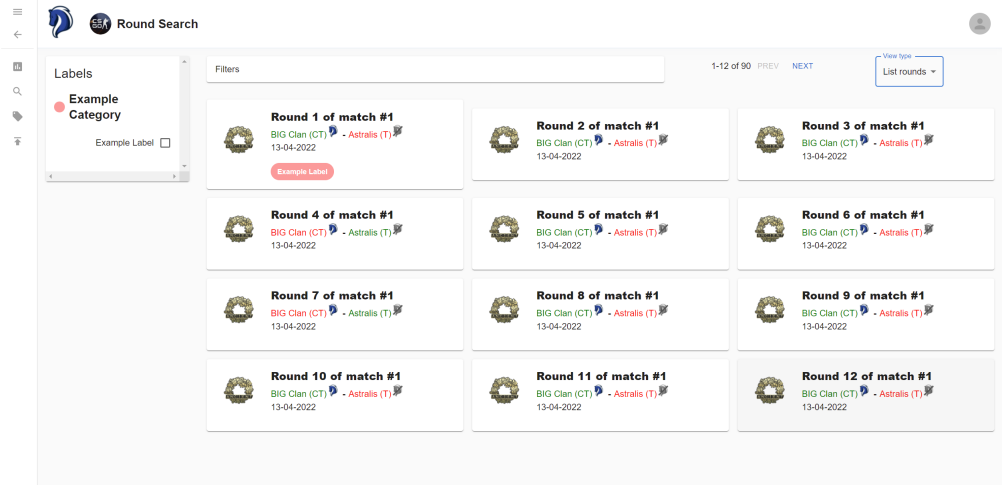

Figure 11.3: Search page

11.4 Prototype Three

The last prototype version received a lot of additions in the back-end. More advanced statistics pertaining to trades and flashes were added. More API endpoints for linking players to users, and teams to lineups were also added.

Lastly, the frontend fleshed out the homepage of the new CS:GO dashboard, giving an overview of recent matches. It also received displays for the advanced statistics added in the backend (Fig. 11.4). Last but not least it received the combined statistics overview, showing the combined and aggregate statistics of an arbitrary set of rounds.

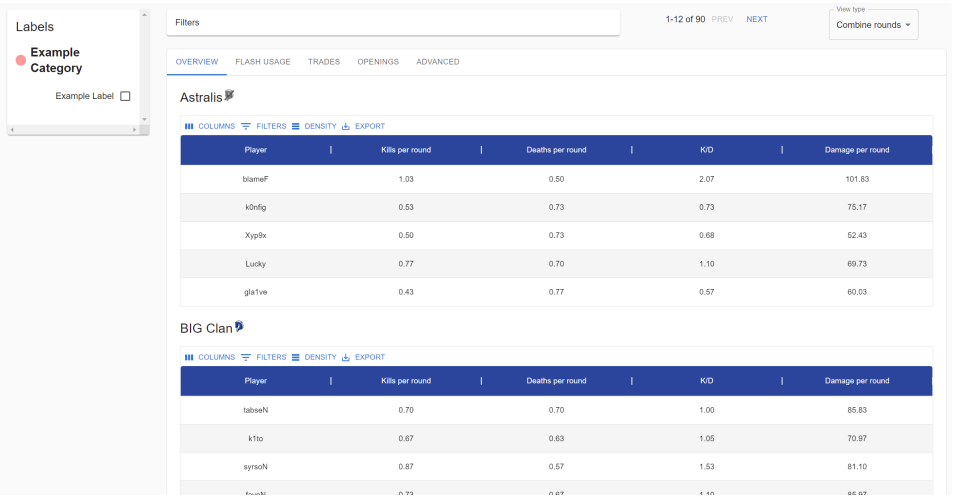

Figure 11.4: Overview of combined statistics of multiple rounds

Chapter 12

Future Work

Whilst many features were completed there is a lot of work that still remains for future implementers to work on. We will cover both the front, and back-end together, as both will be intimately entwined when it comes to new features.

12.1 CS:GO Features

The base for CS is now finished, but many additional features are possible. We will cover a few of them in detail.

Statistics Over Time (Economy)

The statistics over time were a 'could' requirement for us, and we unfortunately ran out of time to implement it. This would involve recording the state of each player's cash flow from the demo file, and aggregate statistics like team total would flow based from that. This could then be displayed in the frontend on graph basis.

Smoke Analysis

A desired feature by the client was smoke analysis/hot spots. This feature, while complex, would be a great addition for future developers to work on. Note that this would most likely involve making modifications to Awpy's analysis to extract additional information.

Demo Persisting

Currently demos are analysed once and the required statistics saved in the database. If one were to add additional statistics this could be troublesome for prior matches, as they would lack these items. Persisting the Awpy

JSON analysis file (which is relatively small, only a few megabytes) could be useful to alleviate the aforementioned issue.

12.2 Dashboard Features

The dashboard as a whole should now be ready for the addition of new games if the need be. The main feature that needs to be implemented is the re-addition of Rocket League. In the old dashboard an initial prototype of Rocket League match analysis was implemented, but this was removed in our restructuring of the project as it was out of scope for us. This could therefore now be re-implemented by future developers.

Responsiveness

While mobile-friendliness was not a focal point of the project, we still made the frontend responsive where possible. However, this was not a high priority, and more emphasis could be placed on responsiveness by future developers.

Authorization

At present the authorization flows are very basic, and not particularly up-to-date with the remainder of the application. They need an overhaul for future development and security.

Bibliography

[1] "Esports team twente." [Online]. Available: https://esportsteamtwente.nl/

[2] "Most played & most popular games in the world (2022)." [Online]. Available: https://gamertweak.com/most-played-popular-games/

[3] "Top global esports games by tournament prize pool in 2021." [Online]. Available: https://www.statista.com/statistics/501853/leading-esports-games-worldwide-total-prize-pool/

[4] "The biggest esports games every gamer should know." [Online]. Available: https://influencermarketinghub.com/biggest-esports-games/

[5] "Counter-strike wiki." [Online]. Available: https://counterstrike.fandom.com/wiki/Competitive

[6] "How to get out of silver in cs:go?!" [Online]. Available: https://www.youtube.com/watch?v=K6XULy_fllw

[7] "Flashbang." [Online]. Available: https://counterstrike.fandom.com/wiki/Flashbang

[8] "He grenade." [Online]. Available: https://counterstrike.fandom.com/wiki/HE_Grenade

[9] "Smoke grenade." [Online]. Available: https://counterstrike.fandom.com/wiki/Smoke_Grenade

[10] "Molotov." [Online]. Available: https://counterstrike.fandom.com/wiki/Molotov

[11] "Understanding roles in csgo." [Online]. Available: https://blog.leetify.com/understanding-roles-in-csgo/

[12] "Cs:go roles — cs:go player roles and functions explained." [Online]. Available: https://www.pinnacle.com/en/esports-hub/betting-articles/cs-go/a-guide-to-csgo-role/ml2jx57tyd6bxr7z

[13] "Understanding the in-game leader role." [Online]. Available: https://dignitas.gg/articles/blogs/CSGO/7529/understanding-the-in-game-leader-role

[14] "Are fragging igls a myth in cs:go?" [Online]. Available: https://www.hotspawn.com/csgo/news/are-fragging-igl-a-myth-in-csgo

[15] "Awp." [Online]. Available: https://counterstrike.fandom.com/wiki/AWP

[16] "Learning roles with dignitas cs:go - being the awper." [Online]. Available: https://dignitas.gg/articles/blogs/CSGO/14279/learning-roles-with-dignitas-csgo-being-the-awper

[17] "So you want to be a support in csgo - a role guide." [Online]. Available: https://dignitas.gg/articles/so-you-want-to-be-a-support-in-csgo

[18] "An in depth guide looking at the lurker position in csgo." [Online]. Available: https://dignitas.gg/articles/blogs/CSGO/7967/an-in-depth-guide-looking-at-the-lurker-position-in-csgo

[19] "Csgo economy guide (2020): How to best manage your money." [Online]. Available: https://news.unikrn.com/article/csgo-economy-guide-money-management-ct-t-terrorists-kill-bonus-win-loss-s_rs

[20] "Understanding economy in cs:go." [Online]. Available: https://www.hotspawn.com/csgo/guides/understanding-economy-in-csgo

[21] "Leetify." [Online]. Available: https://beta.leetify.com/

[22] "Cs:go stats." [Online]. Available: https://csgostats.gg/

[23] B. Meyer, *Object-Oriented Software Construction*. Pretice Hall, 1988.

[24] "Dem format - valve developer community." [Online]. Available: https://developer.valvesoftware.com/wiki/DEM_Format

[25] "What are rest apis? http api vs rest api." [Online]. Available: https://www.educative.io/blog/what-are-rest-apis#http

[26] "Postman api platform." [Online]. Available: https://www.postman.com/

[27] "Python's requests library (guide) - real python." [Online]. Available: https://realpython.com/python-requests/

[28] "Do a simple http request in java — baeldung." [Online]. Available: https://www.baeldung.com/java-http-request

[29] R. B. Dai Clegg, *Case Method Fast-Track: A RAD Approach.* Addison-Wesley, 1994.

[30] "React." [Online]. Available: https://reactjs.org

[31] "Mui." [Online]. Available: https://mui.com

[32] "Swr." [Online]. Available: https://swr.vercel.app

[33] "Formik." [Online]. Available: https://formik.org

[34] "Openapi generator." [Online]. Available: https://openapi-generator.tech/

[35] "Fastapi." [Online]. Available: https://fastapi.tiangolo.com/

[36] "awpy." [Online]. Available: https://awpy.readthedocs.io/en/latest/

[37] "Sqlalchemy - the database toolkit for python." [Online]. Available: https://www.sqlalchemy.org/

[38] "Alembic." [Online]. Available: https://alembic.sqlalchemy.org/en/latest/

[39] "Enforce a schema mongodb." [Online]. Available: https://docs.mongodb.com/realm/schemas/enforce-a-schema/

[40] "Manifesto for agile software development." [Online]. Available: https://agilemanifesto.org/

[41] "Hltv." [Online]. Available: https://www.hltv.org/

[42] "Faceit." [Online]. Available: https://www.faceit.com/

[43] "Figma." [Online]. Available: https://figma.com/

Appendix A

Interviews

A.1 Introduction

This interview was conducted on the 11th of February 2022 **Interviewee**: Vanuit ETT (Esports Team Twente) is er vorig jaar ook een design project geweest voor het bouwen van een dashboard. Dit dashboard wordt gebruikt voor data analyse, data collectie en een hoop statistiek. Daar is documentatie voor geschreven. In dit systeem moeten meerdere games worden toegevoegd.

Nu draaien we binnen ETT met een aantal spellen mee en vooral CS:GO (Counter Strike: Global Offensive) is daar wel een hele mooie voor. Ze hebben demo files zoals ze heten, waar een heleboel informatie inzit. De vraag is om een connectie te maken tussen het dashboard en de demo file zodat dit omgezet kan worden naar iets dat het dashboard kan gebruiken.

Wat ik zelf een beetje zie is dat het maken van die connectie, dus het extracten en importeren in het dashboard, dat is de grootste stap die moet gebeuren. Ik denk dat dit goed te behalen is binnen een aantal weken. Dan zou er nog een scope verandering mogelijk zijn om ook wat moeilijkere statistieken te berekenen.

Interviewer: Voor het begin, wat zijn de statistieken die we uit de demo file moeten halen?

Interviewee: Wat een demo file eigenlijk is, is een registratie van elke frame binnen een match, dus de ticks. Elke tick heeft een lijst aan standaard statistieken, dus player locaties, kills, deaths en damage. De statistieken die we sowieso headshot percentage, average damage, alle kill death statistieken van een bepaald moment en dat soort zaken. Dat zijn de basis statistieken. Ik kan misschien wel een lijst maken van alle statistieken die handig zijn in het spel zelf. De basis statistieken die makkelijk te halen zijn uit de demo files zelf zijn inderdaad headshot percentage, average damage per round, eigenlijk de statistieken die CS:GO zelf al laat zien. Dat zou de eerste stap zijn. Het extracten van die data uit de demo, het importeren van die

informatie naar de pipelines die we al hebben opgezet en daarna weergeven in het dashboard.

Interviewer: Want je zegt dat het dashboard grotendeels al gemaakt is. Is ook de statistic display al gemaakt, of moeten we daar nog wat aan doen?

Interviewee: Daar moet je nog een deel doen. Een aantal zaken zoals standaard tabellen zijn al gedaan, maar als je gebruik wil gaan maken van iets van een heat map zal je daar zelf ook nog veel frontend voor moeten doen.

Interviewer:En de scripts zelf moeten geïntegreerd worden met de back-end repository?

Interviewee: Ja, klopt. Op dit moment wordt er voor de backend gebruik gemaakt van Python met FastAPI en een MongoDB database en er zijn een aantal pipelines binnen Docker en Gitlab. Eentje daarvan staat volgens mij ook beschreven in de documentatie. Je upload aan de ene kant van de pipeline je bestanden, vervolgens kunnen jullie een heleboel scripts schrijven die je informatie extracten en omzetten naar een ander format. Aan het einde van de pipeline kan je dat dan weer pushen.

Interviewer: En dat kan dan weer worden opgeslagen in de MongoDB database?

Interviewee: Ja, dat kan.

Interviewer: Wat is de reden dat iets als Leetify of CSGOStats niet gebruikt wordt maar dat jullie het los willen hebben?

Interviewee: Klopt, Leetify is een prachtige tool voor dit soort dingen. De grootste reden is eigenlijk dat we binnen het ESports team focussen op performance van spelers, maar vooral ook voor onderzoek. Onze eigen tools maken heeft dus heel erg veel nut omdat je eigenlijk alles erop kan loslaten wat je wil. Het is voor ons heel erg nuttig om de kennis te hebben van de data sets voor dit soort spellen en er dus ook veel kunnen verzamelen en ook op een goede manier op onze database op te kunnen slaan en als iemand bijvoorbeeld een machine learning project wil gaan doen, deze data daar bijvoorbeeld voor kan gebruiken.

Interviewer: Ik zie ook dat jullie eye tracking bijvoorbeeld gebruiken, is dat ook data die je bijvoorbeeld wil kunnen combineren met de data die we uit de demo files halen?

Interviewee: Bijvoorbeeld. Het systeem is eigenlijk gebouwd om zoveel mogelijk types informatie op te kunnen slaan en ook te kunnen combineren. Je kan daar Leetify of FaceIt voor gebruiken, maar alleen FaceIt heeft een open API en Leetify niet. Het is dus handig om die informatie zelf te hebben. Dat is eigenlijk de grootste reden dat we het zelf doen.

Interviewer: Verder stond er ook iets in het document over screen recordings. Is dat ook data die we moeten analyseren? Of moeten we het voornamelijk uit de demo files trekken?

Interviewee: We hadden gekeken naar screen recordings of we daar een

stukje koppeling tussen kunnen maken, dus bijvoorbeeld een stukje screen recording met een bepaalde round. Ik zou zeggen dat het later in de scope zit, of waarschijnlijk helemaal niet in de scope zit omdat het echt veel tijd gaat kosten denk ik. In principe zou de focus zijn demo files en een mooie statistische weergave, maar als verder gaande stap zou het mooi zijn als je bijvoorbeeld comments kan toevoegen aan bepaalde rounds of dat je bepaalde moment kan opslaan of dat je bijvoorbeeld een clip van een screen recording kan toevoegen aan een bepaalde round.

Interviewer: Voor die statistics, is dat een lijst die jullie graag zelf willen maken en wij die alleen implementeren? Of hebben jullie liever dat wij daar ook input voor leveren? Je hebt bijvoorbeeld dingen als HLTV ratings. HLTV geeft bijvoorbeeld niet zo heel erg veel om je economy, dat doet Leetify al wel, maar dat is ook niet heel erg fantastisch omdat je iedereen probeert te beoordelen op dezelfde metric, terwijl een ingame leader bijvoorbeeld minder zou hoeven te fraggen. Wil je dat soort metrics bijvoorbeeld per role hebben om te kijken hoe goed iemand bijvoorbeeld zijn utility gebruikt en daar een rating voor support voor krijgt?

Interviewee: Als je hem zou neerzetten als entry fragger dan beoordeelt hij hem slecht, maar als je hem zou neerzetten als support dan beoordeelt hij hem goed?

Interviewer: Precies.

Interviewee: Dat zijn natuurlijk mooie statistieken. Als jullie mee willen denken en willen brainstormen over de statistieken, wat jij nu bijvoorbeeld doet, dat zijn wel hele mooie zaken. Misschien kunnen we volgende week wel even brainstormen over de exacte statistieken die we in de scope zouden willen.

Interviewer: Dan misschien ook nog met wat andere mensen die daar een mening over hebben? Vooral de gebruikers hebben daar waarschijnlijk nog wel een mening over.

Interviewee: Klopt, ik zal een aantal mensen van het CS:GO team ook vragen.

Interviewer: Ik denk dan dat de focus tot die meeting is om te kijken naar de basis extractie, dus de headshots en dergelijken en exploreren hoe het project eruit ziet. Praktisch gezien, hoe wil je op de hoogte worden gehouden? Wil je een keer in de twee weken een presentatie of eens in de week een meeting?

Interviewee: Wat jullie het makkelijkste vinden. Als ik 1x in de week een update krijg is dat top. Dit kan voor kleinere updates per mail, maar een keer in de twee weken een meeting is wel handig denk ik. Voor nu zou ik zeggen dat we volgende week weer even een belletje doen voor de brainstorm en ook zodat jullie even naar het project hebben kunnen kijken.

Interviewer: Kunnen wij dan ook toegang krijgen tot waar we mee moeten integreren?

Interviewee: Ja, ik zal jullie toevoegen aan de Github.

Interviewer: Is het ook mogelijk om toegang te krijgen tot het dashboard?

Interviewee: Ja, ik zal nieuwe accounts voor jullie laten aanmaken.

Interviewer: Is het de bedoeling dat demo files automatisch uit FaceIt worden getrokken? Of moeten ze handmatig worden geüpload?

Interviewee: Het makkelijkst is uploaden. FaceIt is top, maar het liefst willen we uploaden. De reden daarvoor is dat we bijvoorbeeld matchmaking matches erin kunnen zetten en als je een toernooi speelt op custom servers, dan kun je ook zelf je eigen demos recorden. Dat is ook de reden dat we screen recordings noemden, want als je bijvoorbeeld scrims speelt tegen andere teams, dan is het de standaard dat je geen demo record omdat je dan geen analyse kan maken van de strategie van de tegenstander. Dan worden er standaard screen recordings gebruikt, maar dat lijkt me een stap verder.

A.2 Requirements

This interview was conducted on the 16th of February 2022 **Interviewee**: Dus, hoe ging het de laatste paar dagen?

Interviewer: We zijn de backend grotendeels aan het herschrijven om het testbaar te maken, want op het moment is dat niet echt mogelijk. Op de frontend zijn we nog niet echt bezig, dat komt later.

Interviewee: Jullie hebben een eigen Docker opgezet om het te runnen?

Interviewer: Ja. Hoe deployen jullie de applicatie op het moment?

Interviewee: Volgens mij hebben we een Docker instance draaien op een van de servers, maar dat weet ik niet zeker.

Interviewer: Met een losse MongoDB instance?

Interviewee: Met een losse docker-compose. We hebben twee images. Het is een TransIP server namelijk. Even kijken... Ja, op het moment wordt het gehost op een Ubuntu instance met een MongoDB instance eraan.

Interviewer: Toen we de frontend aan het builden waren leek het dat het alle API requests doorstuurt naar de API server van het vorige team dat aan het project werkte.

Interviewee: Echt waar?

Interviewer: Ja, zo zit de code op het moment in elkaar.

Interviewee: Is het niet als environment variabele toegevoegd maar gewoon gehardcode?

Interviewer: Ja.

Interviewee: Oke, dat is niet handig. Ik ga het even vragen aan degene die de server beheert.

Interviewee: Is het voor de rest gelukt om de backend en frontend te draaien?

Interviewer: Ja, dat is gelukt.

Interviewer: We hebben voor vandaag een lijstje met vragen voorbereid, ik weet niet of jij nog iets hebt, maar anders kunnen we die afgaan.

Interviewee: Prima, ik heb een lijstje gemaakt van alle statistics die we graag zouden willen zien, de standaard statistics en een beetje per role wat handig is om te hebben.

Interviewer: Dat is gelijk de eerste vraag, welke statistieken wil je hebben?

Interviewee:

Statistics

- Kills
- Assists
- Deaths
- K/D
- ADR
- HS%

Additional Statistics

- Opening duels - Entry
- Clutches - 2nd entry/lurk
- Potential smoke locations/heatmaps - Overall/support
- Trading - 2nd entry
- Flash assists - Enemies flashed - Flashtime - Friends flashed - Support
- Per role stats -¿ Support, Entry, AWP-er, Lurk/Clutch, 2nd entry

Economy

- Money per round
- Enemy money per round
- Money per person
- Util per person/round
- Weapons per person/round

Interviewee: De eerste stap is om deze data te extracten, maar voor de frontend lijkt het me handig om te bedenken hoe we deze data kunnen indelen, of dat via tabjes gaat zijn, of dat we een overzicht hebben qua round view.

Interviewer: Je hebt het over rollen, wil je dat we mensen kunnen evaluaten als een bepaalde rol? En bijvoorbeeld een role aan iemand assignen en kijken hoe hij binnen die role speelde.

Interviewee: Het handigste is om iedereen te kunnen evalueren als role. Dan kunnen we binnen het team ook wisselen.

Interviewer: Ja, we zaten zelf bijvoorbeeld te denken aan een soort web waarbij je dan voor iedere role ziet hoe je het hebt gedaan.

Interviewee: Dat zou denk ik een hele mooie weergave zijn.

Interviewer: Dan iets technischer, kills en deaths klinken heel erg simpel, maar als je kijkt hoe CS:GO het zelf evaluate als je bijvoorbeeld een teamkill krijgt, haalt hij er een kill vanaf, kijk je in Leetify of CS:GO stats, dan doen ze dat niet. Ik dacht dat er ook zoiets was met deaths van de bom. Hoe willen we dat gaan bepalen?

Interviewee: Het beste is om alle kills te laten staan. Als je een kill krijgt is dat een kill, een teamkill trek je er niets vanaf. Bij een death aan de bom tel je er gewoon een death bij op. Volgens mij wordt het in CS:GO soms wel weergegeven en soms niet, maar het is gewoon een death die de economy verpest. Dat is ook waarom zaken als aim er niet instaan. De grootste reden daarvoor is dat aim niet heel erg relevant is voor een volledige evaluatie. De ene dag is je aim wel goed, de andere niet. Je moet vooral evaluaren of je impact er is. Als je als entry altijd de eerste kill pakt, maar je aimt voor geen meter, heb je nog steeds je impact. Dan maakt het niet uit of je het met een headshot doet. Het kost ook veel moeite om aim te evalueren.

Interviewer: Als we in de toekomst statistics toevoegen, kunnen die niet retroactief in de database worden gezet. Hoe willen we dat doen?

Interviewee: Jullie hebben al gekeken naar de structuur van de demos?

Interviewer: We hebben een library gevonden die het grotendeels doet. Hij zet alles om naar een JSON bestand en importeert dat weer in Python. Daardoor zit een heleboel er al in. We kunnen ook de oude demos saven, maar dat neemt veel ruimte in.

Interviewee: Ik zou dan eerder een demo link opslaan. Voor gedownloade files is dat niet mogelijk, maar een linkje erbij voor een FaceIt game is wel handig.

Interviewer: Hoelangzijn FaceIt demos beschikbaar? Ik dacht dat matchmaking demos 30 dagen waren. Ik weet niet hoelang FaceIt demos beschikbaar zijn.

Interviewee: Even kijken... Oh, volgens mij zijn die ook 30/60 dagen beschikbaar. Daar moet ik even over nadenken. Demo files kunnen namelijk wel 250MB zijn. Dat is wel een zware opslag.

Interviewer: Ja, Mark Kok zou nog even kijken naar of we ze konden compressen, maar dat hielp niet echt. Dan bespaar je ongeveer de helft.

Interviewee: En als we die JSON bestanden opslaan?

Interviewer: Hij gooit een heleboel dingen in de JSON, maar niet alles. Ik weet het niet zeker, maar als je in de toekomst zou willen kijken naar of iemand iemand anders kan zien, dat dat er niet uit te halen is.

Interviewee: Dan zou ik gewoon zeggen dat dat jammer is. De JSON is al een soort van compression. Als je specifieke oude matches wil bekijken, dat dat weinig voorkomt en dat het anders maar gewoon jammer is.

Interviewer: Ja, we hebben hier een match die 350MB was en het demobestand is 2MB, daar gaat nog compression overheen.

Interviewee: Dan kan je die opslaan.

Interviewee: Wat is jullie plan van aanpak voor de database?

Interviewer: Het project gebruikte MongoDB, maar niemand van ons heeft daar eerder meegewerkt. Voor zover we kunnen zien heeft MongoDB een blob storage voor grote bestanden.

Interviewee: Hoe willen jullie het verwerken van de JSON doen? Wil je er losse tabellen aanmaken met de data, of iedere keer het JSON bestand inladen?

Interviewer: Het JSON bestand opslaan, maar de statistics eruit trekken zodat het sneller gaat.

Interviewee: Ja, dus als je het JSON bestand importeert doe je de data extraction al.

Interviewer: Ja, de statistieken komen dan in de database en als er een statistiek nog niet instaat kan die uit de JSON komen die in de database staat.

Interviewee: Dat klinkt prima.

Interviewer: De vorige keer hadden we het over annotations toevoegen aan rounds. Is het belangrijker dat de interface focust op rounds of op matches? Moeten annotations bijvoorbeeld aan matches of aan rounds worden toegevoegd?

Interviewee: De focus ligt op rounds. We willen de rounds terugkijken, daar zal de focus liggen. De matches zijn interessant voor langdurige vergelijkingen. Het gaat vooral om rounds.

Interviewer: Is het dan handig om een pagina te hebben waar je rounds en matches kan vergelijken? En dan bijvoorbeeld rounds van andere matches kan vergelijken?

Interviewee: Ja, de vorige keer hebben we het heel over het vergelijken van bijvoorbeeld pistol rounds gehad. Het mooiste is als je kan zien hoe de round ging, maar dat is belachelijk veel informatie. De structuur om dingen naast elkaar te zetten, zou helpen om in de toekomst meer features toe te voegen, bijvoorbeeld het vergelijken van maps.

Interviewer: We dachten dat comment meer berichten waren, is het ook handig om matches te kunnen taggen? Dan kan je custom tags aanmaken.

Vooral omdat je het voor onderzoek wil gebruiken kan je dan alle rounds met een tag pakken. Bijvoorbeeld om alle eco rounds te pakken.

Interviewee: Precies, dus execute A op Mirage zodat je ze allemaal kan vergelijken. Dat zou top zijn. Ook hierin, stel we gaan door naar het opslaan van demos, of naar een visuele weergave, dan heb je gelijk al je informatie op een rijtje. Dat doet met denken aan het labelen van videos van bijvoorbeeld Amazon.

Interviewer: Dan moeten we het misschien nog even hebben over de tick rates.

Interviewee: Eigenlijk alles wat we gebruiken is 128 ticks. Matchmaking games hebben 64 ticks, maar die matches hebben we nog nooit opgeslagen, dat heb ik even teruggezocht. FaceIt is 128 ticks en onze eigen servers ook.

Interviewer: Oke, dan basseren we het op 128 ticks.

Interviewee: Precies. Er was maar 1 onderzoek waarbij ze willekeurige dingen wilden voor stresstesten, de rest is 128 ticks.

Interviewee: Welk type users zijn er op het moment?

Interviewer: Admin, manager, super admin en player.

Interviewee: Ah ook player, dan is belangrijk.

Interviewee: Wat is jullie plan voor de aankomende weken?

Interviewer: We willen zo snel mogelijk een mockup maken van de frontend en aan jou laten zien. Voor de rest gewoon verder gaan met het parsen van de demo files en kijken hoe we dat goed kunnen integreren in de backend. En vooral de backend ook even restructuren zodat wij ermee kunnen werken.

Interviewee: Want werken jullie ook met sprint?

Interviewer: Ja. We zullen waarschijnlijk elke twee weken laten zien waar we staan.

A.3 First Prototype

This interview was conducted on the 18th of March 2022 **Interviewer**: Op de backend zijn we klaar met het omschrijven naar SQL. Voor de rest zijn de endpoints gemaakt en de analysis werkt.

Interviewer: Op de frontend hebben we eerder een portal zodat het niet meer een geheel is, maar meer portals naar de verschillende games. Momenteel werkt Rocket League nog niet, maar CS:GO wel.

Interviewee: Nice.

Interviewer: Op de CS:GO Dashboard halen we wat statistieken uit de database. Dit is het enige op het moment waar al een verbinding achter zit. Normaal laat hij hier alleen de ETT teams zien, maar voor demo purposes zie je ook andere teams.

Interviewee: Dus jullie hebben een demo geüpload en daar haalt hij de teams uit?

Interviewer: Exact. Hij pakt alle data er al uit, maar de frontend laat het nog niet zien.

Interviewer: Hier hebben we een search pagina waar je rondes op labels, map en win/loss kan filteren.

Interviewer: Voor de matches zie je hier een aantal statistieken. We weten nog niet of een match een naam moet hebben of niet.

Interviewee: Ja, met een naam kan je hem zo noemen als dat je wil, maar je kan ook gewoon een custom label aanmaken.

Interviewer: Dan hebben we hier een pagina voor de statistieken van een ronde. De labels laten we daar ook zien. We willen label categoriën colour coden. En dan hebben we hier ook tabs met informatie. Het is nu vooral zaak om de data daadwerkelijk uit de backend te halen.

Interviewee: Ja, het ziet eruit alsof je de juiste paginas hebben. De knop naar de match vanaf de round werkt ook?

Interviewer: Ja, maar het is nog wel een prototype, dus het is gehard-code, maar het is wel hoe het hoort te werken.

Interviewee: De knoppen zijn er in ieder geval al. Ik vind het heel nice dat je vanaf een round naar een match kan en dat je op rounds kan zoeken. Hoe kom je bij en volledige match uit als ik die upload? Moet je dan altijd via een round klikken, of is er ook een zoekscherm voor volledige matches?

Interviewer: Daar zitten we nog over na te denken, maar op het dash-boardscherm zie je in ieder geval de laatste paar matches.

Interviewee: Prima, ik vind vooral het zoekscherm erg nice, je hebt veel filters. Je kan dan gewoon aanklikken wat je wil. Het is bij filters misschien ook handig om op title te zoeken. Dan kan je een team waar je vaak tegen speelt gelijk vinden zonder labels aan te maken. Datum kan ook wel, maar het liefste zoek je op een match omdat daar instaat tegen wie je hebt gespeeld.

Interviewer: Je kan al een team aangeven.

Interviewee: Ik denk dat dit een mooie flow is om te werken.

Interviewer: We vroegen ons nog af hoe belangrijk mobile friendliness is.

Interviewee: Niet heel erg, onze meeting zijn eigenlijk altijd of online of bij een PC. Er zullen weinig mensen zijn die uitgebreide analyses gaan doen via hun telefoon, maar het moet wel relatief oke scalen. Als je het op een tweede scherm zet, zoals mijn verticale scherm, dan zou het wel fijn zijn als het niet helemaal uit elkaar valt.

Interviewee: Ik vind het er wel al heel erg nice uitzien, het is precies het soort navigatie dat handig is. Als de stats erin komen is het heel mooi.

Interviewer: Ja, dat is het doel voor de volgende sprint. We willen de frontend en de backend iets meer gaan connecten en meer statistieken extracten.

Interviewee: Ik had ook nog een vraag over deployment. Het past ook binnen de Docker deployment structuur?

Interviewer: Ja, de docker-compose werkt ook gewoon, maar die is voornamelijk gericht op development. Waarschijnlijk wil je zelf een PostgresDB opstarten. We gebruiken het zelf voor onze development.

Interviewee: Prima, we moeten er alleen even een mooie structuur voor hebben.

Interviewer: De migrations worden ook automatisch gerunt, dus daar hoe je je geen zorgen om te maken.

Interviewee: Nice, dat is altijd mooi werk.

Interviewee: Het ziet er in ieder geval mooi uit, vooral die custom labels vind ik echt top, ik zou alleen willen toevoegen dat je inderdaad op matches kan zoeken. Voor de statistieken zelf, waar zijn we op uit gekomen? We hebben een heleboel neergezet, maar wouden we uiteindelijk per role statistieken, of zijn we meer op het taggen van rounds gegaan?

Interviewer: De statistics gebasseerd op specifieke roles staan onder de should, dus daar komen we waarschijnlijk wel ja.

Interviewee: Prima, de role statistics hadden inderdaad meer prioriteit. Het is voor de komende weken handig om er een heleboel statistieken in te pompen. Het is handig om daar los nog even een meeting voor te hebben. Hoelang denken jullie dat bepaalde zaken gaan duren?

Interviewer: Maandag gaan we beginnen aan de statistieken, ik denk dat we binnen een week wel basis statistieken hebben, de vraag is alleen of ze dan al op de frontend staan. Binnen een week kunnen we wel meer richting role specifieke statistieken kunnen gaan.

Interviewee: Dus als we over anderhalve week een meeting hebben voor die statistieken, zou dat goedkomen?

Interviewer: Prima, dan zorgen we dat we dan een aantal statistieken hebben.

Interviewee: Ja, de scope is in principe dat de basis statistieken goed werken. Vooral omdat jullie op de backend veel aanpassingen hebben gedaan, dat zijn noodzakelijke dingen die gewoon tijd kosten. Ik zorg er in ieder geval voor dat er wat mensen van het CS:GO team zijn bij de volgende meeting.

Interviewer: Wat voor een dingen wil je op het admin panel?

Interviewee: Elke speler moet matches kunnen zien. We zijn een kleiner team, dus dingen als write en delete permissions zijn niet echt nodig. De coach en de analyst gaan er waarschijnlijk het meeste gebruik van maken en de players zelf kunnen de matches upload. Je hoeft hun toegang niet te blocken omdat ze zelf ook dingen kunnen doen, daarvoor hoeven ze niet te wachten op een coach. Misschien dat je alleen matches die je zelf hebt geüpload kan editten. Ze moeten het sowieso kunnen zien.

Interviewer: Dus de coach moet overal bij, maar een uploader kan alleen de match editten?

Interviewee: Ja. We zijn niet onder gevaar van datalekken, we zijn een onderzoeksorganisatie, dus alles wat wij te weten komen vrij open is. Tenzij het heel specifiek is voor een toernooi. Onze data is beschikbaar voor iedereen binnen ETT die daar mee wil werken.

Interviewer: Op het moment worden teams gemaakt op basis van steam IDs, dus users moeten gelinkt worden aan steam IDs. Dat moet de user zelf doen, of een admin.

Interviewee: Dat is prima. Er zijn altijd wel mensen die dat kunnen doen.

A.4 First Player Meeting

This interview was conducted on the 25th of March 2022 **Interviewee**: How have the last few weeks been? Is everything on schedule?

Interviewer: Yes, we have mainly completed our highest priority requirements and are almost done with the shoulds in the Must Should Could Won't prioritization. The main problem in the start has been that we had to redo the backend and part of the frontend.

Interviewer: For today we would like to clear up some definitions of the statistics such that we are on the same page. We will start with the trade attempts. It is quite difficult to define what counts as a trade attempt. What we are currently thinking of is that we have a specific time in which you can damage the opponent that killed your teammate and if that person gets killed within that timeframe, it's a trade kill, but it is going to be a trade attempt if they hit them only once.

Interviewee: So both sides need to have received damage?

Interviewer: Yes, but it is quite difficult since you could also argue that if you are swinging but missing your shots, it is also a trade attempt, but that is way harder to measure. And also, when should it count as a failed trade attempt?

Interviewee: That is difficult, because a trade attempt can be quite long. Even if you are missing a lot of shots, it should still count as a trade attempt, since you are still trading information. What you could do, is that after a certain point of not shooting, the trade attempt ends. I think 5 seconds is fine. If you don't shoot for 5 seconds you are probably repositioning and starting a new trade attempt. If you are concerned about whether 5 seconds is the right time, you can test it out and see if the players can get information out of that. If they feel like it is a bit unnecessary, you can leave it out, if they want it a little different interpretation, you can change it.

Interviewer: Would you think you prefer 5 seconds of no shooting at all, or 5 seconds of no damage? Because it might be that someone is spamming a location where their opponent previously was.

Interviewee: I would say shooting, because you can miss, because you are always trading information. You don't need to do damage to get an actual trade. If you don't have shooting for a certain period of time, it can stop. Within CS:GO 5 seconds is a lot more important than for example in League of Legends. Do it completely based on shooting. If they start shooting again after 5 seconds, it is a new trade.

Interviewer: I think we are not really on the same page on what a trade is. A trade usually means within CS:GO that it is a trade kill right?

Interviewee: I guess, I'm not that much into CS:GO as Teun is. Forgive me if I am wrong.

Interviewer: No problem, but that does make it harder to ask specific questions.

Interviewee: I guess, so with trades, do you want to talk about trade kills, because I was talking more about information. If you are really focussed on trading kills, than you just need to count the kills that are happening right?

Interviewer: The trade kills are relatively easy, but it is mainly about the trade attempts. I think what you said about taking the 5 second margin after shooting last is a decent way of doing it.

Interviewee: Honestly, I have no answer to it.

Interviewer: We will discuss it with Teun then.

Interviewer: We have the definition of flash assists as well, but if you are not really into CS:GO, that might be difficult.

Interviewee: Flashes get registered nowadays right? So you can just take that.

Interviewer: Yes, but that does not work too well.

Interviewee: How can you then fix that?

Interviewer: We are getting how long someone has been flashed for, so if we for example see that someone has been flashed for longer than half a second when they get flashed.

Interviewee: That would make sense. Even if they are flashed for a short period of time, their sight has been distorted. It's just a case of testing it out with players.

Interviewer: We have some additional statistics like flash assists, opening duels and clutches for now. There is also a request for a potential heat map for smokes. It does seem a bit harder to implement for now, would it be fine if we do not try to add this?

Interviewee: Yes, the things that seem more possible to realize should be implemented first. The ones that you feel like you are not able to do you can keep them in mind, but do not focus on them.

Interviewer: Alright, then we need to plan a next player meeting.

A.5 Second Player Meeting / Second Prototype

This interview was conducted on the 31th of March 2022 **Interviewer**: We gaan eerst even laten zien wat we hebben. Dit is de dashboard. Dit is de belangrijkste pagina. Hier staan alle rounds die ooit zijn gespeeld. Je kan filteren op labels en dan de rondes krijgen. Als je op de ronde klikt ga je naar de statistics en via daar kan je ook naar de game.

Interviewee: Ik zou dat juist andersom doen. 99% van de CS:GO en Valorant spelers zijn gewend om op match te zoeken. Als je dan het overzicht hebt van alle matches en dan dit overzicht hebt met de rounds.

Interviewer: Je kan ook op matches zoeken.

Interviewee: Het is dan alsnog wel handig denk ik om dan via de match naar zo'n overzicht als dit te gaan, maar ga verder.

Interviewer: Je kan van de rondes die je geselecteerd hebt de statistieken bekijken. Op die manier kan je data krijgen van hele specifieke type rounds. Dit zijn alle statistieken die we nu hebben.

Interviewee: Nice.

Interviewer: Dan hebben we nog wat vragen over de implementatie van flash assists en trades. Flash assists zitten wel in CS:GO, maar we vinden dat zelf soms nogal matig.

Interviewee: Ja, soms kijken mensen weg van een flash, maar dan worden ze alsnog een klein beetje geblind, dan ben je voor een frame of 2 wit, maar als je dan de kill pakt zegt hij nog steeds dat hij blind was.

Interviewer: En soms heb je het gevoel dat iemand gigantisch blind was en dan telt het weer niet.

Interviewee: Ja, dan heb je weer damage assists die het weer overwriten, de Valve API is niet altijd even consistent.

Interviewer: Daarom zou het misschien handig zijn om de kijken of iemand nog een halve seconde blind is als hij gekilld wordt.

Interviewee: Ik weet niet precies de details, ik denk dat dat misschien wel handig is. Om eerlijk te zijn denk ik ook dat als iemand wegkijkt, maar nog een beetje geblind wordt, dat eigenlijk ook een flash assist is omdat hij moest wegkijken. Dan is het misschien geen assist omdat hij blind was, maar omdat hij niet blind wilde zijn. Zo kan je een flash assist ook zien. Het makkelijkste is om de Valve API gewoon te pakken als minimal product.

Interviewer: Prima, dan laten we die zo. Voor de trade kills was het lastig om te bepalen wat er telt. We kunnen bijvoorbeeld 5 seconden nemen nadat de eerste kill is gepakt en dat hij binnen die tijd getrade moet worden, of 5 seconden sinds de laatste damage die de twee hebben uitgewisseld.

Interviewee: Nee, een trade kill is gewoon echt A killt B, C killt A en dat binnen een bepaalde tijd. Ik zou eerder 3 seconden nemen. CS:GO is heel langzaam, maar als er actie is, is het snel. Een hoofdje wordt geclickt en dan is het klaar. Dan wil je binnen 3 seconden de trade omdat dat genoeg tijd is voor een reposition of het helpen met een flash.

Interviewee: Ik weet niet of jullie daar al van gehoord hadden, maar misschien kunnen jullie wat dingen van Leetify afkijken. Ik denk dat een goede implementatie ook bijvoorbeeld de voice analysis zou kunnen supporten. Sommige dingen kan je namelijk op 50 verschillende plekken vinden, dus dat is misschien wel interessanter.

Interviewer: Het probleem is een beetje dat we meer wilden doen, maar toen bleek de backend niet echt geschikt dus die hebben we toen lopen rewriten. Dat kostte veel tijd.

Interviewee: Dat snap ik, ik was er vorig jaar bij toen ze het schreven en het zag er inderdaad niet heel erg geweldig uit. Misschien is het nu inderdaad het beste om dit werkend te krijgen en dan volgend jaar naar de andere features te kijken zoals data science en opmerkingen van de psychiatrist. Het minimale voor nu zou bijvoorbeeld iets als notities zijn. Dan heb je een simpel tekstblok en die label je aan dezelfde match. Dan kan de psychiatrist bijvoorbeeld zeggen dat iemand veelte veel praat.

Interviewer: Dat is inderdaad waarom we de labels erin hebben en de comments stonden ook op het lijstje.

Interviewee: Die labels zijn inderdaad ook goed. Een aantal goede labels waar je aan zou kunnen denken zijn: rounds survived, de opening als label per speler. Ik kan dan bijvoorbeeld kijken wat er gebeurd als mijn teammate of ik op de bombsite een kill pakken of als hij als eerste gekillt wordt en hoe dat mijn performance beïnvloed. Een goed label is denk ik ook alle multi-kills. De rondes waarin je een trade kill pakt of een clutch en ook een ronde zonder kills, zodat je ook kan zien of je impact hebt, ookal heb je geen kills. Ik ben een support player, waardoor ik op een hoekje op de map zit te viben, maar misschien heb ik dan wel 3 flash assists. Misschien ook een goed label is de map, CT side en T side. Ik weet bijvoorbeeld dat ik op CT side meer impact heb dan op de T side omdat ik op de T side echt meer support ben. Dan heb ik een stukje op de map en daar vibe ik. Op de CT side ben ik een anchor en dan heb je of heel veel of heel weinig impact. Ik weet nog wel een Mirage game waarin ik voor mijn gevoel elke ronde moest rotaten en moest clutchen op de andere bombsite. Toen ging ik in de statistieken kijken op Leetify en toen zag ik dat mijn presence op de B site heel hoog was, maar ik daar heel weinig kills had, maar op de andere kant van de map had ik weinig presence, maar wel veel kills omdat ik steeds moest clutchen. Misschien ook een goede om bij te houden is de economy, bijvoorbeeld hoeveel geld je hebt. Soms heb ik een game waarin ik wel heel lekker ga, maar wel steeds drops moet vragen, of juist een game waarin ik altijd moet droppen omdat ik niet dood ga. Het is misschien moeilijk om dat helemaal te gaan berekenen, maar misschien de starting equipment en de money aan het begin van de ronde, die kan je wel optellen. Dan kan je ook kijken wat je equipment is en of je vaak dropt of dat je vaak drops ontvangt. Soms heb ik bijvoorbeeld het gevoel dat ik alleen maar de AWP zit te droppen, of dat ik alleen maar zit te clutchen.

Interviewer: Hoe zou je dan denk je een drop definiëren?

Interviewee: Valve heeft volgens mij wel iets om dat te checken, want soms zie je aan het einde van de ronde hoeveel drops je hebt gedaan, maar volgens mij telt de bom daarin mee. Ik denk dat een gun droppen, maar niet zelf in je handen hebt, dat dat een drop is.

Interviewer: Maar stel je zit later in de ronde, je teammate heeft ineens nog maar 10 HP, dus je geeft hem de AWP omdat hij dan nog iets kan. Ga je dat dan niet ook per ongeluk meetellen als drop?

Interviewee: Dat is inderdaad wat ik denk dat de Valve API meetelt. Ik zou alleen de drops in de freeze time meenemen.

Interviewer: Dus als iemand laat uit spawn is, heeft dat maar gewoon even pech?

Interviewee: Soms worden er ook drops gedaan zonder dat ze in de ronde gekocht worden, bijvoorbeeld als je de ronde overleeft. Dat is eigenlijk ook gewoon een drop. Het is misschien wel funky om het te analyzen, maar ik zou het gewoon in de freeze time pakken.

Interviewer: En het dan maar te accepteren dan als mensen guns in de lucht gooien tijdens een timeout als ze zich vervelen?

Interviewee: Ja, maar volgens mij kijkt de Valve API daar wel naar, hij moet wel naar een andere speler gaat. Ik weet niet zeker hoe de API dat trackt. Het belangrijkste is dat als ik een drop doe en iemand anders hem oppakt, dan is dat een drop.

Interviewee: Misschien is het nog een leuke stat om te weten hoeveel util je gooit. Ik weet dat iemand van ons bijvoorbeeld 1 Molotov had per seizoen. Aan de andere kant van het spectrum had je iemand met een gemiddelde van 1.3 flashes per ronde. Die gooide bijna elke ronde een flash en dan elke andere ronde nog een. Dan heb je wel een mooi spectrum van hoeveel nades iemand gooit en ook welke nades. Ik zit nog even door Leetify heen te scrollen, misschien is win percentage nog wel een goede. Misschien is het ook wel leuk om te zien met welke guns je kills krijgt. Als een AWPer dan misschien ziet dat hij weinig kills heeft met de AWP kan hij gaan nadenken waarom dat is. Misschien ging de AWP niet zo goed, maar de AK wel, dus heeft hij maar even een AK gepakt. Het hangt ook een beetje van de map af. Op Inferno T side heb je niet altijd wat aan een AWP. Dan pakt onze AWPer een AK en rent hij er wel in omdat iemand iets van iniatief moet tonen. Soms pak ik even de AWP op omdat ik wel prima een minuut naar een angle kan staren. Map zones zijn misschien moeilijker te checken.

Interviewer: Heatmaps stonden nog wel op een wensenlijstje, maar daar hebben we echt geen tijd meer voor.

Interviewee: Snap ik wel, focus je vooral op de makkelijkere dingen en zorg dat die echt goed werken. Misschien is er ook nog wel de mogelijkheid om zelf labels toe te voegen aan labels.

Interviewer: Die is er. Je kan categorieën en labels aanmaken en toevoegen aan rondes.

Interviewee: Ja precies, want soms is het ook belangrijk om te zien of iets een eco ronde is en of ik veel impact heb op een eco. Dan kan je dus zeggen of iets een eco, full buy of force buy is.

Interviewer: Dat doen we automatisch.

Interviewee: Precies, maar dat je dat nog wel aan kan passen en dat het niet alleen automatic is. Soms klopt een label niet als je eigenlijk denkt dat iets een low buy is, maar het eigenlijk een force buy is. Dan wil je het graag aan kunnen passen. Misschien is het ook wel handig om te kijken hoe je presteert als je de eerste kill pakt. Sommige teams zijn bijvoorbeeld erg goed in een 5v4, maar zijn waardeloos in een 4v5, maar je hebt ook teams die amper last hebben van een 4v5.

A.6 Third Prototype

This interview was conducted on the 8th of April 2022 **Interviewer**: Dit is dus de algemene homepagina. Dan kunnen we naar de CS:GO Dashboard. Daar hebben we een overview van de ETT teams, de recent matches van jouw team en een overzicht van alle geüploadde matches. Als we op de match klikken krijgen we informatie over de match, basis statistieken en je kan zien wie wel en geen ETT teams zijn. Hier staan ook alle rondes van de match. Er is ook de mogelijkheid om comments toe te voegen.

Interviewee: Ah, dan kan je notities maken.

Interviewer: Dan de round overview, daar krijg je de statistieken van een round. Ook de mogelijkheid om labels toe te voegen.

Interviewee: Kan je daar ook de kleuren van aanpassen? Of is dat gewoon blauw?

Interviewer: Nee, dat kan je aanpassen. Dat is de label manager hier. Hier kan je de naam van de categorie en de labels updaten en ook de kleur.

Interviewee: Nice.

Interviewer: Dan hebben we de round search pagina. Je kan erop filter, filteren op labels, datum of specifieke match. We hebben nog plannen voor meer filters. We kunnen op dezelfde pagina ook naar Combine Rounds, dan worden alle statistieken bij elkaar gegooid. We hebben hier de statistieken gesorteerd per team die in die rounds zitten. Hier heb je dan het overzicht van de statistieken. Je kan hier ook gewoon filteren op labels, dus dan pakt hij de statistieken van de matches met die labels.

Interviewee: Dus als je labels selecteert, pakt hij dan een gemiddelde over de geselecteerde rounds? Of wat doet hij dan?

Interviewer: Hij pakt alle rounds die in het round overview komen te staan en berekent daar het gemiddelde van.

Interviewee: Dus nu zien we eigenlijk van de match die we net hebben geüpload alle stats?

Interviewer: Ja, maar als je dus een bepaald label hebt kan je het gemiddelde zien van alle rounds met dat label.

Interviewee: Dus dat kan je makkelijk aanklikken, oke.

Interviewer: Momenteel is het gesorteerd per team, als er veel rounds worden geüpload zijn er meerdere teams, maar we weten niet zo goed of we het moeten sorteren op team, of een grote table met alleen maar players. Ik weet niet wat jouw voorkeur is?

Interviewee: Per player is wel heel nice, want als je het met players doet ga je een nog grotere lijst krijgen. Het enige probleem wat ik zie, is dat je misschien geen teamnamen kan aanpassen. Is dat mogelijk? Want vooral als je FaceIt gebruikt en je bent met zijn vijven, met een random team om dingen te testen, dan wordt de teamnaam bijvoorbeeld Team JaCkz. Dan heb je voor iedere naam een nieuw team en dat kan chaotisch worden.

Interviewer: Daar hebben we rekening mee gehouden. Je kan in het adminpaneel de teamnamen wijzigen.

Interviewee: Oh cool, je voegt dus eigenlijk de games toe aan een team.

Interviewer: Ja en nee, op de uploadpagina zijn een paar opties. Als je wil kan je het winning of losing team selecteren. Als je dat niet wil kan je ook op 'Generate new teams' klikken en dan genereert hij nieuwe teams voor die match. **Interviewee**: Ah, dan pakt hij gewoon de titel van het team. Dat is echt nice, dat is top.

Interviewer: Als je geen van de teams aangeeft pakt hij een null team en linkt hij hem niet aan een team.

Interviewee: Dat is echt top, dat is precies wat ik bedoelde. Per team is denk ik ook wel gewoon goed. Ik kan me ook voorstellen dat je situaties hebt waarin je per speler wil kijken, maar gaan we dat echt gebruiken is de vraag? Voor ons is het het belangrijkste om in de context van een team te kijken. Als ik denk aan de performance van een enkele speler, dan gaan we waarschijnlijk een bestaande tool gebruiken. Leetify werkt bijvoorbeeld heel goed voor enkele spelers, maar niet voor teams. Ik denk dus dat het goed is om dit per team te houden, dat is echt nice.

Interviewer: En als een team meerdere matches heeft met meerdere spelers in die matches, dan hebben we andere lineups, moeten die dan ook apart staan? Of in dezelfde table?

Interviewee: Ik zou het apart zetten. Soms speel je met een substitute, dan is het wel fijn om te zien dat dat een ander team is dan het main team. Waarschijnlijk zal het ook niet belachelijk veel voor komen, maar voor de tegenstanders wel. Wij hebben twee subs, dus dan heb je een paar variaties van teams, maar je kijkt een stuk minder naar die sub games. Je gebruikt dat vaak niet voor de strats maken, dus het losbreken is prima.

Interviewer: Dan breken we ze op lineup, dat is ook wat we nu doen.

Interviewee: En zoals ik zei denk ik ook niet dat dat heel erg vaak gaat gebeuren, als we preparen voor een ander team, dan uploaden we eigenlijk

alleen maar demos met de standaard lineup, want je neemt niet aan dat ze met een sub gaan spelen. Hetzelfd geldt voor ons team.

Interviewer: Er zijn nog een paar dingen waar we aan willen werken. De adminpagina heeft nog wat meer functionaliteit nodig zoals het iets meer managen van de teams. Ook de mogelijkheid om user accounts te linken aan players.

Interviewee: Zodat je echt de lijst krijgt van matches waar jij in speelt?

Interviewer: Ja, zodat je inderdaad jouw matches kan zien. Voor de rest is het vooral een beetje schoonpoetsen en wat statistics toevoegen en aan de round pagina. Alles wat in de combined rounds pagina stond moet hier ook. Wat vind je ervan?

Interviewee: Ik vind het er erg nice uitzien, het werkt echt top. Wat ik echt heel mooi vind is dat we eigenlijk alle statistieken hebben die we willen zien in een ronde. Dan weten we of een ronde nuttig is om te bekijken en dan kunnen we hem makkelijk markeren. Het is heel mooi om matches te bewaren en zoekbaar te maken. Stel we hebben over een halfjaar 40 matches hierin staan kunnen we heel makkelijk kijken hoe een match ging en als we een strategie hebben kunnen we de informatie daarvan snel terugvinden. Dus het is echt top. De feature waar jullie het over hadden zodat je een account kan linken aan een player is heel nice. Dat is helemaal fantastisch. Ik ben heel erg tevreden.

Interviewee: Over anderhalve week spelen we een showmatch tegen het CS:GO team van defensie. Ik vind het wel even leuk om te kijken of ze set strats hebben. Daar heb ik een demo voor gedownload en ik was wel benieuwd of ik die kan uploaden. Gewoon even als test, dan kan ik ook zien hoe het uploaden enzo werkt.

Interviewer: Hij zou moeten redirecten, hij staat ook niet in de database, dat is raar.

Interviewee: Je gaat dus naar upload en daar kan je dus de datum selecteren en zelfs een VOD link, dat is echt top.

Interviewer: Als je niets invult bij de datum pakt hij de huidige datum.

Interviewer: Het werkt in ieder geval niet bij deze demo, we gaan hier even naar kijken. Het heeft iets te maken met de teams. We gaan even een demo uploaden waarvan we weten dat het werkt.

Interviewee: Ah, dat werkt en het heeft ook een prachtige redirect, dus dat zit er gewoon in.

Interviewer: Het is een mooie edge case, daar gaan we even naar kijken.

Interviewee: Het kan best wel zijn dat het vanwege de teamnamen in FaceIt even raar doet. We kijken wel, in ieder geval een mooie edge case. Ook het toevoegen van de comments vind ik nice. Dan kan je ook zeggen dat iets bijvoorbeeld niet zo interessant is. Ik zou voor nu vooral zeggen: focus je op het gladtrekken van alles wat er nu is. Klik er goed doorheen en probeer wat bugs te vinden. Zet vooral de puntjes op de i. Als er iets nog

makkelijk toe te voegen mag dat ook, daar heb je de vrijheid in, maar van mijn kant zit eigenlijk alles erin wat ik erin wilde zien.

Interviewer: Er is in ieder geval genoeg te doen voor een volgend team.

Interviewee: Sowieso. Als we verder gaan zal het inderdaad ook wat meer statistiekanalysise worden, maar daar hebben we nu een mooie basis voor, dat is top. Qua documentatie, hoe zit dat?

Interviewer: Het plan is om de documentatie oveel mogelijk binnen de repo te houden. We willen wel een architecture overview schrijven en in de repo doen. De meeste methodes zijn gedocumenteerd. Als het goed is is dat prima te doen. Op de frontend is het ook het plan om een uitgebreide README te schrijven. Er valt daar ook niet bijzonder veel te documenteren.

Interviewee: Dat is prima, als een volgende groep maar goed verder kan.

Interviewer: En in het Design Report komt nog wel wat over de architectuur, maar ik denk niet dat dat heel relevant zal zijn voor een volgend team.

Interviewee: Ja, je kan natuurlijk al heel veel zien als je naar het project zelf kijkt. Dat lijkt me goed. Heel erg bedankt voor de demo, ik ben heel erg tevreden. Ik ben benieuwd naar de volgende meeting!

Appendix B

Progress By Day

B.1 Sprint 0

7-2-2022

General

- Plan a meeting with the client
- Search for a supervisor
- Research current solutions

Meeting

- Daily Standup

8-2-2022

General

- Research current solutions

Meeting

- Daily Standup
- Visit the client
- Meet with other design project groups

9-2-2022

General

- Research current solutions

Meeting

- Daily Standup

10-2-2022

General

- Research current solutions

Meeting

- Visit the client
- Daily Standup

11-2-2022

Backend

- Enable a local MongoDB instance in the docker-compose

Meeting

- Daily Standup

11-2-2022

Backend

- Enable a local MongoDB instance in the docker-compose

Meeting

- Daily Standup

14-2-2022

Backend

- Clean up unimportant git files

Meeting

- Daily Standup

15-2-2022

Backend

- Update to a new dependency injection system
- Make the database port configurable
- Return a 404 when a user is not found

Meeting

- Daily Standup

16-2-2022

Backend

- Move transactions to a repository

Meeting

- Meeting with the client
- Daily Standup

17-2-2022

Backend

- Add documentation to the tests
- Run the tests with an in-memory database
- Remove redundant async calls

Meeting

- Daily Standup

18-2-2022

Backend

- Add CI
- Refactor tests
- Reformat transactions
- Reformat repositories

Meeting

- Daily Standup

25-2-2022

Backend

- Move to a pipeline structure

26-2-2022

Backend

- Fix the database fill endpoint
- Fix match uploading
- Allow files of up to one gigabyte to be updated

27-2-2022

Backend

- Restructure the pipeline
- Extract basic statistics from the match

B.2 Sprint 1

28-2-2022

Meeting

- Daily Standup

Front-end

- Remove template data
- Add rescaled logos

1-3-2022

Backend

- Add SqlAlchemy
- Add Alembic

Report

- Write the Project Proposal

Meeting

- Daily Standup
- Make the presentation
- Attend the peer-review

2-3-2022

Front-end

- Rename the ApiService to CookieService
- Add a common way to make a get request

Backend

- Add ORM models

Meeting

- Daily Standup

3-3-2022

Front-end

- Downscale images and convert to WebP where beneficial
- Add a wrapper around useSWR
- Use ApiService fetchers

Backend

- Disable the database fill endpoint
- Require explicit commits
- Clean up the repository system

Meeting

- Daily Standup
- Meeting with the client

4-3-2022

Meeting

- Daily Standup

6-3-2022

Frontend

- Add an empty CS:GO dashboard

7-3-2022

Frontend

- Add .env file
- Improve the logout user experience
- Remove the upload from the sidebar
- Remove the base route
- Setup Github Actions
- Update the npm run
- Update the linter

Backend

- Add automatic migration
- Move games to the DTO system
- Move users to the DTO system
- Move authentication to the DTO system
- Move vods to the DTO system
- Move teams to the DTO system

Meeting

- Daily Standup

8-3-2022

Frontend

- Use renamed services
- Add a basic label panel

Backend

- Make sure absolute imports work as well
- Add example unit test
- Run migrations on startup
- Remove the old database fill
- Cleanup Docker Compose
- Move matches to the DTO system
- Move labels to the DTO system
- Move categories to the DTO system
- Migrate the integration tests to the DTO system
- Add parallel testing

9-3-2022

Frontend

- Add a round list
- Make select buttons functional
- Extract components
- Add layout to the dashboard
- Add a custom scroll

Backend

- Remove the last remainders of MongoDB
- Add scripts for (auto)generating migrations
- Restructure to the new feature-based system

Meeting

- Daily Standup
- Meeting with the supervisor

10-3-2022

Frontend

- Update MUI to v5
- Add filters to rounds
- Add placeholders to the dashboard

Backend

- Add the new SQL structure to the pipeline
- Change the names of paginated lists
- Update the query filters to the new SQL system

Meeting

- Daily Standup

11-3-2022

Backend

- Simplify replay upload
- Use the new repository structure in the pipeline
- Optimize round loading
- Deduplicate teams based on their players

Meeting

- Daily Standup

13-3-2022

Frontend

- Move the top bar to a separate file
- Refactor the sidebar

B.3 Sprint 2

14-3-2022

Frontend

- Change map select

Backend

- Add filters to the rounds
- Add more information to teams
- Add more information to maps
- Prevent automatic load of VODs

Meeting

- Daily Standup

15-3-2022

Frontend

- Add a config
- Fix linter
- Refactor label route
- Refactor search route
- Configure scripts
- Update readme
- Update models
- Add profile button to the top bar
- Add cards

Backend

- Add label endpoints
- Add a way to create matches
- Add a way to update teams

Report

- Write the Test Plan
- Update the Requirements

Meeting

- Daily Standup
- Make the presentation
- Attend the peer-review

16-3-2022

Frontend

- Add round page
- Create match layout
- Add round label example

Backend

- Add documentation to the code
- Add endpoints to link users to players
- Add team endpoints
- Add match endpoints

Meeting

- Daily Standup

17-3-2022

Frontend

- Update API models
- Update API calls
- Fix drawer issues
- Fix admin page
- Remove outdated widgets
- Add CS:GO team overview widget

Backend

- Add Git LFS
- Add documentation to the code
- Add integration tests for the teams
- Add integration tests for the rounds

Meeting

- Daily Standup

18-3-2022

Frontend

- Remove user menu from sidebar
- Add ETT icon link to the homepage

Backend

- Rewrite the database to a more general CRUD system
- Remove Git LFS
- Move category to its own feature
- Add endpoints for the categories
- Add endpoints for the labels

Meeting

- Meeting with the client

19-3-2022

Frontend

- Update OpenAPI files

20-3-2022

Frontend

- Select rounds based on labels

21-3-2022

Frontend

- Update the labels
- Remove unused code
- Update upload page
- Add colours to categories in the sidebar
- Add colours to label
- Make rounds clickable
- Add pagination to the rounds

Backend

- Add trade statistics
- Add damage statistics
- Add aggregate statistics
- Move to a lineup structure

Meeting

- Daily Standup

22-3-2022

Frontend

- Update the OpenAPI pages
- Remove unnecessary logs
- Add time to the upload page
- Add updates for categories
- Add updates for labels
- Add OpenAPI generate script

Backend

- Add aggregate statistics for opening duels
- Add aggregate statistics for flashes
- Get a working Minimal Viable Product
- Add a date to the rounds
- Add scores and indexes to the rounds

Meeting

- Daily Standup

23-3-2022

Frontend

- Update the user model
- Add a game-specific admin panel
- Change the base URL
- Update the label model
- Add details to the match page

Backend

- Completed the addition of lineups underneath teams
- Fix general sorting
- Allow the specifying of winner/loser team id when uploading a match
- Add opening duel and initial role specific statistics
- Add match deletion
- Fix category deletion not working

Frontend

- Admin panel split up into general and CS specific
- Work on statistics frontend

Meeting

- Daily Standup
- Meeting with the supervisor

24-3-2022

Frontend

- Update the OpenAPI models
- Update the label model
- Add a basic table
- Add a match delete button
- Add an 'All Team' option

Backend

- Move the statistics to the match instead of a separate endpoint
- Group match statistics in teams
- Fix the sorting of teams
- Add a user to the player DTO
- Add types to the category

Meeting

- Daily Standup
- Meeting with the client

25-3-2022

Frontend

- Add damage dealt statistic

Backend

- Move the statistics to the match instead of a separate endpoint
- Group match statistics in teams
- Fix the sorting of teams
- Add a user to the player DTO
- Add types to the category

Meeting

- Daily Standup
- Players meeting with ETT

B.4 Sprint 3

28-3-2022

Frontend

- Change OpenAPI output directory
- Refactor the frontend
- Add labels to rounds
- Update the category model

Backend

- Add names to lineups
- Fix label creation
- Add names to players

Report

- Move to a one column structure
- Add more references
- Further elaborate the requirements
- Write the test description

Meeting

- Daily Standup

29-3-2022

Frontend

- Add login tests
- Add the url to the env

Backend

- Add names to lineups
- Fix label creation
- Add names to players

Report

- Update the front page
- Write the architecture
- Start the Progress Day by Day

Meeting

- Daily Standup
- Make the presentation
- Attend the peer-review

30-3-2022

Frontend

- Update the match layout
- Update the round view layout
- Update table colours
- Add icons
- Add an ETT lineup indicator
- Fix role tests
- Fix links on the search page

Backend

- Add the match id to rounds
- Add player filtering to matches
- Group statistics by lineups
- Add detailed lineups to the team endpoints
- Add a link between a lineup and teams

Meeting

- Daily Standup

31-3-2022

Frontend

- Implement table for all statistics
- Update the OpenAPI specification
- Fix the basic DataGrid

Backend

- Add lineups to matches
- Add score to matches

Meeting

- Daily Standup

1-4-2022

Backend

- Improve the load time of statistics

Meeting

- Daily Standup

2-4-2022

Frontend

- Add map icons
- Remove the comparison of rounds

4-4-2022

Frontend

- Add date filters
- Add a map filter
- Add the recent matches to the dashboard

Backend

- Add a match filter
- Add a date filter
- Automatically create teams on a replay upload

Meeting

- Daily Standup

5-4-2022

Frontend

- Modify the round card
- Add match filters to the combined rounds
- Add lineup filters to the combined rounds
- Make sure non-hover tables don't change colour on mouse hover
- Add labels to rounds

Backend

- Group the aggregate statistics by lineup
- Add endpoints to retrieve players linked to users
- Add various tests
- Add documentation to various parts of the code
- Add comments

Meeting

- Daily Standup

6-4-2022

Frontend

- Reload the round when it has a new label
- Update endpoint call

Backend

- Add various tests
- Add documentation to various parts of the code
- Fix a flacky test

Meeting

- Daily Standup
- Meeting with the supervisor

7-4-2022

Frontend

- Add user team data to the dashboard
- Add comments
- Add teams to the upload

Backend

- Add users to comments
- Restructure comment DTOs

Meeting

- Daily Standup

8-4-2022

Backend

- Remove players with a steam id of 0
- Resolve warnings

Meeting

- Daily Standup
- Meeting with the client

9-4-2022

Frontend

- Remove the user CS:GO icon
- Add a create team functionality
- Add a delete team functionality
- Enable label filters on combined statistics
- Styling update

10-4-2022

Frontend

- Change label delete colour
- Change hover table colour
- Update match round display
- Add sidebar back to the upload page
- Update the maximum file size to 1GB
- Add more statistics to rounds
- Add ratings to matches

B.5 Sprint 4

11-4-2022

Frontend

- Restructure the admin directory
- Change the dashboard team list to a flex component

Backend

- Add an endpoint to query all maps

Meeting

- Daily Standup

Poster

- Create a first version of the poster

Report

- Update the progress

12-4-2022

Backend

- Add an endpoint to query all maps

Meeting

- Daily Standup
- Attend the peer-review

Poster

- Update the poster with feedback from the peer-review and the standup

Report

- Add more references
- Write a more detailed Introduction
- Further explain the Mission
- Add the Tools
- Add the Team
- Add minutes of the Requirements interview
- Add minutes of the First Prototype interview
- Add minutes of the First Player Meeting
- Add minutes of the Second Player Meeting

13-4-2022

Frontend

- Remove in-game name field from register form
- Add unfinished lineup editing
- Add player search

Meeting

- Daily Standup

14-4-2022

Frontend

- Add skeletons to player management
- Add progress indicator to player management
- Implement player management functionality
- Change round buy type display
- Add proper SWR mutation to team (un)linking

Meeting

- Daily Standup

15-4-2022

Frontend

- Update API spec
- Fix various runtime warnings
- Add link to upload page from CS:GO section
- Add more statistics to match overview
- Add match VODs

Backend

- Add VODs to the pipeline

General

- Work on the poster

Meeting

- Daily Standup

18-4-2022

Frontend

- Remove sorting by date
- Make the application more mobile friendly

Backend

- Fix the trade statistics
- Add uploaders to matches
- Add users to the DetailedMatch
- Restructure the application.py
- Automatically add labels

General

- Finish the poster

Meeting

- Daily Standup

19-4-2022

Frontend

- Add uploaders to the match overview
- Update the OpenAPI spec
- Handle the 409 from the form
- Improve responsiveness
- Add a dynamic number of lineups
- Display the percentage at the upload
- Add permissions to matches and rounds
- Make the tables consistent
- Require admin permissions for labels

Backend

- Add more comments
- Explain the project structure
- Add or/and feature for teams in a match

Meeting

- Daily Standup
- Meeting with the Data Science team

20-4-2022

Frontend

- Loosen restrictive permissions
- Cleanup the admin
- Update API calls for matches
- Fix the data picker
- Add a map icon to rounds
- Fix the textfield sizing for the filter bar

Backend

- Fix tests
- Add missing permissions

General

- Create the slides for the presentation

Meeting

- Daily Standup
- Poster presentation

21-4-2022

Backend

- Split off the architecture

Report

- Finishing touches

Meeting

- Daily Standup
- Presentation for the chair of the supervisor

Appendix C

Source Code

The source code of the backend can be found at GitHub. The manual describing the architecture and layout of the backend project can be found in Architecture.md in the backend repository.
The source code of the frontend is hosted on GitHub. The frontend repository also includes a README.md and HANDOVER.md that contain general information about the project and explanations on how to set it up.