UNIVERSITY OF CALIFORNIA,
IRVINE

CapturEsports: A Multiplayer Data Aggregation Tool for Facilitating Esports Research and Analysis

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Software Engineering

by

Daniel Yao

Dissertation Committee:
Professor André van der Hoek, Chair
Professor Constance Steinkuehler
Professor John Joseph
Professor James A. Jones

2020

DEDICATION

To my parents, family, and friends.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude and sincere appreciation to Professor André van der Hoek for his dedicated guidance and exceptionally contagious positivity and excitement for software engineering, research, and life in general. This thesis would not have been possible without his brilliant advice, unique insight, and honest feedback.

I would like to thank Professor Constance Steinkuehler and Professor John Joseph for providing the resources, motivation, and purpose to drive this thesis in the best direction possible.

I would also like to especially thank my team Jason Reitman, Je Seok Lee, and Sandy Pan who were integral throughout the development of this thesis. Throughout the design, implementation, and testing phases, their continued feedback, developmental support, and research insight made this thesis a success.

Finally, I would like to give special thanks to my parents for always believing in me and giving me their undying support and especially Kelsey for her encouragement and moral support through the ups and downs of this thesis.

ABSTRACT OF THE THESIS

CapturEsports: A Multiplayer Data Aggregation Tool for Facilitating Esports Research and Analysis

by

Daniel Yao

Master of Science in Software Engineering

University of California, Irvine, 2020

Professor André van der Hoek, Chair

The growing prominence of esports and competitive gaming has not only attracted a global audience but captured the curiosity of academic researchers. With increased research interest, the necessity for tools to collect contextual game data has subsequently risen significantly. Currently, a combination of generalized software solutions such as keyloggers, screen recorders, and audio recorders can be used to gather artifacts regarding an individual player. However, these solutions are inadequate when used to support research questions concerning player interactions in a multiplayer context. To address this problem, I introduce CapturEsports, a software tool to facilitate research in esports by supporting the capture and organization of multiple streams of player data. Focusing on League of Legends developed by Riot Games as a proof of concept, CapturEsports enables researchers to capture screen video recordings, voice communication, keystrokes, and in-game contextual events (such as player kills, buildings destroyed, and item purchases) for multiple players simultaneously, thereby facilitating analysis of team behavior. CapturEsports performs data collection and provides access to the data remotely, allowing studies to be conducted in a distributed manner. In this thesis, I present the

motivation behind CapturEsports, an examination of alternative software tools, and a detailed description of the tool I developed. I also discuss the primary design decisions and underlying requirements, the overall architecture and organization of data structures, and a preliminary evaluation and analysis of CapturEsports as an esports research application.

Chapter 1

Introduction

The esports industry has been on a continual rise to success in recent years, and its growing popularity has reached audiences worldwide. One hundred one esports tournaments were held in 2009, which grew to 696 tournaments in 2012. Annual prize money also rose, from $2 million to $10 million, respectively (Phillips, 2020). Throughout this growth, multiple new game titles emerged, including Overwatch, Starcraft 2, Hearthstone, Dota 2, and League of Legends, each solidifying a unique and ever-expanding community around the game.

As esports is gaining cultural popularity in society, its domain stretches into previously established industries unassociated with esports. For example, simulation racing (also known as "Sim Racing") recreates the experience of real-world racing as an electronic sport, catching the attention of professional drivers, one of whom described it as getting "the same buzz as racing for real" (Barretto, 2020).

In the world of cycling, professional cyclists are turning to virtual means, as it provides a greater, more versatile approach to practice. As the president of the Union Cycliste Internationale stated, "It is a new way of practicing cycling that is expanding rapidly and enables more athletes, whether beginners or more experienced, to train and race regardless of what the weather is like and where they live" (Cyclingnews, 2019).

Given this popularity in professional industries, it is no surprise that esports has garnered the attention of the academic community. There are numerous opportunities of interest surrounding

the study of esports, whether it is understanding it as a cultural phenomena (Hamari, Sjöblom, 2017), as a psychological dependency (Bányai, Griffiths, Király, Demetrovics, 2019) or as a source of data to examine human connection, interaction and team play (Lee, Ramler, 2019). With a spreading excitement for esports in professional contexts, researchers are motivated to better understand how the learned concepts can be expanded to various domains. Therefore, the significance of supporting such research is amplified to further this field of knowledge.

In this thesis, I focus on League of Legends (League), a competitive game in the multiplayer online battle arena genre. League is a game in which two teams of five players compete to destroy each other's base. Each player selects a character with unique abilities and must work together with others on their team to defeat the opposing team. League, created by Riot Games, has grown from a small, budding computer game to a blooming internet sensation enticing more and more players, in the process creating a global community of eight million concurrent daily players (Goslin, 2019). Since the inception of professional play in 2012, the League of Legends World Championship Finals, broadcasted online worldwide each year, grew from roughly 1 million concurrent viewers in 2012 (Auxent, 2016) to a substantial 44 million concurrent viewers in 2019 (Staff, 2019).

Many kinds of esports studies will require an analysis of gameplay data, such as sequences of actions a player makes or interpersonal communication with teammates. Therefore, it is important that we have the ability to collect data in a convenient and reliable manner. For instance, studying team performance and coordination might require observations at a level that clearly exemplifies interaction between players, such as a player verbally explaining their plan of attack to their teammates. As another example, to study individual behaviors and responses to integral moments of competitive stress, one may want to examine keystrokes that occur within a

short time frame to capture moments where a player frustratingly presses a key repeatedly when their character is killed.

Existing tools already make it possible to capture certain data points from single-player play in League of Legends (see section 2). From a research perspective, however, this would be insufficient, as the overall context of decision-making processes as well as impetuses of team coordination are unavailable. Data regarding a single player is inadequate when asking questions that involve multiple player interactions. While it may be plausible to manually combine data traces from multiple individual players, subtle but important hurdles exist that make this a daunting and involved exercise. The most straightforward method involves physically transferring the observation files to an external storage device for every single machine used. Consequently, not only is this approach immensely time-consuming, the study would also require participants to be co-located within a close proximity. Thus, a cohesive automatic method of collecting raw observational data in a multiplayer context is needed to effectively support researchers interested in esports data.

In this thesis, I present CapturEsports, a software tool that collects and synchronizes raw temporal data (such as player kills, ability usages, and item purchases), as well as audio recordings and screen capture videos from multiple players for the game League of Legends (League). The captured data is automatically brought together, synchronized, and stored in a central networked location.

The remainder of this thesis is organized as follows: in section 2, I introduce the background motive and explore a few existing tools in the space of capturing esports gameplay data. In section 3, I describe the software requirements for CapturEsports. Section 4 discusses the high-

level design decisions and implementation details and is followed in Section 5 by a preliminary assessment of CapturEsports. In the final section, I discuss prospective capabilities and future extensions for CapturEsports to improve upon the current prototype.

Chapter 2

Background

Electronic sports (or more commonly known as esports) has been a massively growing industry in recent years. Video games have evolved from casual play with friends to a highly competitive profession. With many roots from traditional sports that feature a heavier emphasis on physiological fitness, esports expands the gameplay of sports to incorporate deeper cognitive factors that affect decision-making at the team and individual levels. Typically, competitive esports is comprised of a small two to five-person team where the players must work together towards a common objective or end goal. The players are not required to be in the same physical proximity, which emphasizes the necessity for clear and efficient communication among teammates.

2.1 Esports Research

Within the expanding realm of competitive esports, the academic community has also experienced a fast-growing interest in the research aspects this environment can provide. A prominent area of interest is team dynamics and the sociological facets surrounding esports teams, e.g., profiling successful team behaviors (Nascimento, Melo, Costa, Marinho, 2017). To support this type of research, different data must be captured depending on the type of phenomenon of interest. In the section below, we provide examples of prior studies, the data on which they relied, and the data collection techniques they used.

One study looked into measuring and utilizing collective intelligence or group ability between team members to predict performance (Kim et al., 2017). As social interactions are beginning to shift from the real-world to an online virtual world, developing methods of cultivating relationships that contribute to collectively intelligent teams provides an understanding of teams in diverse domains. Through an assessment of player statistics in League of Legends (such as rank, total wins, average number of kills, and hours played) as well as a test of collective intelligence, Woolley et al. discovered that collective intelligence predicted competitive performance and positively correlated with the average social perceptiveness of the team (Woolley, Chabris, Pentland, Hashmi & Malone, 2010). However, the dataset consisted of summative statistics and profile information, and was only available to the study through an approved collaboration with the developers of League.

Although partnering with game developers is not always a viable option, certain games permit harvesting data through an API. In a paper examining team formations in the game Dota 2, Pobiedina et al. utilized a statistical approach to identify aspects of successful teams (Pobiedina, Neidhardt, Moreno, Grad-Gyenge, & Werthner, 2013). Through an API, game information was extracted from roughly 87,000 matches, including player profile data (signup date, country, list of friends, etc.) and match details (in-game damage, gold earned, skill level, etc.). The resulting dataset was used to train a logistic regression model to identify significant factors of successful teams. The authors found evidence positively influencing team success when players embraced diverse roles, played with friends, and selected a proper team leader. This study demonstrates the potential use of esports data and statistical techniques to discover patterns of social behavior.

In another study examining player behavior, machine learning was utilized to identify distinct clusters of players exhibiting similar patterns of playing strategies and behavioral progression

(Sapienza, Bessi, & Ferrara, 2018). The League of Legends API was used to obtain historical game data for roughly 900 players, and this data was then used to investigate features such as gold income, types of damage dealt, and health restoration. Using k-means clustering from machine learning, players were characterized by groups of features such as kills and earned gold or assists and earned gold. The researchers then employed a technique called Non-negative Tensor Factorization (Sapienza et al., 2018) to discover data correlations in various dimensions of time. One group of players they identified would focus on earning gold and were characterized by both a high number of kills and an above-average death rate of themselves. Another group would focus on assists and was characterized by stronger social behavior resulting in higher team collaboration. From the analysis, this study revealed distinct playstyles for each group showing the potential of assessing chronologically indexed data to derive player behavior.

As a final example, in a study of the game Starcraft 2, Bosc and colleagues investigated how employing sequential pattern mining algorithms can reveal hidden strategies to players and coaches (Bosc, Kaytoue, Raissi & Boulicaut, 2013). By gathering timestamped in-game events through web scraping game replays, ordered sequences of actions were compiled into a large database in which pattern mining algorithms, such as PrefixSpan (Han et al., 2001), discovered unique strategies from repeating sequences. Considering an example, one of the top fifty sequences involved rapidly building multiple instances of a defensive structure in the opponent's base causing fatal damage. The pattern mining also revealed the opponent's inability to effectively defend regardless of the number of resources dedicated in response. This indicated an imbalanced strategy within the game which could be heavily exploited. As a result, these kinds

of analyses not only aid game analysts but can also provide insight for developers to balance the game.

2.2 Free-to-use and Commercial Tools

The software tools currently used to capture esports data are limited to the collection of individual performance statistics. As game companies and developers typically do not build software to assist players with personal performance, third-party tools have been developed to reinforce player mechanics and game knowledge to the individual. These tools can output useful information to the player, such as average Actions per Minute (APM), instant short video replays of crucial moments in the game, and individual comparisons to relative community statistics.

Despite having convenient and advantageous functionalities for individual play, these tools fall short when examining the context of the game from a team perspective. While gathering individual statistics is vital to assess performance, much of the team dynamics and decision-making processes are lost to an inability to easily capture multiple data streams simultaneously as well as to an inability to provide context for interpersonal communication among team members.

2.2.1 LoLWiz

LoLWiz is a third-party application running on the Overwolf platform. LoLWiz provides in-game strategies (such as suggested item purchases), enables users to quickly scout the performance statistics of their teammates and opponents, and composes a short post-game video with highlights of important moments. While this tool is excellent for supporting players through a live game, the post-game output only contains summative and descriptive statistics regarding the most recent game. Since LoLWiz does not output specific user inputs or communication, it would be difficult to capture full observations, as it leaves gaps of contextual information required for a concrete analysis.

Figure 2.1: LoLWiz displaying opponent statistics and suggested item purchases

2.2.2 Mobalytics

Mobalytics is a comprehensive analysis software designed for competitive players to analyze and improve their performance. Equipped with advanced filtering and responsive visualizations, Mobalytics presents users with clear metrics of where a player needs improvement. These include a user's improvement of gold income rate, contribution to team damage, or survivability throughout a match.

Additionally, by using the League of Legends API, this tool also implements machine learning to calculate unique statistics, such as player aggressiveness, consistency, and versatility. While the advantage of Mobalytics resides in its ability to identify player weaknesses, it does not provide raw unmanipulated data and only presents post-processed data.

Figure 2.2: Personal statistics output from Mobalytics

2.2.3 Shadow.gg

Shadow.gg is a paid application geared towards professional esports teams to analyze video replays and prepare for scrimmages against another team. Shadow.gg is capable of presenting complex data visualizations and breakdowns of a team's performance as well as an individual player's performance. Users can view the tendencies of opponents based on historical match information, such as typical item purchases or character navigation around the map. This tool is exceptionally useful to team coaches and analysts as it streamlines the logistic overhead of fetching player performance statistics, aggregates past game data, and provides specialized tools for analyzing video replays, such as gold earned within a time period. While the full capabilities of Shadow.gg are unavailable without a full license, its main advantage stems from its clear data visualizations and auto-compilation of historical context. However, a possible limitation of this tool is that it captures player actions as decided by the game, not keystrokes. As a result, a player could begin to cast an ability then cancel the ability cast by right-clicking, which would not be captured by Shadow.gg but may be important for researchers to understand. A second concern lies in its ability to support detailed analysis of the data, as it allows the user to breakdown video streams but does not permit the user to run important queries on the captured data. While Shadow.gg is useful for professionals to look into opponents and how players play, actual analysis of the captured data with the mentioned limitations makes this tool difficult to use as a basis for research purposes.

Chapter 3

Requirements

While the existing esports data capture tools described in section 2 each have their advantages and disadvantages, none provide the ability to support data collection in a multiplayer context, store unmanipulated raw data, and submit database queries. In this section, I provide a list of requirements to fulfill the need to accurately capture multiplayer data in League matches.

3.1 List of Software Requirements

Each of the following requirements is inspired by informal conversations with a range of current researchers in esports, focusing on the types of studies they would like to perform and the observed shortcomings of existing research goals.

1. **Support data collection of multiple players in the same match**

   As League pits five players against an opposing team of five players, CapturEsports needs to support gathering data for ten players simultaneously. The basic process of collecting data for a single player involves recording a stream of data from the beginning to the end of a match. CapturEsports must extend this functionality to capture ten streams of data simultaneously within the same match and aggregate all streams to a centralized location. This requirement enables researchers to observe player interactions in a distributed manner, discarding the need for participants to be located within close physical proximity.

2. **Capture granular or raw data**

   While League offers statistics at the end of each match, it is comprised of aggregated metrics that cannot be decomposed into smaller pieces of data. A core aspect of data collection is the ability to gather information at a high level of detail. We refer to this data as granular or raw data in which no analysis or manipulation has occurred between the time of collection and the time the data is stored. Examples of this data in the context of League include user keystrokes and mouse clicks or in-game events such as types of objectives taken, players killed, items used, etc.

3. **Compose media recordings of a match to bridge contextual holes**

   Granular data stored, per the previous requirement, resides in a purely textual format, which can leave several unfilled contextual holes. In League, for instance, a player could hide in a bush without moving and wait to ambush an unsuspecting opponent. In this situation, the raw data being captured would be empty for some time since there is no actual input from the player. However, the player's behavior indicates a deliberate decision to wait and not move. As a result, it is necessary to include the ability to record the player's screen to capture the context of their interactions and output a video with audio that includes voice data.

4. **Time must be synchronized in the data across multiple players**

   A major component of aggregating raw data for each individual involves synchronizing the timestamp of each data point to accurately reflect the events of a match. For instance, explosive action in a match can result in numerous events and inputs occurring in rapid succession. Logging the exact timestamps of interactions that occur in an accurate manner

relative to one another empowers researchers to sequence player interactions chronologically in short periods of time and investigate team dynamics at a microscopic level.

5. **Minimal interference with gameplay experience**

   To decrease noise in the data, observed players must be performing in their natural state, and any disruptions that could interfere with their gameplay experience must be minimized. Once a match has started, CapturEsports should disappear from view and use as few resources as possible. It should be noted that on machines with lower performance capabilities, running multiple programs at the same time will potentially lower frame rates or cause slower network speeds. Therefore, CapturEsports intends to follow the optimal hardware requirements recommended by Riot Games, the developer of League, specifying at least 3GHz of processing power and 4GB of RAM (Horse, 2014).

6. **Extensible support for evolving gameplay and data**

   Unlike traditional sports where rule changes are infrequent, the rules of esports games can be relatively fluid. For example, in previous versions of League, killing a dragon rewarded a team with a flat amount of gold. However, in recent versions, killing a dragon rewards a team with a myriad of possible advantages, such as faster health regeneration, increased damage, and quicker movement. Our database schema must be flexible enough to support updated changes without modifying the structure of previously collected data.

7. **Support for non-game generated data**

   Support for data beyond game generated data may hold significant value for certain types of research. Physiological and biometric data, such as heart rate or facial expressions, can also be variables of importance to consider. Although we currently do not have the capacity to

perform actual collection of this type of data, CapturEsports must be future designed to enable such capture seamlessly with collected game data.

8. **The database can be queried using custom statements**

   Since manually parsing through a large dataset can become cumbersome, researchers must have the ability to submit custom queries to our database. This creates a more sophisticated method of retrieving the collected data in addition to retrieving the data as a single batch load. Custom queries also permit users to retrieve subsets of data by specifying parameters, such as getting all data collected between two dates or all data for a single team.

Chapter 4

Software Design and Architecture

In this section, I discuss the primary design decisions made that formed the structure of CapturEsports. I also provide further details about the different components and how each fulfills a role in the system architecture.

4.1 Primary Design Decisions

The overall organization of CapturEsports consists of a three-tiered architecture. As shown in Figure 4.1, the tiers are divided into the Client (or Presentation) Tier, the Service Tier, and the Database Tier. All three tiers work together to communicate and fulfill the requirements listed in the previous section. By implementing this separation of concerns, we are able to consolidate our design decisions by function, subsequently increasing the degrees of freedom for our system. This sets up a modular foundation that ensures our components are robust and decoupled and eases future development and maintenance. We begin our explanation of design decisions with the Database Tier, then discuss the Service Tier, and finally present the Client Tier.

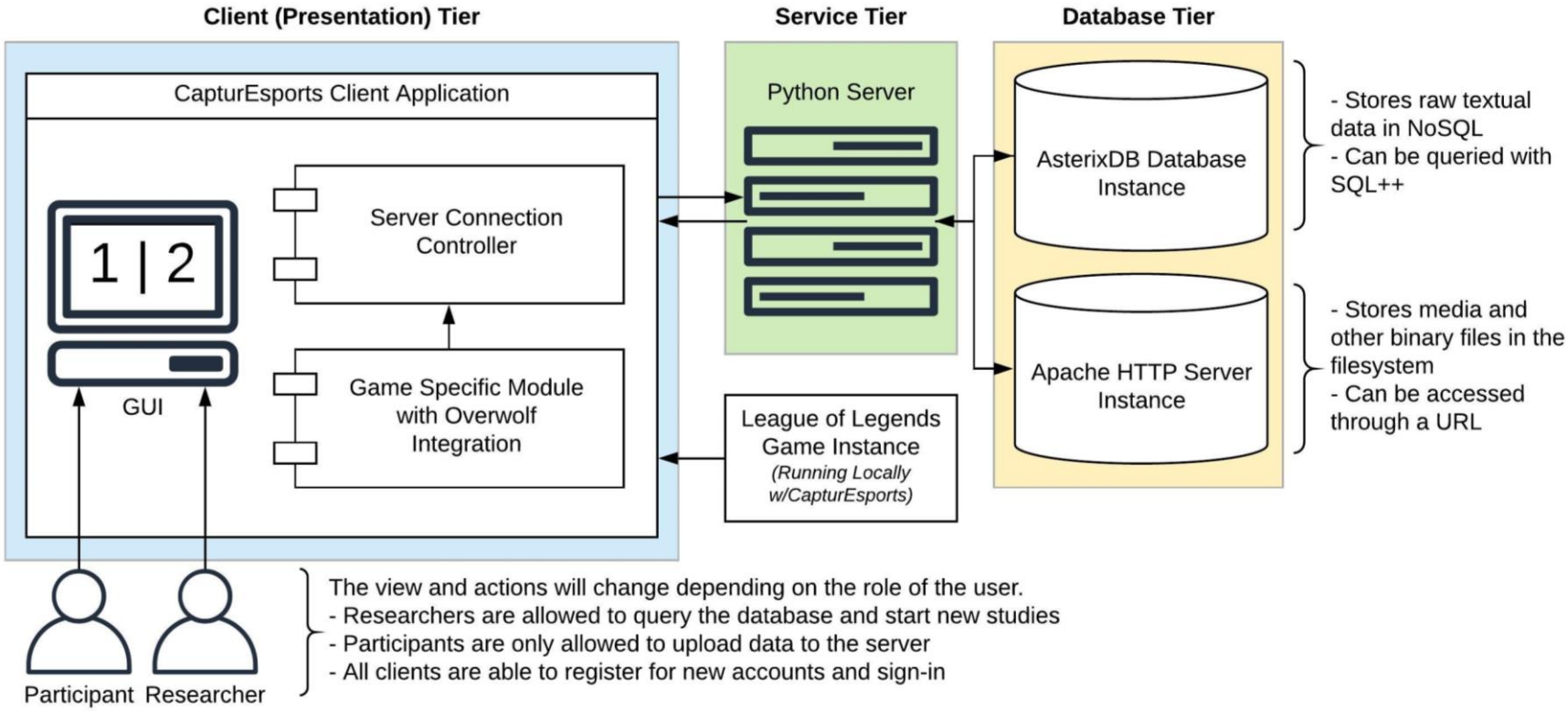


Figure 4.1: High-level overview of the CapturEsports architecture

4.1.1 The Database Tier

This tier is responsible for storing and organizing all the collected data and presenting users with methods for retrieval.

- **The database is semi-structured in order to handle multimodal data.** Since the structure of our data is not constant, managing a database using a strict schema would prove to be difficult. Drawing from our requirements, the CapturEsports database storage must handle textual raw data, video and audio media recordings, and physiological data, such as heart rate and facial expressions. Considering that a use case is retrieving the collected data, our data model is designed to easily link new multimodal data with each corresponding participant. To implement this decision, the Apache AsterixDB Big Data Management System is employed. AsterixDB is a data storage platform with the purpose of analyzing and storing data pertaining to "evolving world models" (Behm, Borkar, Carey, Grover, Li, Onose, Tsotras, 2011). Using a NoSQL semi-structured model, the data can be queried using the language SQL++, which is derived from the traditional semantics of SQL. While traditional SQL syntax interacts with a flat relational schema, SQL++ is modified to better handle nested data types (Alsubaiee, et al., 2014).

- **Collected data can be retrieved through submitting custom queries or downloading through the browser.** Our AsterixDB instance is purposefully designed to ease the amount of administrative overhead necessary for a researcher to supervise a study. After data collection, we enable researchers to submit custom SQL++ queries to retrieve textual raw data. These queries permit researchers to retrieve partitions of data or cross-reference with earlier stored data providing greater analytical power. However, due to the multimodal nature

of our data, storing large binary files (such as video) in the database would be resource-intensive and highly inefficient. Thus, we employ Apache HTTP Server to run in conjunction with AsterixDB. Apache Web Server offers the ability to download files through the browser from a specific URL. While the internal organization of AsterixDB and Apache Web Server is different, the raw data is the same and accessible through both methods.

- **Collected data is anonymized to protect user identities.** Due to the sensitive nature of our collected data, any personally identifiable information in our database is hashed. This includes a user's login email as well as their password. Although we anonymize the data, we maintain an auto-assigned identifier used to associate data with a specific user. This methodology will be explained in more depth in later sections.

4.1.2 The Service Tier

This tier is responsible for facilitating the communication between the Database and Client tiers. A few examples of these responsibilities include managing user login and logout, retrieving collected data from clients, and submitting custom queries to the database.

- **The server communicates with the other tiers and components using the REST protocol.** Each of our three tiers resides in separate locations, requiring us to communicate over a network. While there are many ways of sending and receiving messages across a network, we chose to use the REST protocol as it is a proven protocol that allows us to distinguish the valid entry points into the server.
- **Server access is limited through local network restrictions.** We globally limit access to our service components to permit only clients within the same network. An extensive amount

of sensitive data tends to be stored within our system, and researchers may not necessarily want to share the data with the world. In addition to our efforts to protect user privacy, we built a safety valve where only certain users have access. Currently, the CapturEsports server resides in the network of an academic institution and only accepts connections originating from the same network. The collected data is also password protected to further prevent unauthorized access.

- **The server can support at least ten simultaneous client connections for data upload.** Since a League match involves two teams of five players each, to collect observations from all players for a single match requires ten simultaneous connections to the server. While a constant stable connection is not needed, simultaneous data uploads would require a single-process server to queue and write data synchronously. As a result, large data files are uploaded in small chunks and reconstructed by the server, and our server instance is multi-processed to manage multiple data uploads asynchronously and prevent functionality from being blocked by a single connection.

4.1.3 The Client Tier

This tier is responsible for performing the actual data collection and runs on the local machine a participant is using. As displayed in Figure 4.1, the client tier runs in parallel with the game instance on the same machine.

- **The client collects data with minimal interference with the player's gameplay experience.** To provide researchers with the most accurate observations, we minimize the interference that CapturEsports could potentially reflect in the data. Since a subset of the data

pertains to human interactions within the observed game, any interactions with other installed software throughout the duration of a match can be mistakenly captured as well. Thus, we design CapturEsports to run in the background during a match and present a UI that only requires input before and after data collection. To achieve this, we use the Overwolf SDK, a platform for developing third-party applications to run alongside many different games.

Overwolf provides a sophisticated API to hook into actively running games enabling access to real-time game data which is otherwise difficult to capture. A vital aspect of this platform is its ability to facilitate non-obtrusive gameplay to assure data integrity. By running in the background, events triggered throughout the game are logged.

- **Timestamps of collected data points are synchronized locally before uploading to the server.** A major facet of gathering and organizing multiplayer data is synchronizing the temporal aspect of collected data. Because CapturEsports utilizes a machine's internal epoch time to timestamp captured data points, typically negligible variations in clock time between machines become significant when precisely mapping the exact sequence of multiplayer interactions and events. For example, if the clocks of two machines are 300 milliseconds apart and two participants left-clicked at the exact same moment, our unsynchronized data would incorrectly reflect one player who left-clicked 300 milliseconds after the other player.

  To solve this time inconsistency among all players within a single match, we identify a point in time in the match that becomes our absolute truth. For League, the point we identified is a "minion spawn time," which occurs exactly 65 seconds after the match starts. Using this fact, we apply an offset formula to synchronize every data point relative to the official match start

time retrieved from the Riot API (provided by the developers of League). With this information, our offset formula to resynchronize the data is:

$$Synced\ Time\ ms = (Data\ Point\ Timestamp) + (Riot\ API\ Start\ Time + 65000ms) - (Local\ Minion\ Spawn\ Time)$$

Once the collected data has been resynchronized, it is sent to the server to be inserted into the database. We chose to synchronize data locally first due to the varying differences specific to each local machine and to offload processing power from the server to the client.

4.2 CapturEsports Database Architecture

The organization of our AsterixDB instance currently contains five datasets with three core datasets required for CapturEsports to function properly. Presented in Figure 4.2, the three core datasets are EsportsUserSet, StudySet, and LeagueDataSet.

From a high-level perspective, a CapturEsports participant creates an account and joins a study where the data from every match played is stored and linked with the study. The EsportsUserSet creates an anonymized identifier for each CapturEsports user and the LeagueDataSet is a comprehensive repository of all collected match data for every match observed by CapturEsports. The StudySet keeps track of a specific subset of CapturEsports users and links the collected data to the LeagueDataSet. Each dataset will be discussed in more detail in the subsections below.

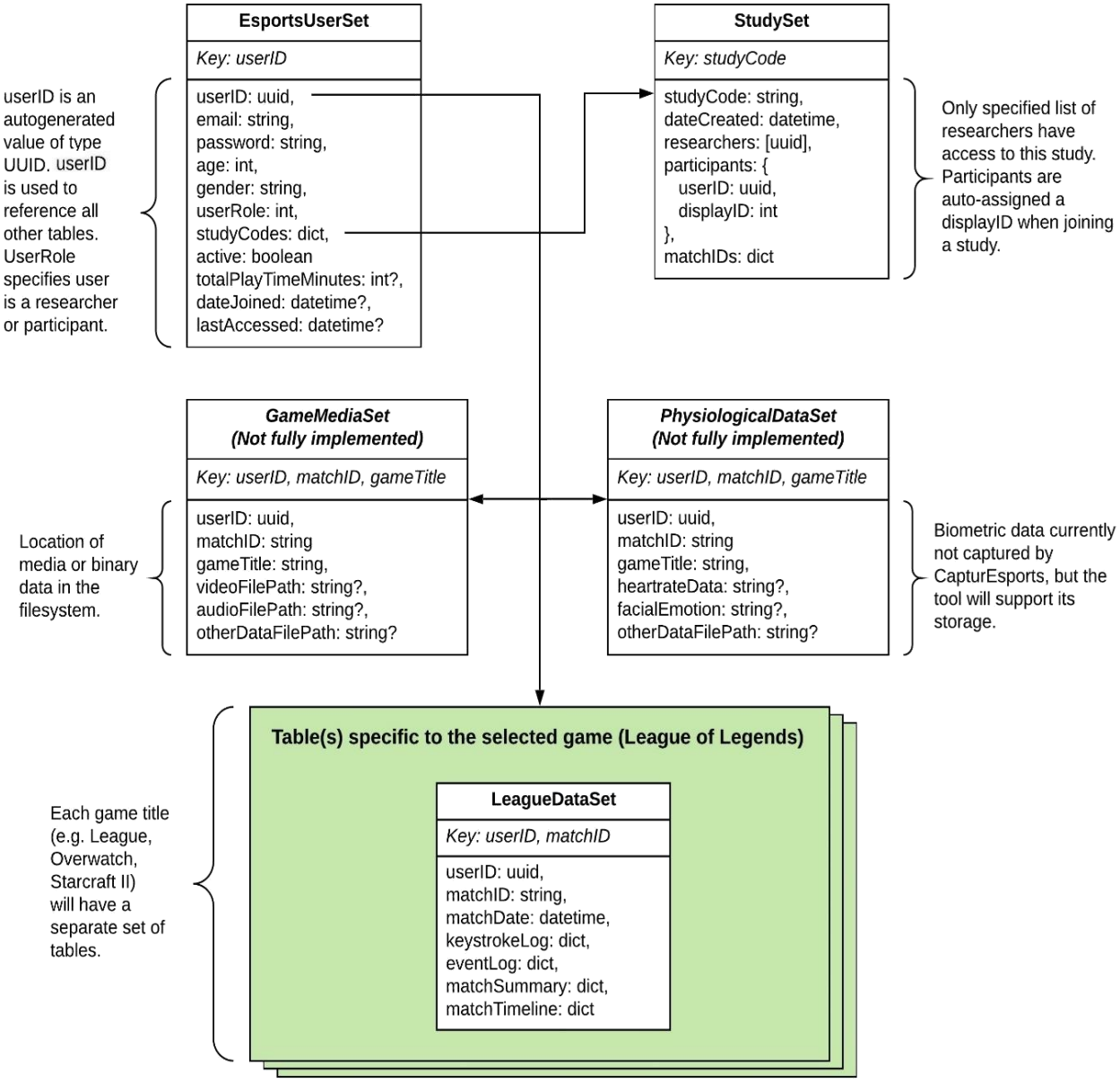


Figure 4.2: Overall database architecture for CapturEsports

4.2.1 Database Schemas

In this section, I will explain the type definitions of each dataset and clarify how certain fields are associated with each dataset. It is important to note that the type definitions in our AsterixDB instance are defined as “open,” meaning datasets are permitted to carry extra fields that are not explicitly defined.

- **EsportsUserSet**

  The EsportsUserSet stores all information regarding participants and researchers who have registered accounts through CapturEsports. An email and password are required to create an account with optional information that can be entered at the user’s discretion, such as age or preferred gender. To ensure the privacy of our participants, emails and passwords are hashed using the MD5 algorithm to obfuscate the plain-text user input before inserting it into the database.

  Once a user has created an account, an autogenerated universally unique identifier (UUID) userID is assigned to their account, which becomes a foreign key to associate the user with other datasets. A userRole is also assigned to identify if a user is a researcher or a participant. Researcher roles are given full access to the collected data, whereas participant roles can only upload their own data. To restrict access, newly created users default to a participant role and researcher roles must be authorized and manually created by an existing researcher or CapturEsports developer. The remaining fields totalPlayTimeMinutes, dateJoined, and lastAccessed are optional fields that provide basic metadata regarding a user.

- **LeagueDataSet**

  The LeagueDataSet contains the granular data of every study collected from each participant, as well as game data retrieved from the Riot API regarding each League match played while being captured by CapturEsports. Each match has an associated match identifier number assigned by Riot and is used in conjunction with a user's UUID to identify a participant's collected match data. For each participant in each match, their raw textual data includes four data fields: MatchSummary, MatchTimeline, EventLog, and KeystrokeLog.

  The MatchSummary holds the end game statistics for the entire game. This includes items such as total gold, largest damage, and dragons killed. MatchTimeline contains data regarding the timeline of major events that occur throughout the game such as when a player is killed or when an objective is taken. Both of these fields are provided by the game developers and retrieved through the Riot API.

  The EventLog and KeystrokeLog are different from the previous fields as they are collected on the client machine using the Overwolf SDK. The EventLog contains event data performed by an individual player. While similar to MatchTimeline, the EventLog differs by including all events in addition to major events, such as when an ability is used or when an item is consumed. Unlike the other data fields, the KeystrokeLog contains data non-specific to League. Throughout a match, it logs every time a key is pressed and released regardless of the input result. All raw data is stored in a JSON format (or as a dictionary) meaning that each raw data element is stored as a key/value pair.

- **StudySet**

  The StudySet consists of specific logistic information regarding a conducted research study. A researcher generates an alphanumeric study code that is used to identify which players participated and which collected data is associated. The creation of study codes is restricted to only registered researcher users to limit access to the database, and each created study code is an inputted string that is automatically converted to all uppercase to enforce case-insensitivity and avoid study code name collisions. Within a single StudySet entry, a list of researcher userIDs is stored to specify authorized researchers from accessing the study data. A dictionary of participants is also stored where each participant is assigned an autogenerated display ID as their unique identifier throughout the duration of the study. Finally, a dictionary of match IDs is stored in chronological order with each match ID retrieved from the Riot API used to reference the collected data in the LeagueDataSet.

- **Other Datasets**

  The CapturEsports database also contains two additional datasets, the GameMediaSet and the PhysiologicalDataSet. Currently, both datasets are not implemented or populated, as they are not essential for fulfilling the system requirements. The presence of these datasets primarily sets up the database to support new data types in the future. GameMediaSet holds the information for locating additional media data associated with participants in the filesystem, which could include video recordings of body language or pre-game team discussions. Although our usage of Apache HTTP Server allows users to download video recordings directly from the filesystem, handling several media files per user can become cumbersome and this dataset organizes multiple files in order for CapturEsports to scale.

The purpose of the PhysiologicalDataSet is to store any biometric data that researchers may have collected. While CapturEsports is not capable of capturing this data type, we want to enable the storage of physiological data collected through other means. Examples of physiological data include, but are not limited to, a participant's heart rate, facial expressions and emotion data, or skin conductivity sensors. Although the current format of the data is unknown, the "heartrateData" and "facialEmotion" fields in the PhysiologicalDataSet are string file paths to the location of a binary file.

4.3 CapturEsports Server

The CapturEsports server is the sole point of network interaction, responsible for routing connection requests received from users and maintaining authorized access to the database. The web server is implemented in Python and utilizes the Flask library to abstract core server functionalities, such as binding to ports and listening for requests. This section describes the main responsibilities of the server in greater detail.

Employing REST calls, all requests and responses are formatted in JSON due to its straightforward integration with web applications. When a participant first logs into CapturEsports, their hashed email, hashed password, and plain-text study code are sent to the server. The server then checks the existence of the hashed email and study code in the database to ensure the user is indeed a valid participant as logged by the study organizer. Once verified, the user's CapturEsports UUID and their assigned display ID specific to the provided study code are returned to the client. These two IDs anonymously identify the user throughout the entire

CapturEsports system and duration of a study, respectively. For all following requests, the client includes the UUID for the server to verify the authenticity of the user request.

CapturEsports has the option to record gameplay video and audio, typically outputting a file of a size roughly 300-500MB for a 40-minute game of League. The server is built with the Resumable library to enable large files to be uploaded in smaller chunks to a temporary folder. Each chunk is uniquely identified and linked to a specific output file by a substring in the filename. Once the server receives the final piece to a file, all chunks are consolidated, joined together, and moved to a directory accessible through the Apache HTTP Server. Uploading files as such provides researchers with remote accessibility to observational data in an organized manner almost immediately after a study has concluded.

While the architecture of our server component does not immediately impact the end user, it greatly affects the future maintenance of CapturEsports. Implementation using the Python programming language enables our server to easily separate functionalities into distinct modules. Currently, the server is divided into five modules, with Figure 4.3 providing an explanation of each individual server component. Because of this modular structure and the inherent nature of Python, common functionality can be developed in separate files which can be imported easily. This facilitates the integration of new features and support for separate games in distinct modules.

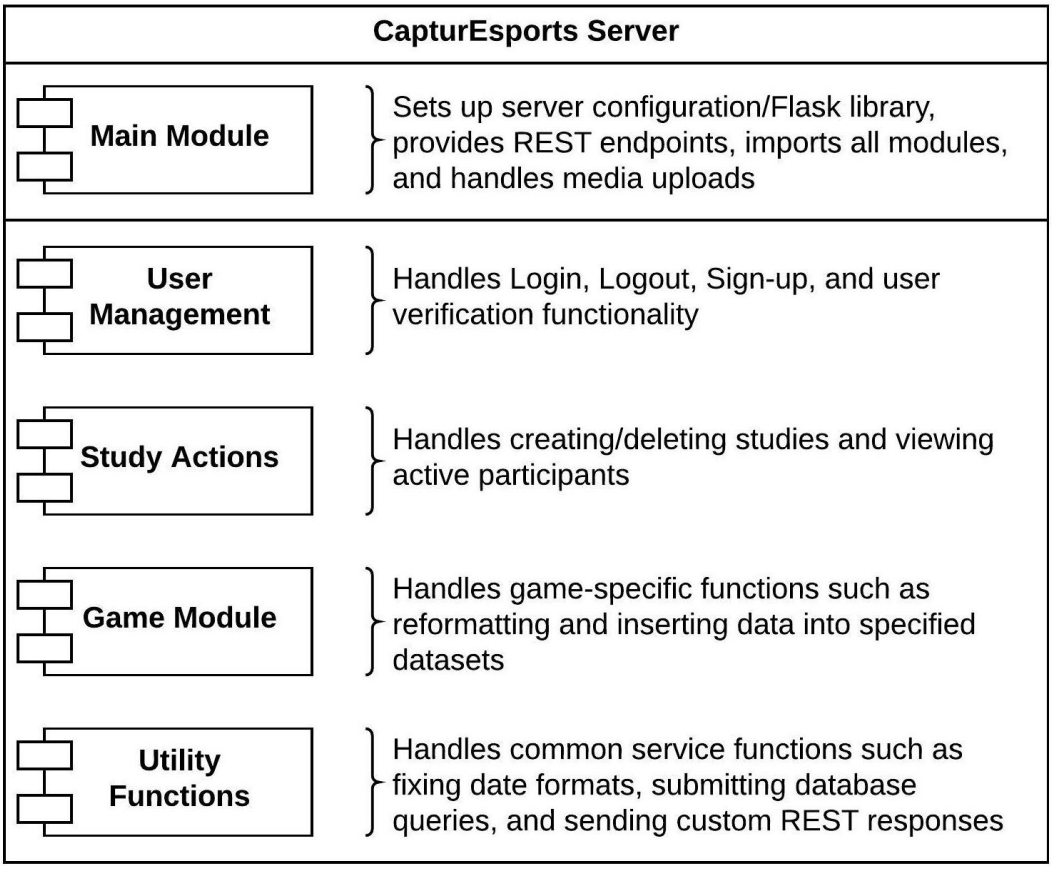


Figure 4.3: Explanation of Individual Server Components

4.4 CapturEsports Client Application and Software Components

Our client application runs on a participant's local machine concurrently with League. Following the classic Model-View-Controller design paradigm, the client application offers two different interfaces: one for a participant user and one for a researcher user. Taking a closer look at the Client Tier mentioned in Figure 4.1, Figure 4.4 provides a detailed diagram of the client architecture, entailing three major software components: the App Connection Controller, the League Collector Module, and the Researcher Module, each with a corresponding UI view to handle user input and interactions. Since participant and researcher roles have distinct sets of use cases, distinguishing each role with separate components organizes our system by use case rather than functionality. Common operations utilized by both roles exist in the App Connection controller and the more specialized methods for each user role reside in their respective modules. As a result, the components loaded at runtime are based upon the role of the user after they login.

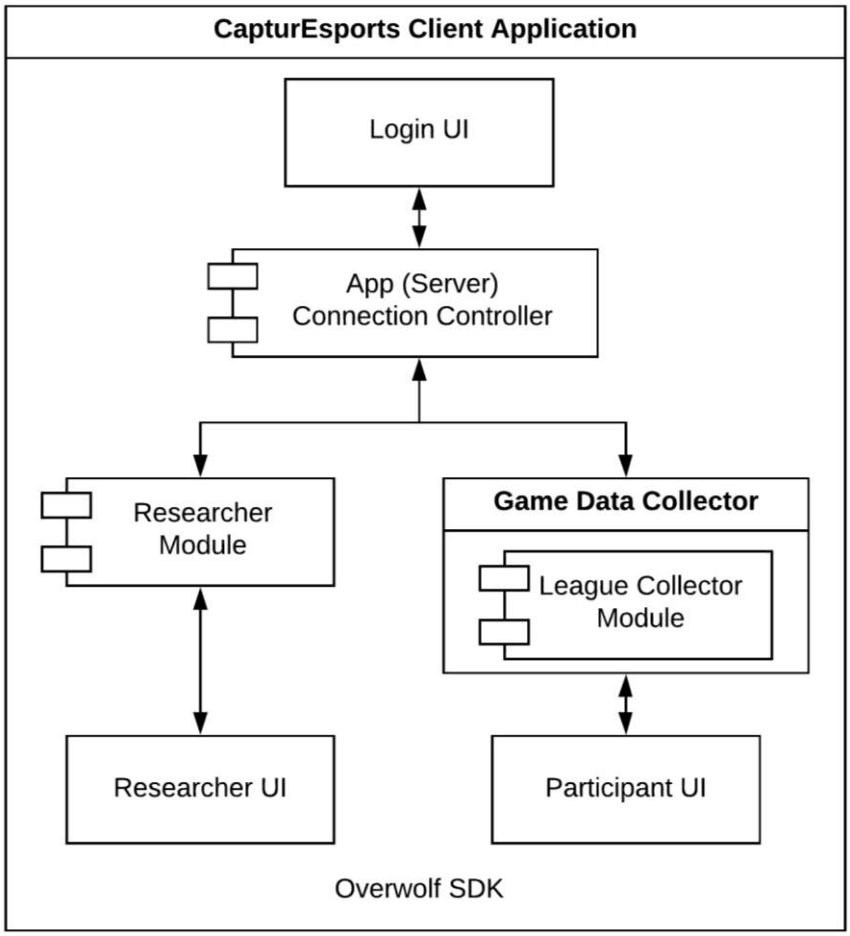


Figure 4.4: CapturEsports client architecture built using the Overwolf SDK

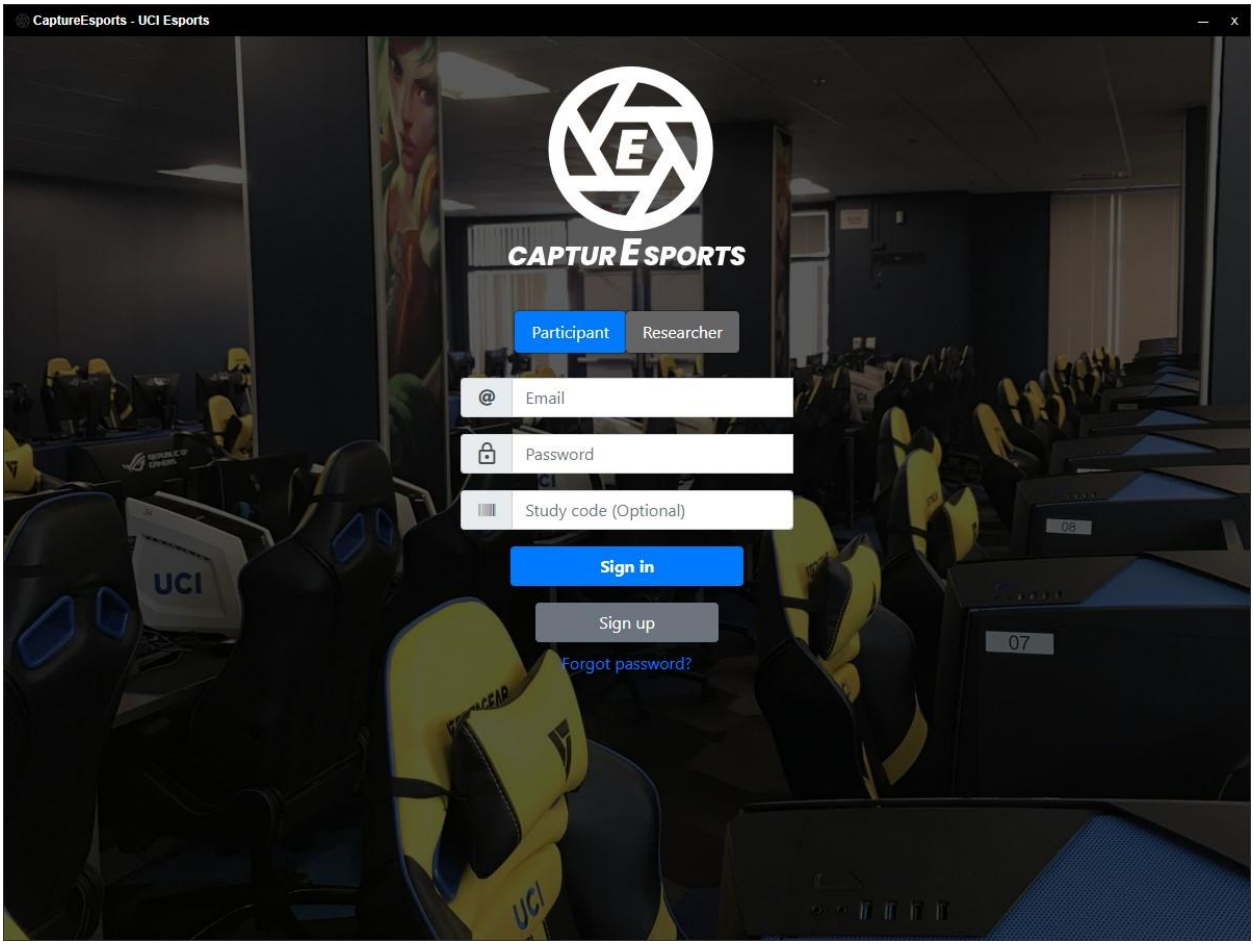


Figure 4.5: CapturEsports Login Page

The App Connection controller is the main component for connecting the client to the server instance to enable the functionality of sending and receiving REST calls. Correspondingly, this component is responsible for networking actions such as logging in, registering new users, and uploading data. The CapturEsports login page in Figure 4.5 allows users to login as a participant, login as a researcher, or register a new account. Depending on the login, the user is redirected to the view corresponding with their role. Along with the login, participants can join a specific study using a code created by the researcher. If no study code is entered, the user is defaulted to the study “NOSTUDY” in which data can still be collected in an unsupervised setting. This

allows participants to contribute to our growing database of games for analysis at their own discretion.

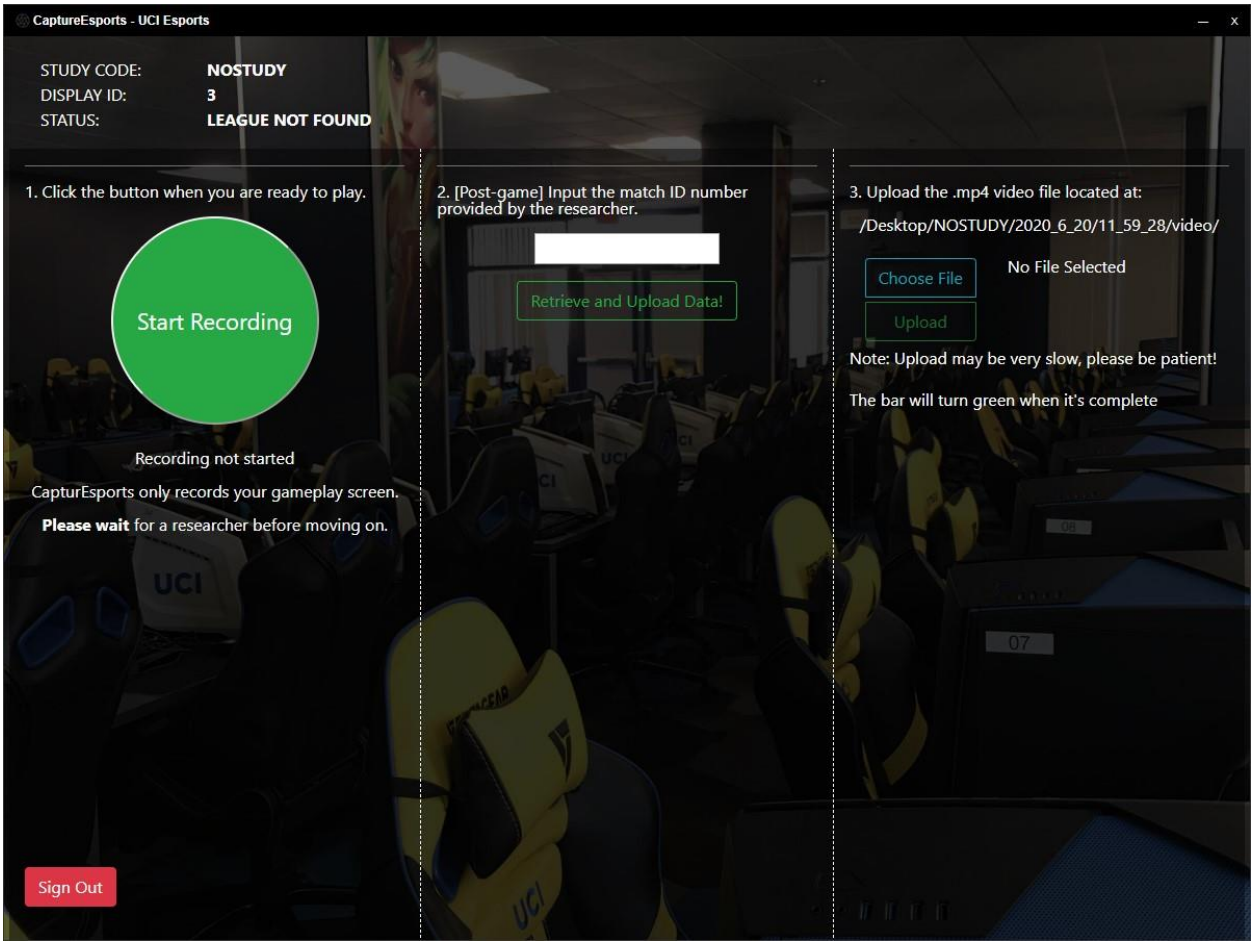


Figure 4.6: CapturEsports Participant Interface

The League Collector Module is the component that performs the actual in-game data collection and is loaded at runtime by the Game Data Collector. While our current implementation only supports the game League, this design permits future iterations to easily support other game titles by swapping out the League Collector Module. However, our current participant view as displayed in Figure 4.6 is specific to League to simplify the development of our prototype. The top header section provides logistic information to the participant, showing the study they joined, their temporary assigned display identifier, and the status of whether League is running on their

machine. In step 1, pressing the large green circular button indicates a user is ready to begin the study, however CapturEsports does not begin recording until an actual League match begins. Step 2 requires the participant to input the match ID number provided by the researcher to correctly pull data from the Riot API and upload textual data. Finally, step 3 prompts the participant to navigate to the location in their filesystem where their recorded gameplay is saved and upload the video.

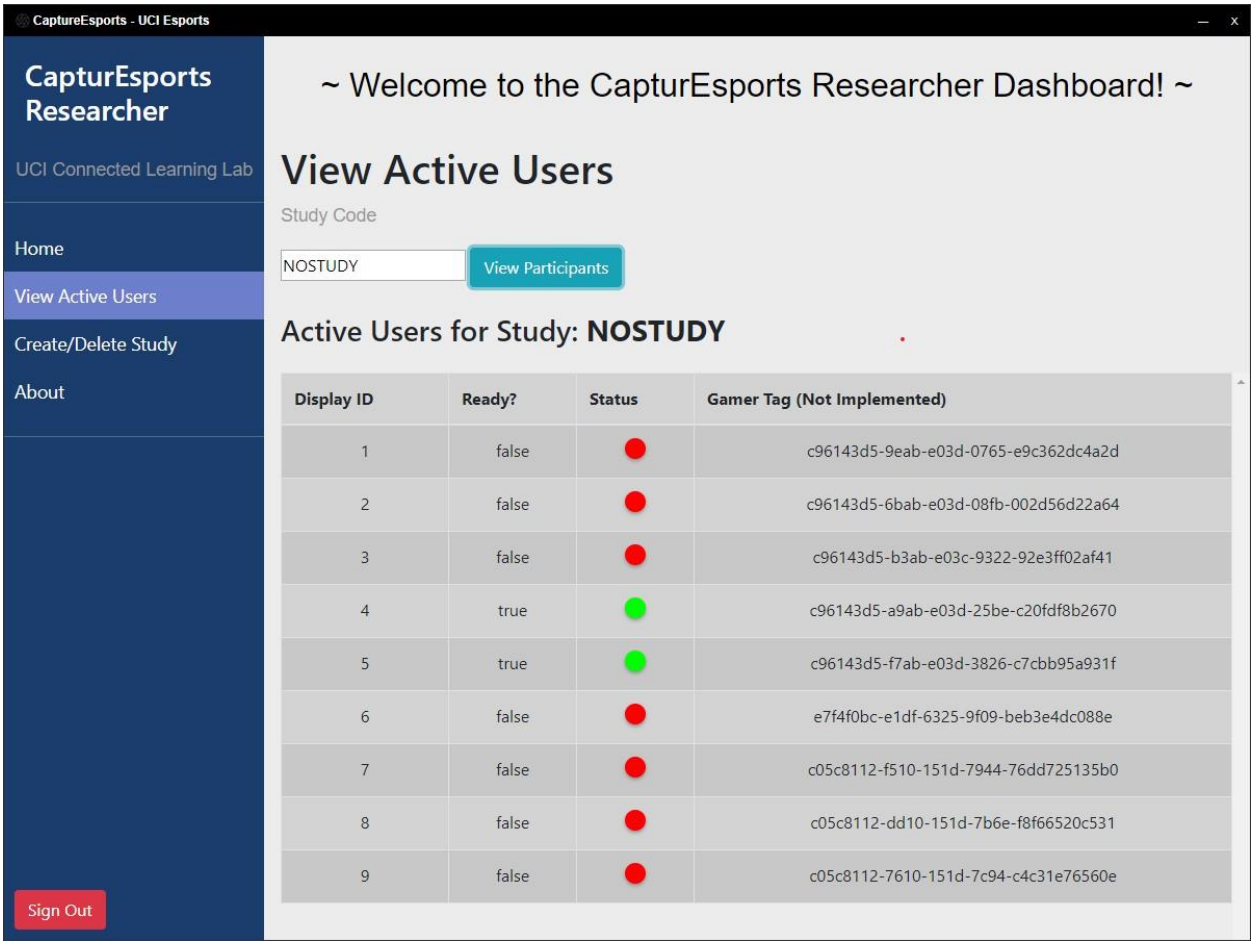


Figure 4.7: CapturEsports Researcher Interface

The Researcher Module provides basic functionality for researchers to conduct a study with CapturEsports. When a user with a researcher role logs in, they are currently presented with two performable actions. Figure 4.7 illustrates the ability to View Active Users for a study. When a

participant clicks the “Start Recording” button in their interface, their ready status is updated in this researcher view, changing their ready value and status color signifying the participant is ready for the study to commence. In addition, the Create/Delete Study option enables the ability to create new studies and delete existing studies with the caveat that they are a researcher on the study and no data had been collected yet. This is enforced by the “researchers” field in the StudySet presented by Figure 4.2 indicating users with access to the study.

Chapter 5

Preliminary Evaluation

In this section, I present a preliminary evaluation of the requirements implemented by CapturEsports. First, I describe an alpha test of CapturEsports involving current game researchers at the University of California, Irvine and examine the tangible output that resulted. Afterwards, I explore the potential data analyses made possible by CapturEsports.

5.1 CapturEsports Alpha Test

Throughout the development of CapturEsports, we periodically conducted various trials to verify the correctness of individual components and ensure proper system communication. This gradually led to a trial of eight volunteers, which allowed us to discover reliability and performance bugs. Although many trivial bugs were discovered, such as small syntax errors and conflicting filenames, the majority of critical errors were uncovered as a result of supporting multiple players. For instance, in previous builds, when multiple participants attempted to upload data, an incorrect configuration caused the server to unintentionally overwrite existing data. Through incremental testing, CapturEsports improved upon server stability, client-side functionality, and overall security and robustness.

At the culmination of our testing process, we conducted an alpha test consisting of ten players over the course of two hours of gameplay to simulate a real study setting. All players were located in their homes and installed CapturEsports on their machines. The players then created accounts and logged into CapturEsports with a specific study code. Once all players were logged

in and ready, they were randomly assigned to two teams and tasked to compete against each other. At the end of the match, five files were produced for each player, four of which were the raw textual data fields for the LeagueDataSet: MatchSummary, MatchTimeline, EventLog, and KeystrokeLog. The last file was a video file containing the player's screen recording and their voice communication. Following the actions outlined earlier in Figure 4.5, each player uploaded all collected textual and video data streams (with a total size averaging 300-500MB per player) within fifteen minutes from the end of the match. Figure 5.1 illustrates a raw data entry in the KeystrokeLog indicating a single keystroke, which includes the name of the key, whether the key was pressed or released, the XY mouse coordinates relative to the player's monitor, and timestamp data provided in the standard epoch format.

Figure 5.1: Raw data indicating a logged keystroke

5.1.1 Integrity of Captured Data

Given that our data is primarily used for research analysis, the accuracy of the collected granular observations is a crucial requirement to fulfill. Considering the variability of game states in League, we employed manual verification of data accuracy.

Figure 5.2: Visual representation of a dragon kill in the video recording


```
{
  "type": "ELITE_MONSTER_KILL",
  "timestamp": 773516,
  "position": {
    "x": 9913,
    "y": 4461
  },
  "killerId": 1,
  "monsterType": "DRAGON",
  "monsterSubType": "WATER_DRAGON"
},
```


Figure 5.3: Textual representation of a dragon kill in the MatchTimeline file

Figure 5.2 and Figure 5.3 present an example of an in-game event from the captured video and how it is reflected in the granular data. Figure 5.2 observes two players moments before killing a dragon at time 12 minutes 53 seconds. The glowing circle symbol on the left of Figure 5.2 specifically denotes the dragon type is water. Figure 5.3 corroborates this event where a water dragon was killed at time 773516 milliseconds, converted to 12 minutes 53 seconds and 516 milliseconds. Since the precision of the displayed video time is in seconds and the precision of the granular data time is in milliseconds, there exists a one-second ambiguity when verifying the data. Throughout multiple games, random spot-checking was performed on at least 20 instances of in-game events and cross-referenced with the recorded video. All instances had correctly reflected in-game actions within the bounds of the one-second ambiguity. This confirmed that despite the varying levels of detail, all checked instances showed the video timestamp was equivalent to the granular data timestamp down to the minute and second.

5.2 Potential Data Analysis

We also considered how CapturEsports permits a researcher to view and query the collected data after the conclusion of a study. To demonstrate, I present three potential scenarios of how researchers can query the granular data and perform exploratory analyses. It is important to emphasize that these three scenarios are not based on known or proven phenomena; they represent early manifestations of potential approaches of studying the data based on assumptions and hypotheses that may or may not be true. Nevertheless, CapturEsports allows the verification of unanswered hypotheses in a convenient manner and empowers researchers to structurally capture and study data across multiple games.

Scenario 1: A researcher aims to explore differences in playstyles among participants of varying expertise through specific aggregate statistics.

The disparity between experienced and novice participants is often most apparent to people with existing knowledge of League and may not be entirely discernable to others. However, there exist some hypotheses by which others have shown that the behavior of experts and novices differ. While these could be evident in varying durations of visual fixations (Kasarskis, Stehwien, Hickox, Aretz, & Wickens, 2001) or differing physical responses in keyboard presses, the distinction of expert players from novice players can also manifest empirically in their interactions and playstyles. Figure 5.4 shows a SQL++ query submitted to AsterixDB that calculates the total number of keystrokes for every player in a match with ID number 3402192685.

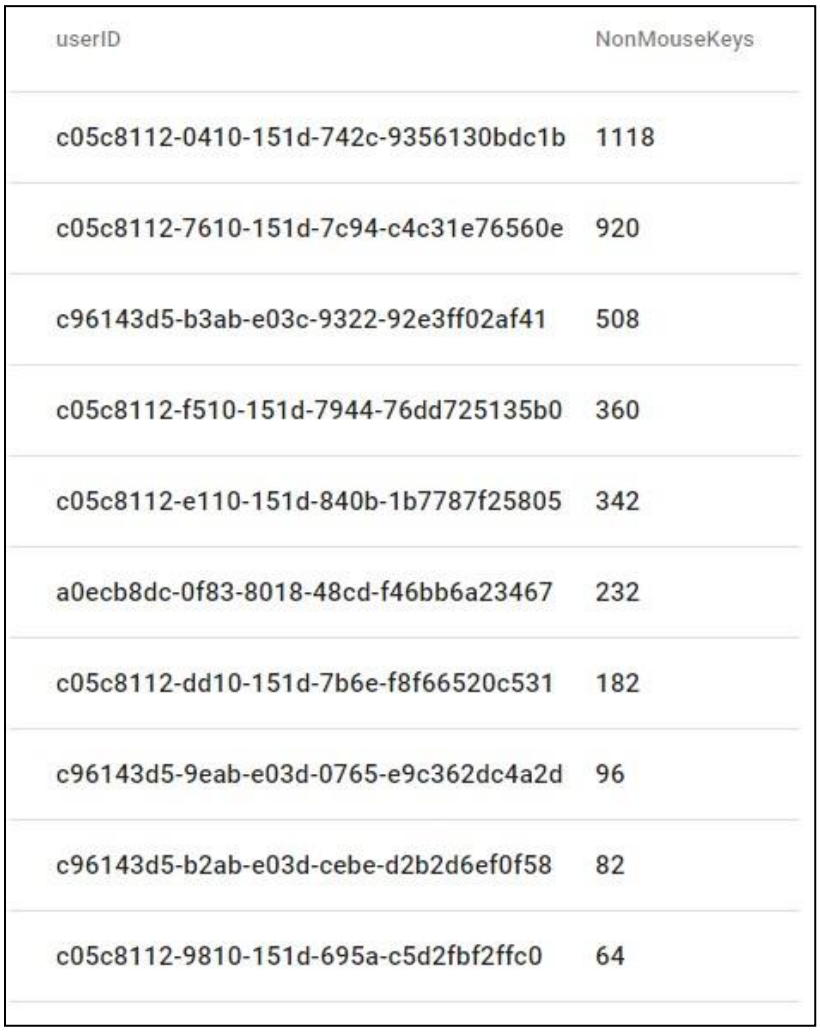
The top participant exhibits a keystroke count almost double or triple the count of other players in the game, implicating a different playstyle. Although expertise cannot be derived exclusively from keystroke count, additional SQL++ queries can be constructed to further explore this hypothesis. Figure 5.5 presents a query counting all keystrokes excluding mouse clicks in the first ten minutes of the game. Notice that the top participant has changed in Figure 5.5, indicating keystroke count may not be sufficient enough to distinguish an expert playstyle from a novice playstyle as other undiscovered factors are involved. This shows that to a researcher, CapturEsports can be useful when exploring these types of patterns since multiple queries can be submitted in succession. This also shows that the preliminary conclusion drawn from Figure 5.4 that expert players have higher keystroke count may not be correct. Thus, further investigation is necessary to discover any correlations between keystrokes and expertise.

```
SELECT s.userID, ARRAY_COUNT((keys)) AS keyCount
FROM LeagueDataSet AS s, s.keystrokeLog AS keys
WHERE s.matchID="3402192685"
GROUP BY s.userID ORDER BY keyCount DESC;
```

userID	keyCount
c05c8112-7610-151d-7c94-c4c31e76560e	13122
c05c8112-0410-151d-742c-9356130bdc1b	8589
c05c8112-e110-151d-840b-1b7787f25805	7793
c96143d5-b3ab-e03c-9322-92e3ff02af41	6109
c05c8112-f510-151d-7944-76dd725135b0	5994
a0ecb8dc-0f83-8018-48cd-f46bb6a23467	5794
c05c8112-dd10-151d-7b6e-f8f66520c531	5443
c96143d5-b2ab-e03d-cebe-d2b2d6ef0f58	5437
c96143d5-9eab-e03d-0765-e9c362dc4a2d	4474
c05c8112-9810-151d-695a-c5d2fbf2ffc0	4130

Figure 5.4: Player keystroke count for an entire match

```
SELECT s.userID, COUNT((keys)) AS NonMouseKeys
FROM LeagueDataSet AS s, s.keystrokeLog AS keys
WHERE s.matchID="3402192685"
AND NOT CONTAINS(keys.keyName, "MOUSE")
AND keys.syncedInGameTime<"10:00:00"
GROUP BY s.userID ORDER BY NonMouseKeys DESC;
```



userID	NonMouseKeys
c05c8112-0410-151d-742c-9356130bdc1b	1118
c05c8112-7610-151d-7c94-c4c31e76560e	920
c96143d5-b3ab-e03c-9322-92e3ff02af41	508
c05c8112-f510-151d-7944-76dd725135b0	360
c05c8112-e110-151d-840b-1b7787f25805	342
a0ecb8dc-0f83-8018-48cd-f46bb6a23467	232
c05c8112-dd10-151d-7b6e-f8f66520c531	182
c96143d5-9eab-e03d-0765-e9c362dc4a2d	96
c96143d5-b2ab-e03d-cebe-d2b2d6ef0f58	82
c05c8112-9810-151d-695a-c5d2fbf2ffc0	64

Figure 5.5: Player keystroke count excluding mouse clicks in the first ten minutes

Scenario 2: A researcher is interested about player behaviors through specific interactions taken during stressful moments of the game or within a certain period of time.

As objectives are often key moments of stress, the time around securing an objective can provide insight into a player's behavior. Observing the same match in the previous scenario, a turret is destroyed at 17:04 (17 minutes 4 seconds into the match). Using this event as a stressful moment, specific interactions surrounding this period of time can be examined through SQL++ queries and manual inspection.

In League, a player is able to use abilities or issue commands, and typically each command requires a single keystroke. Figure 5.6 queries AsterixDB for all abilities used by each player one minute before and after the 17:04 timestamp. Queries empower the researcher to ask questions based on specific conditions and specific time periods. While the query provides an aggregated view of abilities used, flattening the data by converting to .CSV files presents a clearer temporal perspective of the data. After the turret is destroyed, observing the "usedAbility" events in the EventLog in Figure 5.7 reflects ability usage in roughly ten-second intervals, whereas the "MOUSE right" key in the KeystrokeLog in Figure 5.8 reflects repeated keystrokes within milliseconds. Since issuing commands require only a single input, spamming or repeatedly issuing keystrokes in quick succession can be indicative of erratic behavior in response to a stressful event.

```
SELECT s.userID, COUNT(evt) AS abilitiesUsed
FROM LeagueDataSet AS s, s.eventLog AS e,
     e.events AS evt
WHERE s.matchID="3402192685"
AND evt.name="usedAbility"
AND e.syncedInGameTime>="16:04:00"
AND e.syncedInGameTime<="18:04:00"
GROUP BY s.userID ORDER BY abilitiesUsed DESC;
```

userID	abilitiesUsed
c05c8112-7610-151d-7c94-c4c31e76560e	74
c05c8112-0410-151d-742c-9356130bdc1b	48
c96143d5-b3ab-e03c-9322-92e3ff02af41	26
c05c8112-f510-151d-7944-76dd725135b0	14
c05c8112-dd10-151d-7b6e-f8f66520c531	13
a0ecb8dc-0f83-8018-48cd-f46bb6a23467	12
c05c8112-e110-151d-840b-1b7787f25805	12
c96143d5-9eab-e03d-0765-e9c362dc4a2d	11
c96143d5-b2ab-e03d-cebe-d2b2d6ef0f58	9
c05c8112-9810-151d-695a-c5d2fbf2ffc0	8

Figure 5.6: Abilities used one minute before & after a stressful event

syncedInGameTime	events/0/name	events/0/data
17:04:02	announcer	{ "name": "turret_destroy", "data": "enemy"}
17:31:35	ability	2
17:31:38	usedAbility	{ "type": "2"}
17:39:35	ability	3
17:39:47	usedAbility	{ "type": "3"}
17:45:32	assist	{ "count": "5"}
17:46:08	assist	{ "count": "6"}
17:47:20	ability	1
17:47:35	usedAbility	{ "type": "1"}
17:49:22	kill	{ "label": "kill", "count": "9", "totalKills": "9"}
18:00:46	ability	2
18:01:23	usedAbility	{ "type": "2"}

Figure 5.7: CSV EventLog of a player's ability usage after a turret is destroyed

syncedInGameTime	keyValue	keyName	press	mouseX	mouseY
17:04:08	0	MOUSE right	keyUp	1856	862
17:04:01	0	MOUSE right	keyDown	1856	862
17:04:01	0	MOUSE right	keyDown	1856	862
17:04:44	0	MOUSE right	keyUp	1856	862
17:04:48	0	MOUSE right	keyDown	1856	862
17:05:03	0	MOUSE right	keyUp	1856	862
17:05:34	89	Y	keyDown	1856	862
17:05:34	89	Y	keyUp	1856	862
17:05:51	0	MOUSE right	keyDown	1856	862
17:06:01	65	A	keyDown	1856	862
17:06:01	0	MOUSE right	keyUp	1856	862
17:06:34	0	MOUSE right	keyDown	1863	865
17:06:59	0	MOUSE right	keyUp	1863	865
17:07:00	65	A	keyUp	1863	866
17:07:17	0	MOUSE right	keyDown	1863	867
17:07:35	0	MOUSE right	keyUp	1863	867
17:08:12	65	A	keyDown	1868	866
17:08:33	65	A	keyUp	1868	866

Figure 5.8: CSV KeystrokeLog of a player's keystrokes after a turret is destroyed

Scenario 3: A researcher would like to perform an exploratory analysis to potentially infer significant moments in play.

There are moments within a match where the intentionality of players can be observed. However, due to the high variability of the game, the intention of a player or team can be inferred in numerous ways. Moving towards an enemy could indicate an intention to attack or buying healing items could indicate a rationale to support teammates. By examining the average number of abilities used, we investigate the possibility of defining a significant moment based on players' intentions to spend larger amounts of resources.

In this particular match, the Red team was comprised of players ranked Silver and lower and the Blue team was comprised of players ranked Silver and higher. After employing a combination of multiple SQL++ queries based on the previous scenarios, Table 5.1 presents the average number of abilities used per player on a team, partitioned into time periods distinguished by secured key objectives. Considering the second entry of Table 5.1, the Blue team averaged 23.2 ability uses per player and as each ability requires a limited resource (such as energy, mana, or cooldown time), multiple abilities used between destroying the Top Outer Turret and killing the Rift Herald suggest a moment had occurred where the team determined an objective as valuable enough to expend a large number of resources. Although a significant moment cannot be concretely deduced from Table 5.1, the high number of abilities used implies players had identified the existence of a substantial in-game event relative to the other in-game events.

Objective A	Objective A Time	Objective B	Objective B Time	Avg abilities used (Blue)	Avg abilities used (Red)	Objective B Taken
Mid Outer Turret	12:14	Top Outer Turret	13:01	8.4	2.4	Red
Top Outer Turret	13:01	Rift Herald	14:32	23.2	8	Blue
Rift Herald	14:32	Mid Inner Turret	15:39	14.4	5.2	Blue
Mid Inner Turret	15:39	Air Dragon	16:30	16.2	6.2	Blue
Air Dragon	16:30	Top Inner Turret	17:04	11	2	Blue

Table 5.1: Average ability use per player on the same team

Chapter 6

Conclusion and Future Work

In this thesis, I contributed CapturEsports as a novel software tool for capturing and storing multiple data streams to support esports researchers conducting studies involving team dynamics, such as cognition, motivation, player communication, and performance under stress. As current commercial tools mainly focus on individual performance and lack the ability to collect raw unmanipulated data, I explored the feasibility of capturing gameplay observations of entire teams at a high level of detail by developing a prototype with respect to the popular online game League of Legends.

Aimed at becoming an extensible and openly available research tool, CapturEsports currently rests on several key design decisions. Due to the evolving nature of esports, I utilized AsterixDB as a database management system to handle frequently changing data structures. The Apache HTTP server and a custom web server were employed to enable secure remote access to the collected data through a web browser and to support multiple network connections in order to conduct studies in a distributed manner. A user interface was also developed to allow participants and researchers to interact with the system in an organized manner.

Through an alpha test, CapturEsports was shown to succeed in its purpose of enabling researchers the ability to collect player keystrokes, in-game events, and video recordings for ten players simultaneously within the same match. This data was captured at millisecond intervals, synchronized chronologically across all ten players, and uploaded without issues to a centralized server.

In the future, the development of CapturEsports is devoted to facilitating research in the realm of competitive gaming. Although SQL++ enables customizable queries, the learning curve for inexperienced users can be daunting. Thus, building out a comprehensive and domain specific query interface would empower researchers with better analysis functionality with a lower barrier to entry. Furthermore, as researchers explore multiple facets of data when testing various hypotheses, expanding the capability of capturing additional multimodal data, such as physiological or biological responses, would increase the overall amount of empirical evidence. Finally, with a large selection of game titles in esports, extending CapturEsports to support competitive games other than League of Legends opens up opportunities to conduct research in other virtual contexts.

References

[1] Alsubaiee, S., Altowim, Y., Altwaijry, H., Behm, A., Borkar, V., Bu, Y., ... & Gabrielova, E. (2014). AsterixDB: A scalable, open source BDMS. arXiv preprint arXiv:1407.0454.

[2] Auxent, A. (2016, December 8). Riot Games World Championship Numbers Show Slowing in Viewership. Retrieved April 8, 2020, from https://esportsobserver.com/riot-games-world-championship-numbers-show-slowing-viewership/

[3] Bányai, F., Griffiths, M. D., Király, O., & Demetrovics, Z. (2019). The psychology of esports: A systematic literature review. Journal of gambling studies, 35(2), 351-365.

[4] Barretto, L. (2020, April 19). 'You get the same buzz as racing for real' – Lando Norris on the thrill of sim racing: Formula 1®. Formula 1. https://www.formula1.com/en/latest/article.you-get-the-same-buzz-as-racing-for-real-lando-norris-on-the-thrill-of-sim.lFM26RLpKxViWXTYwSz0J.html.

[5] Bosc, G., Kaytoue, M., Raïssi, C., & Boulicaut, J. F. (2013, November). Strategic Patterns Discovery in RTS-games for E-Sport with Sequential Pattern Mining. In MLSA@ PKDD/ECML (pp. 11-20).

[6] Cyclingnews. “UCI Adds Esports World Championships in 2020.” Cyclingnews.com, Cyclingnews, 26 Sept. 2019, www.cyclingnews.com/news/uci-adds-esports-world-championships-in-2020/.

[7] Goslin, A. (2019, September 17). League of Legends has nearly 8 million peak daily concurrent players globally, says Riot. Retrieved from https://www.riftherald.com/2019/9/17/20870382/league-of-legends-player-numbers-active-peak-concurrent

[8] Hamari, J. and Sjöblom, M. (2017), "What is eSports and why do people watch it?", Internet Research, Vol. 27 No. 2, pp. 211-232. https://doi.org/10.1108/IntR-04-2016-0085

[9] Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., & Hsu, M. (2001, April). Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In proceedings of the 17th international conference on data engineering (pp. 215-224). IEEE Washington, DC, USA.

[10] Horse, P. of. (2014, February 10). Minimum and Recommended System Requirements. Minimum and Recommended System Requirements. https://support-leagueoflegends.riotgames.com/hc/en-us/articles/201752654-Minimum-and-Recommended-System-Requirements.

[11] Kasarskis, P., Stehwien, J., Hickox, J., Aretz, A., & Wickens, C. (2001, March). Comparison of expert and novice scan behaviors during VFR flight. In Proceedings of the 11th international symposium on aviation psychology (Vol. 6, pp. 325-335).

[12] Kim, Y. J., Engel, D., Woolley, A. W., Lin, J. Y. T., McArthur, N., & Malone, T. W. (2017, February). What makes a strong team? Using collective intelligence to predict team performance in League of Legends. In Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing (pp. 2316-2329). ACM.

[13] Lee, C. S., & Ramler, I. (2019). A Data Science Approach to Exploring Hero Roles in Multiplayer Online Battle Arena Games. Data Analytics Applications in Gaming and Entertainment, 49.

[14] LoLwiz. (2019). Dominate Ranked Play With The #1 League Companion App. Retrieved from https://lolwiz.gg/

[15] Mobalytics. (n.d.). Summoner Stats and Match History for League of Legends. Retrieved from https://app.mobalytics.gg/

[16] Myers, J. (n.d.). MD5 Algorithm. Retrieved April 15, 2020, from http://www.myersdaily.org/joseph/javascript/md5-text.html

[17] Nascimento Junior, F. F. D., Melo, A. S. D. C., da Costa, I. B., & Marinho, L. B. (2017, October). Profiling successful team behaviors in League of Legends. In Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web (pp. 261-268).

[18] Overwolf. “Tech for Developers Who Love Gaming.” Overwolf, www.overwolf.com/.

[19] Phillips, L. (2020, April 1). The History Of Esports. Hotspawn.com.

[20] Pobiedina, N., Neidhardt, J., Moreno, M. D. C. C., Grad-Gyenge, L., & Werthner, H. (2013, July). On successful team formation: Statistical analysis of a multiplayer online game. In 2013 IEEE 15th Conference on Business Informatics (pp. 55-62). IEEE.

[21] Sapienza, A., Bessi, A., & Ferrara, E. (2018). Non-negative tensor factorization for human behavioral pattern mining in online games. Information, 9(3), 66.

[22] Staff, L. (2019, December). 2019 World Championship Hits Record Viewership. Retrieved April 8, 2020, from https://nexus.leagueoflegends.com/en-us/2019/12/2019-world-championship-hits-record-viewership/

[23] Woolley, A. W., Chabris, C. F., Pentland, A., Hashmi, N., & Malone, T. W. (2010). Evidence for a collective intelligence factor in the performance of human groups. science, 330(6004), 686-688.