*Department of Computer Science & Engineering*

*University Of Nevada-Reno*

Project Glimmer

Team 15

**Mike DesRoches**

**Christian Garcia**

**Jake Lahr**

**Alexandra Pasinski**

Instructors

**Dr. David Feil-Seifer**

**Devrin Lee**

External Advisor

**Dr. Engin Arslan**

November 1, 2019

**Table of Contents**

Introduction …………………………………………………………………………………….... 2

Summary of stakeholders’ interviews ………………………………………………………….... 2

Dr. Alireza Tavakkoli ……………………………………………………………………. 2

Alexandra Pasinski ………………………………………………………………………. 4

Nathan Griffin …………………………………………………………………………… 5

High level business requirements ……………………………………………………………….. 7

Technical requirements specification ……………………………………………………………. 8

Functional requirements …………………………………………………………………. 8

Non-functional requirements ……………………………………………………………. 9

Use case modeling……………………………………………………………………………….10

Requirement traceability matrix ……………………………………………………………….. 14

Initial snapshots………………………………………………………………………………….14

Glossary………………………………………………………………………………………….18

References………………………………………………………………………………………..20

Contributions of team members………………………………………………………………….21

**3. Introduction**

Streaming services, such as Twitch, have become an important part of the video game industry. With millions of people watching video game streams every day, game developers are realizing the potential for marketing their games on these streaming platforms. Project Glimmer’s objective is to create an application programming interface (API) that game developers can utilize to create audience interaction between players live streaming their games and viewers watching those lives streams. Streamers would also be able to use this API to modify their favorite games to allow audience interaction in games that do not already have this interactive feature. Live stream viewers would be able to interact with game streamers in a real-time fashion and have an impact on the current game state.

Our objective is that, by integrating our API into a game, the life-span of a video game can be increased by the increase in possible interactions from external sources such as live stream viewers. Our main software design goal is to create an API that game developers can easily integrate with their existing software, whether they are using their own game engines or utilizing an existing engine, such as Unreal or Unity. Our product will have a unique significance in the video game development industry as, while live stream chat integration in video games is not a completely new concept, most solutions need to be developed in house and few resources exist to allow independent developers the ability to easily achieve this type of interaction in their games.

**4. Summary of Stakeholders’ Interviews**

**Dr. Alireza Tavakkoli**

We interviewed Dr. Tavakkoli first as we had planned for him to be our advisor. He has extensive experience in the game development field which would be the main market for our product. As a potential user of our product, we believed he would offer us valuable insight into features we could focus on and, as an experienced software engineer, he could give us guidance on managing our project.

Q. Could you summarize your experience in game development?

A. Dr. Tavakkoli has been teaching game development for the past 10 years and develops small pieces of games as a hobby.

Q. How might you use our product in your projects?

A. He would find great use in our product if he decided to create a game which required interaction from remote viewers of the game through web live streams.

Q. What part of our product excites you the most?

A. He is interested in the challenges our team will have to face in making sure that only valid interactions are allowed from the viewers of the live streams. He is also interested in what new types of interactions our product will allow as games made with our product would not be multiplayer in the traditional sense but would have several remote users interacting at once.

Q. Which features should we focus most on?

A. Dr. Tavakkoli was not sure what features would be best to focus on as he saw our product as very new and believed that the biggest challenge for us would be discovering what features worked and what did not.

Q. Which features should we remove, if any?

A. He recommended that we start small and focus only on features which we believed could be accomplished quickly. He encouraged us to avoid spending time on designing more complex interactions until we could get our networking systems functioning properly with simple live stream chat commands.

Q. What do you think about the market potential for this product?

A. He believes that live streamers would actually be a potential group of users for our product. This was due to the potential use of our product in game modding with live streamers potentially using it to create custom interactions in games that they enjoy playing for their audiences. He also mentioned that we could potentially build plugin versions of our product for popular game engines, such as Unreal or Unity, which we could sell to game developers as an easy way to integrate live stream interaction into their games.

Q. What do you expect us to accomplish in the time we have?

A. As we are going into an unexplored area in terms of game development, Dr. Tavakkoli was unable to be very specific on what he believed we could accomplish in the time we have. He did believe that the simpler features mentioned earlier should be possible in the time allotted. He recommended that we split our time into three phases: 10% in the design phase, 60% in the prototyping phase, and 30% in the testing phase.

Q. How would you start this project if it was yours?

A. He recommends spending most of our free time, that is not dedicated to other classes, working as a group to design our systems, decide on what features we want to include, and figure out who we are marketing to.

Q. Do you have any other contacts that would be of some use? Marketing, implementation, etc.

A. He recommended speaking with experts in the networking field for the main phase of our product development as most of our work will be spent gathering commands from an external web service, the live stream chat log. If we do eventually start to market our product to game developers, he would be happy to introduce us to his contacts working on the Unreal Engine.

Q. We have an idea and an end goal but are there any strategies you can suggest that will help us focus our goals into more manageable pieces?

A. Dr. Tavakkoli advised that we should focus on splitting our intended features into three categories: easily completed features, more complex but useful features for users, and experimental features that we are not certain are possible but would provide a great benefit to users of our product.

Q. What external resources, such as books, articles, websites, etc. do you think would help us develop our project?

A. He told us about a website called Monday.com which provides excellent group management software that lets teams manage deadlines and send automatic update emails to other team members.

Q. Are there any products you know of currently on the market that provide similar features to ours?

A. He does not know of any other products that are similar to ours.

Q. What phase should we consider licensing issues?

A. He believes that we should start considering licensing issues as soon as possible and pointed us to the possibility of filling for a provisional patent to safeguard our IP.

**Alexandra Pasinski**

We interviewed Alexandra as she is currently trying to complete her Digital Interactive Games minor at UNR and is interested in implementing various forms of player interaction for developers to use in game engines.

Q. Could you summarize your experience in game development?

A. She has been learning how to create game engines for several years now.

Q. How might you use our product in your projects?

A. She would want to integrate our product as a core component of a game engine she is designing. She wants to provide it as one of many options available to developers for enabling player interaction.

Q. What part of our product excites you the most?

A. She sees the ability to easily integrate our product into almost any application to be the most exciting part of it.

Q. Which features should we focus most on?

A. She believes that focusing on allowing developers the freedom to specify their own syntax that their game would accept for commands would be the most important feature to focus on.

Q. Which features should we remove, if any?

A. She did not believe we should remove any features from our product currently. Instead, she recommended focusing most on those features that we could complete for the prototype.

Q. What do you think about the market potential for this product?

A. She says that streaming services such as Twitch.tv are really popular right now but that there are no apparent solutions to the problem we are attempting to solve. Therefore, while we may not have a completely polished product by the beginning of May, she sees our product as potentially having great value among game developers.

Q. What do you expect us to accomplish in the time we have?

A. She believes that we should be able to design a basic version of our product that allows for single word commands. She also believes that we should be able to integrate our product into a simple command line application at least.

Q. Do you have any other contacts that would be of some use? Marketing, implementation, etc.

A. She knows some students in the Journalism and Business departments that may be able to help guide us on marketing our product to a larger audience.

Q. We have an idea and an end goal but are there any strategies you can suggest that will help us focus our goals into more manageable pieces?

A. She recommends taking the time as a team to sketch out many diagrams of our internal systems before we ever touch a keyboard. This way, our entire team can be on the same page during development.

Q. What external resources, such as books, articles, websites, etc. do you think would help us develop our project?

A. She suggested looking into any web apis that Twitch.tv exposes to developers as those would be our main entry point for many of our product’s features.

Q. Are there any products you know of currently on the market that provide similar features to ours?

A. She only knows of several in-house implementations of live stream integration in games such as Twitch Plays Pokemon and the ability for Cluster Truck’s developers to connect to any live streamer’s game session and change the properties of the game on the fly.

Q. What phase should we consider licensing issues?

A. She believed that a product like ours would be better suited being completely open sourced for the developer community. However, she did recommend using a less-restrictive license such as the MIT license over the GPL to allow for commercial developers to use our product.

**Nathan Griffin**

We interviewed Nathan as he has extensive experience developing video games and we believed he would be a great resource for us as we try to find out what features our users would want most.

Q. Could you summarize your experience in game development?

A. He has been a hobbyist game developer for over a decade and specializes in virtual reality game design.

Q. How might you use our product in your projects?

A. If he wanted to create a game that integrated with different live streaming services, he says our api would be a useful tool.

Q. What part of our product excites you the most?

A. The potential for an api to allow ease of integration across multiple platforms is what excites him the most.

Q. Which features should we focus most on?

A. Cross-game-system support is the feature he would like us to focus most on so that our api could be used for any deployment system a game developer wants.

Q. Which features should we remove, if any?

A. He did not see any of our features as worthy of removal.

Q. What do you think about the market potential for this product?

A. He believes our market would be mostly independent game developers and hobbyists.

Q. What do you expect us to accomplish in the time we have?

A. He fully expects Twtich.tv integration to be achieved but is unsure if we would have enough time to do multi-platform support.

Q. Do you have any other contacts that would be of some use? Marketing, implementation, etc.

A. He recommends posting our project to development boards to gain attention in those communities and let other users market our product for us.

Q. We have an idea and an end goal but are there any strategies you can suggest that will help us focus our goals into more manageable pieces?

A. He recommends having our most senior developer create a structured environment for the team to work within and allow for the management of all phases of the project.

Q. What external resources, such as books, articles, websites, etc. do you think would help us develop our project?

A. He told us to look at existing Twitch.tv api wrappers and then consider how we could improve on them in terms of reliability and ease of use.

Q. Are there any products you know of currently on the market that provide similar features to ours?

A. He knows of a wrapper called TwitchLib which is popular but there are almost no apis that allow for integration with other streaming services like YouTube or Mixer.

Q. How could we make our product more unique?

A. He recommended trying to integrate with multiple streaming services at once so that streamers could stream to several sites and their viewers could still interact no matter what site they are on.

**5. High Level Business Requirements**

|  |  |  |
| --- | --- | --- |
| ID | Description | Rationale |
| HLBR-001 | Create an API that can be integrated into existing games. | Our team wants to make the process of using our API as simple as possible. |
| HLBR-002 | Create an API that can allow live stream viewers to have an impact on the game being viewed | We want to explore a new type of player interaction where many external users have an impact on the game being played by one local player. |
| HLBR-003 | Create a simple game to test our API on. | We want to be able to show potential users the useful features of our API and can only do that by demoing it with a game. |

Table 1: High Level Business Requirements

**6. Technical requirements specification**

|  |  |  |
| --- | --- | --- |
| **ID** | **Priority** | **Description** |
| **Functional Requirements** | | |
| FR1 | 1 | Project Glimmer shall connect to Twitch.tv’s live stream chat |
| FR2 | 1 | Project Glimmer shall parse live stream chats via Twitch chat bots for specific chat commands |
| FR3 | 1 | Project Glimmer shall process and send the data as packets to the users of Project Glimmer. |
| FR4 | 1 | Project Glimmer shall use TCP to connect to web services. |
| FR5 | 1 | Project Glimmer shall utilize chatbots to control and manage stream chat commands |
| FR6 | 1 | Project Glimmer shall allow its users to use these commands to affect their application any way they choose. |
| FR7 | 1 | Project Glimmer shall allow its users to send messages back to the live stream chat. |
| FR8 | 1 | Project Glimmer shall keep track of what users are sending commands |
| FR9 | 1 | Project Glimmer shall allow the user to pause/unpause the receiving of commands |
| FR10 | 2 | Project Glimmer shall be able to keep track of repeated chat commands to provide feedback to the user. |
| FR11 | 2 | Project Glimmer shall allow the user to specify how many parameters a command has and what they are. |
| FR12 | 2 | Project Glimmer shall provide controls to the user to limit the number of commands being processed and the number of commands that one viewer can send. |
| FR13 | 3 | Project Glimmer shall be integrated into a multiplayer game with each player acting as a separate streamer with their own chat command stream. |
| FR14 | 3 | Project Glimmer shall allow users to specify commands that require stream viewers to pay to use (for example, only viewers who support the streamer are allowed to send messages to them). |
| **Non-Functional Requirements** | | |
| NFR1 | 1 | Project Glimmer shall Run on Linux |
| NFR2 | 1 | Project Glimmer shall be implemented as an Open Source Project. |
| NFR3 | 1 | Project Glimmer shall be written in C/C++ |
| NFR4 | 1 | Project Glimmer shall provide complete documentation of its functionalities |
| NFR5 | 1 | Project Glimmer shall provide complete documentation of its functionalities |
| NFR6 | 2 | Project Glimmer can be used on other live streaming platforms than just twitch |
| NFR7 | 2 | Project Glimmer shall be integrated into a simple game developed by the team. |
| NFR8 | 3 | Project Glimmer can be used by non software engineers/developers |
| NFR9 | 3 | Project Glimmer shall support Linux, Windows, Android deployment platforms |
| NFR10 | 3 | Project Glimmer shall be implemented as a plug-in for common game engines. |
| NFR11 | 3 | Project Glimmer shall be able to run on several different game platforms (XBox One, Playstation 4, etc.) |

Table 2: Functional and Non-Functional Requirements

**7. Use Case Modeling**

Use Case Diagram

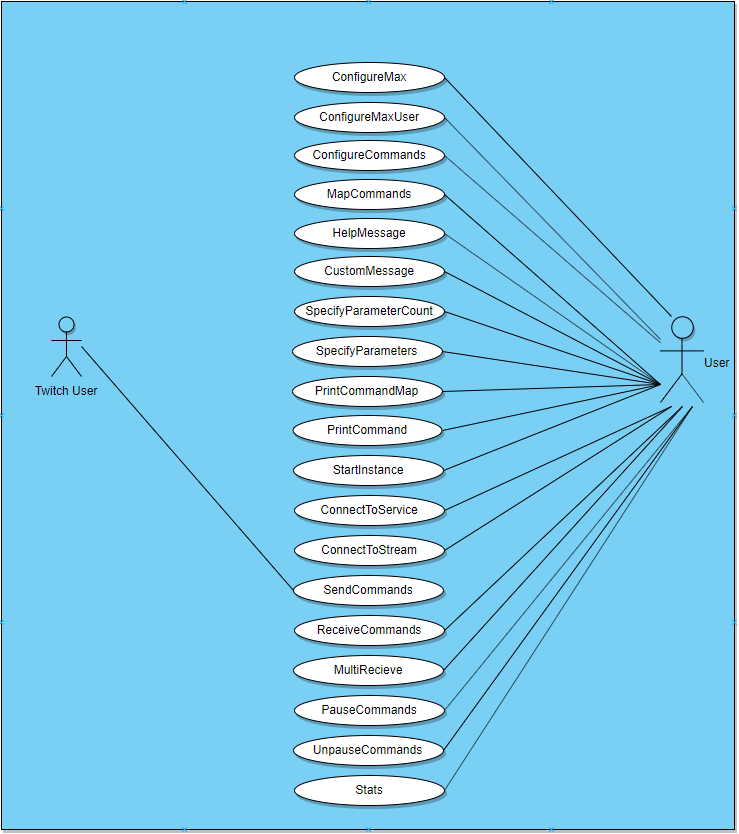
****

Figure 1: Use Case Diagram

Use Case Descriptions

|  |  |  |
| --- | --- | --- |
| **ID** | **UseCase** | **Description** |
| UC1 | ConfigureMax | Configure the maximum number of commands allowed per second to prevent viewers from slowing down or disrupting game performance. |
| UC2 | ConfigureMaxUser | Configure and track the number of commands sent per user to prevent single users from spamming commands and allowing other users to send commands. |
| UC3 | ConfigureCommands | Configure what commands are allowed to be accepted by the live stream chat bot. |
| UC4 | MapCommands | Configure what developer defined functions are called for each command defined. |
| UC5 | HelpMessage | Configure what help message is displayed for each command allowing for live stream viewers to see what command does what. |
| UC6 | CustomMessage | Define specific commands to allow custom messages being sent to the live stream viewing window. |
| UC7 | SpecifyParameterCount | Specify if a command has zero or more parameters to allow for more complex commands to be sent. |
| UC8 | SpecifyParameters | Specify and define what parameters a command can accept for more complex commands that have more than one parameter. |
| UC9 | PrintCommandMap | Print out a mapping of each command to the function it calls and displays the output for visual representation of what each command is mapped to. |
| UC10 | PrintCommand | Print out a detailed view of each command being received from the stream chat. |
| UC11 | StartInstance | Start an instance of the project api enabling functionality of the api to start reading and processing commands. |
| UC12 | ConnectToService | API connects to the streaming service allowing for commands to be taken from the live stream chat. |
| UC13 | ConnectToStream | API connects to a specific live stream allowing a specific streamer to use api functionality within the game. |
| UC14 | SendCommands | API allows viewers to have their commands read and sent to the game developer. |
| UC15 | ReceiveCommands | API can be used to receive commands from multiple streaming services |
| UC16 | MultiReceive | API can be used to receive commands from multiple live streams (one for each player in the game). |
| UC17 | PauseCommands | The stream of commands from the live streams can be paused to no longer affect the users game. |
| UC18 | UnpauseCommands | The stream of commands from the live streams can be unpaused. |
| UC19 | Stats | The API will return statistics about each command (how many times it is being called, how many users are calling it, etc. |

Table 3: Detailed Use Cases

Detailed Use Case Templates

|  |
| --- |
| **Use Case:** ConfigureMax |
| **ID:** UC1 |
| **Actor(s):** User |
| **Precondition(s):**   1. User has defined what commands can be accepted 2. Game and API and been Initialized 3. Stream may have too many viewers and can potentially slow the game |
| **Flow of Events:**   1. User sets an appropriate amount of commands to be processed per second |
| **Postcondition(s):**   1. The game receives a set amount of commands per second that does not affect the game in a destructive manner |

Table 4: Template or Use Case *ConfigureMax*

|  |
| --- |
| **Use Case:** HelpMessage |
| **ID:** UC5 |
| **Actors(s):** User |
| **Precondition(s):**   1. Establish and define commands to be used in game. 2. Live Streamer has enabled game commands. |
| **Flow of Events:**   1. User writes descriptions for each command to be displayed when function is called. 2. Twitch user wants to know what each command does 3. Twitch user types command |
| **Postcondition(s):**   1. Chat bot displays what each command does |

Table 5: Template for Use Case *HelpMessage*

|  |
| --- |
| **Use Case:** SendCommands |
| **Actor(s):** Twitch User |
| **ID:** UC14 |
| **Precondition(s):**   1. Live Streamer has enabled in game commands 2. Parameters of commands have been established |
| **Flow of Events:**   1. List of commands can be displayed 2. Command input is monitored 3. Chat box from live stream service pushes command 4. Command is received by Project Glimmer and mapped accordingly to appropriate function |
| **Postcondition(s):**   1. Command is read, processed and affects the state of the game |

Table 6: Template for Use Case *SendCommands*

**8. Requirement traceability matrix**

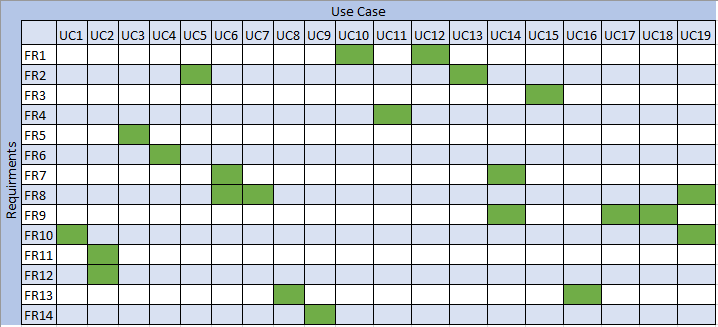


Figure 2: Traceability Matrix

**9. Initial Snapshots**

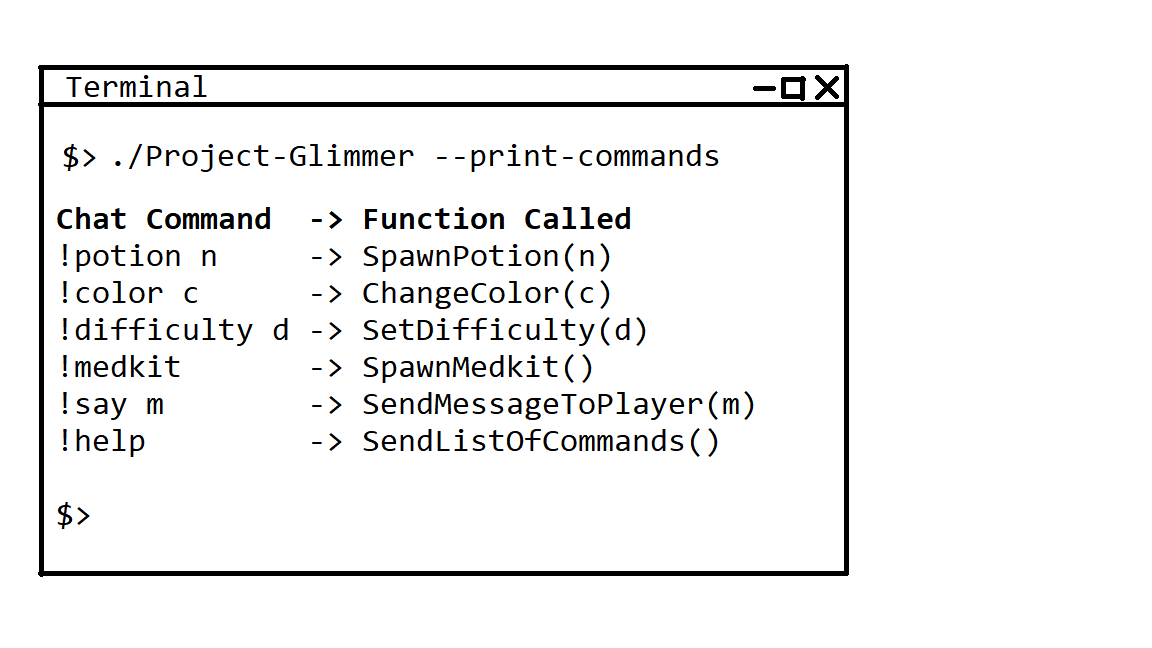


Figure 3. The game developer uses the API to print out the mapping of commands to game functions.

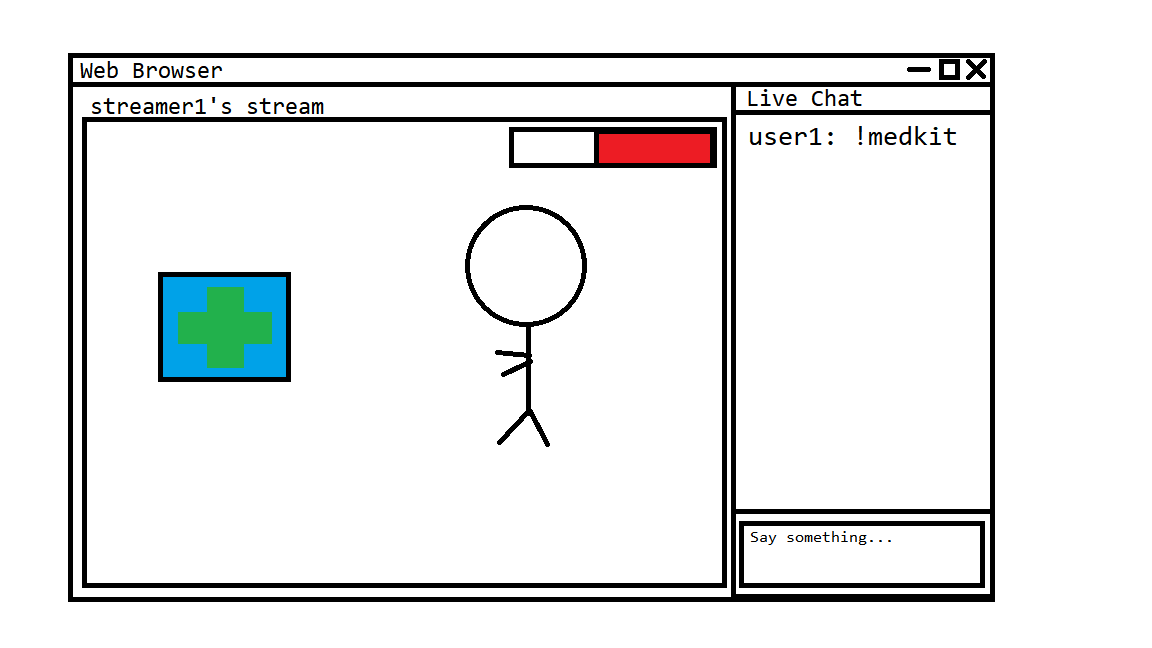


Figure 4. Viewer, user1, enters zero argument command, !medkit, on a live stream service and a medkit is created in the game.

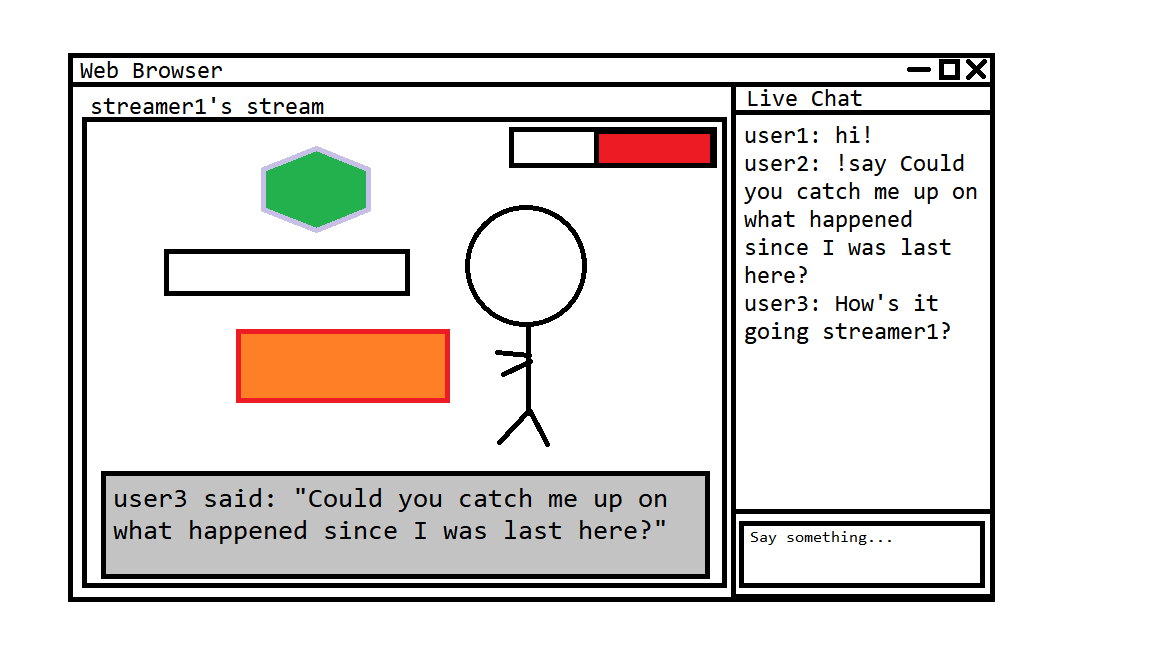


Figure 5. Viewer, user2, enters single argument command, !say m, on a live stream service and the message ‘m’ is displayed in the game.

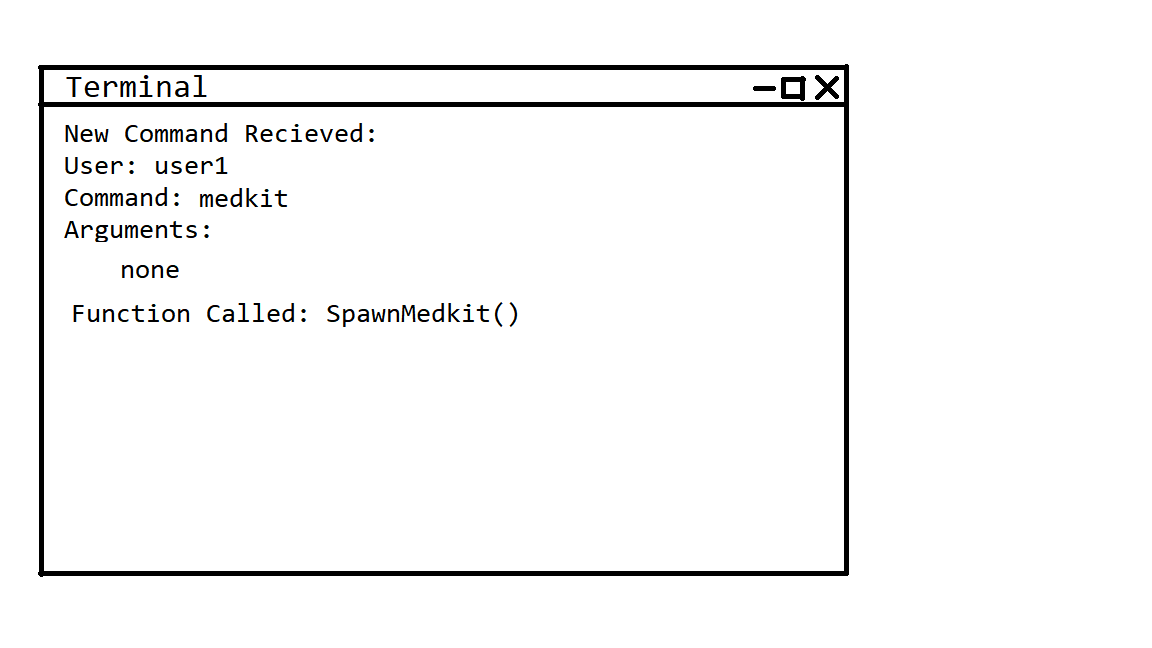


Figure 6. The API prints out the command information, for a zero argument command that was received, to the terminal.

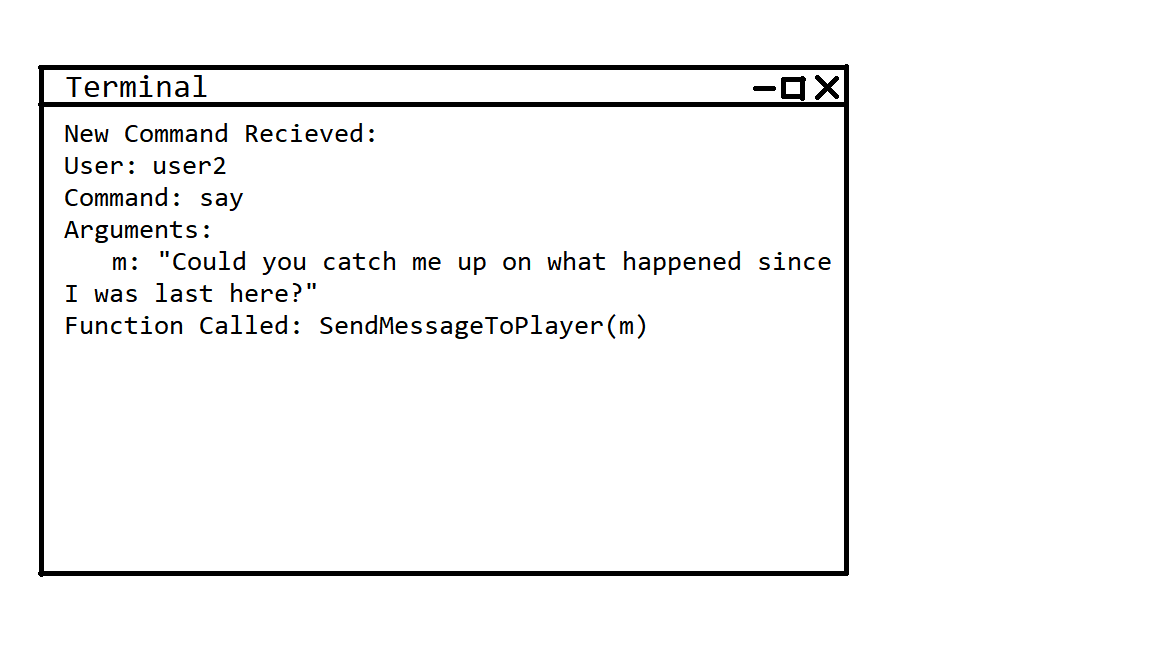


Figure 7. The API prints out the command information, for a single argument command that was received, to the terminal.

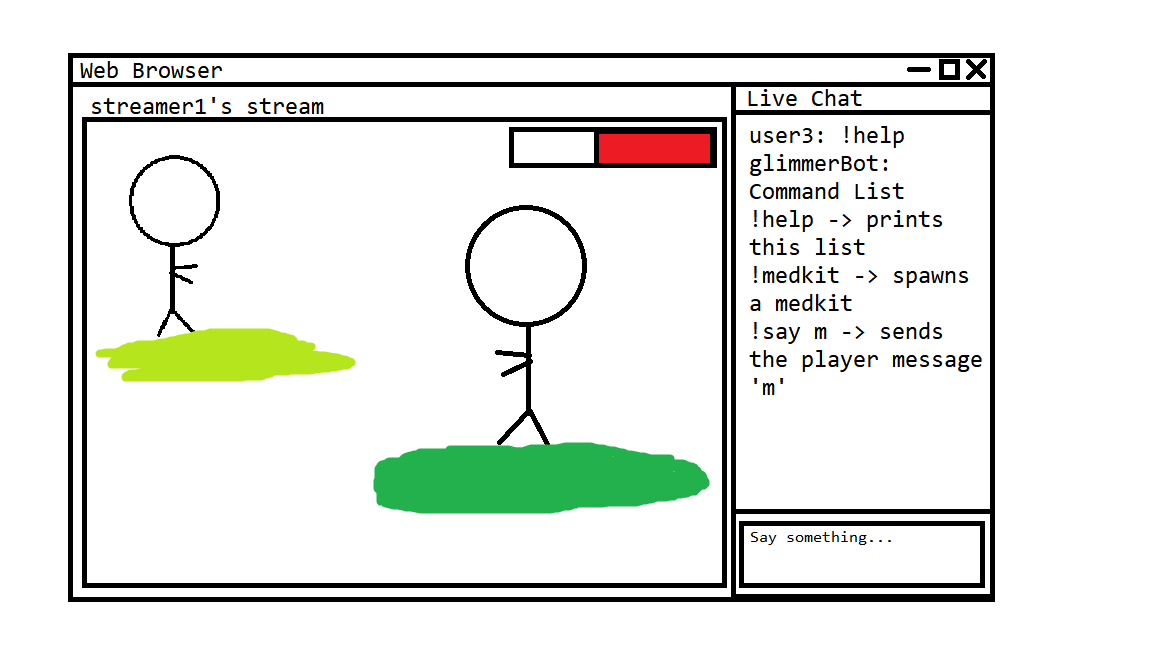


Figure 8. The api responds to the !help command by printing a list of available commands in the live stream chat log.

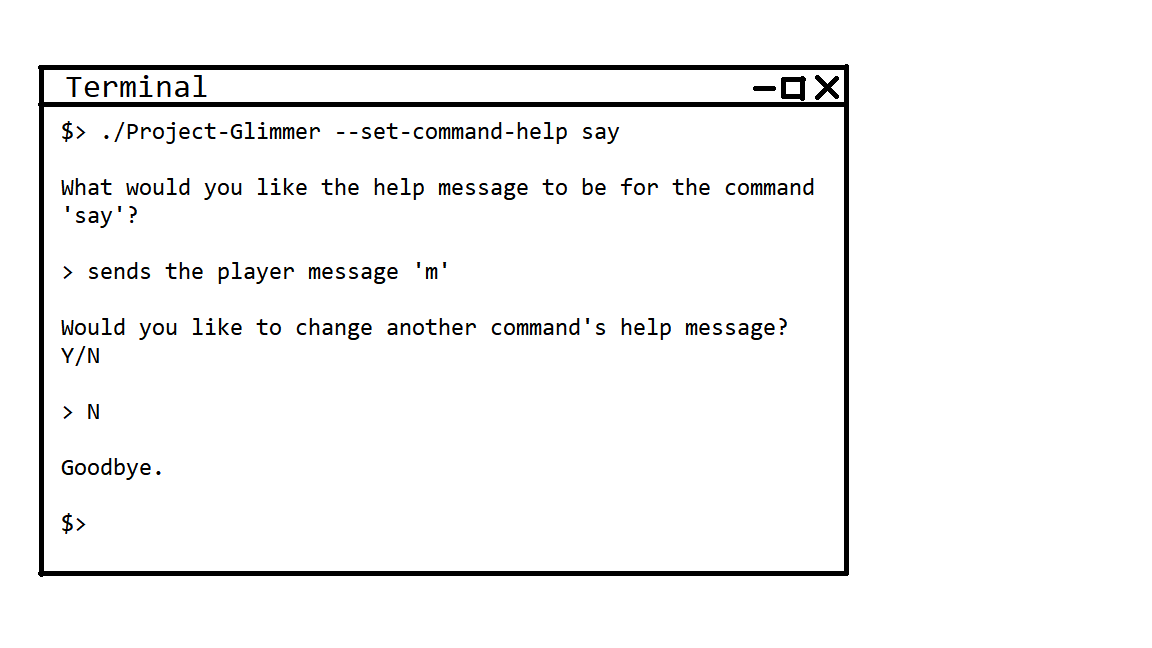


Figure 9. A developer uses the api to set the message that is displayed, for a specific command, when a viewer sends the !help command.

**10. Glossary**

**Application programming interface (API):** A system of tools and resources enabling developers to create software applications.

**Chatbots:** Special programs that can moderate a Twitch.tv channel's chat log, greet new viewers, post scheduled messages, and add extra functionality to a livestream.

**Commands:** Chat comments that can be read and interpreted by the API.

**Features:** Characteristics or abilities, things that can be done with the use of the API.

**Game engine:** The basic software, or foundation, of a video game. Contains features and software that can be shared across multiple different games.

**Game state:** Description of the parameters of a game at a given point in time.

**Interaction:** The act of a viewer changing the game state while the streamer is playing the game.

**Live streams:** A live video transmission of a game over the internet.

**Modding:** Alteration by players or fans of a video game that changes one or more aspects of the game, such as how it looks or behaves.

**Multiplayer:** Denoting or relating to a video game designed for or involving several players**.**

**Networking system:** Two or more computers that are linked in order to share resources, exchange files, or allow electronic communications.

**One-to-Many Player Game:** A single player streams live footage of their game to an online video streaming service. The many viewers of that stream then interact by sending commands through the stream chat that modify the game state in some way.

**Packets:** A formatted unit of data carried by a packet-switched network. A packet consists of control information and user data

**Plug-in:** A software component that adds a specific feature to an existing computer program.

**Prototype:** A first, typical or preliminary model of the software from which other forms are developed or copied.

**Streamer:** Someone who streams video games live for an audience.

**Streaming services:** A web site or application that allows for the live transmission video.

**TCP:** The Transmission Control Protocol which is a protocol for transmitting data over the internet.

**TCP sockets:** An internal endpoint for sending or receiving data within a node on a computer network using TCP.

**Twitch.tv:** An online live streaming video platform with a focus on gaming.

**Video game:** A game played by electronically manipulating images produced by a computer program on a television screen or other display screen.

**Viewers:** The people who watch live streamers play games on a live streaming service.

**11. References**

Journal References

Haque, A. Twitch Plays Pokemon, Machine Learns Twitch: Unsupervised Context-Aware Anomaly Detection for Identifying Trolls in Streaming Data.*University of Texas at Austin.* 2014. Retrieved From: <https://arxiv.org/abs/1902.06208>

Annotation: Scholarly journal from Cornell University. This paper will be useful because it discusses the problem of detecting and identifying internet “trolls” in the stream chat. A “troll” is someone who deliberately angers or upsets someone else on the internet. If these trolls can be properly located and dealt with using our API, developers will be more likely to use it.

Hilvert-Bruce, Z. Neill, J. Sjoblom, M. Hamari, J. Social motivations of live-streaming viewer engagement on Twitch. *Computers in Human Behavior*. 2018. Retrieved From:

<https://www-sciencedirect-com.unr.idm.oclc.org/science/article/pii/S0747563218300712>

Annotation: Social motivations of live-streaming viewer engagement on Twitch. This paper studies the motivations behind viewer engagement in streaming services. This reference might be useful because it will give us an idea of how viewers would like to interact with their favorite streamers and why.

Johnson, M. Woodcock, A. The impacts of live streaming and Twitch.tv on the video game industry. *Media, Culture, & Society*. 2018. Retrieved From: <https://journals-sagepub-com.unr.idm.oclc.org/doi/full/10.1177/0163443718818363?utm_source=summon&utm_medium=discovery-provider&>

Annotation: The impacts of live streaming and Twitch.tv on the video game industry. This article will be useful because it covers the importance of live streaming sites, such as Twitch, to the gaming industry. It covers the impact of streaming newly released games, the added lifespan from streaming a game, and the live streaming of the creation of games.

Website Resources

Business Requirements Example

Retrieved from: <https://www.businessphrases.net/business-requirements-example/>

Annotation: This website provides a business requirements example and definition. It starts with business rules examples and explanation. The page also contains examples of stakeholder requirements, solution requirements, transition requirements, assumptions, constraints, and use cases. This reference will help with the business aspect of the project.

Spark Streaming Programming Guide

Retrieved From: <https://spark.apache.org/docs/latest/streaming-programming-guide.html>

Annotation: This is the Spark streaming programming guide. We might use Spark for secure processing of the stream chat commands. Using this API, data can be ingested from many sources like Kafka, Flume, Kinesis, or TCP sockets, and can be processed using complex algorithms expressed with high-level functions

Twitch Developer Documentation

Retrieved From: <https://dev.twitch.tv/docs/>

Annotation: This is the Twitch developer documentation page. This reference will help us with connecting games to the streamers’ Twitch channels, integrating the chatbot, and reading and interpreting the chat commands. The site shows everything that is needed to integrate Twitch services into our project.

**12. Contribution of Team Members**

Alexandra Pasinski

Hours Worked: 10

Parts Worked On: Wrote interview questions; performed interviews; summarized interview answers; contributed to high level business requirements; contributed to functional requirements; contributed to non-functional requirements; contributed to use cases; created snapshots; contributed to glossary; contributed to references.

Jake Lahr

Hours Worked: 9

Parts Worked On: Technical Requirements, Use Case Diagram, Use Case Descriptions, Use Case Templates, Join in interview with Professor Tavakkoli, Organized Advisor meetings.

Mike DesRoches

Hours Worked: 9

Parts Worked On: Partook in interview with Professor Tavvakoli. High Level Business Requirements, Use case Modeling, Requirement traceability Matrix, and formatted FR and NFR tables, contributed to references.

Christian Garcia

Hours Worked: 9

Parts Worked On: Cover page, table of contents, introduction, glossary, list of references,