Chat plays

“Twitch Plays” as a programmable multiplatform user-friendly consumer product and network marketplace.

Executive Summary

“Twitch Plays Pokémon” was a cultural phenomenon in 2014 - it was a social experiment that allowed anyone to play the game “Pokémon Red” in a Twitch livestream by sending simple commands through the livestream chat. The project became an overnight success, and the biggest thing happening on the Internet at the time, breaking multiple world records in the process. This project inspired people knowledgeable in software development to make their own “Twitch Plays” software tools for various other games and activities, thus creating a new category of livestreaming: the “Twitch Plays” category. Nowadays, when popular live streamers make a “Twitch Plays” for their live stream - in other words, when they allow their livestream audience to play a videogame by sending commands through the livestream chat - it is usually met with an overwhelmingly positive response, and seen as a great source of entertainment.

Despite the community demand, making a “Twitch Plays” right now in the year 2021 is too hard. There is a high technical barrier to entry, it isn’t easy to use, and the tools available on the Internet either don’t work or aren’t very advanced - and when they do work, they only work for one streaming platform: Twitch. These shouldn’t be issues for a concept that has been around for so long, which has seen so many use cases and major successes. If only there were a product that solved all these issues at an affordable price - this is the idea that led to the creation of ChatPlays.

The ChatPlays company aims to solve these issues by making a computer program with a user-friendly graphic interface that gives anyone the ability to recreate the magic and anarchic fun of the original “Twitch Plays” on their own livestream with any game they would like, on any major streaming platform they would like, not just Twitch; while also allowing users to buy and sell digital assets, which are made in the ChatPlays program, to each other on a marketplace, which is operated by our company, in exchange for money or for cryptocurrency. The ChatPlays program also aims to gradually implement artificial intelligence and other quality-of-life features in future iterations in order to improve the user experience, and in order to make the program easier and more efficient to use for both streamers and audiences alike.

Background

What is a “Twitch Plays”?

“Twitch Plays” livestreams are a type of stream on the twitch streaming platform where the audience can directly interact with what is happening on the live video stream by sending commands through the live chat. Typically, a game is played where the audience has to reach some sort of goal.

Twitch Plays Pokémon

The first “Twitch Plays” was a social experiment called “Twitch Plays Pokémon”. In 2014, an anonymous developer made a script that allowed anyone to come into his live stream and collectively play Pokémon Red through commands in the live chat. This quickly became a cultural phenomenon and the Internet flooded into the live stream to try and beat the game. After 16 days, 9 hours, and 55 minutes the audience managed to complete the game. While beating the game, the “Twitch Plays Pokémon” channel became the biggest channel around at the time, with accomplishments such as having 100,000 people watching at the same time, obtaining a Guinness World Record of "the most participants on a single-player online videogame" with 1,165,140, along with various other incredible achievements. This channel became a pop culture hit and would go on to beat every main series Pokémon game, all with the audience steering the wheel the whole time.

The new Twitch Plays category

After seeing what was possible, many other “Twitch Plays” started popping up. Streams where the audience could play other video games like the Legend of Zelda, play on a claw machine, etc. were all over the place. A new genre of live streams had been created, the “Twitch Plays” category.

Problems

Barrier to Entry, Not User-Friendly, Limitation to one Streaming Platform,

The biggest problem with making a “Twitch Plays” right now is the barrier to entry. If a regular person today in 2021 wanted to make their own “Twitch Plays”, in other words, if they wanted to allow their audience to play a video game through the live chat on a streaming platform, they would have to scour the Internet to find the right tools, learn how to use these complicated tools, watch YouTube tutorials, find their OAuth key, etc. This is a time-consuming and inconvenient process which isn’t user-friendly. Not to mention that most of the tools that perform this function on the Internet either don’t work, are frustrating to use, haven’t been updated in years, don’t have a wide variety of functions, and are made for only one platform, Twitch.tv. The only alternative for someone to avoid all this would be to hire a personal developer to make a customized “Twitch Plays”, which no one with a normal income stream could afford.

Stale Format, No interaction with Streamer Personalities

Another problem is that despite the success of the “Twitch Plays Pokémon” channel and the initial boom of other “Twitch Plays” that were popping up, eventually it got stale. If you go to the original Twitch Plays Pokémon today, you will see that they have approximately or less than 200 viewers. It is a husk of what it once was. Alternative Twitch Plays channels struggle to get more

than 30 viewers. Why? Because Twitch streaming is about personality. Consistent Twitch viewers see their “streamer” as their virtual friend, and form communities around one or several Twitch personalities. There is no personal interaction in a traditional “Twitch Plays,” it’s just the game on stream and a random collective playing in the live chat.

Solutions

Eliminating Barrier to entry, User-Friendly Graphic Interfaces, Multi-platform Compatibility

A program with a user-friendly graphical interface that lets the everyday consumer or live streamer make their own “Twitch Plays” on any major streaming platform of their choice, not just Twitch, for a game of their choice that is compatible with the program. This would eliminate the barrier to entry and make “Twitch Plays” - or rather now “Chat Plays,” since it isn’t tied to one streaming platform - accessible to everyone.

New ways to play, Playing with a Streamer Personality and their Communities

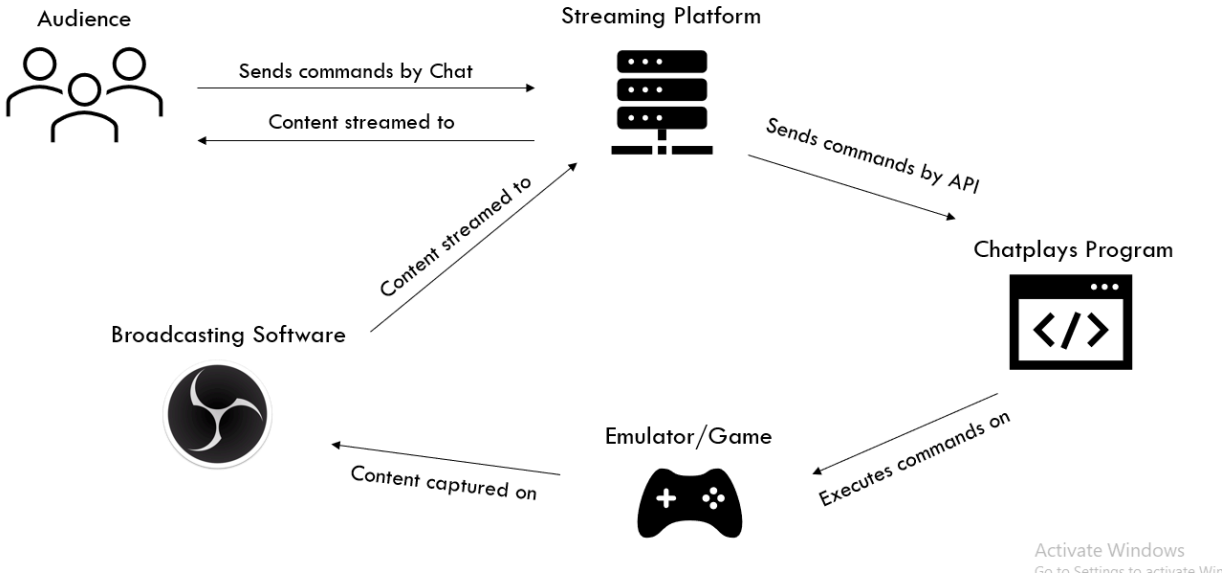
This program would also expand on the concept of the original Twitch Plays to match the strengths of what live streaming is. Now, on this program, instead of it just being the chat playing against a game, the chat could also play against/along with a live streamer, against each other, or against the live chat of another live stream. There’s more fun and attachment to what is happening on screen now since it’s not just a random collective that are trying to overcome and reach a goal; now it’s a specific community of people trying to overcome the goal, overcome their favorite streaming personality, overcome each other, or overcome another community of people watching a live stream.

How would it work?

Ecosystem

The audience would send commands through the live chat to control the game, the ChatPlays program would receive those commands through the streaming platform’s API, then those commands would be executed on a game/emulator, the audio & video output of which would be captured by streaming software and sent back to the streaming platform, which would broadcast the content back to the audience. The streamer/user has control over the ChatPlays program,

game/emulator, and the streaming software.



Compatibility and Rollout

To make as many games as possible compatible with the program, instead of having to work game-by-game, the ChatPlays program could interface with emulator programs for popular videogame consoles such as Dolphin (GameCube/Wii emulator), Visual Boy Advanced (GameBoy Advanced emulator), ePSXe (Playstation emulator), etc. When you make an emulator compatible, you make every single game on that system available, adding thousands of games to the compatibility library. The efficiency speaks for itself.

How about computer games not on emulators? For computer games, the ChatPlays program could be used as a virtual controller and treated as if it were an additional controller in-game. This is the easiest way to make thousands of these games compatible, by simply telling the computer to treat the program as if it were an additional controller for the game.

ChatPlays modes and types

What are ChatPlays modes and types?

ChatPlays modes & types are the ways a streamer sets how they want the ChatPlays program to execute commands.

Modes:

The *Modes* set the rules on how the commands are chosen executed. These *Modes* are the same as the governance used in the classical Twitch Plays which are *Democracy*, *Anarchy*, and *a poll between Anarchy and Democracy*. *Democracy* chooses commands based on a poll which lasts for a set time - once this set time is expired the top command is executed on the game/emulator. *Anarchy* executes every command sent in consecutive order. The *poll between the two* lets the audience in the live chat choose between Anarchy and Democracy on a live poll and after a certain threshold is met, say 60% of the poll or so want to switch modes, the ChatPlays mode is automatically changed to that mode. In this poll mode the streamer must

choose which mode to start on and how often the poll resets the votes. Various other modes, including monetizable modes, would be added later on.

Types:
ChatPlays Types tell the program what type of game is being played and who is playing it. As of right now there are 4 game Types: *Chat vs Game*, *Chat vs Streamer/Co-op, Chat vs Chat, Community vs Community. Chat vs Game* is the classical Twitch Plays wherein only the live chat is playing the game. *Chat vs Streamer/Co-op* is when the live chat plays a game against or along with the streamer. *Chat vs Chat* is when the chat is divided by some variable or by choice into a team or number of teams and they battle it out against each other. *My Chat vs Your Chat*, or *Community vs Community*, is when a live streamer's live chat plays a game against another streamer's live chat.

ChatPlays terminology and language

How ChatPlays is controlled can be described with these terms: *Inputs, Input Sequences, Commands, Command Categories/Subcategories,* and *game/universal Command sets.* These terms form a sort of language that makes it easy for both streamers and audiences to use and interact with the ChatPlays program.

Inputs:
Inputs are the most basic unit of control. They correspond to the action of someone pressing a button on a video game controller, like A, B, Start, Select, etc.

Inputs can be tapped, pressed, or held for a length of time. The length of time determined for each will have a default but will also be changeable. If no length of time is typed, the program by default will view it as a "press", which can also be changed if needed. The syntax would be something like:

Tap	Press	Hold
Ta	Pa	Ha
"Tap A"	"Press A"	"Hold A"

To be more specific on the exact time pressed there could be another syntax like this:

(1.6)a
"press for A for 1.6 seconds"

An *Input* measure can also be several buttons hit at the same time. The program will default to 2, but it can be changed to hitting up to all the buttons on the controller/keyboard. Syntax:

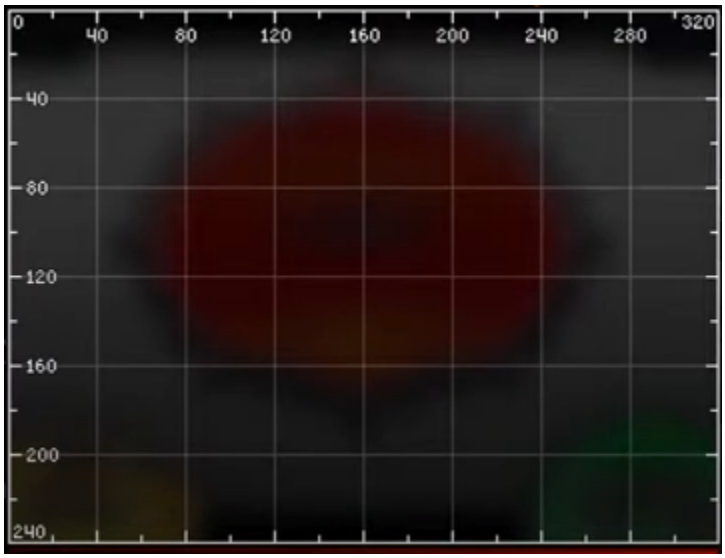
B+A
"Press B and A at the same time"

To press an *Input* multiple times, put a number up to 10 after the button name.

Up10

“Press up 10 times”

Some consoles such as the Nintendo DS or Wii use touch screen or motion controls. For these types of controls, there will be a mapping system with an x- and y-axis like the one used by Twitch Plays Pokémon here:



The *Input* for pressing the screen would look something like this:

(240,200)
“Press 240 on the X axis and 200 on the Y axis”

To drag the virtual cursor on the screen:

(240,200)>(40,40)
“Drag from 240 on the X axis and 200 on the Y to 40 on the X and Y axis”

Of course all of these rules can be mixed together in some way like:

(2.5)a10+b2
“Press A 10 times for 2.5 seconds while simultaneously pressing B twice”

The syntax for the program must NOT be case-sensitive because of the r9k function in Twitch chat.

Input Sequences:

*Input Sequences*are series of Inputs which are executed in sequential order with some sort of dividing syntax like a comma:

Up10, b+a, down+left, right

"Press up 10 times, then press b and a at the same time, then press down and left at the same time, then press right"

Commands:

Commands are defined *Input Sequences.* Instead of having to type out complicated sequencesof inputs to get things done, people can type out a simple Command in its place. Here is an example:

You want to choose the move on the top left of the touch screen for a Pokémon during a battle in a Nintendo DS mainline Pokémon game

Name	Description	Command	Input Sequence
Move 1	Orders the move on the top left of the touch screen to the Pokémon battling	move1	(160,120),(80,80)

*Input Sequencess (*including a simple Input) and *Commands* are the two controls that the audience in the live chat can invoke. The rest of the terminology we are going to discuss are only controllable by the streamer/main-userand mainly exist for comprehension, functional, and organizational purposes.

Command Categories/Subcategories:

Command Categories are Commands that can go together under the same idea or are connected by being part of a particular graphic interface or phase in a game. For example, in Pokémon there are different phases: the overworld, a Pokémon battle, naming a Pokémon, etc. We could use the "Pokémon battle" phase as a Command Category and then have function Subcategories such as "battle", "switching", and "items", which all contain the various Commands. Categories don't necessarily need Subcategories since they only exist for organizational purposes; and just because a Category has a Subcategory doesn't necessarily mean that all the Commands will be in the Subcategories; there can also be Commands directly under a Category and not under a Subcategory. Using the example from before, the "run" command in Pokémon doesn't really fit into the hypothetical Subcategories for "battle", "switching", and "items", so it doesn't have to be listed under any of them and can simply be a Command in the "Pokémon battle" Command Category. The organization of the above Command Categories would look something like this:

1. Pokémon Battle Phase ***Category***
    a. Battle *Subcategory*
        i. Move 1 *Command*
        ii. Move 2 *Command*…
    b. Switching *Subcategory*
        i. Party member 2 switch *Command*

ii. Party member 3 switch *Command*…

c. Items *Subcategory*

i. Potion *Command*

ii. Super Potion *Command*…

d. Run *Command*

The purpose of Categories and Subcategories are so that streamers can easily turn various Commands on and off quickly while they are using the program.

Game/Universal Command Sets:
Command Sets are a collection of Commands organized by Command Category and Subcategory. These Command Sets can be made for either a specific game or for universal use. One reason someone might want to make a "Universal" set is because there are game series, like Pokémon, that have gameplay that use the same Inputs, functions, and phases in most of their games.

How would users make their commands?

Command Studio
Users would be able to make *Command Sets* in the *Command Studio.* When a user enters the Command Studio and wants to make a new set, they are prompted for whether they want a universal or game specific set, what emulator(s) and game(s) the set is intended for, etc. Once they enter this menu, they will be able to make Commands by typing out Input Sequences manually or by loading up the game, adding the Inputs they need in-game via a keyboard or controller, and recording their Inputs used through a ChatPlays "record command" function. They will also be able to test out/modify their Commands on the game through the Studio. Command Categories and Subcategories are optional.

Command Marketplace
Making Command Sets, no matter how accessible it is, takes time, and not everyone will want to take the time to make Commands themselves. People will also naturally want to share the Commands and novel things they've made with Commands Sets. So, ChatPlays couldhave a Command Marketplace where people can share, buy, and sell Command Sets and digital assets to each other, in exchange for money or possibly in exchange for cryptocurrency.

Artificial intelligence in ChatPlays – How artificial intelligence can improve user experience

Game Phase Command Automation - eliminating tediously turning Command Categories on and off:
When using the ChatPlays program without any artificial intelligence, on certain games, a user would have to turn Command Categories, Subcategories, etc. on and off manually for certain phases in a game. Although a minor inconvenience, manually turning these Commands on and off can quickly become tedious in these situations. Here is an example:

Someone is using ChatPlays to let their audience play Pokémon Stadium against them in the Democracy mode. The user has a Command Category for the phase of the game where the Pokémon teams are chosen and a different Command Category for the phase of the game where there is a battle. When it comes time to do so, and the audience has to choose the Pokémon they want to use, the streamer would switch on the Command Category. Once the audience finishes choosing their Pokémon and the two teams enter battle, the streamer has to switch off the Command Category that lets the audience choose the Pokémon and has to switch on the Command Category for the battle phase. Doing this over and over again for each battle would quickly get tiring.

Now, let's imagine that the ChatPlays program could identify screens and associate those screens with the "phases" of the game. Applying it to the same scenario:

When the user enters the "choose your Pokémon" section, the ChatPlays program recognizes the screen and automatically turns on the Command Category associated with that phase, and turns off the Categories that are not used during that phase of the game. Then, when the audience is done with that phase and enters the battle phase, the same process occurs automatically. This eliminates the tedious need to turn Command Categories on and off constantly.

Resolving Game Circumstantial Commands:

Another hurdle that can be resolved through AI are problems concerning the circumstance that the game is in. Let's give another example:

Here's how the Commands on ChatPlays would work without AI

Imagine a Command in the mainline "Pokémon" series where someone wanted to change the place of Pokémon "party members":

Name: Party member #6 to #1
Description: Changes the Pokémon on the bottom right of the party in slot #6 to become the leader of the party in slot #1.
Command: t6-t1
Input Sequence: start, down, a3, down2, right, a3, b2

Because of the nature of ChatPlays, these types of Circumstantial Commands wouldn't work in its current form. In this example, if the user's audience used this Command anywhere that isn't the main Pokémon overworld screen in a very specific circumstance, it would do something wonky.

If ChatPlays could track the phases and screens of a game using AI, we could order the AI through a Command to go and perform a specific action regardless of circumstance. If the AI recognizes that a Command is not possible during a certain game phase, then it turns that Command off automatically. If the AI knows a Command can be executed in a phase of the game, the Command will be on and if asked, it will navigate through the game UI to execute the Command.

AI recognizing game elements and helping visualize possible commands:
Another way we could use AI in ChatPlays would be by helping the audience see the possible Commands on screen. Take for example the AI-driven overlay illustrated below:

How AI would label a war in “Advanced Wars” for the Gameboy Advanced:

In this example, the AI could also recognize game elements such as the different types of army units, the different places an army unit can move, the opposing army units, actions, etc. The AI essentially knows how to play the game, how the game works, and is taking orders from the audience directly in order to overcome the hurdle of Command circumstance.

Command Studio and ChatPlays AI
The aim is to eventually make programming these AIs user-friendly enough that anyone could do it in the Command Studio.

Game Phase Automation in command studio
The first step would be to allow users to define “Game Phases” by letting them take screen shots of a certain phase of a game and feeding it to the AI so it recognizes what a certain phase in the game “looks like” and pairing it with the associated Command Category to turn off and on and game modes to switch to when it gets to that phase. i.e.:

Say hypothetically there was an AI that was able to type things out in game, say the AI could only do this in Democracy mode, and say someone wanted to use this AI for a mainline Pokémon series game when a Pokémon is captured and nicknamed. How would he set this up in the Command Studio?

The user would feed screenshots to the ChatPlays program of what the “name your Pokémon” phase looks like, then he would set it up so when the program recognizes the phase and

triggers, the program automatically changes the mode to Democracy and enables the AI. Then, the user would set the program up to return to its prior state when the AI is done being used and the game is no longer in the “name your Pokémon” phase.

Circumstantial Commands

The way users would start making Circumstantial Commands would be by first creating Game Phases for the game which they want to make Circumstantial Commands for. Then, they would tell the program which Game Phases the Circumstantial Commands are possible in. Next, the ChatPlays program would put the user in different random places inside the labeled Game Phases where the Commands are possible and observing what the user does to complete the task they want for the Circumstantial Command. By doing this, the AI observes how a user is navigating through an interface and what in-game information needs to change for a task to be complete. The AI takes this information and figures out how to do it itself in the quickest and most efficient way possible. While this is happening, the user might see a “calculating” loading screen or something similar. Once the AI figures out how to do the task and navigate the screens, the Circumstantial Command is created, and can be tested and modified by the user.

Creating a Keyboard Game Phase

Many games have a game phase where the user needs to type something out. It’s hard to type things out in Anarchy and Democracy, so a solution would be an AI that can type out whatever the chat would like. Doing something like this in the Command Studio would be similar to the way Circumstantial Commands work. The streamer tells the program which phase is a “keyboard phase”, the program prompts the streamer to type some things out in the phase, the program observes it and learns what the user is doing, and now the AI can type things out.

Anarchy and Democracy in a Keyboard phase

Democracy would work exactly like one would expect: the result with the most votes gets typed out after a set time. Anarchy would work a little differently than usual. here would be two variations. The first variation gives everyone a set time to submit what they want typed out, and once the time runs out, the AI chooses a result at random. The second variation has a set time, and whatever is on queue when the time expires gets executed by the AI as the result.

ChatPlays “Minigames”

One of the problems with “Twitch Plays” type streams is that they take too long. It may take several days to get through a game, and no one wants to sit through that. “Minigames” aim to solve this problem by making short minigames inside of games that already exist. Take this video of Dougdoug letting his chat control a chicken to cross a road in GTA 5 - this is a perfect example of what a “minigame” could be.

The way this would be done would be by loading up a save state of a game, giving a goal for the audience, and an AI that knows what the goal is, and will reset or end the game once the chat either fails or succeeds. These “minigames” could be shared and sold in the marketplace. We have yet to figure out how this concept would work in the Command Studio, but we are certain that it would be similar to the training described with the other AIs.

Additional quality-of-life features

- The Ability to turn ChatPlays on and off at any time
- The ability to turn certain Inputs on and off
- Quick access to a chat poll
- Subscribers/Members only and the ability to distinguish subscribers/members from non-subscribers/non-members
- Visual overlay of people sending their Commands and various other program information overlays for streaming software such as OBS and streamlabs
- Browser extension that hides ChatPlays Commands in the live chat for both streamers and audience members, so the chat is more usable.

- Auto-save states for games

How will users in the live chat know which Commands are available?

Quick Copy and Paste

The easiest way to let the audience know what Commands are available to use would be to allow the user to click a button that copies all available Commands that includes descriptions and paste it wherever they would like. Whether that be the Bio of their channel, a chatbot command like those used with the nightbot chatbot, etc.

Chatbot

A chatbot that connects to the ChatPlays program linked to the user's channel that sends usable Commands when someone types out something like “!commands”

A Personal Link for each user

Each user of ChatPlays has a link that they can give their followers so they can see the Command Sets they are using, along with descriptions of what they do, and the ability to see which ones are on/off in real time.

Recent Use Cases and testimonials of people who made “Twitch Plays”

There have already been several popular live streamers who have made their own “Twitch Plays”-like video content and have been rewarded with thousands or millions of views.

Dougdoug

Dougdoug is at the top of the list. He’s made many videos and has made chat playing on the live stream a large part of his shtick. He is, however, a professional programmer.

Mizkif

Mizkif is another recent one that comes to mind. He's one of the most popular streamers on Twitch, and he isn't a professional programmer, but there's no doubt that he got help from some developers to pull of his "Twitch Plays" streams.

Mizkif has praised this type of "Twitch Plays" content on his twitter after having a run of Pokémon FireRed on his channel while he was on vacation in December 2020.

These are some good use cases that show that what we're talking about works, engages the audience, results in views, and is enjoyable for the content creators as well as the viewers.

Conclusion

We have proposed a computer program that gives anyone the ability to make their own "Twitch Plays" on any major streaming platform, that gives anyone the ability to play with or against their live stream audience, that gives the livestream audience the ability play amongst themselves or against another livestream audience by typing out commands in the live stream chat. All this while being compatible with a wide variety of games. We have proposed a marketplace where people can buy and sell digital assets made in our program in exchange for cryptocurrency or real-world money, and a way to use artificial intelligence to improve user experience. All of this is well and great, and we're sure the concepts we've written about excite a lot of people, but now we have to actually make the damn thing which is the toughest part.