Applying for: Cloudflare Internship Application: Product Management
Name: Jasveen S

Cloudflare Workers for Gaming

Thesis abstract

Cloudflare Workers for Gaming is one of the substantial approaches for game developers to go beyond the limit of passive conventional gaming development with the help of internal cloud network optimizations, reverse proxy abilities, and making the best use of its coding execution environment. The furious competition in the market today needs a new alternative which Cloudflare Workers provide!

User research (market, and needs)

One of the best ways to do the market research for the user and the company would be a constant iterative feedback looping mechanism. Surveying system, along with rewards for the end-user along with a dedicated server storing the market needs by capturing user's data in a secured manner would help. In order to find the USP of the product in the market, gaming exhibition centres can have a dedicated lounge for the user request along with the needs that need to be satisfied for what is missing in the current gaming scenarios.

As an experiment, I'd like to talk about my own product. I'm the CEO and founder of a startup company which basically is gamified coding. We intend to teach people coding through gaming. So as a real-life hypothesis, the market needs innovators. They look for something far beyond the ordinary and should capitalise them in the first look.

The market is already aggressive for gaming thanks to the sophisticated pipelines, algorithm development, and software development processes that the competitors are using. The need in the market is for a different, better, and unique product better than these which can be implemented using the Cloudflare Workers for Gaming.

Product changes or additions

Product changes are required for the better visualisation and data restoration abilities for a particular user. Offline movements of the intricate movements of the game and the user, being done using an API, and an offline handler for the changes to be made can handle issues well. A few changes or tweaks in the workers may help. Better adaptability to the serverless function can help. Implementing the worker implementations along with a slight change of the platform from FaaS to PaaS where the Cloudflare's software of serverless functionalities can change, will help change things for better. Viewing the server caches building over time in the game's serverless background can help view the changes or additions we might need.

As an experiment, let's assume that there're lags and glitches when the object tends to go to a specific place. This addition will be done by keeping in mind that an alternative solution exists within this problem. The lag can be resolved (hopefully) by decreasing the load pressure on the object at that particular instance, and/or by keeping a mix between the visual being shown, and the change require (serverless function deployed by the Workers).

The creation of infrastructure within the Cloudflare's gaming development will prove helpful.

Along with focusing on the serverless cache treatment, we can shift to multiplayer modes improvements with in the infrastructure. VAC systems which already, or anti-cheating algorithms can be detected within the infrastructure. Handling of large concurrent traffic within the Cloudflare workers can be tough to handle if the current load limit is not increased.

Improving the quality (before release)

Constant, and frequent testing along with the builds that are being updated can help. Before release, the quality can be increased by checking similar builds which were done before. Unit testing along with integration deployment and consistency checks will help in this improvisation (which is done generally in software development). The API development check which was mentioned earlier along with the server's database integration can help, as it'll help in strengthing the overall game development. As a hypothesis, if the server deployment doesn't work efficiently, we'd then need to look out for an alternative within the Workers. A couple of important tweaks and infrastructural deployments would then be required. A number of testings can be implemented which can help in proper improvisation in the quality before the release of the product.

Goals to measure success (of what we've built)

Once a game has been in the market for a certain amount of time, a 'break down' period should occur where the whereabouts of success (and, of course, failures) can be discussed. User API data collected from the serverless functions can be used within time indefinite in order to check the milestone that has been achieved. Post-release developments, great advertising, in house bug (and, problem) checker from time to time can help. Capability to scale to large level concurrent user experience should also help. Specific product releases (like alpha, beta, gamma) can test a particular set of users can be used as a yardstick for our success. The anti-cheating systems prevalent can also be of great use which can determine how far the network and firewall security is effective.

Failure-risk ratio, next steps (Risks which might lead to its failure)

Concomitant traffic of several users using the Cloudflare workers can lead to heavy congestion on the network which might some thinking. The game development should be so good that words should go around praising the game because people tend to believe what's vocally told than the visuals. The chain deployment at the Workers reinforcement system can be a worry since it has not handled that much traffic before, and the users can be messing around with the software by using unfiltered access. The heavy competition in the market is another issue to worry about. The market is literally bombarding with hundreds of thousands of games every weekend (some, which are literally awestruck), and our product should be very unique, and oblivious to the given current scenario. Packet data loss should be minimized over the severs while maintaining the high performance of the game.