GDC | March 20-24, 2023
San Francisco, CA

Creating Scalable and Engaging User Generated Content in 'Saints Row Boss Factory'

Ispanji Pratama

Sr. Software Engineering Manager

ACCELBYTE

#GDC23

A bit about me

- *From product management to engineering*
- *10+ years working in online services across multiple verticals (messaging, news, e-commerce, etc.)*
- *4+ years specifically in the game backend*
- *Help shipped 7+ game within the past 2 years, powered by AccelByte backend*

Agenda

- ➔ *The tour of UGC system that we built for Saints Row*
- ➔ *How we built it*
- ➔ *Content Moderation*
- ➔ *Cross Platform Implementation*
- ➔ *Some Learnings from running this at scale*

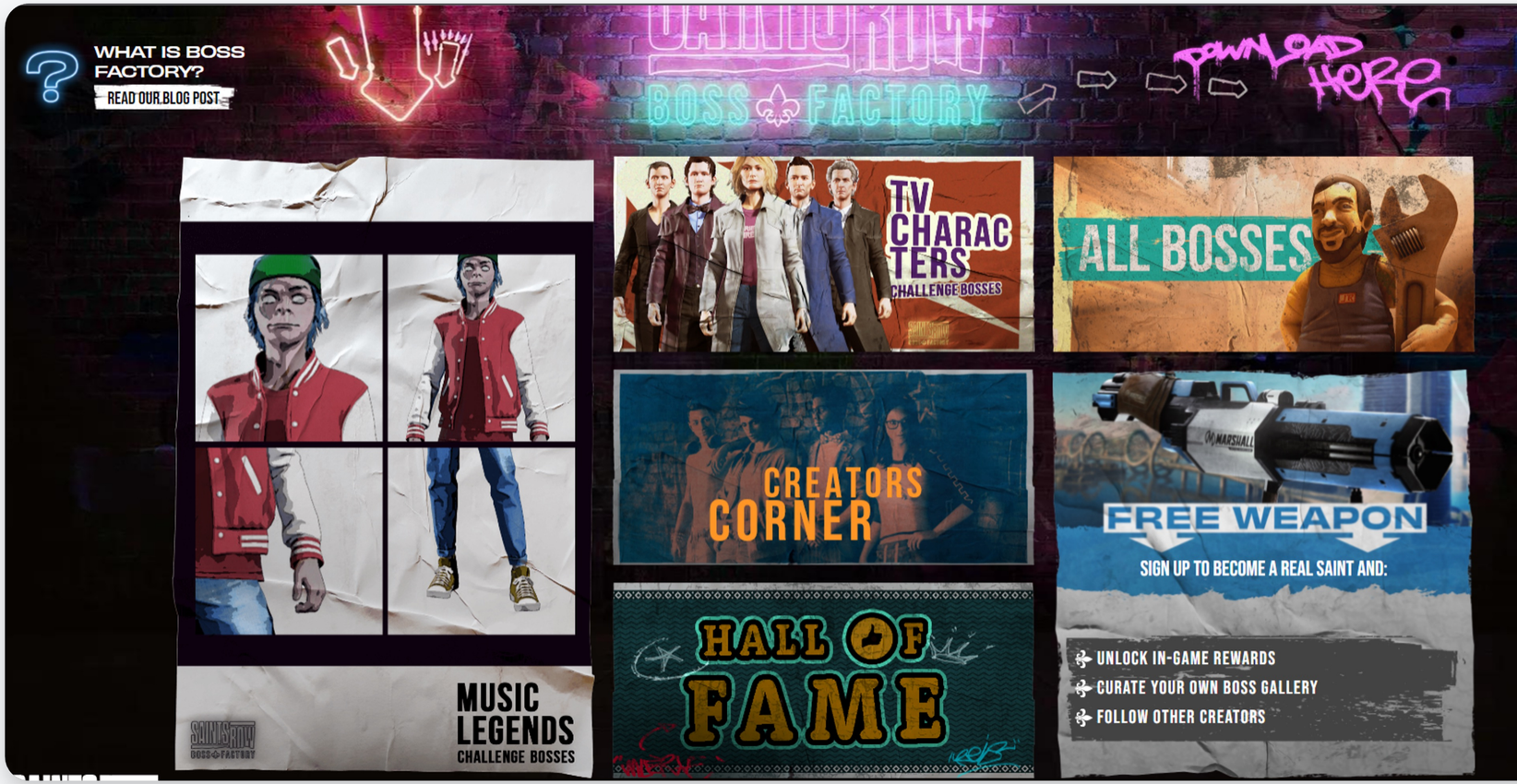
Saints Row Boss Factory

Extensive Character creation and customization tool

Community sharing experiences

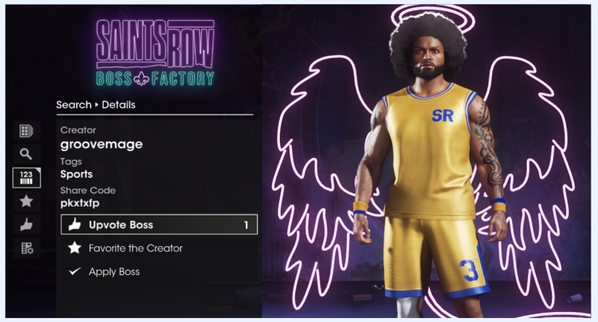
Web Showcase Experience

Showcasing popular ***contents*** *and* ***creators****, that are accessible from any browsers and devices*

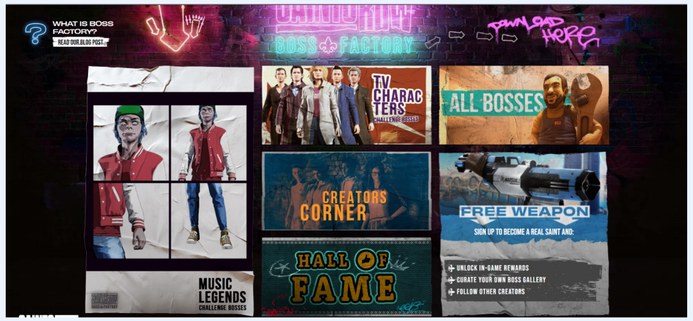


The Timeline

June 2022



+



Free Cross Platform Character Customization Demo

Web Showcase Experience
https://bossfactory.saintsrow.com/

Aug 2022

Main Game Launch with the same character customization feature

Some Metrics from the Demo Launch

1st week — **67XXX** — *Average Bosses created per day*

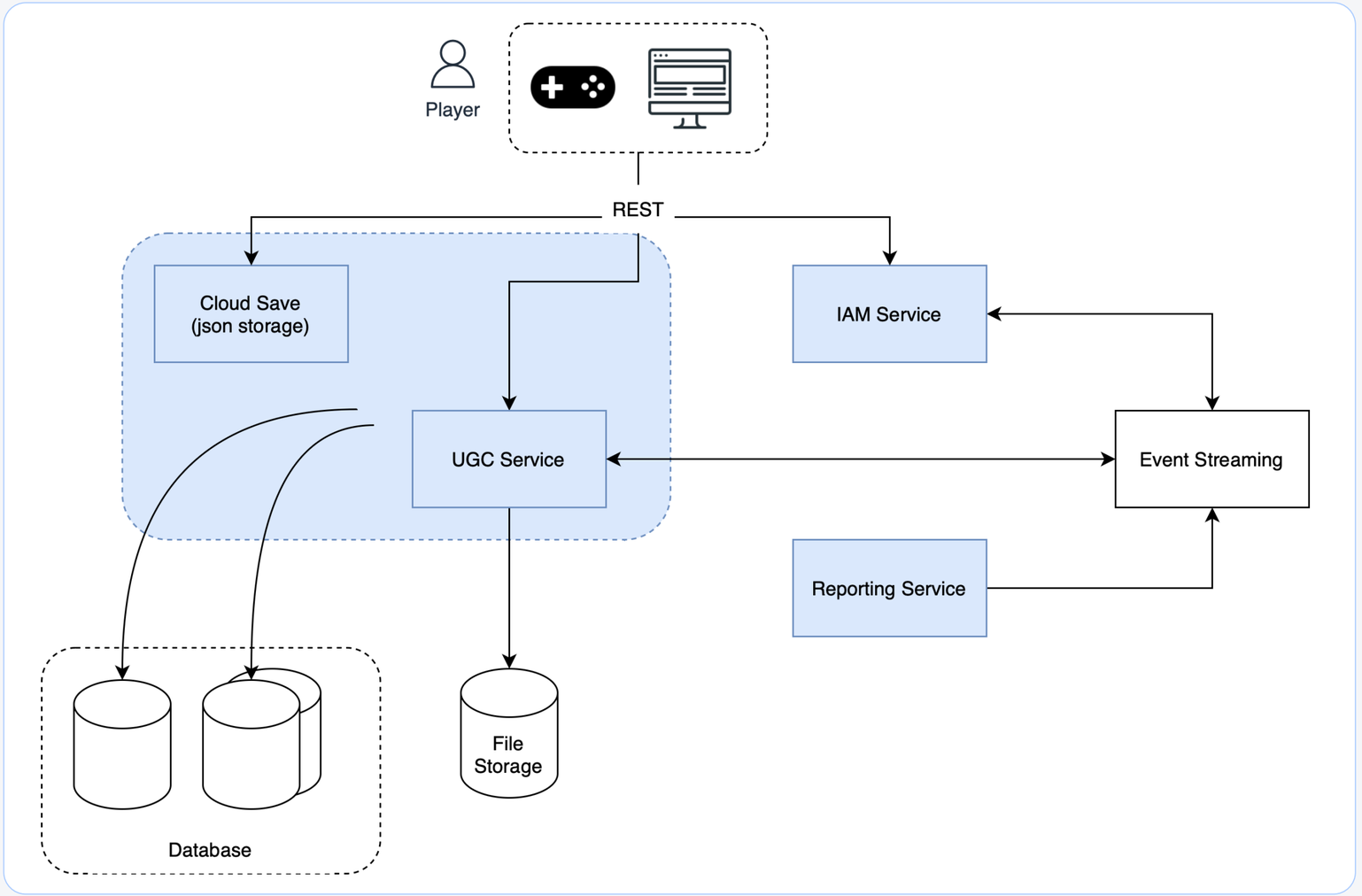
1st month — **860XXX** — *Total Bosses created*

130XXX — ***Highest total bosses created in 1 day***

How we built it?

Service Architecture

➔ ***The essentials***



How we built it?

Service Architecture

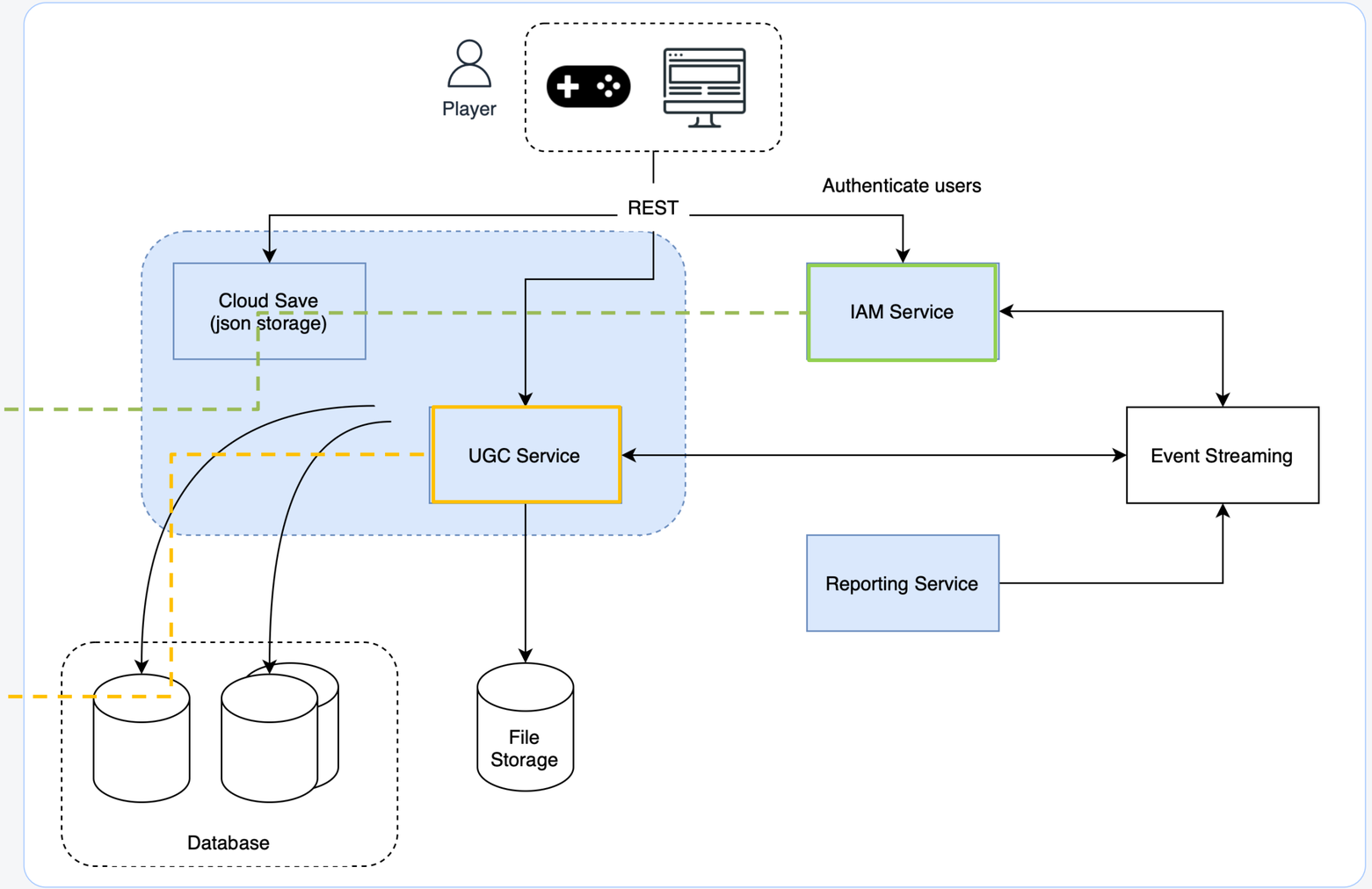
➔ The essentials

IAM Service

- User authentication
- User management
- User Banning

UGC Service

- Content sharing
- Content browsing
- Social interactions (like, follow, etc.)



How we built it?

Service Architecture

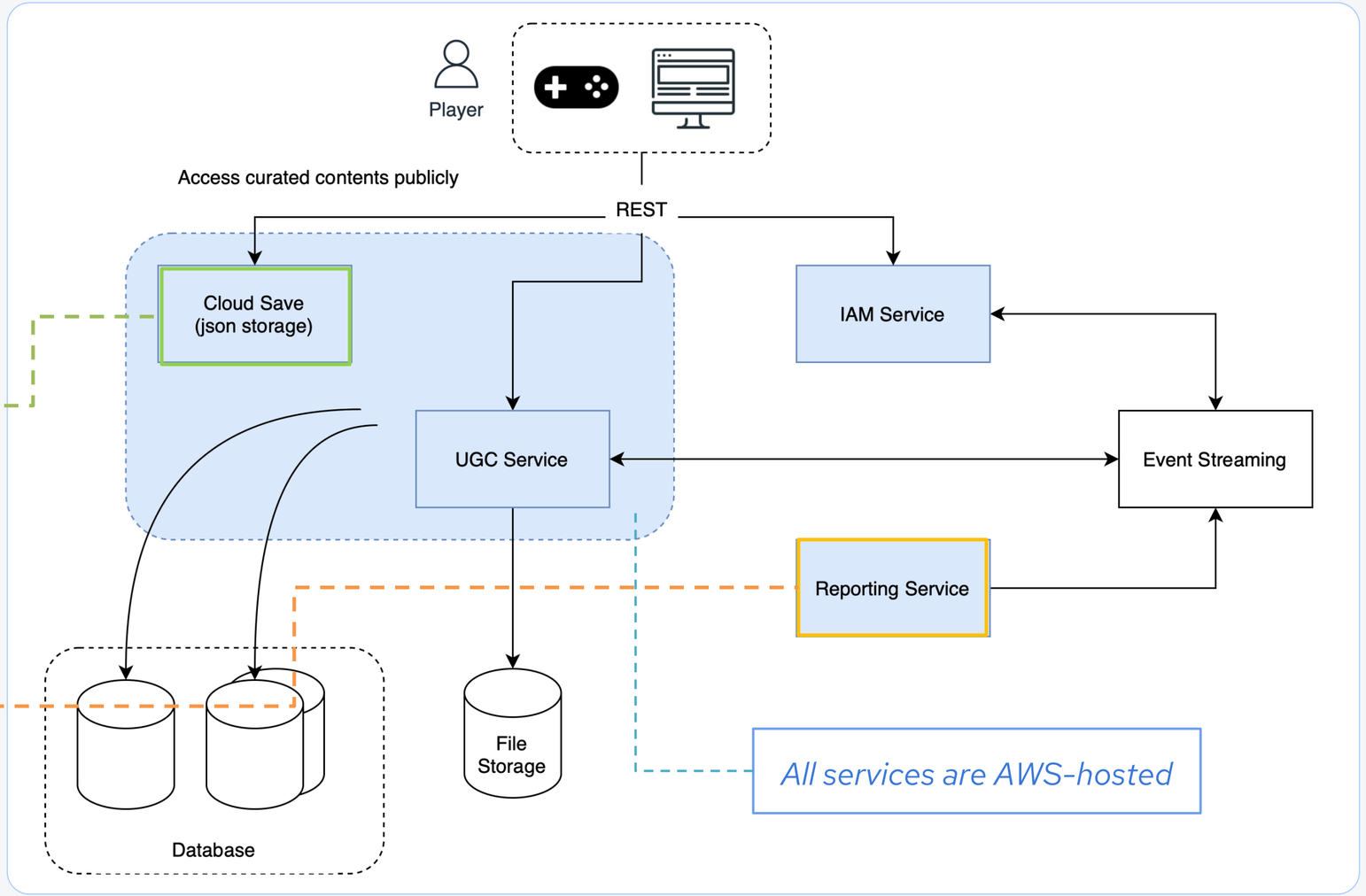
➔ The essentials

Cloud Save (json storage)

- Store curated contents
- Store curated creators

Reporting Service

- Report content
- Content moderation



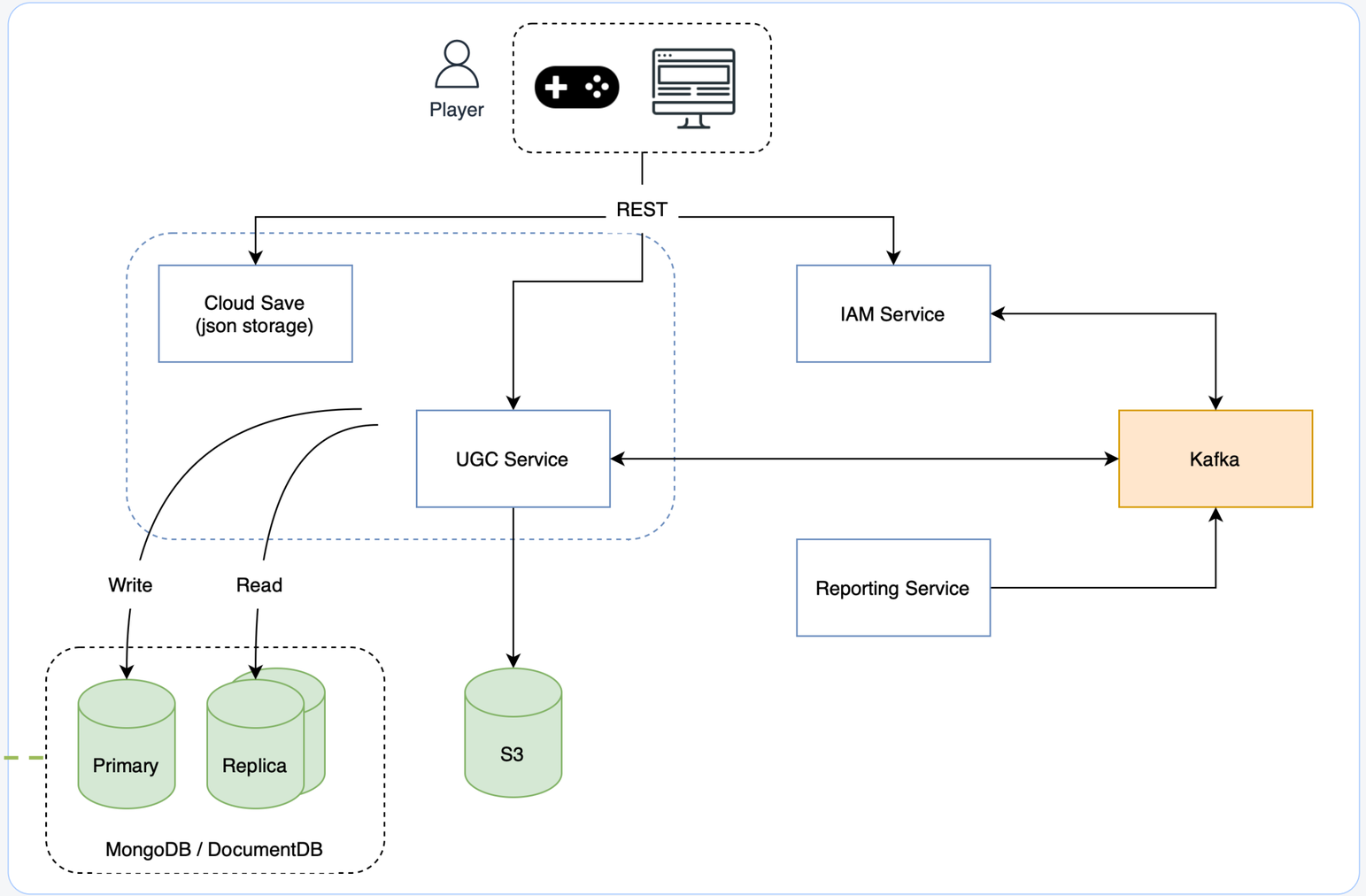
How we built it?

Service Architecture

➔ ***The stacks***

MongoDB - AWS DocumentDB
(For storing content metadata)

- *Flexible unstructured data for faster feature iterations*



How we built it?

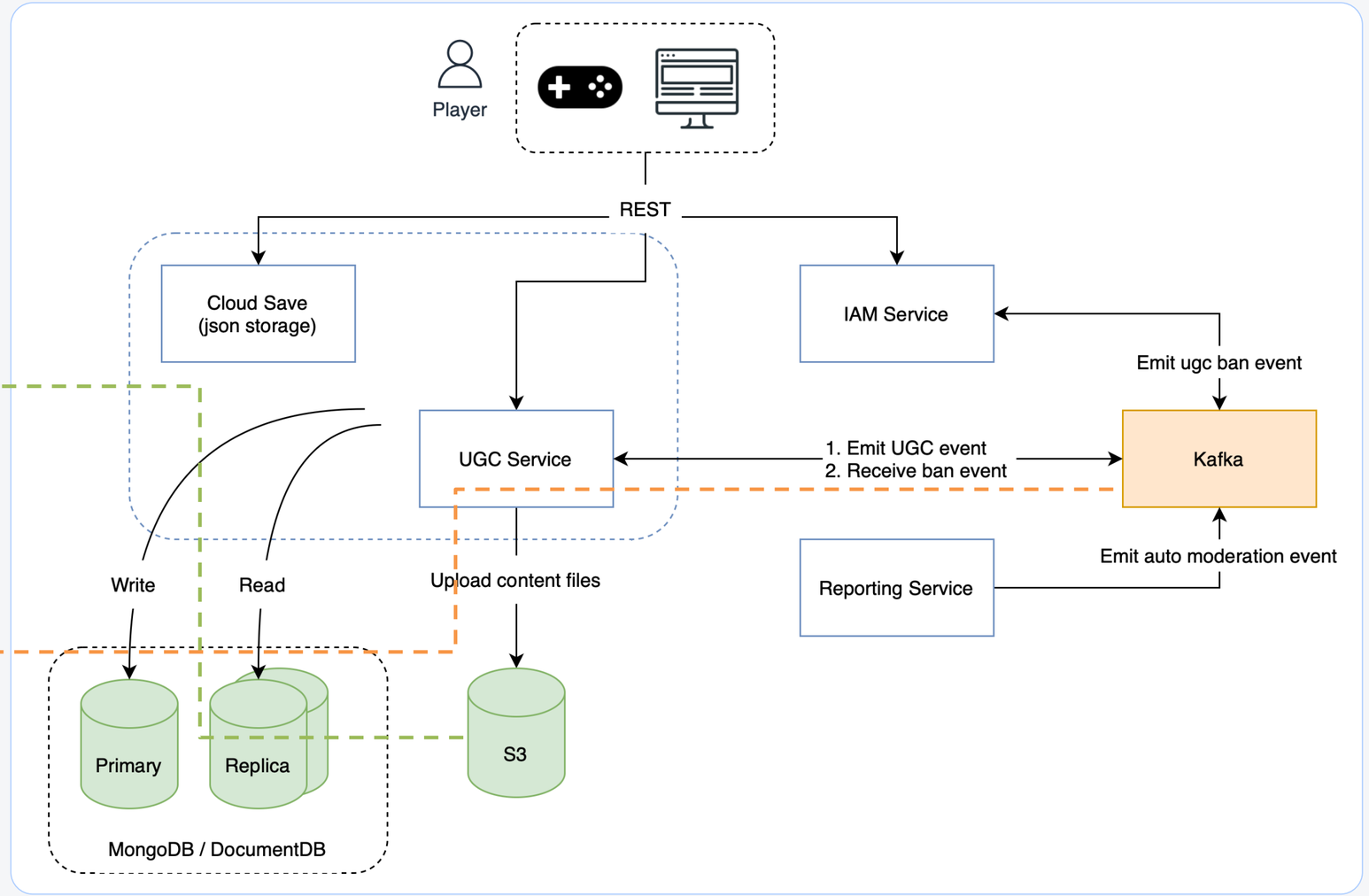
Service Architecture

➔ The stacks

AWS S3 (Object Storage)
(For storing content files)

- *Reliability and availability (CDN setup)*

Kafka - AWS MSK
(For asynchronous communication between services)



Feature/Service Anatomy

Data Model

➔The essentials

Player supplied

Read only system supplied

System calculated

Content associated payload

Feature/Service Anatomy

Data Model - **Example**

Read Only System Supplied

```
"id": "string"
"userId": "string"
"createdTime": "string"
"updatedTime": "string"
"shareCode": "string"
```

System Calculated

```
"downloadCount": 0
"likeCount": 0
```

Player Supplied

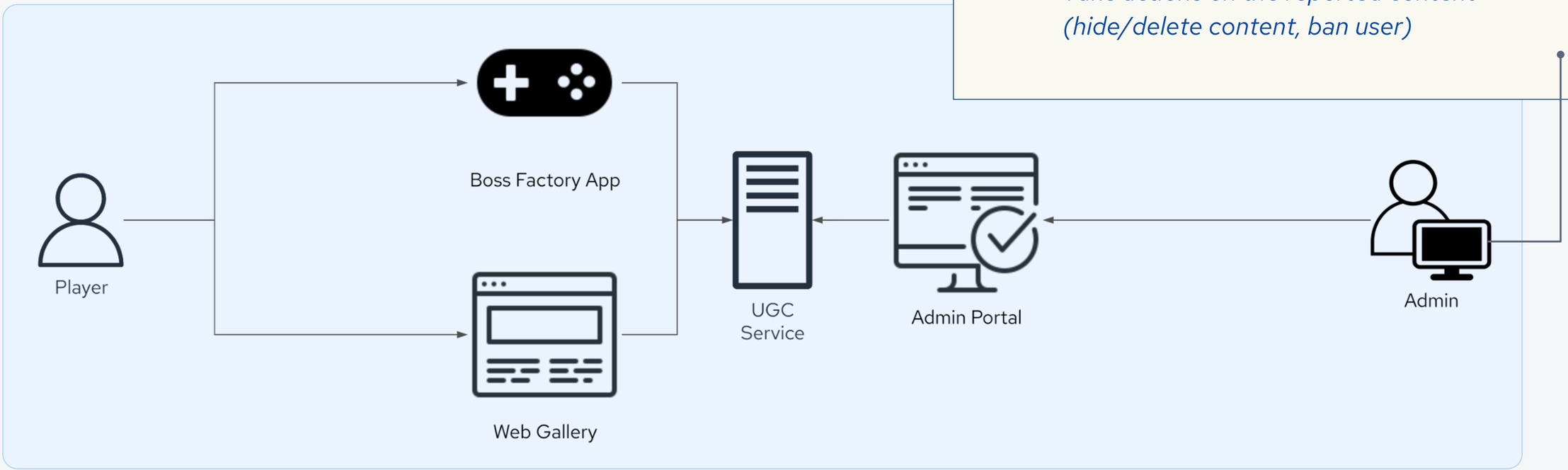
```
"tags": ["string"]
```

Content Associated Payload

```
"payload": "string",
"fileExtension": "string",
"payloadURL": [{
      "source": "string",
      "url": "string"
}],
"previewURL": [{
      "source": "string",
      "url": "string"
}],
"screenshots": [{
      "contentType": "string",
      "fileExtension": "string",
      "screenshotId": "string",
      "url": "string"
}]
```

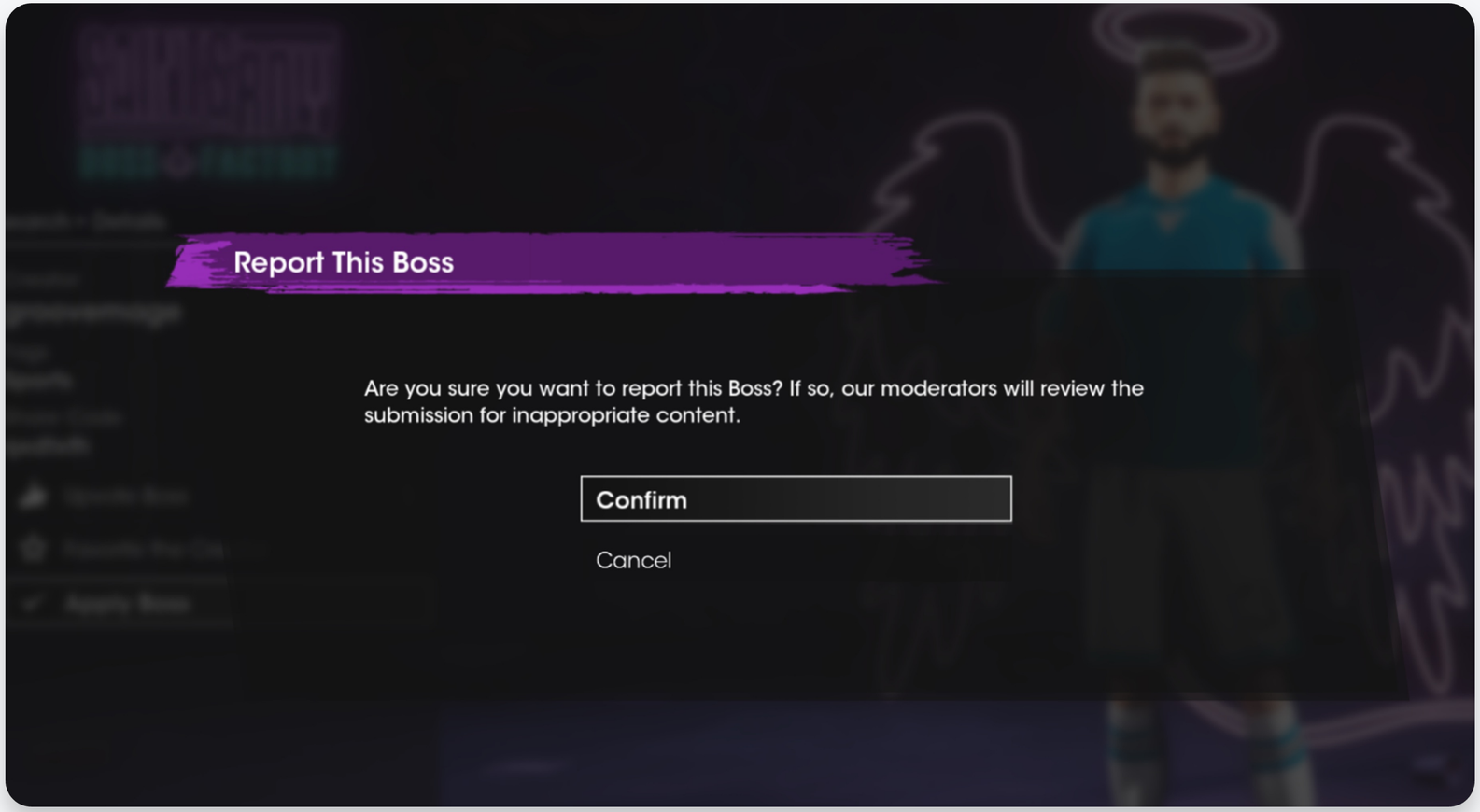
Operations

- *Search and browse content with more control*
- *Curate content for web showcase*
- *Review reported content*
- *Take actions on the reported content (hide/delete content, ban user)*



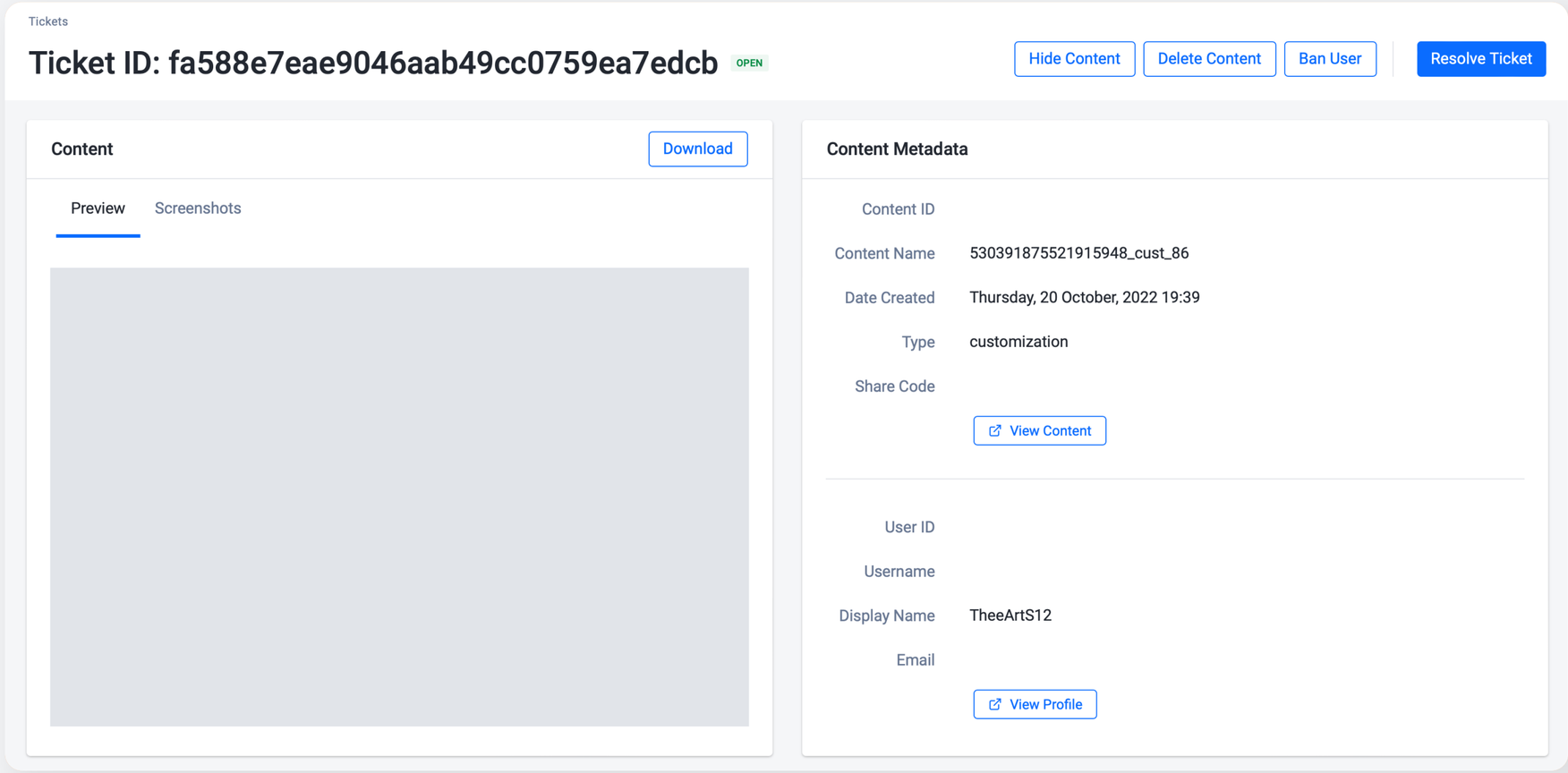
Content Moderation

Straight forward content reporting to track potential offensive contents



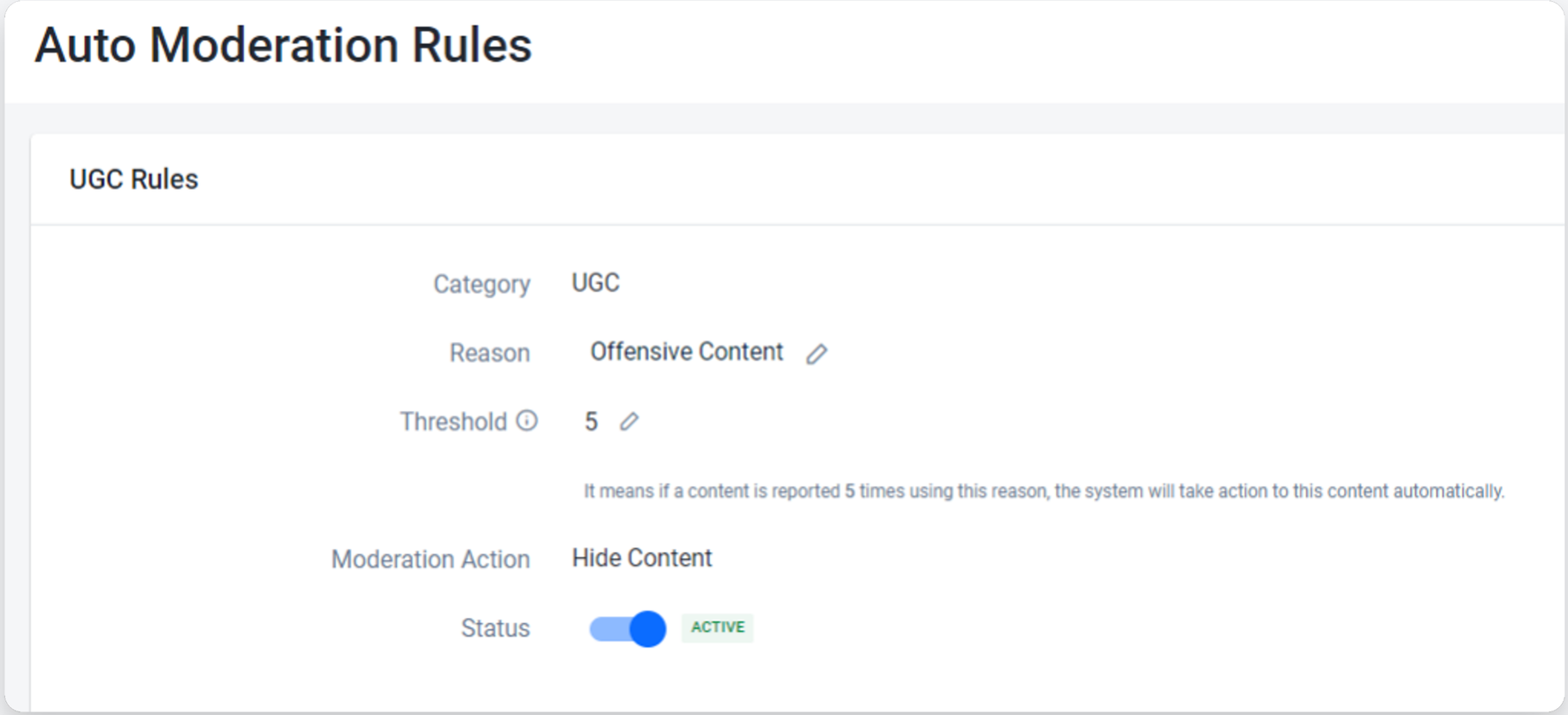
Content Moderation

Several actions that moderator can take



Content Moderation

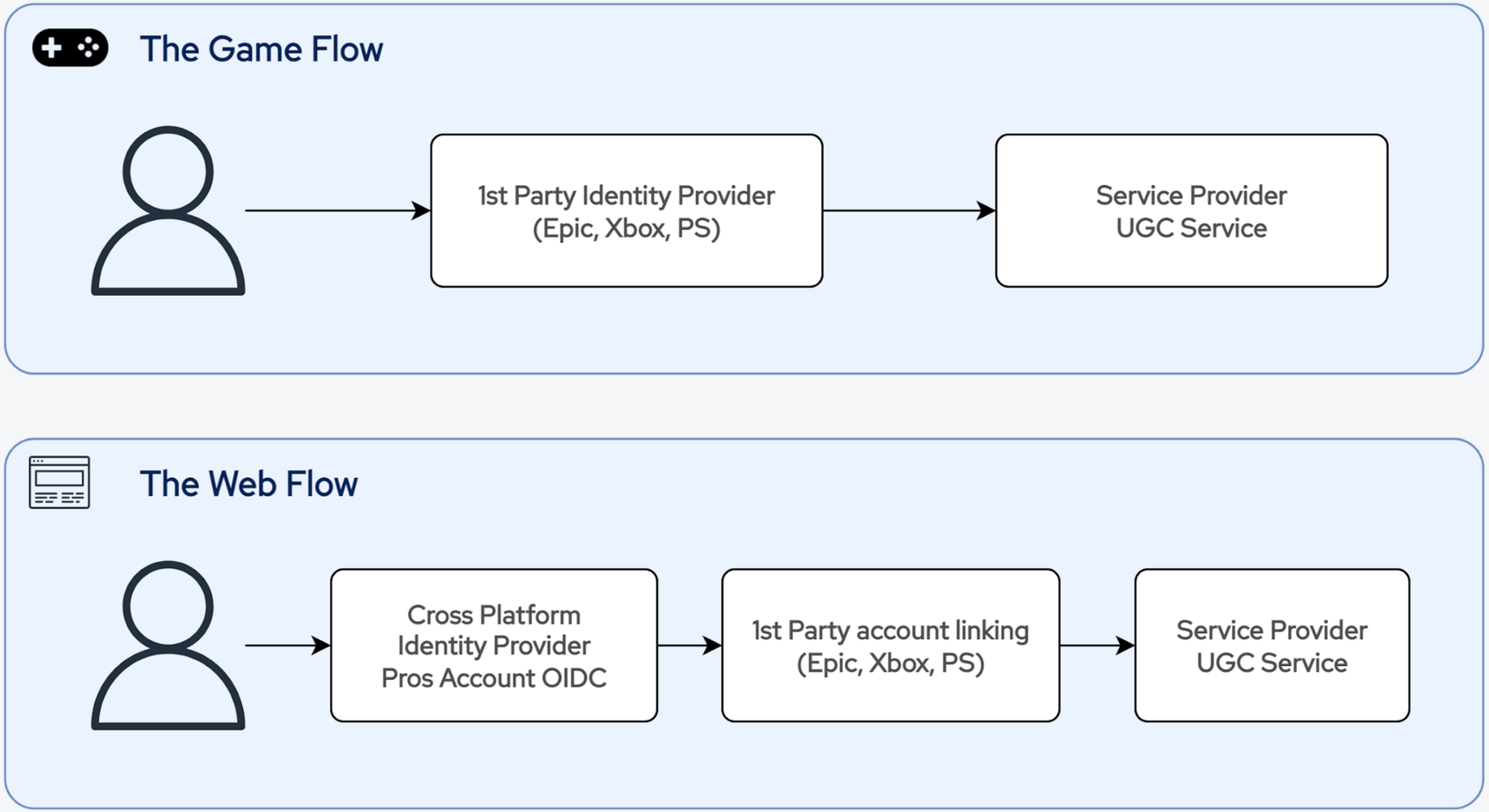
Having a little bit automation to make moderation easier



Cross Platform Implementation

User Login Flow:

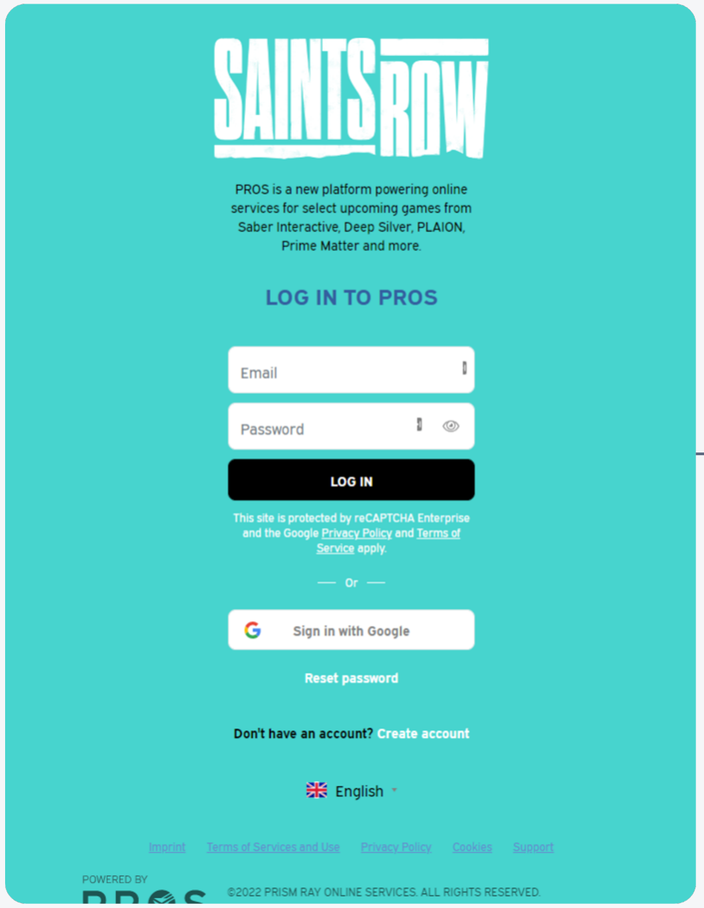
Game vs Website

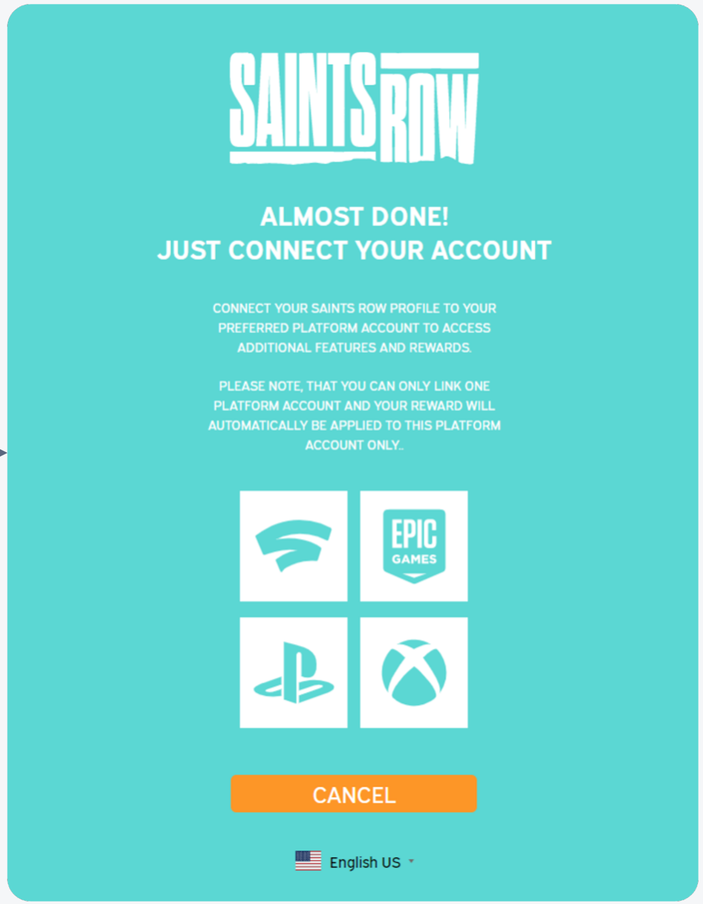


Cross Platform Implementation

- *Use **OIDC** for the **cross platform** user identifier*

- *Must do an account linking to the **account** used from the game*





Learnings from the launch

Main Challenge

Aligning development between teams with multiple different milestones and target

Launch Stages

Learnings – Prelaunch

Problem 1: **Slow query caused by lengthy entry stored in the database**

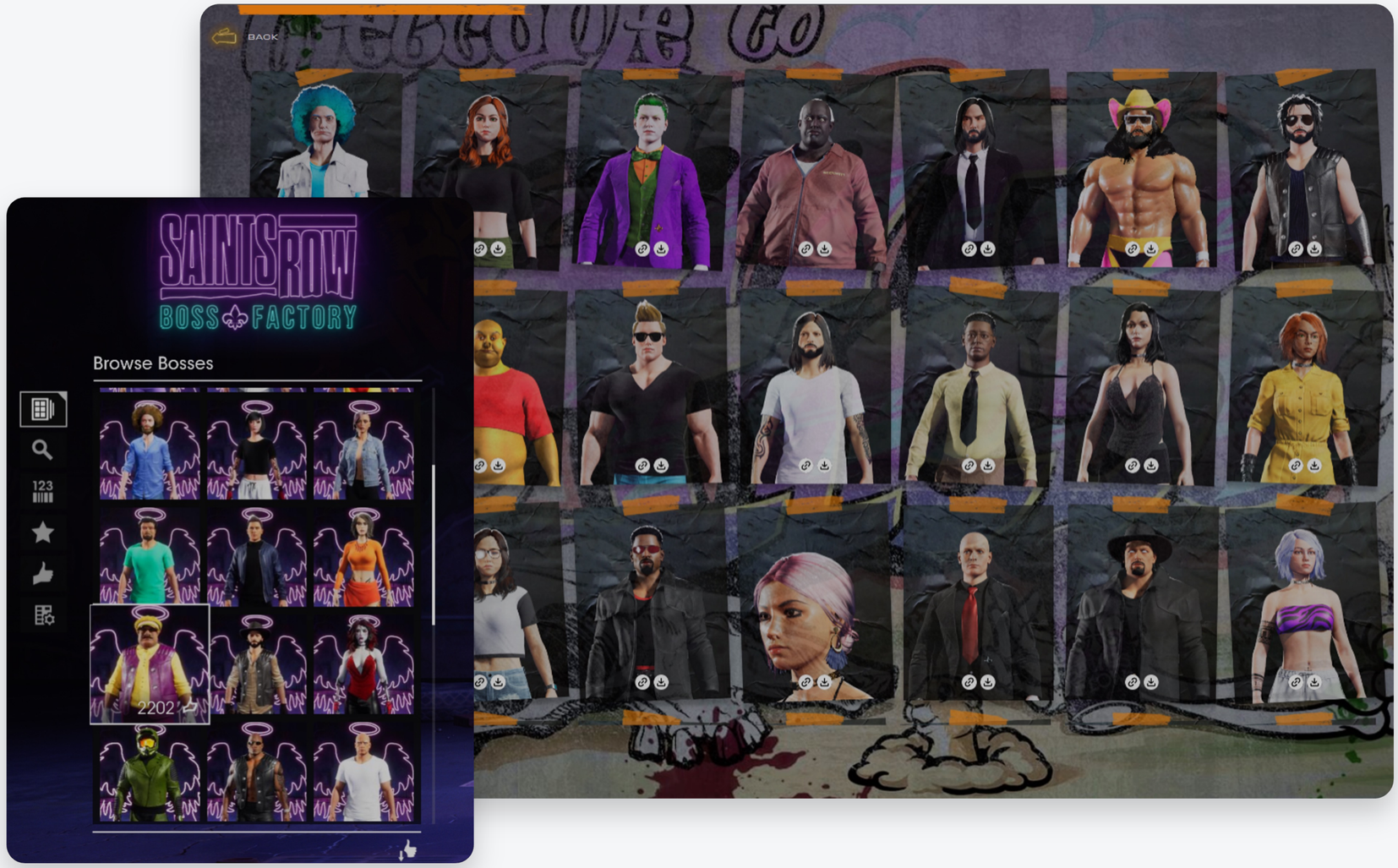
There's a field that stores base64 encoded image format in the database

Learnings – Prelaunch

Problem 1:

Slow query caused by lengthy entry stored in the database

The initial implementation queried and returned that field from most requests



Learnings – Prelaunch

Problem 1:

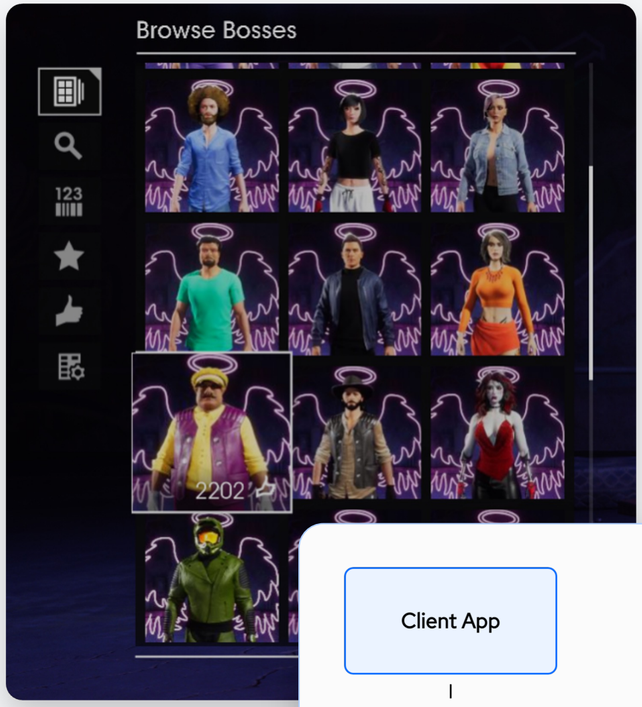
Slow query caused by lengthy entry stored in the database

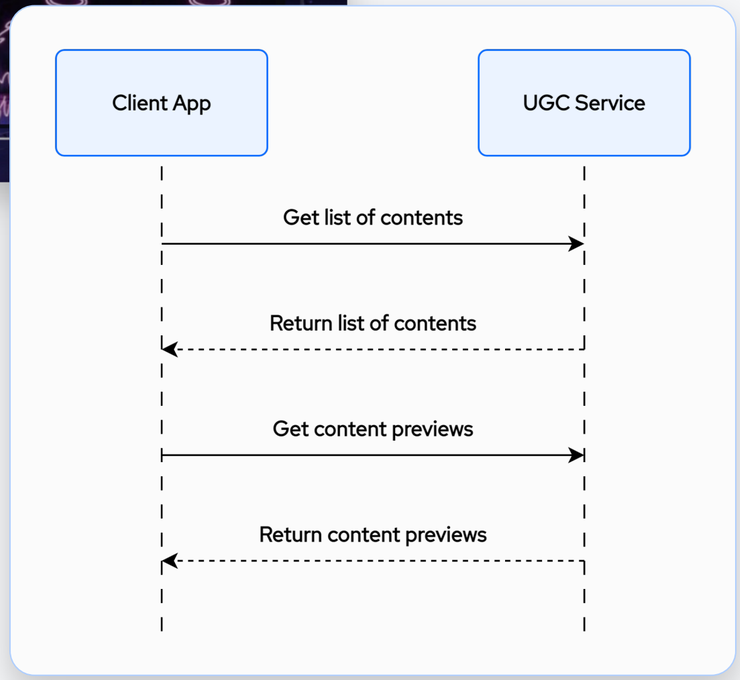
Resolution:

- *Remove the field from the queries & responses*
- *Separate the query to access the preview data*

Further suggestion:

- *Store this type of data in another form of storage e.g. AWS S3*





Learnings – Demo launch

Problem 2:

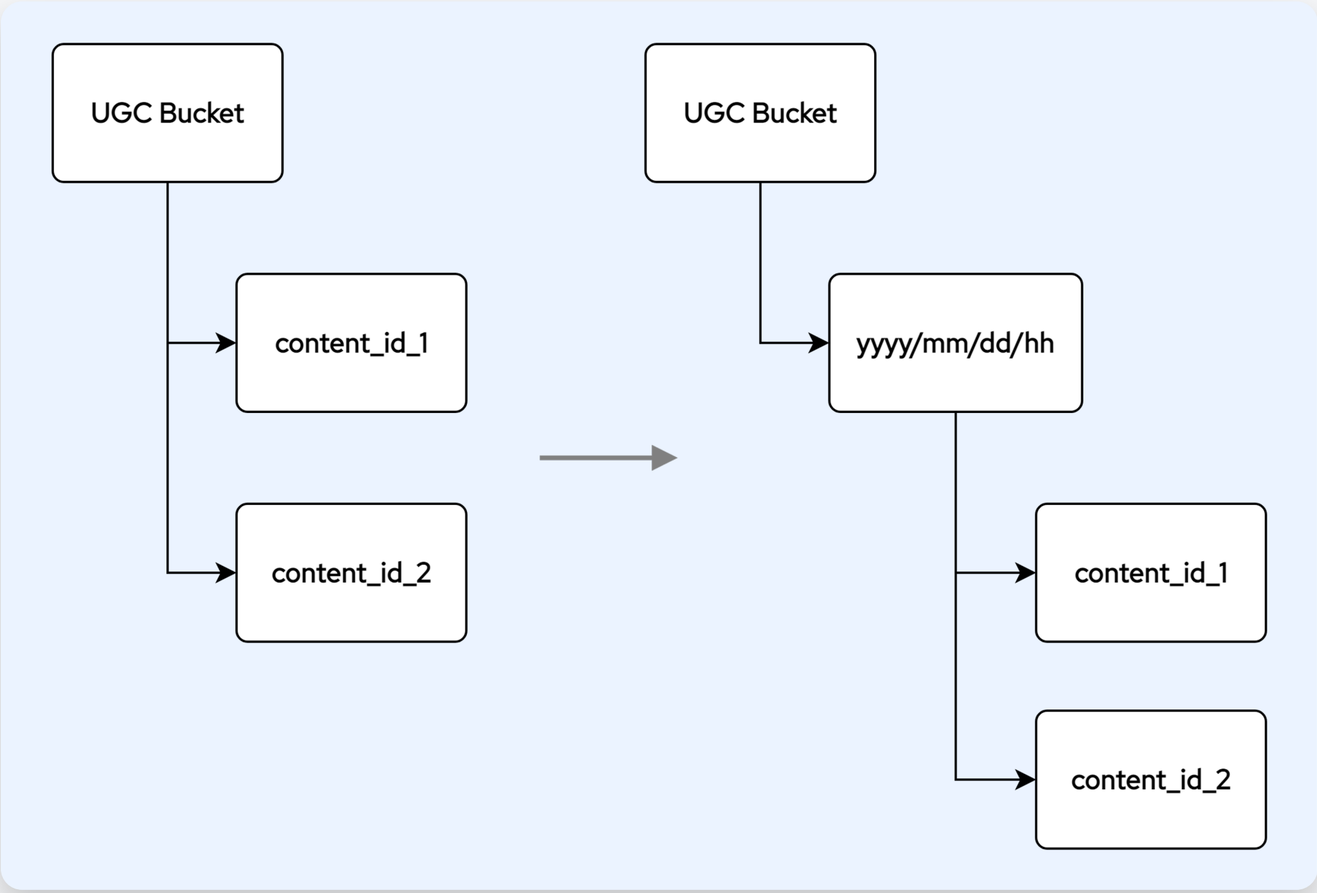
Lack of distribution

in the Storage (S3) partition

- *Up to 600K objects in a single bucket*
- *1 partition only able to handle 5K requests/second MAX*

Resolution:

- *Add storage partition by upload time*
- *Reduce the size to around 1K objects / bucket*

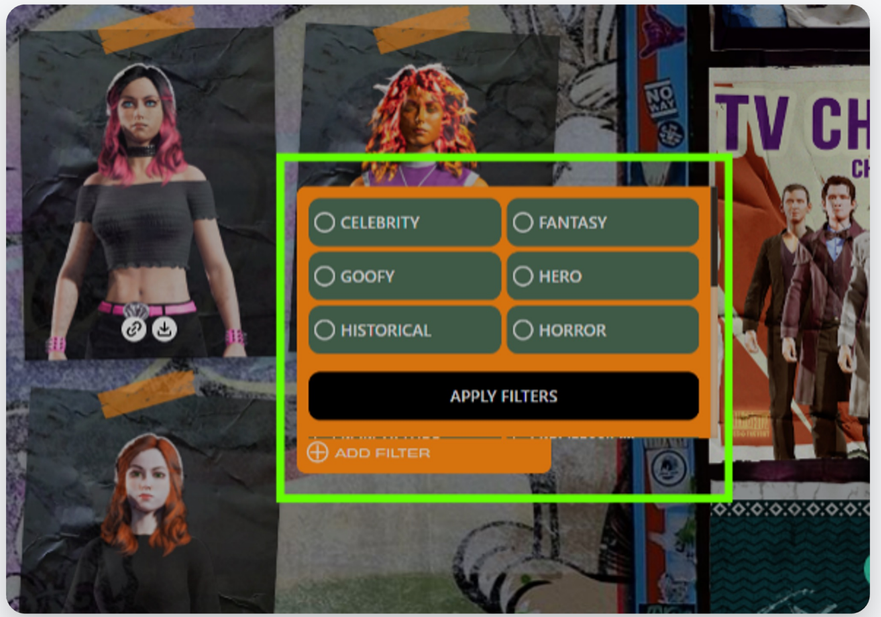


Learnings – Main Launch

Problem 3:

Too little or too many database index might cause slow queries

Tag filter is one of the most used queries. And It just makes sense to create database index based on tag.

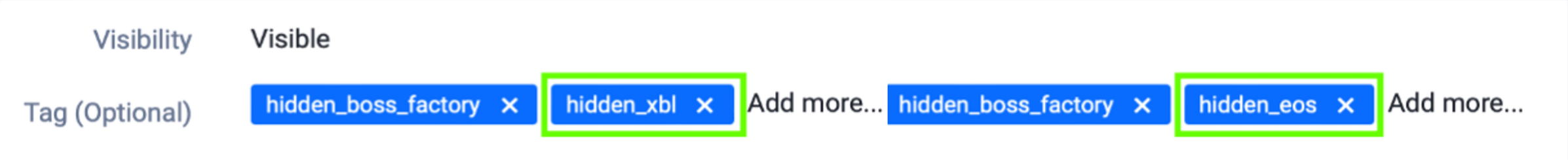


Learnings – Main Launch

Problem 3:

Too little or too many database index might cause slow queries

The tag is used for global flags to differentiate contents between platforms or builds. The query will return large number of data based on the tags.



Resolution:

- *Removing index for tag speed up the query process*

Further suggestion:

- *Implement database index based on the actual live data and implementation*

Recommendations

- ➔ *Ensure load test with real user data and game scenario to minimize the unknowns.*
- ➔ *Potentially offensive UGC is a real concern, make sure to have a player/content reporting and moderation solution in place.*
- ➔ *Avoid freeform user input for content name, description or tags to avoid moderation nightmares.*

Thank you