Perancangan Fitur *Achievement* pada Backend *Game Portal* Berbasis PlayFab dan Directus Headless CMS

PT Triple One Global
Jl Rungkut Industri Raya No.1 Blok F-11
Periode: 10 Februari – 10 Mei 2025

Oleh:

Adyuta Prajahita Murdianto 5025221186

Pembimbing Departemen
Aldinata Rizky Revanda, S.Kom., M.Kom

Pembimbing Lapangan
Novian Citantio Widodo

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika
Cerdas Institut Teknologi Sepuluh Nopember
Surabaya 2025

[Halaman ini sengaja dikosongkan]

KERJA PRAKTIK

Perancangan Fitur *Achievement* pada *Backend Game Portal* Berbasis PlayFab dan Directus Headless CMS

PT Triple One Global
Jl Rungkut Industri Raya No. 1 Blok F-11
Periode: 10 Februari – 10 Mei 2025

Oleh:

Adyuta Prajahita Murdianto 5025221186

Pembimbing Departemen
Aldinata Rizky Revanda, S.Kom., M.Kom

Pembimbing Lapangan
Novian Citantio Widodo

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika
Cerdas Institut Teknologi Sepuluh Nopember
Surabaya 2025

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN
KERJA PRAKTIK

Perancangan Fitur Achievement pada Backend Game Portal BerbasisPlayFab dan Directus Headless CMS

Oleh:

Adyuta Prajahita Murdianto 5025221186

Disetujui oleh Pembimbing Kerja Praktik:

1. Aldinata Rizky Revanda, S.Kom., M.Kom.
   NIP. 199806262024061002

(Pembimbing Departemen)

2. Novian Citantio Widodo

(Pembimbing Lapangan)

[Halaman ini sengaja dikosongkan]

Perancangan Fitur *Achievement* pada Backend Game Portal Berbasis PlayFab dan Directus Headless CMS

Nama Mahasiswa	:	Adyuta Prajahita Murdianto
NRP	:	5025221186
Departemen	:	Teknik Informatika FTEIC-ITS
Pembimbing Jurusan	:	Aldinata Rizky Revanda, S.Kom., M.Kom.
Pembimbing Lapangan	:	Novian Citantio Widodo

ABSTRAK

Laporan ini membahas perancangan, implementasi, dan evaluasi fitur *achievement* pada *backend Game Porta*l di divisi KMB Studio, PT Triple One Global, menggunakan *PlayFab* sebagai *Backend-as-a-Service* dan Directus sebagai Headless CMS. Pada bab pendahuluan dikemukakan latar belakang kebutuhan integrasi data game, tujuan kerja praktik, dan metodologi yang meliputi studi literatur, analisis & perancangan, implementasi, hingga pengujian serta evaluasi.

Pada tinjauan pustaka dijelaskan konsep *Headless CMS (Directus)*, layanan *backend game (PlayFab)*, serta teknologi pendukung seperti *JavaScript*, API, dan *Unity*. Implementasi meliputi pembuatan katalog *achievemen*t di Directus (level 1 & 2), alur logika pembaruan progress melalui *Cloud Script PlayFab*, validasi data, hingga mekanisme klaim reward. Diagram alur, dan potongan kode memperlihatkan detail proses *fetch* katalog, formatting data, *update user* data, dan *rollback error*.

Pengujian dilakukan dengan berbagai skenario, mulai dari sukses *update progress* dan *claim achievement*, hingga beberapa kasus gagal (*invalid request body*, kode tidak terdaftar, duplikasi, *over max point*, dan *rollback* saat *reward* gagal diberikan). Hasil uji menunjukkan semua kriteria terpenuhi: sistem berjalan sesuai rancangan, *error handling* efektif, dan *reward* terdistribusi dengan benar. Kesimpulan laporan menegaskan keberhasilan implementasi fitur *achievement* yang stabil dan dapat diandalkan, serta saran perbaikan meliputi optimasi performa, perluasan jenis reward, dan penambahan monitoring sistem.

Kata Kunci: Backend, Game Portal, PlayFab, Directus, Headless CMS, Achievement, Cloud Script.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT atas limpahan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan kegiatan Kerja Praktik (KP) sebagai salah satu kewajiban akademik di Departemen Informatika.

Laporan ini disusun berdasarkan pengalaman yang penulis peroleh selama melaksanakan kerja praktik. Penulis menyadari bahwa dalam proses pelaksanaan maupun penyusunan laporan ini masih terdapat kekurangan. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang bersifat membangun untuk penyempurnaan di masa mendatang. Penulis juga berharap laporan ini dapat memberikan manfaat serta menjadi bahan referensi yang berguna.

Sebagai bentuk penghargaan, penulis menyampaikan terima kasih kepada pihak-pihak yang telah membantu, baik secara langsung maupun tidak langsung, antara lain:

1. Kedua orang tua penulis.
2. Bapak Aldinata Rizky Revanda, S.Kom., M.Kom., selaku dosen pembimbing dan koordinator program Kerja Praktik.
3. Bapak Novian Citantio Widodo, selaku pembimbing lapangan.

Surabaya, 30 Mei 2025

Adyuta Prajahita Murdianto

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR KODE SUMBER

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1. Latar Belakang

Dalam era transformasi digital yang terus berkembang, industri teknologi informasi dan komunikasi (TIK) menuntut sumber daya manusia yang tidak hanya menguasai teori, tetapi juga memiliki pengalaman praktis yang relevan. Untuk menjembatani kesenjangan antara teori dan praktik, program kerja praktik (KP) menjadi bagian integral dalam kurikulum pendidikan tinggi, memberikan kesempatan bagi mahasiswa untuk terlibat langsung dalam dunia industri dan memahami dinamika kerja yang sesungguhnya.

Penulis berkesempatan melaksanakan kerja praktik di PT Triple One Global, sebuah perusahaan yang didirikan pada tahun 2011 dan dikenal sebagai penyedia layanan konten (content provider). Perusahaan ini menawarkan berbagai layanan, termasuk penyediaan konten digital, secure messaging, dan solusi komunikasi bisnis lainnya. Dengan komitmen terhadap inovasi dan integrasi teknologi komunikasi informasi terkini, PT Triple One Global bertujuan untuk menjadi mitra pilihan dalam menyediakan konten terbaik dan solusi komunikasi yang andal bagi pelanggan di Indonesia.

Selama periode kerja praktik, penulis ditempatkan di cabang Surabaya, tepatnya pada divisi KMB Studio, yang berfokus pada pengembangan game. Dalam divisi ini, penulis berperan sebagai Backend Developer, bertanggung jawab dalam merancang dan mengembangkan fitur *achievement* pada backend dari sistem Game Portal, sebuah aplikasi terpusat yang menyediakan berbagai game dengan data yang terintegrasi.

Dalam proyek ini, penulis menggunakan teknologi Playfab dan Directus. Playfab digunakan sebagai *backend as a service* untuk mengelola data game, sementara Directus digunakan sebagai *headless* CMS yang memungkinkan pengelolaan konten secara fleksibel. Penulis bertanggung jawab dalam

pengembangan fitur *achievement*, yang mengintegrasikan Playfab dan Directus untuk memberikan penghargaan kepada pemain berdasarkan pencapaian tertentu. Selain itu, penulis juga mengerjakan beberapa *task* lain diluar project ini, yakni *fixing* beberapa *bug* pada aplikasi lain yang dirancang oleh perusahaan. Namun pada laporan kali ini, penulis akan fokus untuk membahas perancangan sistem *achievement* yang dimana merupakan task paling besar yang dikerjakan oleh penulis.

Melalui keterlibatan dalam proyek ini, penulis memperoleh pengalaman berharga dalam pengembangan backend aplikasi game, integrasi layanan pihak ketiga, serta pengelolaan konten dinamis. Pengalaman ini memberikan pemahaman yang lebih mendalam tentang praktik industri dalam pengembangan perangkat lunak, khususnya dalam konteks pengembangan game.

Oleh karena itu, laporan ini disusun untuk mendokumentasikan kegiatan selama kerja praktik, serta merefleksikan pembelajaran dan keterampilan yang diperoleh, dengan harapan dapat memberikan kontribusi positif bagi pengembangan diri penulis dan sebagai referensi bagi pihak-pihak yang berkepentingan.

1.2. Tujuan

Pelaksanaan kerja praktik ini bertujuan untuk memenuhi salah satu syarat akademik dalam program studi di Institut Teknologi Sepuluh Nopember, yakni menyelesaikan mata kuliah Kerja Praktik dengan beban empat SKS. Selain itu, tujuan lain dari kerja praktik ini adalah untuk :

1. Merancang katalog *achievement* di Directus untuk mendukung permintaan rancangan alur *achievement* yang telah ditentukan
2. Mengembangkan sistem fitur *achievement* dalam proyek Game Portal, yang dirancang untuk menjadi platform terintegrasi bagi berbagai jenis permainan digital
3. Merancang mekanisme validasi data dan penanganan error saat pembaruan *progress achievement*

1.3. Manfaat

Manfaat yang diharapkan dari pelaksanaan kerja praktik ini meliputi:

1.3.1 Bagi Penulis

- Memperoleh pengalaman langsung dalam pengembangan backend aplikasi game menggunakan teknologi terkini seperti Playfab dan Directus.
- Meningkatkan pemahaman dan keterampilan teknis dalam pengelolaan backend dan integrasi sistem.
- Menjalin pengalaman kerja di lingkungan industri yang sesungguhnya sebagai bekal karier professional.

1.3.2 Bagi Perusahaan

- Mendapatkan dukungan dalam pengembangan fitur achievement yang saat penulis pertama kali masuk, masih belum dibangun.
- Memperoleh dokumentasi hasil kerja praktik yang dapat digunakan sebagai referensi pengembangan lebih lanjut.

1.3.3 Bagi Institusi Teknologi Sepuluh Nopember

- Memperkuat hubungan kerja sama antara institusi dengan dunia industri melalui program kerja praktik.
- Menjadi bahan evaluasi dan pembelajaran bagi mahasiswa lainnya terkait aplikasi ilmu di dunia kerja.

1.4. Rumusan Masalah

Berikut rumusan masalah yang akan dibahas dalam laporan ini:

1. Bagaimana merancang katalog *achievement* di Directus untuk mendukung permintaan rancangan alur *achievement* yang telah ditentukan?
2. Bagaimana proses pengembangan fitur *achievement* pada backend Game Portal menggunakan PlayFab dan Directus?
3. Bagaimana mekanisme validasi data dan penanganan error saat pembaruan progress *achievement*?

1.5. Lokasi dan Waktu Kerja Praktik

Kerja praktik ini dilaksanakan pada waktu dan tempat sebagai berikut:

Lokasi : Jl. Rungkut Industri Raya No. 1 Blok F-11
Waktu : 10 Februari – 10 Mei 2025
Hari Kerja : Senin – Jumat (10 Februari – 23 Februari 2025), Rabu dan Jumat (24 Februari – 10 Mei 2025)
Jam Kerja : 8 Jam/Hari (9-5)

1.6. Metodologi Kerja Praktik

1.6.1 Perumusan Masalah

Pada tahap ini, dilakukan identifikasi terhadap kebutuhan dan permasalahan yang akan diselesaikan selama kerja praktik. Fokus utama diarahkan pada pengembangan fitur *achievement* platform Game Portal, serta integrasi dengan teknologi yang digunakan, yaitu Playfab dan Directus.

1.6.2 Studi Literatur

Selanjutnya penulis melakukan pencarian dan kajian terhadap sumber-sumber literatur yang relevan, baik berupa dokumentasi resmi, artikel teknis, maupun referensi akademik. Studi ini bertujuan untuk memahami arsitektur Playfab, konsep *headless CMS* seperti Directus, serta praktik terbaik dalam pengembangan backend sistem game.

1.6.3 Analisis dan Perancangan

Tahap ini mencakup analisis kebutuhan sistem serta perancangan struktur backend yang mendukung fitur-fitur yang akan dikembangkan. Desain alur data, struktur koleksi dalam Directus, endpoint API, serta logika pengelolaan *achievement* pada Playfab dirancang agar sesuai dengan kebutuhan proyek.

1.6.4 Implementasi Sistem

Setelah rancangan disetujui, penulis mulai melakukan implementasi secara bertahap dalam pengembangan sistem *achievement* pada backend Game Portal.

1.6.5 Pengujian dan Evaluasi

Pengujian dilakukan untuk memastikan bahwa sistem berjalan sesuai dengan rancangan dan bebas dari kesalahan fungsional. Evaluasi dilakukan bersama tim senior yang berada disana.

1.6.6 Kesimpulan dan Saran

Pada bagian ini, dipaparkan kesimpulan yang dapat diambil dan juga saran dalam pengerjaan kerja praktik.

1.7 Sistematika Laporan

Laporan kerja praktik ini terdiri dari tujuh bab dengan rincian sebagai berikut:

1.7.1 Bab I Pendahuluan

Bab ini memuat uraian mengenai latar belakang, perumusan tujuan, waktu pelaksanaan kerja praktik, serta sistematika dalam pelaksanaan dan penulisan laporan kerja praktik.

1.7.2 Bab II Profil Perusahaan

Bab ini menjelaskan secara rinci mengenai profil PT Triple One Global, perusahaan tempat penulis melaksanakan kerja praktik.

1.7.3 Bab III Tinjauan Pustaka

Bab ini menyajikan referensi dan dasar teori yang digunakan sebagai acuan dalam pelaksanaan kerja praktik.

1.7.4 Bab IV Implementasi Sistem

Bab ini menjelaskan tahapan-tahapan implementasi yang dilakukan penulis selama kerja praktik, termasuk proses pengembangan fitur *achievement* pada backend Game Portal.

1.7.5 Bab V Pengujian dan Evaluasi

Pada bab ini, dijelaskan tentang hasil pengujian dan evaluasi dari fitur *achievement* sudah penulis buat.

1.7.6 Bab VI Kesimpulan dan Saran

Pada bab ini, akan dipaparkan kesimpulan yang dapat diambil dan juga saran selama pengerjaan kerja praktik.

[Halaman ini sengaja dikosongkan]

BAB II

PROFIL PERUSAHAAN

2.1. Profil PT Triple One Global

PT Triple One Global adalah perusahaan penyedia layanan telekomunikasi dan teknologi informasi yang didirikan pada Mei 2011. Perusahaan ini menyediakan berbagai layanan, termasuk Application-to-Person (A2P) messaging, platform digital, dan penyedia konten, dengan komitmen terhadap peningkatan berkelanjutan dan penyediaan layanan digital terbaik di setiap negara tempat mereka beroperasi.

PT Triple One Global melayani berbagai klien dari sektor korporat dan institusi, termasuk operator telekomunikasi besar seperti Indosat, Telkomsel, XL, Smartfren, dan Tri. Selain itu, perusahaan ini juga menjalin kemitraan dengan berbagai perusahaan ternama seperti Agung Podomoro Group, KFC, Grab, Bank Indonesia, PSSI, dan Djarum.

Dalam operasionalnya, PT Triple One Global memiliki kantor pusat yang berlokasi di Jalan Aren No.29, Kelurahan Jati Pulo, Kecamatan Palmerah, Jakarta Barat, DKI Jakarta 11430. Perusahaan ini juga memiliki kantor cabang di Ruko Section One, Jl. Rungkut Industri Raya No.1 Blok F-11, Kelurahan Kendangsari, Kecamatan Tenggilis Mejoyo, Surabaya, Jawa Timur. Di cabang inilah penulis melaksanakan kerja praktiknya.

Cabang Surabaya memiliki suatu devisi bernama KMB Studio yang bergerak dibidang pembuatan game. Pada saat penulis melaksanakan kerja praktik, project ini sedang mengembangkan sebuah platform portal game yang memungkinkan beberapa game dapat saling berintegrasi data, sehingga data-data player dapat di Kelola secara terpusat. Dalam sebagian besar masa kerja praktiknya, penulis ikut andil dalam membangun sistem *achievement* pada backend Game Portal yang dirancang oleh KMB Studio.

2.2. Logo Perusahaan

Gambar 2.1. Logo PT Triple One Global

Sumber: https://www.linkedin.com/company/triple-one-global-indonesia/posts/?feedView=all

2.3. Visi Misi Perusahaan

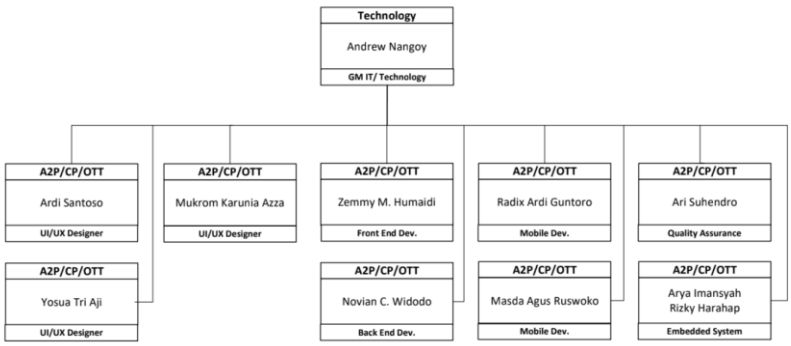
2.3.1. Visi

Menjadi mitra pilihan dalam menyediakan konten terbaik, dengan kepuasan pengguna sebagai tujuan utama.

2.3.2. Misi

- Mengubah bisnis secara digital melalui inovasi dan integrasi produk serta layanan teknologi komunikasi informasi terkini.
- Memberikan layanan tepat waktu, sesuai anggaran, dan dengan standar kualitas tertinggi di industri is penyedia konten.
- Memastikan kepuasan pelanggan yang berkelanjutan serta kemakmuran bagi karyawan dan perusahaan.

2.4. Struktur Organisasi



Gambar 2.2. Struktur organisasi PT Triple One Global (Devisi IT)

Sumber: HRD PT Triple One Global

BAB III
TINJAUAN PUSTAKA

Pada bab ini, akan dijelaskan mengenai dasar teori yang digunakan selama proses kerja praktik.

3.1. Headless CMS

Headless CMS adalah sistem manajemen konten yang memisahkan antara penyimpanan konten (back-end) dan penyajian konten (front-end). Dalam arsitektur ini, konten disimpan dalam repositori dan disajikan melalui API ke berbagai platform, seperti website, aplikasi mobile, atau perangkat IoT.

Keunggulan utama dari Headless CMS adalah fleksibilitas dalam pengembangan frontend, memungkinkan penggunaan berbagai teknologi dan framework tanpa terikat pada struktur backend tertentu. Selain itu, pendekatan ini mendukung distribusi konten secara omnichannel, mempercepat waktu pemuatan halaman melalui integrasi dengan Static Site Generators (SSG), serta meningkatkan skalabilitas dan keamanan sistem secara keseluruhan.

3.2. Directus

Directus adalah platform open-source yang mengubah basis data SQL menjadi Headless CMS dengan menyediakan antarmuka pengguna yang intuitif dan API instan (REST dan GraphQL). Platform ini memungkinkan pengelolaan konten tanpa perlu menulis kode, mendukung berbagai jenis basis data seperti MySQL, PostgreSQL, dan SQLite, serta dapat dihosting secara mandiri atau di cloud. Dengan Directus, misalnya, tim pengembang game dapat membuat koleksi “Achievements” di database, mendefinisikan field seperti nama, deskripsi, dan tipe reward, lalu Directus akan menyediakan API untuk mengakses data *achievement* tersebut. Data (item) yang diunggah atau diubah melalui panel admin Directus langsung tercermin di API, dan dapat diambil oleh aplikasi game atau Playfab (seperti yang akan penulis gunakan) menggunakan metode hit API seperti pada umumnya.

Perkembangan terbaru Directus menunjukkan penambahan fitur-fitur canggih di versi 11 (rilis 2024–2025). Di antaranya modul Visual Editor (beta) yang memungkinkan penyuntingan konten langsung di halaman web, integrasi dengan Elasticsearch/Algolia untuk pencarian, plugin SEO, dan komponen AI chat pada editor. Directus juga terus mengikuti perkembangan platform lain: misalnya, versi terbaru sudah mendukung Node.js 22. Dengan demikian, Directus saat ini menjadi platform full-stack yang mengotomatiskan penyediaan backend siap pakai (autogenerasi API, otentikasi, interface admin), dan terbukti diadopsi di berbagai proyek skala besar (seperti aplikasi Weber Grill yang melayani jutaan sesi pengguna).

Keunggulan Directus terletak pada kemampuannya untuk beradaptasi dengan kebutuhan proyek yang kompleks, memberikan fleksibilitas dalam pengembangan frontend, dan memastikan keamanan data melalui control akses yang ketat.

3.3. Playfab

PlayFab adalah platform layanan backend untuk pengembangan dan pengoperasian game secara langsung (live). Disediakan oleh Microsoft Azure, PlayFab menawarkan layanan seperti server multiplayer, analitik real-time, dan alat operasi langsung (LiveOps) yang membantu pengembang dalam meluncurkan game lebih cepat, memperpanjang siklus hidup game, dan mengurangi biaya operasional. Sebagai contoh, PlayFab memungkinkan developer menyimpan data progress pemain, membuat leaderboard global, dan mengelola pencapaian (*achievement*) serta reward secara terpusat. PlayFab banyak diadopsi oleh industri game; misalnya, lebih dari 2,5 miliar akun pemain telah didukung di lebih dari 5000 game menggunakan PlayFab. PlayFab menyediakan SDK resmi (termasuk SDK C# untuk Unity) dan API REST, memudahkan integrasi ke dalam game engine.

3.4. Javascript

JavaScript adalah bahasa pemrograman tingkat tinggi yang awalnya dikembangkan oleh Brendan Eich di Netscape pada tahun 1995. Dirancang untuk meningkatkan interaktivitas halaman web, JavaScript memungkinkan manipulasi elemen HTML secara dinamis, pengelolaan peristiwa pengguna, dan komunikasi

asynchronous dengan server melalui teknologi seperti AJAX. Seiring perkembangan teknologi, JavaScript telah berevolusi menjadi bahasa pemrograman yang dapat digunakan di sisi klien maupun server, dengan dukungan dari berbagai framework dan pustaka seperti Node.js, React, dan Angular. Standarisasi JavaScript dilakukan oleh ECMA International melalui spesifikasi ECMAScript, yang terus diperbarui untuk mencerminkan kebutuhan dan praktik terbaik dalam pengembangan web modern.

3.5. API

API (Application Programming Interface) adalah antarmuka yang memungkinkan dua aplikasi perangkat lunak untuk saling berkomunikasi dengan cara yang terstruktur melalui serangkaian ptotokol dan definisi. API bekerja seperti perantara yang menerima permintaan (request) dari satu sistem, kemudian mengirimkan permintaan tersebut ke sistem lain, dan akhirnya mengembalikan respons kepada pemohon.

API sangat penting dalam pengembangan aplikasi modern karena memungkinkan pengembang untuk mengakses data dan layanan dari aplikasi atau sistem lain tanpa harus mengetahui bagaimana sistem tersebut bekerja secara internal. Sebagai contoh, aplikasi cuaca di ponsel tidak menyimpan data cuaca itu sendiri, melainkan mengakses data dari sistem pihak ketiga seperti BMKG atau layanan cuaca lainnya melalui API.

3.6. Unity

Unity adalah sebuah *game engine* lintas platform yang dikembangkan oleh Unity Technologies. Pertama kali dirilis pada tahun 2005, Unity telah menjadi salah satu platform pengembangan game paling populer di dunia. Unity memungkinkan pengembang untuk membuat game dan aplikasi interaktif dalam format 2D, 3D, Virtual Reality (VR), dan Augmented Reality (AR) yang dapat dijalankan di berbagai platform seperti Windows, macOS, Linux, iOS, Android, serta konsol seperti PlayStation, Xbox, dan Nintendo Switch.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI SISTEM

Pada bab ini akan menjelaskan tahap implementasi yang dilakukan dalam merancang sistem *achievement* pada backend Game Portal.

4.1. Alur Sistem

Pada bagian ini kita akan membahas mengenai alur sistem *achievement* yang sudah dibuat oleh penulis secara sederhana. Harapannya, sistem ini akan selalu memperbarui *achievement progress* dari *player*. Setiap kali *player* telah membuat progress tertentu ke suatu *achievement*, game yang berbasis unity akan mengirimkan request ke Playfab untuk mengupdate progress *player* pada *achievement* tersebut. Namun, Playfab juga akan melakukan pengecekan terlebih dahulu terhadap katalog *achievement* yang berada di Directus.

Gambar 4.1. Alur proses update progress *achievement*

Sumber: Dokumentasi Penulis

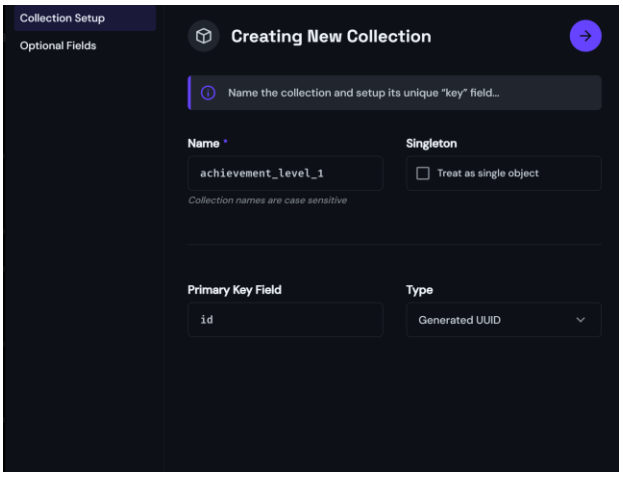
4.2. Katalog *Achievement*

Katalog *achievement* digunakan sebagai acuan Playfab dalam memperbarui progress *achievement* dari seorang player. Disini kita akan menggunakan directus untuk menyimpan data achievement yang akan disediakan.

Achievement sendiri terdiri dua level, yang dimana level pertama adalah sebagai parent *achievement* dan level kedua merupakan *achievement-achievement* yang harus diselesaikan untuk menyelesaikan suatu parent *achievement.*

Setiap collection didalam Directus dapat dibuat melalui Setting -> Data Model. Lalu kita dapat menekan tombol + yang terdapat dibagian pojok kanan atas

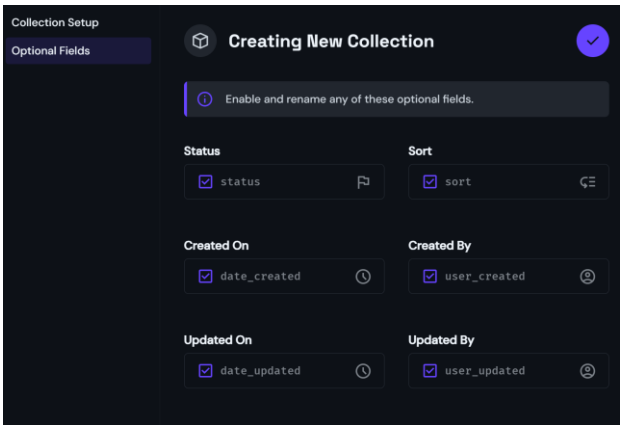
untuk membuat Data Model baru. Pertama disini penulis membuat entity untuk *achievement l*evel 1.



Gambar 4.2. Collection Setup *achievement_level_1*

Sumber: Dokumentasi Penulis

Untuk primary key dari collection tersebut penulis memutuskan untuk menggunakan UUID yang merupakan string unik yang tidak akan pernah sama setiap kali di *generate*.



Gambar 4.3. Optional Fields *achievement_level_1*

Sumber: Dokumentasi Penulis

Disini penulis juga mengaktifkan opsi Optional Fields yang diantaranya ada Status (untuk melacak status publikasi data), Sort (untuk otomatis mengurutkan data berdasarkan waktu pembuatan), Created On (untuk melacak kapan suatu data terbuat), Created By (untuk melacak siapa yang membuat data), Updated On (untuk

melacak kapan suatu data diubah), dan Updated By (untuk melacak siapa yang mengubah suatu data). Setelah itu, untuk membuat collection yang telah di setup, tekan tombol panan di kanan atas.

Untuk menambahkan *field*, kita bisa membuka Data Model yang sudah kita buat, dalam kasus ini *achievement*_level_1. Lalu kita bisa menekan tombol *Create Field*. Disini kita bisa memilih tipe field apa saja yang ingin kita buat. Berikut penjelasan setiap tipe *field* yang penulis pilih beserta *field* apa saja yang menggunakan tipe tersebut,

1. Tipe Input

Tipe ini adalah tipe paling sederhana, yang dimana data yang disimpan hanya berupa sebuah text. Berikut *field* yang menggunakan tipe tersebut,

- Name: Menyimpan nama setiap *achievement*, tipe data integer.
- Code: Code unik yang dimana akan menyimpan nama game dengan huruf kecil dan "_" sebagai pengganti spasi, digunakan agar program game tau ingin update *achievement* yang mana, tipe data string.
- Level: Menyimpan informasi level *achievement,* tipe data integer.
- Description: Menyimpan deskripsi untuk menyelesaikan *achievement*.
- Max Point: Menyimpan batas *progress* maksimal dari setiap *achievement* agar Playfab berhenti mengupdate *progress achievement* dari *player* ketika *player* sudah menyelesaikan achievement tersebut, tipe data integer.

2. Tipe Many to One

Merupakan tipe *field* yang menghubungkan *relationship* dari satu atau lebih data koleksi saat ini (achievement_level_1) dengan salah satu data pada koleksi target. Kolom ini akan menyimpan *primary key* dari salah satu data yang dipilih di koleksi target. Berikut *field* yang menggunakan tipe ini,

- Game_ID: menyimpan id dari salah satu game yang dipilih, sebagai identifikasi *achievement* merupakan *achievement* dari game apa. Disini satu game bisa memiliki banyak *achievement.*

3. Image

Merupakan tipe *field* untuk menyimpan data berupa gambar. Berikut *field* yang menggunakan tipe ini,

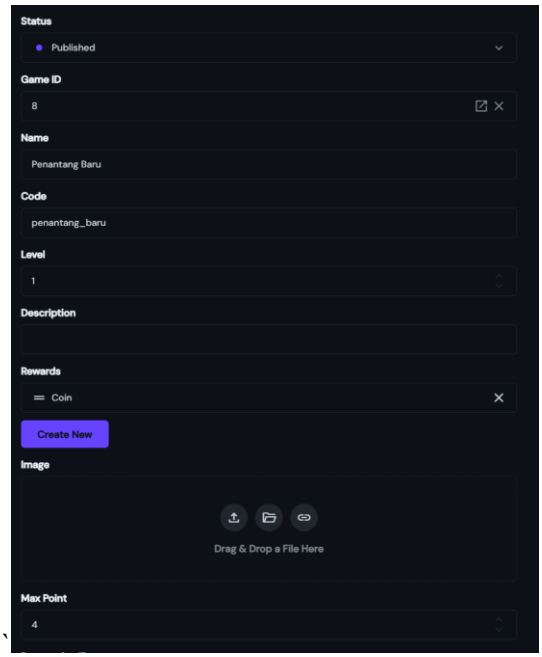
- Image: Menyimpan gambar dari *achievement*.

4. Repeater

Tipe *field* ini unik. Dengan menggunakan tipe ini, kita bisa menyimpan satu atau *sub-field* tanpa menggunakan relasi *One-to-Many.* Berikut *field* yang menggunakan tipe ini,

- Rewards: Menyimpan beberapa hadiah yang akan diberikan kepada *player* ketika berhasil menyelesaikan *achievement. Sub-field* yang ada pada field ini diantara lain ada *name* (nama hadiah), *type* (tipe hadiah), *value* (jumlah hadiah), dan *code* (kode identifikasi yang merujuk kepada kode barang yang akan diberikan pada *player* yang berada pada *playfab*).

Berikut adalah contoh dari salah satu data yang dibuat pada koleksi *achievement_level_1,*

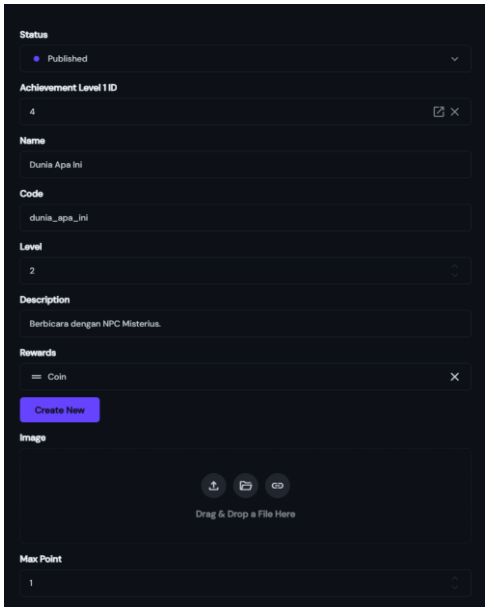


Gambar 4.4. Contoh data pada koleksi *achievement*_level_1

Sumber: Dokumentasi Penulis

Untuk *achievement*_level_2 memiliki struktur yang mirip dengan *achievement*_level_1. Namun, disini alih-alih menggunakan game_id, kita akan menggunaakn *achievement*_level_1_id yang memiliki tipe many to one yang menghubungkan banyak *achievement*_level_2 dengan satu *achievement*_level_1

sebagai parent *achievement.* Berikut contoh salah satu data yang dibuat pada koleksi *achievement*_level_2,



Gambar 4.5. Contoh data pada koleksi *achievement*_level_2

Sumber: Dokumentasi Penulis

Berikut adalah daftar *achievement* yang diberikan oleh *senior* penulis untuk diinputkan kedalam katalog yang sudah dibuat dengan Directus,

Tabel 4.1. Daftar *achievement* game Sapi Kebluk

Nama *Achievement*	**Syarat *Achievement***	**Reward**	**Group *Achievement***	**Group Reward**
sapi_pemula	Berbicara dengan NPC Penjaga Sapi.	20 Koin	jagoan_sapi	Sapi Kebluk (Pet) & 10 Gem
lompatan_pertama	Melompat sebanyak 1x.	10 Koin	jagoan_pemula	100 Koin
keluarkan_amarahmu	Melompat (suara keras) sampai mentok ke atas.	10 Koin		
aku_pemalu	Melompat dengan menggunakan suara yang sangat rendah.	10 koin		

Nama *Achievement*	Syarat *Achievement*	Reward	Group *Achievement*	Group Reward
hati_hati_dijalan	Kena obstacle 1x.	10 Koin		
pelompat_handal	Melewati rintangan sebanyak 7x tanpa gagal	10 koin		
aku_kaya	Mendapatkan coin 1x.	10 Koin		
kolektor_koin	Mendapatkan coin sebanyak 10x dalam 1 lintasan.	10 Koin	jagoan_ menengah	100 Koin
aku_kalah	Kalah sebanyak 3x.	10 Koin		
aku_hebat	Bertahan selama 2 menit dalam 1x main.	10 Koin		

Sumber: Dokumentasi Penulis

Tabel 4.2. Daftar *achievement* game Acade Arena

Nama *Achievement*	Syarat *Achievement*	Reward	Group *Achievement*	Group Reward
dunia_apa_ini	Berbicara dengan NPC Misterius.	10 Koin	penantang_ baru	50 Koin
kamu_pelupa	Berbicara dengan NPC sebanyak 10x.	10 Koin		
kamu_kepo	Berbicara dengan NPC sebanyak 20x.	10 Koin		
kamu_bawel	Berbicara dengan NPC sebanyak 30x.	10 Koin		
duit_siapa_ini	Mengambil koin tersembunyi di map.	10 Koin	penjelajah_ baru	50 Koin
portal_ke_dunia_lain	Memasuki portal pertama kali.	10 Koin		
awalan_baik	Memainkan salah satu mini game.	10 Koin		

Nama *Achievement*	Syarat *Achievement*	Reward	Group *Achievement*	Group Reward
menjadi_turis	Mengeliling seluruh arena.	10 Koin		
raja_arena	Menjadi peringkat 1 di Leaderboard.	10 Koin	puncak_ kejayaan	100 Koin
tak_terkalahkan	Menjadi peringkat 1 selama 7 hari berturut-turut.	10 Koin		
ksatria_arena	Mencapai 50 besar di Leaderboard.	10 Koin		
penantang_ penghabisan	Berhasil naik ke peringkat 1 dalam waktu 24 jam.	10 Koin		

Sumber: Dokumentasi Penulis

Tabel 4.3. Daftar *achievement* game Tali Tantangan

Nama *Achievement*	Syarat *Achievement*	Reward	Group *Achievement*	Group Reward
sang_penantang	Berbicara dengan NPC anak kecil.	10 Koin	penantang_tali	10 Koin
penerus_tradisi_1 penerus_tradisi_2	Bermain sebanyak sekali Bermain sebanyak 10 kali	5 Koin 10 Koin	penerus_tradisi	50 Koin
penerus_tradisi_3	Bermain sebanyak 25 kali	15 Koin		
penerus_tradisi_4	Bermain sebanyak 50 kali	25 Koin		
penerus_tradisi_5	Bermain sebanyak 100 kali	50 Koin		
batu_lompatan_1	Berhasil melakukan 10 kali lompatan	5 Koin	batu_lompatan	100 Koin
batu_lompatan_2	Berhasil melakukan 25 kali lompatan	10 Koin		

Nama *Achievement*	Syarat *Achievement*	Reward	Group *Achievement*	Group Reward
batu_lompatan_3	Berhasil melakukan 50 kali lompatan	15 Koin		
batu_lompatan_4	Berhasil melakukan 75 kali lompatan	25 Koin		
batu_lompatan_5	Berhasil melakukan 100 kali lompatan	50 Koin		
lompatan_katak_1	Berhasil melakukan 10 kali lompatan berturut-turut	5 Koin	lompatan_katak	100 koin
lompatan_katak_2	Berhasil melakukan 20 kali lompatan berturut-turut	10 Koin		
lompatan_katak_3	Berhasil melakukan 30 kali lompatan berturut-turut	15 Koin		
lompatan_katak_4	Berhasil melakukan 40 kali lompatan berturut-turut	25 Koin		
lompatan_katak_5	Berhasil melakukan 50 kali lompatan berturut-turut	50 Koin		
salah_tekan_1	Salah menekan arah sebanyak sekali	10 Koin	si_kikuk	20 koin
salah_tekan_2	Salah menekan arah sebanyak 5 kali	10 Koin		
salah_tekan_3	Salah menekan arah sebanyak 10 kali	10 Koin		
kesempatan_kedua_1	Hidup kembali sebanyak sekali	10 Koin	demi_tradisi	20 koin
kesempatan_kedua_2	Hidup kembali sebanyak 5 kali	10 Koin		
kesempatan_kedua_3	Hidup kembali sebanyak 10 kali	10 Koin		

Nama *Achievement*	Syarat *Achievement*	Reward	Group *Achievement*	Group Reward
jangan_menyerah	Kalah sebanyak 100 kali	50 Koin	berhati_kuat	50 koin

Sumber: Dokumentasi Penulis

Tabel 4.4. Daftar *achievement* game Mandor Kue

Nama *Achievement*	Syarat *Achievement*	Reward	Group *Achievement*	Group Reward
menara_kue	Berbicara dengan NPC Kue Jahe	20 Koin	sayembra_kue	10 Koin
penantang_ sayembara_1	Bermain sebanyak sekali	5 Koin	penantang_sayembara	50 Koin
penantang_ sayembara_2	Bermain sebanyak 10 kali	10 Koin		
penantang_ sayembara_3	Bermain sebanyak 25 kali	15 Koin		
penantang_ sayembara_4	Bermain sebanyak 50 kali	25 Koin		
penantang_ sayembara_5	Bermain sebanyak 100 kali	50 Koin		
menara_ penjaga	Berhasil membuat 10 tingkat kue	5 Koin	menara_ tertinggi	100 Koin
menara_listrik	Berhasil membuat 20 tingkat kue	10 Koin		
menara_pencakar_ langit	Berhasil membuat 50 tingkat kue	15 Koin		
menara_babel	Berhasil membuat 75 tingkat kue	25 Koin		
menara_ angkasa	Berhasil membuat 100 tingkat kue	50 Koin		

Nama *Achievement*	Syarat *Achievement*	Reward	Group *Achievement*	Group Reward
pekerja_keras_1	Berhasil membuat 10 tingkat kue berturut-turut	5 Koin	pekerja_keras	100 Koin
pekerja_keras_2	Berhasil membuat 20 tingkat kue berturut-turut	10 Koin		
pekerja_keras_3	Berhasil membuat 30 tingkat kue berturut-turut	15 Koin		
pekerja_keras_4	Berhasil membuat 40 tingkat kue berturut-turut	25 Koin		
pekerja_keras_5	Berhasil membuat 50 tingkat kue berturut-turut	50 Koin		
arsitektur_asimetris	Menata 3 kue yang tidak simetris satu sama lain	10 Koin	tidak_terprediksi	10 Koin
awas_kepala	Tertabrak bangunan selama satu kali	5 Koin	resiko_pekerja	25 Koin
jaga_keselamatan	Tertabrak bangunan sebanyak 5 kali	5 Koin		
bahaya_jatuh	Tertabrak bangunan sebanyak 10 kali	5 Koin		
variabel_tak_terduga	Tertabrak burung/angin sebanyak satu kali	5 Koin		
kecelakaan_dari_alam	Tertabrak burung/angin sebanyak 5 kali	5 Koin		
rawan_kecelakaan	Tertabrak burung/angin sebanyak 10 kali	5 Koin		
bangkit_dari_reremp ahan_1	Hidup kembali sebanyak sekali	10 Koin	kue_yang_panta ng_menyerah	20 Koin

Nama *Achievement*	Syarat *Achievement*	Reward	Group *Achievement*	Group Reward
bangkit_dari_rerempahan_2	Hidup kembali sebanyak 5 kali	10 Koin		
bangkit_dari_rerempahan_3	Hidup kembali sebanyak 10 kali	10 Koin		
lalai_dalam_bekerja	Kalah sebanyak 100 kali	25 Koin	kelalaian_sang_mandor	25 Koin

Sumber: Dokumentasi Penulis

Tabel 4.5. Daftar *achievement* game Giring Sapi

Nama *Achievement*	Syarat *Achievement*	Reward	Group *Achievement*	Group Reward
sang_penggiring	Berbicara dengan NPCPenggembala	20 Koin	giring_sapi_GA	10 Koin
penggiring_sapi_1	Bermain sebanyak sekali	5 Koin	penggiring_tetap	50 Koin
penggiring_sapi_2	Bermain sebanyak 10 kali	10 Koin		
penggiring_sapi_3	Bermain sebanyak 25 kali	15 Koin		
penggiring_sapi_4	Bermain sebanyak 50 kali	25 Koin		
penggiring_sapi_5	Bermain sebanyak 100 kali	50 Koin		
gembala_berpengalaman_1	Berhasil menggiring sejauh 10 langkah	5 Koin	gembala_berpengalaman	100 Koin
gembala_berpengalaman_2	Berhasil menggiring sejauh 20 langkah	10 Koin		
gembala_berpengalaman_3	Berhasil menggiring sejauh 50 langkah	15 Koin		

Nama *Achievement*	Syarat *Achievement*	Reward	Group *Achievement*	Group Reward
gembala_berpengalaman_4	Berhasil menggiring sejauh 75 langkah	25 Koin		
gembala_berpengalaman_5	Berhasil menggiring sejauh 100 langkah	50 Koin		
sahabat_sapi_1	Berhasil menggiring sapi selama 10 langkah berturut-turut	5 Koin	si_sahabat_sapi	100 Koin
sahabat_sapi_2	Berhasil menggiring sapi selama 20 langkah berturut-turut	10 Koin		
sahabat_sapi_3	Berhasil menggiring sapi selama 50 langkah berturut-turut	15 Koin		
sahabat_sapi_4	Berhasil menggiring sapi selama 75 langkah berturut-turut	20 Koin		
sahabat_sapi_5	Berhasil menggiring sapi selama 100 langkah berturut-turut	50 Koin		
tidak_fokus	Salah tekan QTE sebanyak 3 kali dalam satu permainan	10 Koin	kurang_minum	10 Koin
penggiring_yang_gagal_1	Sapi marah sebanyak sekali	5 Koin	gembala_kikuk	20 Koin
penggiring_yang_gagal_2	Sapi marah sebanyak 5 kali	5 Koin		
penggiring_yang_gagal_3	Sapi marah sebanyak 10 kali	5 Koin		

Nama *Achievement*	Syarat *Achievement*	Reward	Group *Achievement*	Group Reward
serangan_25ngags_1	Tertabrak angsa atau ayam sebanyak satu kali	5 Koin		
serangan_unggas_2	Tertabrak angsa atau ayam sebanyak 5 kali	5 Koin		
serangan_unggas_3	Tertabrak angsa atau ayam sebanyak 10 kali	5 Koin		
penggiring_gigih_1	Hidup kembali sebanyak sekali	10 Koin	penggiring_ gigih	20 Koin
penggiring_gigih_2	Hidup kembali sebanyak 5 kali	10 Koin		
penggiring_gigih_3	Hidup kembali sebanyak 10 kali	10 Koin		
gembala_kewalahan	Kalah sebanyak 100 kali	25 Koin	gembala_kebing ungan	25 Koin

Sumber: Dokumentasi Penulis

Tabel 4.6. Daftar *achievement* game Puncak Keberanian

Nama *Achievement*	Syarat *Achievement*	Reward	Group *Achievement*	Group Reward
menuju_puncak	Berbicara dengan NPC Anak Kecil Pemberani	20 Koin	si_pemberani	10 Koin
pemanjat_berulang_1	Bermain sebanyak sekali	5 Koin	pemanjat_ berulang	50 Koin
pemanjat_berulang_2	Bermain sebanyak 10 kali	10 Koin		
pemanjat_berulang_3	Bermain sebanyak 25 kali	15 Koin		
pemanjat_berulang_4	Bermain sebanyak 50 kali	25 Koin		

Nama *Achievement*	Syarat *Achievement*	Reward	Group *Achievement*	Group Reward
pemanjat_berulang_5	Bermain sebanyak 100 kali	50 Koin		
pemanjat_hebat_1	Berhasil memanjat sebanyak 10 tingkat	5 Koin	keberanian_tanpa_batas	100 Koin
pemanjat_hebat_2	Berhasil memanjat sebanyak 20 tingkat	10 Koin		
pemanjat_hebat_3	Berhasil memanjat sebanyak 50 tingkat	15 Koin		
pemanjat_hebat_4	Berhasil memanjat sebanyak 75 tingkat	25 Koin		
pemanjat_hebat_5	Berhasil memanjat sebanyak 100 tingkat	50 Koin		
panjat_rintangan_1	Berhasil memanjat sebanyak 10 tingkat berturut-turut	5 Koin	berani_menuju_puncak	100 Koin
panjat_rintangan_2	Berhasil memanjat sebanyak 20 tingkat berturut-turut	10 Koin		
panjat_rintangan_3	Berhasil memanjat sebanyak 50 tingkat berturut-turut	15 Koin		
panjat_rintangan_4	Berhasil memanjat sebanyak 75 tingkat berturut-turut	20 Koin		
panjat_rintangan_5	Berhasil memanjat sebanyak 100 tingkat berturut-turut	50 Koin		
kesalahan_di_awal	Menabrak sesuatu sebelum memanjat 10 tingkat ketinggian	10 Koin	belum_saatnya	10 Koin
berani_jatuh_1	Jatuh sebanyak sekali	5 Koin	gagal_memanjat	20 Koin

Nama *Achievement*	Syarat *Achievement*	Reward	Group *Achievement*	Group Reward
berani_jatuh_2	Jatuh sebanyak 5 kali	5 Koin		
berani_jatuh_3	Jatuh sebanyak 10 kali	5 Koin		
lihat_atas_1	Kejatuhan ranting sebanyak sekali	5 Koin		
lihat_atas_2	Kejatuhan ranting sebanyak 5 kali	5 Koin		
lihat_atas_3	Kejatuhan ranting sebanyak 10 kali	5 Koin		
pemanjat_gigih_1	Hidup kembali sebanyak sekali	10 Koin	gigih_menuju_puncak	20 Koin
pemanjat_gigih_2	Hidup kembali sebanyak 5 kali	10 Koin		
pemanjat_gigih_3	Hidup kembali sebanyak 10 kali	10 Koin		
memanjat_tanpa_kenal_lelah	Kalah sebanyak 100 kali	25 Koin	menuju_puncak_dan_melampauinya	25 Koin

Sumber: Dokumentasi Penulis

Tabel 4.7. Daftar *achievement* game Hujan Mangga

Nama *Achievement*	Syarat *Achievement*	Reward	Group *Achievement*	Group *Reward*
musim_mangga	Berbicara dengan NPC Penangkap Mangga	20 Koin	27anga_jatuh	10 Koin
kembali_untuk_mangga_1	Bermain sebanyak sekali	5 Koin	kembali_untuk_mangga	50 Koin
kembali_untuk_mangga_2	Bermain sebanyak 10 kali	10 Koin		
kembali_untuk_mangga_3	Bermain sebanyak 25 kali	15 Koin		

Nama *Achievement*	Syarat *Achievement*	Reward	Group *Achievement*	Group *Reward*
kembali_untuk_ 28anga_4	Bermain sebanyak 50 kali	25 Koin		
kembali_untuk_ 28anga_5	Bermain sebanyak 100 kali	50 Koin		
ketagihan_28anga_1	Akumulasi menangkap 10 mangga	5 Koin	ketagihan_ mangga	100 Koin
ketagihan_28anga_2	Akumulasi menangkap 20 mangga	10 Koin		
ketagihan_28anga_3	Akumulasi menangkap 50 mangga	15 Koin		
ketagihan_28anga_4	Akumulasi menangkap 75 mangga	25 Koin		
ketagihan_28anga_5	Akumulasi menangkap 100 mangga	50 Koin		
rentetan_mangga_1	Berhasil menangkap 28anga sebanyak 10 buah berturut-turut	5 Koin	rentetan_ mangga	100 Koin
rentetan_mangga_2	Berhasil menangkap 28anga sebanyak 20 buah berturut-turut	10 Koin		
rentetan_mangga_3	Berhasil menangkap 28anga sebanyak 50 buah berturut-turut	15 Koin		
rentetan_mangga_4	Berhasil menangkap 28anga sebanyak 75 buah berturut-turut	20 Koin		
rentetan_mangga_5	Berhasil menangkap 28anga sebanyak 100 buah berturut-turut	50 Koin		

Nama *Achievement*	Syarat *Achievement*	Reward	Group *Achievement*	Group *Reward*
sedang_sial	Kejatuhan ranting sebelum menangkap 10 buah mangga	10 Koin	kurang_ beruntung	10 Koin
target_terlewat_1	Kalah karena membiarkan 29anga jatuh sebanyak sekali	5 Koin	penangkap_sial	20 Koin
target_terlewat_2	Kalah karena membiarkan 29anga jatuh sebanyak 5 kali	5 Koin		
target_terlewat_3	Kalah karena membiarkan 29anga jatuh sebanyak 10 kali	5 Koin		
salah_tangkap_1	Kejatuhan ranting sebanyak sekali	5 Koin		
salah_tangkap_2	Kejatuhan ranting sebanyak 5 kali	5 Koin		
salah_tangkap_3	Kejatuhan ranting sebanyak 10 kali	5 Koin		
demi_mangga_1	Hidup kembali sebanyak sekali	10 Koin	demi_mangga	20 Koin
demi_mangga_2	Hidup kembali sebanyak 5 kali	10 Koin		
demi_mangga_3	Hidup kembali sebanyak 10 kali	10 Koin		
konsistensi_penangkap_mangga	Kalah sebanyak 100 kali	25 Koin	bertahan_untuk_ mangga	25 Koin

Sumber: Dokumentasi Penulis

Tabel 4.8. Daftar *achievement* game Tupai Kepak

Nama *Achievement*	**Syarat *Achievement***	***Reward***	**Group *Achievement***	**Group *Reward***
si_tupai_cerdik	Berbicara dengan NPC Tupai	20 Koin	tupai_lompat	10 Koin
menjadi_tupai_1	Bermain sebanyak sekali	5 Koin	menjadi_tupai	50 Koin
menjadi_tupai_2	Bermain sebanyak 10 kali	10 Koin		
menjadi_tupai_3	Bermain sebanyak 25 kali	15 Koin		
menjadi_tupai_4	Bermain sebanyak 50 kali	25 Koin		
menjadi_tupai_5	Bermain sebanyak 100 kali	50 Koin		
tupai_lintas_1	Akumulasi mencapai ke sisi lain sebanyak 10 kali	5 Koin	tupai_lintas	100 Koin
tupai_lintas_2	Akumulasi mencapai ke sisi lain sebanyak 20 kali	10 Koin		
tupai_lintas_3	Akumulasi mencapai ke sisi lain sebanyak 50 kali	15 Koin		
tupai_lintas_4	Akumulasi mencapai ke sisi lain sebanyak 75 kali	25 Koin		
tupai_lintas_5	Akumulasi mencapai ke sisi lain sebanyak 100 kali	50 Koin		
melintas_tanpa_henti_1	Berhasil mencapai ke sisi lain sebanyak 10 kali berturut-turut	5 Koin	konsistensi_sempurna	100 Koin

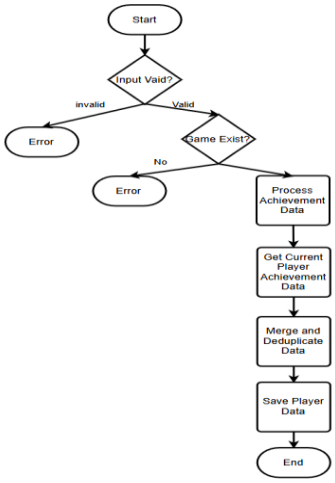
Nama *Achievement*	Syarat *Achievement*	*Reward*	Group *Achievement*	Group *Reward*
melintas_tanpa_henti_2	Berhasil mencapai ke sisi lain sebanyak 20 kali berturut-turut	10 Koin		
melintas_tanpa_henti_3	Berhasil mencapai ke sisi lain sebanyak 50 kali berturut-turut	15 Koin		
melintas_tanpa_henti_4	Berhasil mencapai ke sisi lain sebanyak 75 kali berturut-turut	20 Koin		
melintas_tanpa_henti_5	Berhasil mencapai ke sisi lain sebanyak 100 kali berturut-turut	50 Koin		
belum_panas	Tertangkap atau jatuh sebelum mencapai 10 skor	10 Koin	latihan_jatuh	10 Koin
latihan_lompat_1	Jatuh sebanyak sekali	5 Koin	magnet_ kegagalan	20 Koin
latihan_lompat_2	Jatuh sebanyak 5 kali	5 Koin		
latihan_lompat_3	Jatuh sebanyak 10 kali	5 Koin		
tertangkap_basah_1	Tertangkap jaring sebanyak sekali	5 Koin		
tertangkap_basah_2	Tertangkap jaring sebanyak 5 kali	5 Koin		
tertangkap_basah_3	Tertangkap jaring sebanyak 10 kali	5 Koin		
nyawa_cadangan_1	Hidup kembali sebanyak sekali	10 Koin	nyawa_ cadangan_ terpakai	20 Koin
nyawa_cadangan_2	Hidup kembali sebanyak 5 kali	10 Koin		

Nama *Achievement*	Syarat *Achievement*	*Reward*	Group *Achievement*	Group *Reward*
nyawa_cadangan_3	Hidup kembali sebanyak 10 kali	10 Koin		
akrab_dengan_kegagalan	Kalah sebanyak 100 kali	25 Koin	dedikasi_100_kegagalan	25 Koin

Sumber: Dokumentasi Penulis

4.3. Update Progress *Achievement*

Seluruh proses pembaruan progress *achievement player* dilakukan seluruhnya di Playfab. Disini penulis memanfaatkan fitur *Cloud Script* yang disediakan oleh Playfab untuk membuat program berupa fungsi-fungsi untuk menjalankan proses tertentu yang nantinya fungsi-fungsi tersebut dapat dipanggil dari sisi game (Unity). Namun sebelum masuk ke penjelasan setiap fungsi nya, kita akan melihat terlebih dahulu bagaimana alur keseluruhan proses pembaruan *progress achievement player* pada diagram dibawah,



Gambar 4.6. Flowchart proses pembaruan *achievement*

Sumber: Dokumentasi Penulis

Pada dasarnya, keseluruhan proses pembaruan *progress achievement player* berjalan pada fungsi berikut,

```
handlers.updatePlayerData = function (args, context) {
    // try {
    if (!args.game_code || !args.achievements || !Array.isArray(args.achievements)) {
        throw new Error(`Input harus memiliki 'game_code' dan 'achievements' yang merupakan array.`);
    }

    const gameExists = fetchGameByCode(args.game_code);
    if (!gameExists || !gameExists.results || !gameExists.results.data || gameExists.results.data.length === 0) {
        throw new Error(`Game dengan kode '${args.game_code}' tidak ditemukan.`);
    }

    let addedLevel1Achivements = new Map();
    const achievementCatalog = fetchAllAchievementsByGameCode(args.game_code).results;

    let updatedData = [];

    for (let achievement of args.achievements) {
        let reqAchievement = {};
        const matchingAchievement = achievementCatalog.find(a => a.code === achievement.code);
        if (!matchingAchievement) {
            throw new Error(`Achievement dengan kode '${achievement.code}' tidak ditemukan dalam katalog.`);
        }

        if (achievement.value > matchingAchievement.max_point) {
            throw new Error(`Nilai untuk achievement ${achievement.code} ${achievement.value} melebihi max_point
${matchingAchievement.max_point}.`);
        }

        reqAchievement.code = matchingAchievement.code;
        reqAchievement.type = matchingAchievement.type;
        reqAchievement.name = matchingAchievement.name;
        reqAchievement.max_point = matchingAchievement.max_point;
        reqAchievement.value = achievement.value;

        updatedData.push(reqAchievement);

        if (matchingAchievement.achievement_level_1_id) {
            const level1Achievement = achievementCatalog.find(a => a.id === matchingAchievement.achievement_level_1_id && a.level === 1);
            if (level1Achievement) {
                if (addedLevel1Achivements.has(level1Achievement.id)) {
                    const existingLevel1 = addedLevel1Achivements.get(level1Achievement.id);
                    const newValue = existingLevel1.value + achievement.value;
                    if (newValue > level1Achievement.max_point) {
                        throw new Error(`Nilai untuk achievement ${level1Achievement.code} ${existingLevel1.value} melebihi max_point.
${level1Achievement.max_point}`);
                    }
                    existingLevel1.value = newValue;
                } else {
                    let newLevel1Achievement = {
                        code: level1Achievement.code,
                        type: level1Achievement.type,
                        name: level1Achievement.name,
                        max_point: level1Achievement.max_point,
                        value: achievement.value,
                    };
                    addedLevel1Achivements.set(level1Achievement.id, newLevel1Achievement);
                }
            }
        }
    }

    updatedData.push(...addedLevel1Achivements.values());
    const key = `achievement_${args.game_code}`;

    let existingData = server.GetUserData({ PlayFabId: currentPlayerId, Keys: [key] });
    let currentData = [];
    if (existingData.Data && existingData.Data[key] && existingData.Data[key].Value) {
        try {
            currentData = JSON.parse(existingData.Data[key].Value);
        } catch (error) {
            log.error("Gagal mengurai JSON data yang ada:", error);
            throw new Error("Data pencapaian yang tersimpan rusak.");
        }
    }

    const filteredData = updateDataDuplicationCheck(currentData, updatedData);
    const dataJson = {};
    dataJson[key] = JSON.stringify(filteredData);

    const updateRequest = {
        PlayFabId: currentPlayerId,
        Data: dataJson
    };

    try {
        const result = server.UpdateUserData(updateRequest);
        log.info("Berhasil menyimpan atau memperbarui data pencapaian.", result);
        return { success:true, message: "Achievements saved/updated successfully.", result };
    } catch (error) {
        log.error("Gagal menyimpan atau memperbarui data pencapaian:", error);
        throw new Error("Terjadi kesalahan saat menyimpan atau memperbarui data pencapaian.");
    }
    // } catch (error) {
    //     return {success:false,error};
    // }
};
```

Kode Sumber 4.1. Fungsi *updatePlayerData*

Awalnya, setiap *player* berhasil meningkatkan *progress achievement*, game yang dimainkan akan memanggil fungsi tersebut dengan input parameter seperti contoh berikut,

```
{
    "game_code": "sapi_kebluk",
    "achievements": [
        {
            "code": "aku_kaya",
            "value": 50
        },
        {
            "code": "keluarkan_amarahmu",
            "value": 50
        }
    ]
}
```

Gambar 4.7. Contoh *request body* fungsi *updatePlayerData*

Sumber: Dokumentasi Penulis

Lalu, pertama fungsi akan mengecek apakah bentukan input sudah valid. Jika sudah valid, maka lanjut mengecek apakah game yang ingin diperbarui data *achievement* nya ada di katalog yang berada di Directus. Dalam prosesnya, fungsi akan memanggil fungsi lain yang lebih jelasnya seperti gambar dibawah,

```
function fetchGameByCode(code) {

  if (!code) {
      return {
          message: "Missing required parameter: code",
          results: null
      };
  }

  var url = "https://devdirectus.kreatifmaju.com/items/games?filter[code][_eq]=" + encodeURIComponent(code);
  var method = "get";
  var contentBody = "";
  var contentType = "application/json";
  var headers = { "Content-Type": "application/json" };

  try {
      var responseString = http.request(url, method, contentBody, contentType, headers);

      // Coba parse ke JSON
      var results;
      try {
          results = JSON.parse(responseString);
      } catch (parseError) {
          results = { error: "Failed to parse JSON", rawResponse: responseString };
      }

      return {
          results
      };

  } catch (error) {
      return {
          message: "Failed to fetch game data",
          results: error.Message ? error.Message : "Unknown error"
      };
  }
};
```

Kode Sumber 4.2. Fungsi *fetchGameByCode*

Fungsi ini (Kode Sumber 4.2) akan mengambil data game dengan *code* yang sama dengan parameter input, yang dimana jika kita lihat pada fungsi *updatePlayerData* (Kode Sumber 4.1), parameter input yang diberikan berupa *game_code* yang ada pada parameter input fungsi *updatePlayerData*. Untuk melakukan pengambilan data, kita akan melakukan *hit endpoint* yang sudah disediakan oleh Directus, dengan format "{*url* Directus}/items/{koleksi yang ingin diambil datanya}". *Endpoint* tersebut (Kode Sumber 4.2) digunakan untuk memfilter data dari koleksi games berdasarkan *field code* yang nilainya harus sama persis dengan parameter code yang telah dienkode menggunakan *encodeURIComponent*, guna memastikan karakter khusus dalam *code* tidak menyebabkan *error* dalam *url*. Dengan memanggil endpoint ini, kita akan mendapatkan data game yang memiliki code sesuai dengan parameter input.

Jika lolos pada pengecekan diatas, sekarang program akan lanjut untuk melakukan *pre-processing* data yang akan digunakan untuk memperbarui data *achievement player* (Kembali pada Kode Sumber 4.1). Tahapan ini diawali dengan mengambil seluruh katalog achievement yang berkaitan dengan game tersebut,

menggunakan fungsi *fetchAllAchievementsByGameCode*, yang hasilnya disimpan ke dalam variabel *achievementCatalog*.

```
function fetchAllAchievementsByGameCode(code) {
  // Pastikan args berisi gameCode
  if (!code) {
      return {
          message: "Missing required parameter: code",
          results: null
      };
  }

  var gameCode = encodeURIComponent(code); // Pastikan gameCode aman untuk dimasukkan ke dalam URL
  var url = "https://devdirectus.kreatifmaju.com/items/achievement_level_2"
      + "?fields=*,achievement_level_1_id.*,achievement_level_1_id.game_id.code"
      + "&filter[achievement_level_1_id][game_id][code][_eq]=" + gameCode;

  var method = "GET";
  var contentBody = "";
  var contentType = "application/json";
  var headers = { "Content-Type": "application/json" };

  try {
      var responseString = http.request(url, method, contentBody, contentType, headers);

      // Parsing response ke JSON
      var results;
      try {
          results = JSON.parse(responseString);
      } catch (parseError) {
          return {
              message: "Failed to parse JSON response",
              rawResponse: responseString
          };
      }

      if (!results.data || !Array.isArray(results.data)) {
          return {
              message: "Invalid data structure",
              results: results
          };
      }

      // Panggil fungsi pemrosesan data
      let formattedData = processAchievementsData(results.data);

      return {
          message: "Request successful",
          results: formattedData
      };

  } catch (error) {
      return {
          message: "Request failed",
          results: error.Message ? error.Message : "Unknown error"
      };
  }
}
```

Kode Sumber 4.3. Fungsi *fetchAllAchievementsByGameCode*

Fungsi ini (Kode Sumber 4.3) bertugas mengambil seluruh data *achievement level* 1 dan 2 berdasarkan *game_code* yang diberikan. Pemanggilan ini penting agar sistem bisa membandingkan data *achievement* yang dikirim oleh pengguna (melalui parameter *args.achievements*) dengan katalog resmi yang tersimpan di Directus. Proses pemangambilan data berpusat pada pemanggilan *endpoint* yang sejatinya akan memanggil data-data *achievement_level_2.* Pada bagian query-nya, kita meminta semua field (*), termasuk field relasional *achievement_level_1_id* (yaitu *parent* dari level 2), dan juga field relasional lebih dalam *achievement*_level_1_id.game_id.code, yang dipakai untuk melakukan filter berdasarkan game_code. Setelah data didapat, lalu fungsi tersebut memanggil fungsi *processAchievementsData* untuk merapikan data agar lebiih mudah di proses.

```
function processAchievementsData(data) {
  let formattedData = [];
  let addedAchievements = new Set(); // Menghindari duplikasi

  data.forEach(item => {
      // Ekstrak level 2 dengan menyimpan ID dari level 1-nya
      formattedData.push(extractAchievementsByLevel(item, 2, item.achievement_level_1_id ? item.achievement_level_1_id.id : null));

      // Ekstrak level 1 jika belum ditambahkan
      if (item.achievement_level_1_id && !addedAchievements.has(item.achievement_level_1_id.id)) {
          formattedData.push(extractAchievementsByLevel(item.achievement_level_1_id, 1, item.achievement_level_1_id.game_id ?
item.achievement_level_1_id.game_id.code : null));
          addedAchievements.add(item.achievement_level_1_id.id);
      }

      // Jika ada level 3 dan seterusnya, cukup tambahkan pemanggilan extractAchievementsByLevel() di sini.
  });

  return formattedData;
}
```

Kode Sumber 4.4. Fungsi *processAchievementsData*

Pada awal fungsi (Kode Sumber 4.4), variabel *formattedData* disiapkan sebagai penampung hasil akhir berupa *array* dari *achievement* yang telah diformat. Sementara itu, *addedAchievements* merupakan sebuah objek *Set* yang berfungsi sebagai pencatat ID *achievement level* 1 yang sudah diproses, untuk mencegah duplikasi entri. Setiap elemen item dalam array data kemudian diekstraksi dua kali, pertama untuk *level* 2 (yang merupakan entri utama dalam data dari *api*), dan kedua untuk *parent* nya yaitu *level* 1, tetapi hanya jika ID-nya belum diproses sebelumnya. Ekstraksi ini dilakukan melalui pemanggilan fungsi *extractAchievementsByLevel*.

```
function extractAchievementsByLevel(item, level, referenceId = null) {
  let achievement = {
      id: item.id,
      type: `achievement_level_${level}`,
      name: item.name,
      level: item.level,
      max_point: item.max_point,
      rewards: item.rewards,
      description: item.description,
      code: item.code
  };

  // Jika level 2, simpan ID level 1
  if (level === 2 && referenceId) {
      achievement.achievement_level_1_id = referenceId;
  }

  // Jika level 1, simpan game_code (jika ada)
  if (level === 1 && referenceId) {
      achievement.game_code = referenceId;
  }

  return achievement;
}
```

Kode Sumber 4.5. Fungsi *extractAchievementsByLevel*

Fungsi ini (Kode Sumber 4.5) bekerja sebagai penyusun struktur final dari masing-masing data *achievement*, berdasarkan *level* yang ditentukan oleh parameter. Untuk *achievement level* 2, fungsi ini (Kode Sumber 4.5) akan menyisipkan properti *achievement_level_1_id* sebagai penanda referensi ke *parent* nya. Sementara untuk *level* 1, fungsi ini (Kode Sumber 4.5) akan menyisipkan

properti *game_code*, yang berasal dari parent tertingginya yaitu objek game. Output dari fungsi ini (Kode Sumber 4.5) kemudian dikembalikan ke fungsi *processAchievementsData* (Kode Sumber 4.4), sehingga pada akhir prosesnya, *formattedData* akan berisi gabungan dari entri *level* 1 dan *level* 2 yang telah dinormalisasi.

Sekarang balik lagi ke Kode Sumber 4.1. Setelah mendapatkan data *achievement* yang telah di format, program akan membuat array *updatedData* yang nantinya akan berisi data *achievement* yang telah divalidasi dan siap disimpan. Untuk proses validasi, program akan membuat perulangan untuk setiap item di *args.achievements* (Kode Sumber 4.1). Untuk setiap *achievement*, dilakukan pencarian pada *achievementCatalog* berdasarkan kode *achievement*. Jika tidak ditemukan, proses dihentikan dengan *error*.

Selanjutnya, program memeriksa apakah nilai *achievement* (*value*) melebihi batas maksimum (*max_point*). Jika iya, juga dilempar *error*. Jika validasi lolos, data *achievement* seperti *code*, *type*, *name*, dan *max_point* disalin ke objek baru, lalu nilai *value* dari input juga disalin ke objek tersebut, dan dimasukkan ke *updatedData*.

Selain itu, jika *achievement* memiliki hubungan ke *achievement level* 1 (*achievement_level_1_id*), program mencari data *level* 1 terkait yang juga dipastikan memiliki *level* sama dengan 1. Jika sudah ada di koleksi *addedLevel1Achivements*, nilainya dijumlahkan dan divalidasi kembali agar tidak melebihi batas maksimal. Jika belum ada, dibuat objek baru untuk *level* 1 tersebut dan ditambahkan ke koleksi.

Setelah semua data *achievement level* 2 selesai diproses, semua *achievement level* 1 yang sudah dikumpulkan juga ditambahkan ke *updatedData*. Dengan begitu, *updatedData* berisi data lengkap *achievement* yang siap diperbarui untuk pemain.

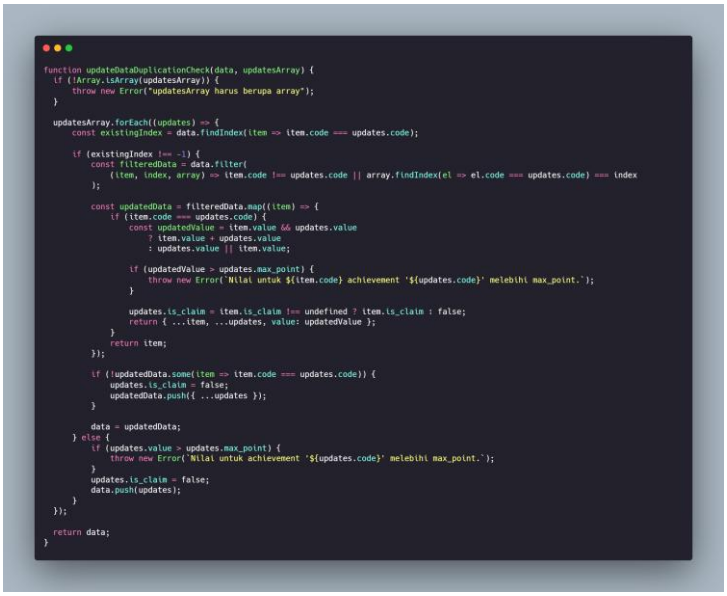
Setelah semua data *achievement level* 2 selesai diproses, semua *achievement level* 1 yang sudah dikumpulkan juga ditambahkan ke *updatedData*. Dengan

begitu, *updatedData* berisi data lengkap *achievement* yang siap diperbarui untuk pemain.

Setelah seluruh data *achievement* berhasil dikumpulkan dan disusun dalam *updatedData*, langkah selanjutnya adalah mengambil data pencapaian pemain yang sudah tersimpan sebelumnya di server. Hal ini dilakukan dengan memanggil *server.GetUserData*, yang mengambil data berdasarkan PlayFabId milik pemain saat ini dan kunci data tertentu (*key*) yang telah ditentukan, dalam hal ini adalah kunci data *achievement player* sesuai game yang ditentukan (Berdasarkan *game_code)*. Data yang dikembalikan disimpan dalam *existingData*.

Jika *existingData* mengandung nilai pada *key* tersebut, program mencoba mengurai isinya dari bentuk *string JSON* menjadi *array* JavaScript. Proses ini dibungkus dalam blok *try-catch* untuk menghindari *error parsing*. Bila *parsing* gagal, artinya data rusak atau bukan format yang valid, maka akan ditampilkan pesan *error* dan proses dihentikan.

Setelah data lama berhasil diuraikan (atau dibiarkan kosong jika belum ada data sebelumnya), program akan memanggil fungsi *updateDataDuplicationCheck(currentData, updatedData)* untuk menyaring dan menggabungkan data pencapaian baru dengan data yang sudah tersimpan. Tujuan utama dari fungsi ini adalah mencegah terjadinya duplikasi data pencapaian dan memastikan nilai yang tersimpan tidak melebihi batas maksimal (*max_point*) yang telah ditentukan untuk tiap pencapaian.



```
function updateDataDuplicationCheck(data, updatesArray) {
  if (!Array.isArray(updatesArray)) {
      throw new Error("updatesArray harus berupa array");
  }

  updatesArray.forEach((updates) => {
      const existingIndex = data.findIndex(item => item.code === updates.code);

      if (existingIndex !== -1) {
          const filteredData = data.filter(
              (item, index, array) => item.code !== updates.code || array.findIndex(el => el.code === updates.code) === index
          );

          const updatedData = filteredData.map((item) => {
              if (item.code === updates.code) {
                  const updatedValue = item.value && updates.value
                      ? item.value + updates.value
                      : updates.value || item.value;

                  if (updatedValue > updates.max_point) {
                      throw new Error(`Nilai untuk ${item.code} achievement '${updates.code}' melebihi max_point.`);
                  }

                  updates.is_claim = item.is_claim !== undefined ? item.is_claim : false;
                  return { ...item, ...updates, value: updatedValue };
              }
              return item;
          });

          if (!updatedData.some(item => item.code === updates.code)) {
              updates.is_claim = false;
              updatedData.push({ ...updates });
          }

          data = updatedData;
      } else {
          if (updates.value > updates.max_point) {
              throw new Error(`Nilai untuk achievement '${updates.code}' melebihi max_point.`);
          }
          updates.is_claim = false;
          data.push(updates);
      }
  });

  return data;
}
```

Kode Sumber 4.6. Fungsi updateDataDuplicationCheck

Fungsi ini (Kode Sumber 4.6) dimulai dengan validasi bahwa *updatesArray* harus berupa *array*. Kemudian, fungsi melakukan iterasi untuk setiap *item* baru yang akan diperbarui. Jika pencapaian dengan kode yang sama sudah ada di data sebelumnya (*currentData*), maka data tersebut akan diperbarui dan nilai *value* lama ditambahkan dengan nilai baru, selama hasilnya tidak melebihi *max_point*. Fungsi juga menjaga agar hanya satu entri unik per *code* yang tersisa, dan menambahkan atribut *is_claim* jika belum ada.

Jika pencapaian belum pernah tercatat sebelumnya, maka item tersebut langsung ditambahkan ke dalam data akhir dengan nilai awal dan is_claim diset false.

Hasil akhir dari proses ini adalah kumpulan data pencapaian yang sudah diperbarui dan tervalidasi, bebas dari duplikasi, dan siap disimpan ke dalam sistem. Data tersebut kemudian diubah kembali ke format JSON string dan dibungkus ke dalam objek updateRequest yang akan dikirim ke server menggunakan server.UpdateUserData()(Balik ke Kode Sumber 4.1). Jika proses penyimpanan berhasil, maka sistem akan mencatat keberhasilan melalui log dan mengembalikan hasil sukses. Jika tidak, maka akan dicatat sebagai error dan fungsi akan melemparkan pesan kegagalan penyimpanan data.

4.4. *Claim Achievement*

Setelah menyelesaikan suatu *achievement*, *player* dapat memperoleh hadiah yang ditawarkan oleh *achievement* tersbut. Untuk melakukan proses ini, penulis membuat suatu fungsi bernama *claimAchievement.*

```
handlers.claimAchievement = function (args, context) {
    // Validasi input
    if (!args.game_code || !args.code) {
        throw new Error("Diperlukan 'game_code' dan 'code' untuk mengklaim achievement.");
    }

    // Fetch game berdasarkan game_code
    const gameExists = fetchGameByCode(args.game_code);

    // Periksa apakah game ditemukan
    if (!gameExists || !gameExists.results || !gameExists.results.data || gameExists.results.data.length === 0) {
        throw new Error(`Game dengan kode '${args.game_code}' tidak ditemukan.`);
    }

    const key = `achievement_${args.game_code}`;

    // Ambil data pencapaian pengguna dari PlayFab
    let existingData = server.GetUserData({ PlayFabId: currentPlayerId, Keys: [key] });

    let currentData = [];
    if (existingData.Data && existingData.Data[key] && existingData.Data[key].Value) {
        try {
            currentData = JSON.parse(existingData.Data[key].Value);
        } catch (error) {
            log.error("Gagal mengurai JSON data yang ada:", error);
            throw new Error("Data pencapaian yang tersimpan rusak.");
        }
    }

    // Cari achievement berdasarkan code
    let achievement = currentData.find(a => a.code === args.code);

    if (!achievement) {
        throw new Error(`Achievement dengan kode '${args.code}' tidak ditemukan.`);
    }

    // Periksa apakah achievement sudah diklaim sebelumnya
    if (achievement.is_claim) {
        throw new Error(`Achievement '${args.code}' sudah diklaim sebelumnya.`);
    }

    // Periksa apakah nilai telah mencapai max_point
    if (achievement.value < achievement.max_point) {
        throw new Error(`Achievement '${args.code}' belum mencapai poin maksimum untuk diklaim.`);
    }

    // Fetch achievement dari katalog untuk mendapatkan rewards
    const achievementFromCatalog = findAchievementByCode(args.game_code, args.code);
    if (!achievementFromCatalog || !achievementFromCatalog.result) {
        throw new Error(`Achievement dengan kode '${args.code}' tidak ditemukan di server.`);
    }

    let rewards = achievementFromCatalog.result.rewards || [];
    if (rewards.length === 0) {
        log.warn(`Achievement '${args.code}' tidak memiliki rewards.`);
    }

    // Tandai achievement sebagai diklaim
    achievement.is_claim = true;

    // Simpan kembali data yang diperbarui sebelum memberikan reward
    const dataJson = {};
    dataJson[key] = JSON.stringify(currentData);

    const updateRequest = {
        PlayFabId: currentPlayerId,
        Data: dataJson
    };

    try {
        // Simpan perubahan ke PlayFab sebelum memberikan reward
        const result = server.UpdateUserData(updateRequest);
        log.info(`Achievement '${args.code}' berhasil diklaim.`, result);

        // Memberikan Rewards ke Pemain
        rewards.forEach(reward => {
            if (reward.type === "Currency") {
                // Tambahkan currency
                server.AddUserVirtualCurrency({
                    PlayFabId: currentPlayerId,
                    VirtualCurrency: reward.code, // Contoh: "CN" untuk Gold Coins
                    Amount: reward.value
                });
                log.info(`Menambahkan ${reward.value} ${reward.name} ke user.`);
            } else if (reward.type === "Item") {
                // Berikan item ke inventory
                server.GrantItemsToUser({
                    PlayFabId: currentPlayerId,
                    ItemIds: [reward.code],
                    Annotation: `Reward dari achievement ${args.code}`
                });
                log.info(`Menambahkan item '${reward.name}' ke inventory user.`);
            }
        });

        return {success:true, message: `Achievement '${args.code}' berhasil diklaim dan reward diberikan!`, result };

    } catch (error) {
        log.error("Gagal menyimpan data klaim achievement atau memberikan reward:", error);

        // Rollback jika pemberian reward gagal
        achievement.is_claim = false;
        dataJson[key] = JSON.stringify(currentData);
        server.UpdateUserData({ PlayFabId: currentPlayerId, Data: dataJson });

        throw new Error("Terjadi kesalahan saat menyimpan klaim achievement atau memberikan reward. Klaim achievement dibatalkan.");
    }
//     } catch (error) {
//         return {success:false,error}

//     }
};
```

Kode Sumber 4.7. Fungsi *claimAchievement*

Fungsi ini (Kode Sumber 4.7) dimulai dengan melakukan validasi terhadap parameter yang diberikan. Program memastikan bahwa game_code dan code telah diisi, karena kedua parameter ini sangat penting untuk menentukan game dan a*chievement* yang akan diklaim. Jika salah satu dari parameter tersebut tidak tersedia, maka proses akan langsung dihentikan dengan pesan error yang sesuai.

Setelah validasi awal, program menggunakan fungsi *fetchGameByCode* (Kode Sumber 4.2) untuk mengambil data game berdasarkan *game_code* yang

diberikan. Hal ini sama dengan proses *updatePlayerData* sebelumnya (Kode Sumber 4.1). Pengecekan ini dilakukan agar *achievement* game yang ingin diupdate benar-benar berada di katalog.

Jika data game berhasil ditemukan, program kemudian membuat key data *achievement* berdasarkan kode game dengan format *achievement_<game_code>*. Key ini digunakan untuk mengambil data pencapaian user dari PlayFab melalui pemanggilan *server.GetUserData*. Data yang didapatkan dari server kemudian diperiksa, jika data tersebut valid dan memiliki nilai, maka akan diurai menjadi bentuk array menggunakan *JSON.parse*. Namun, jika proses parsing gagal misalnya karena data rusak atau format tidak sesuai program akan mencatat pesan kesalahan di log dan menghentikan proses dengan pesan bahwa data pencapaian rusak.

Selanjutnya, program mencari item pencapaian yang sesuai di dalam array *currentData* berdasarkan kode yang diberikan. Jika tidak ditemukan, maka muncul pesan bahwa pencapaian tersebut tidak ada dalam data pengguna. Setelah ditemukan, program memeriksa apakah pencapaian tersebut sudah pernah diklaim sebelumnya. Jika sudah, maka klaim akan ditolak. Setelah itu, dilakukan pengecekan apakah nilai pencapaian (*value*) telah mencapai batas maksimum (*max_point*) yang ditentukan. Jika belum mencapai nilai maksimum, maka klaim tidak bisa dilakukan karena pencapaian dianggap belum selesai. Jika semua kondisi ini terpenuhi, maka proses klaim dapat dilanjutkan ke tahap berikutnya.

Setelah memastikan bahwa sebuah pencapaian memang layak untuk diklaim, program kemudian mengambil informasi lebih lanjut mengenai *achievement* tersebut dari katalog server. Hal ini dilakukan dengan memanggil fungsi *findAchievementByCode*. Fungsi ini bertanggung jawab untuk mencari data pencapaian berdasarkan kode game dan kode *achievement* yang diberikan.

Kode Sumber 4.8. Fungsi *findAchievementByCode*

Fungsi ini (Kode Sumber 4.8) terlebih dahulu memeriksa apakah parameter *gameCode* dan *achievementCode* sudah tersedia. Jika salah satu dari parameter ini tidak ada, maka fungsi akan langsung mengembalikan objek dengan pesan kesalahan dan nilai result berupa null. Selanjutnya, fungsi akan memanggil fungsi lain bernama *fetchAllAchievementsByGameCode*, yang sudah penulis bahas sebelumnya (Lihat Kode Sumber 4.3). Setelah data berhasil diambil, fungsi akan memeriksa apakah struktur data yang diterima valid dan dalam bentuk array.

Jika semua validasi terpenuhi, fungsi akan mencari data pencapaian yang memiliki kode yang sama persis dengan *achievementCode* yang dikirim. Apabila pencapaian tersebut ditemukan, maka data pencapaian yang sesuai akan dikembalikan dalam objek *result*. Namun jika tidak ditemukan, maka fungsi akan mengembalikan pesan bahwa pencapaian dengan kode tersebut tidak ada.

Kembali ke fungsi utama *claimAchievement* (lihat kembali Kode Sumber 4.7), jika data dari katalog tidak ditemukan atau hasil dari fungsi *findAchievementByCode* (Kode Sumber 4.8) tidak memiliki *result*, maka akan dilemparkan error yang menyatakan bahwa *achievement* dengan kode tersebut tidak tersedia di server. Validasi ini penting karena sistem hanya akan memberikan reward berdasarkan data resmi dari katalog server untuk memastikan bahwa reward yang diberikan sah dan telah dikonfigurasi sebelumnya.

Setelah data pencapaian berhasil ditemukan dari katalog dan dinyatakan valid, langkah selanjutnya adalah menandai pencapaian tersebut sebagai sudah diklaim. Hal ini dilakukan dengan mengubah properti *is_claim* pada objek *achievement* menjadi *true*. Perubahan ini penting untuk mencegah pencapaian yang sama diklaim lebih dari satu kali oleh *player*.

Setelah perubahan dilakukan, seluruh data pencapaian yang telah diperbarui dikonversi kembali menjadi format JSON dan disimpan dalam objek *dataJson*, dengan *key* yang sesuai berdasarkan kode game. Objek ini kemudian dibungkus lagi ke dalam *updateRequest*, yang berisi PlayFabId pemain dan data pencapaian yang siap disimpan ke server.

Langkah selanjutnya adalah mencoba menyimpan data ke server PlayFab melalui pemanggilan *server.UpdateUserData(updateRequest)*. Jika proses penyimpanan berhasil, sistem mencatat log keberhasilan, dan proses dilanjutkan ke tahap pemberian hadiah (*rewards*) kepada pemain.

Hadiah diberikan satu per satu sesuai dengan daftar reward yang telah ditentukan dalam katalog *achievement*. Jika reward bertipe *Currency*, maka server akan memanggil fungsi *AddUserVirtualCurrency* yang merupakan fungsi bawaan dari PlayFab untuk menambahkan sejumlah mata uang virtual ke akun pemain, berdasarkan kode dan jumlah yang ditentukan. Log juga akan dicatat untuk menunjukkan nilai dan jenis mata uang yang diberikan. Jika reward bertipe *Item*, maka sistem akan menggunakan *GrantItemsToUser* yang juga merupakan bawaan dari PlayFab untuk memberikan *item* tertentu ke *inventory player*, dengan anotasi yang menjelaskan bahwa *item* tersebut berasal dari pencapaian.

Setelah seluruh reward berhasil diberikan, fungsi akan mengembalikan respon berhasil, menyertakan pesan keberhasilan serta hasil dari proses penyimpanan. Namun, jika terjadi kegagalan saat menyimpan klaim achievement atau saat memberikan *reward*, sistem akan menangkap *error* tersebut, mencatatnya ke log, dan melakukan proses *rollback*. *Rollback* dilakukan dengan mengembalikan nilai *is_claim* menjadi *false*, kemudian menyimpan kembali data yang sudah dikembalikan ke kondisi sebelumnya ke server. Dengan demikian, proses klaim

achievement dibatalkan agar tidak ada inkonsistensi data atau pemberian *reward* yang tidak sah.

[Halaman ini sengaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

Bab ini menjelaskan tahap uji coba dan evaluasi yang dilakukan terhadap fitur yang telah dibuat.

5.1 Tujuan Pengujian

Pengujian dilakukan terhadap fitur yang telah dibuat untuk menguji apakah fitur tersebut sudah sesuai dengan alur sistem *achievement* yang sudah di *briefing* sebelumnya.

5.2 Kriteria Pengujian

Penilaian atas pencapaian tujuan pengujian fitur dapat dilihat dari bagaimana fitur dapat bekerja dalam baik dan dapat menangani error dengan baik.

5.3 Skenario Pengujian

Skenario pengujian melibatkan seorang pembimbing lapangan yaitu Novian Citantio Widodo yang mengarahkan penulis untuk mencoba fitur yang sudah dibuat dengan berbagai skenario.

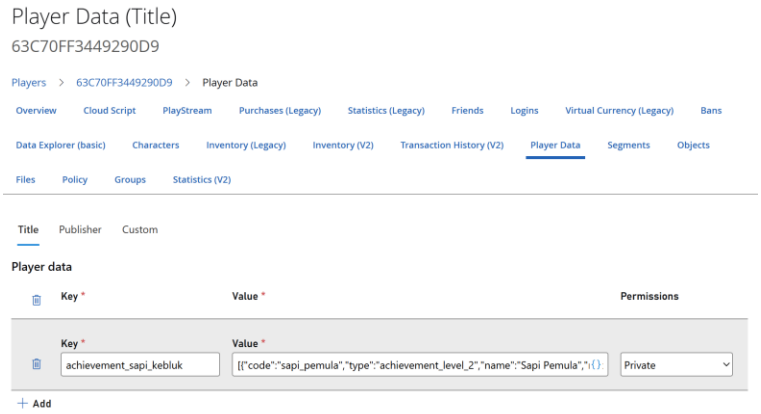
Pertama penulis menguji fitur pembaruan *progress achievement player* yang dijalankan oleh fungsi *updatePlayerData.* Untuk mengetahui keberhasilan fitur, kita dapat mengakses data *player* yang dapat dibuka pada bagian *Player* pada bagian *Build.* Kebetulan disini penulis sudah disediakan satu *player* dummy yang diperuntukan untuk *testing* fitur. Saat kita membuka *player* tersebut, akan memunculkan halaman seperti dibawah,

Gambar 5.1. Pilihan menu informasi *player*

Sumber: Dokumentasi penulis

Dibagian *Cloud Script* kita dapat melakukan *testing* terhadap fungsi yang sudah kita buat sebelumnya, diantaranya adalah *updatePlayerData* dan *claimAchievement*.

Untuk melihat keberhasilan fungsi *updatePlayerData*, kita dapat membuka bagian *Player Data*. Disana adalah tempat dimana data-data *player* disimpan sesuai *key* nya.



Gambar 5.2 Menu *Player Data*

Sumber: Dokumentasi Penulis

Berikut skenario-skenario yang diberikan oleh pembimbing lapangan terhadap fitur yang sudah dikembangkan oleh penulis. Disini menggunakan game Sapi Kebluk sebagai pengujian, dimana pertama penulis akan mengosongkan setiap *achievement* yang sudah pernah ditambahkan sebelumnya saat debugging.

Gambar 5.3. *Achievement* game Sapi Kebluk yang sudah dikosongkan

Sumber: Dokumentasi Penulis

1. Success Update Player Achievement Progress

Pertama penulis diberi skenario saat fitur berhasil memperbarui *progress achievement* player. Disini kita akan memanggil fungsi *updatePlayerData* dengan request body seperti berikut.

Gambar 5.4. *Request body* pengetesan fungsi updatePlayerData

Sumber: Dokumentasi Penulis

Disini, *max_point* dari *achievement* "Keluarkan Amarahmu" sendiri memiliki nilai sebesar satu, makanya disini kita hanya bisa memperbarui *point* sebesar satu. Keberhasilan dari skenario ini dilihat dari respon saat fungsi tersebut di jalankan dan perubahan pada data *achievement* yang sebelumnya telah dikosongkan.

```
{
    "FunctionResult": {
        "success": true,
        "message": "Achievements saved/updated
successfully.",
        "result": {
            "DataVersion": 230
        }
    },
    "Logs": [
        {
            "Level": "Info",
            "Message": "Berhasil menyimpan atau
memperbarui data pencapaian.",
            "Data": {
                "DataVersion": 230
            }
        }
    ],
    "ExecutionTimeSeconds": 1.0420307,
    "MemoryConsumedBytes": 84968,
    "APIRequestsIssued": 2,
    "HttpRequestsIssued": 2,
    "Error": null
}
```


Gambar 5.5. Respon sukses updatePlayerData

Sumber: Dokumentasi Penulis

Gambar 5.6. Data *Achievement* yang berhasil terupdate

Sumber: Dokumentasi Penulis

Berdasarkan Gambar 5.5, fungsi mengembalikan respon sukses untuk keberhasilan pembaruan *progress achievement player.* Pada Gamber 5.6 juga terlihat bahwa data *achievement player* berhasil diperbarui, yang dimana *achievement target* (Keluarkan Amarahmu) berhasil diperbarui dan karena *achievement* target berhasil diselesaikan, data *achievement parent* nya juga ikut terperbarui.

2. *Failed Update Player Achievement (Invalid Request Body)*

Penulis tidak hanya diminta untuk menguji skenario sukses. Disini penulis juga diminta untuk mensimulasikan bebrapa simulasi gagal. Pertama, disini penulis diminta untuk mensimulasikan dengan skenario *request body* yang tidak valid. Berikut contoh *request body* yang tidak valid.

Gambar 5.7. Invalid *request body*

Sumber: Dokumentasi Penulis

Jika *request body* yang tidak valid seperti diatas digunakan, maka akan mendapatkan respon *error* seperti berikut,


```
{
    "FunctionResult": null,
    "Logs": [],
    "ExecutionTimeSeconds": 0.0008171,
    "MemoryConsumedBytes": 21496,
    "APIRequestsIssued": 0,
    "HttpRequestsIssued": 0,
    "Error": {
        "Error": "JavascriptException",
        "Message": "JavascriptException",
        "StackTrace": "Error: Input harus memiliki
'game_code' dan 'achievements' yang merupakan
array.\n    at handlers.updatePlayerData (37637-
main.js:61:13)\n    at Object.invokeFunction
(Script:118:33)"
    }
}
```


Gambar 5.8. Respon *error* dari *request body* yang tidak valid

Sumber: Dokumentasi Penulis

Disini data *achievement player* juga tidak berhasil diperbarui dan tetap persis seperti Gambar 5.6.

3. *Failed Update Player Achievement (Invalid Achievement Code)*

Skenario berikutnya adalah ketika kita menginputkan *achievement* yang tidak ada di katalog. Beriku *request body* dengan *invalid achievement,*

Gambar 5.9. *Request body* dengan invalid *achievement*

Sumber: Dokumentasi Penulis

Berikut respon server jika kita menggunakan *request body* dengan invalid *achievement* seperti diatas,


```
{
    "FunctionResult": null,
    "Logs": [],
    "ExecutionTimeSeconds": 1.1154887,
    "MemoryConsumedBytes": 81896,
    "APIRequestsIssued": 0,
    "HttpRequestsIssued": 2,
    "Error": {
        "Error": "JavascriptException",
        "Message": "JavascriptException",
        "StackTrace": "Error: Achievement dengan
kode 'invalid_achievement' tidak ditemukan dalam
katalog.\n    at handlers.updatePlayerData (37637-
main.js:78:17)\n    at Object.invokeFunction
(Script:118:33)"
    }
}
```


Gambar 5.10. Respon *error* untuk *invalid achievement*

Sumber: Dokumentasi Penulis

4. *Failed Update Player Achievement* (Player Sudah Menyelesaikan Achievement Yang Ingin Di Update)

Saat *player* sudah menyelesaikan *achievement* tertentu, sistem tidak memperbolehkannya untuk terus memperbarui *progress* nya. Keadaan yang manandakan *player* sudah menyelesaikan suatu *achievement* adalah dimana *point* yang berada pada data *achievement player* sudah sama dengan *max_point* dari *achievement.* Sebelumnya kita sudah melakukan perbaruan data *achievement keluarkan_amarahmu* dengan satu *point,* yang dimana *max_point* nya juga berjumlah satu. Jika kita memperbarui lagi *achievement* tersebut, maka server akan melemparkan *error* seperti berikut,


```
{
    "FunctionResult": null,
    "Logs": [],
    "ExecutionTimeSeconds": 1.1745802,
    "MemoryConsumedBytes": 81888,
    "APIRequestsIssued": 0,
    "HttpRequestsIssued": 2,
    "Error": {
        "Error": "JavascriptException",
        "Message": "JavascriptException",
        "StackTrace": "Error: Nilai untuk
achievement keluarkan_amarahmu 50 melebihi
max_point 1.\n    at handlers.updatePlayerData
(37637-main.js:82:17)\n    at
Object.invokeFunction (Script:118:33)"
    }
}
```

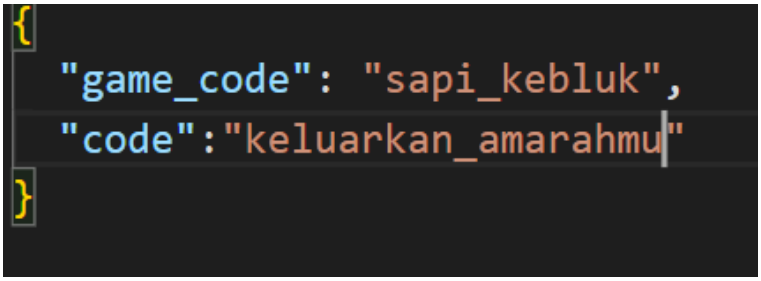

Gambar 5.11. Respon *error* untuk perbaruan *achievement* yang sudah dicapai

Sumber : Dokuemntasi Penulis

Hal ini juga berlaku pada kasus dimana kita memperbarui data *achievement* dengan jumlah *point* yang melebihi jumlah *max_point* nya.

5. Success Claim Achievement

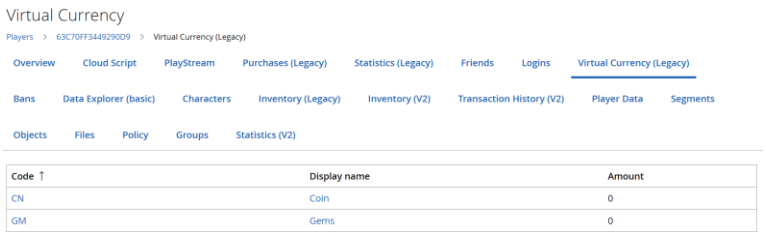
Selain menguji fitur pembaruan *achievement,* penulis juga diminta untuk menguji fitur *claim_achievement* yang juga dibangun olehnya. Berikut contoh *request_body* yang digunakan untuk memanggil fungsi *claim_achievement*,



Gambar 5.12. Contoh *request body* fungsi *claimAchievement*

Sumber: Dokumentasi Penulis

Untuk melihat keberhasilan fungsi, kita akan melihat pada menu *Virtual Currency*,



Gambar 5.13. Menu *Virutal Currency*

Sumber: Dokumentasi Penulis

Menu ini adalah bagian yang menyimpan informasi total *virtual currency* yang dimiliki oleh player. Pada pengujian ini, kita akan menggunakna *achievement* yang bernama *keluarkan_amarahmu* yang sebelumnya sudah digunakna dalam pengujian *updatePlayerData.* Berikut merupakan respon sukses dari server dalam menjalankan fungsi *claimAchievement,*

```
{
    "FunctionResult": {
        "success": true,
        "message": "Achievement
'keluarkan_amarahmu' berhasil diklaim dan reward
diberikan!",
        "result": {
            "DataVersion": 249
        }
    },
    "Logs": [
        {
            "Level": "Info",
            "Message": "Achievement
'keluarkan_amarahmu' berhasil diklaim.",
            "Data": {
                "DataVersion": 249
            }
        },
        {
            "Level": "Info",
            "Message": "Menambahkan 10 Coin ke
user.",
            "Data": null
        }
    ],
    "ExecutionTimeSeconds": 1.0927283,
    "MemoryConsumedBytes": 85776,
    "APIRequestsIssued": 3,
    "HttpRequestsIssued": 2,
    "Error": null
}
```


Gambar 5.14. Respon sukses fungsi *claimAchievement*

Sumber: Dokumentasi Penulis

Disini penulis juga menambahkan *log* untuk melihat apakah 10 *coin* berhasil ditambahkan kedalam data *player*. Berikut total *virtual currency* yang sekarang dimiliki oleh *player* setelah melakukan *claim achievement,*

Code ↑	Display name	Amount
CN	Coin	10
GM	Gems	0

Gambar 5.15. Total *virtual currency player* setelah *claim achievement*

Sumber: Dokumentasi Penlulis

6. Failed Claim Achievement (Invalid Request Body)

Penulis juga diminta untuk menguji fungis *claimAchievement* dengan *request body* yang tidak valid. Berikut *request body* yang digunakan,

Gambar 5.16. Invalid *request body* fungsi *claimAchievement*

Sumber: Dokumentasi Penulis

Berikut respon server untuk *request body* yang tidak valid pada fungsi *claimAchievement,*


```
{
    "FunctionResult": null,
    "Logs": [],
    "ExecutionTimeSeconds": 0.0006519,
    "MemoryConsumedBytes": 20648,
    "APIRequestsIssued": 0,
    "HttpRequestsIssued": 0,
    "Error": {
        "Error": "JavascriptException",
        "Message": "JavascriptException",
        "StackTrace": "Error: Diperlukan
'game_code' dan 'code' untuk mengklaim
achievement.\n    at handlers.claimAchievement
(37637-main.js:157:13)\n    at
Object.invokeFunction (Script:118:33)"
    }
}
```


Gambar 5.17. Respon *error* untuk *request body* yang tidak valid pada *claimAchievement*

Sumber: Dokumentasi Penulis

7. *Failed Claim Achievement* (*Achievement* Sudah Diclaim).

Jika suatu *achievement* sudah di *claim*, tentunya *achievement* tersebut tidak boleh di *claim* lagi. Untuk itu, penulis juga diminta untuk menguji apakah fitur yang dibuat juga mempertimbangkan hal tersebut. Disini, kita sudah melakukan *claim* terhadap *achievement kerahkan_amarahmu*. Jika kita melakukan *claim* ulang terhadap *achievement* tersebut, berikut respon dari server,

```
{
    "FunctionResult": null,
    "Logs": [],
    "ExecutionTimeSeconds": 0.7664496,
    "MemoryConsumedBytes": 40848,
    "APIRequestsIssued": 1,
    "HttpRequestsIssued": 1,
    "Error": {
        "Error": "JavascriptException",
        "Message": "JavascriptException",
        "StackTrace": "Error: Achievement
'keluarkan_amarahmu' sudah diklaim sebelumnya.\n
at handlers.claimAchievement (37637-
main.js:192:13)\n    at Object.invokeFunction
(Script:118:33)"
    }
}
```


Gambar 5.18. Respon *error* kasus *claim* ulang *achievement* yang sudah di *claim*

Sumber: Dokumentasi Penulis

Untuk jumlah *coin* yang dimiliki oleh pemain akan tetap berjumlah sama seperti Gambar 5.15.

8. *Failed Claim Achievement* (Jika Achievement Berhasil Diclaim, Namun *Currency Player* Gagal Diperbarui)

Terakhir, penulis diminta untuk menguji sistem *rollback* dalam *handling* kasus dimana jika suatu *achievement* terlanjur ditandai telah di *claim,* namun terjadi *error* saat memperbarui jumlah *currency player.* Untuk kasus ini, pada fungsi *claimAchievement* yang ditunjukkan pada Kode Sumber 4.7, penulis akan menambahkan *error* buatan di setelah perbaruan status *claim achievement* dan di sebelum perbaruan *currency player.* Disini penulis juga akan mereset status *claim keluarkan_amarahmu* menjadi *false* supaya bisa digunakan lagi pada pengujian kasus kali ini. Berikut respon dari server terhadap *error* buatan yang dibuat oleh penulis,

```
{
    "FunctionResult": null,
    "Logs": [
        {
            "Level": "Info",
            "Message": "Achievement
'keluarkan_amarahmu' berhasil diklaim.",
            "Data": {
                "DataVersion": 251
            }
        },
        {
            "Level": "Error",
            "Message": "Gagal menyimpan data klaim
achievement atau memberikan reward:",
            "Data": {}
        }
    ],
    "ExecutionTimeSeconds": 1.1573794,
    "MemoryConsumedBytes": 97176,
    "APIRequestsIssued": 3,
    "HttpRequestsIssued": 2,
    "Error": {
        "Error": "JavascriptException",
        "Message": "JavascriptException",
        "StackTrace": "Error: Terjadi kesalahan
saat menyimpan klaim achievement atau memberikan
reward. Klaim achievement dibatalkan.\n    at
handlers.claimAchievement (37637-main.js:259:13)\n
at Object.invokeFunction (Script:118:33)"
    }
}
```


Gambar 5.19. Respon *error* kasus berhasil memperbarui *achievemen* dan gagal memperbarui *currency player*

Sumber: Dokumentasi Penulis

Disini terlihat bahwa status *claim* berhasil di perbarui, namun gagal dalam memperbarui *currency player.* Setelah dicek pada bagian data *achievement player,* status *claim keluarkan_amarahmu* yang sebelumnya *false*, masih tetap *false* yang menunjukan keberhasilan sistem *rollback*.

JSON Editor (achievement_sapi_kebluk)

Gambar 5.20. Data *achievement player* yang berhasil di *rollback*

Sumber: Dokumentasi Penulis

5.4 Evaluasi Pengujian

Berdasarkan pengujian yang telah dipaparkan di atas, maka pembimbing lapangan telah menyetujui fitur yang telah dibuat oleh penuli. Berikut ini adalah rangkuman dari pengujian yang dilakukan:

Tabel 5.1. Tabel Evaluasi Pengujian Fitur

Kasus	Ekspektasi
Success Update Player Achievement Progress	Terpenuhi
Failed Update Player Achievement – Invalid Request Body	Terpenuhi
Failed Update Player Achievement – Invalid Achievement Code	Terpenuhi
Failed Update Player Achievement – Achievement Sudah Diselesaikan	Terpenuhi
Success Claim Achievement	Terpenuhi
Failed Claim Achievement – Invalid Request Body	Terpenuhi
Failed Claim Achievement – Achievement Sudah Diclaim	Terpenuhi
Failed Claim Achievement – Gagal *Update Currency* Setelah Claim	Terpenuhi

Sumber: Dokumentasi Penulis

Dengan hasil pengujian pada tabel di atas, maka dapat disimpulkan bahwa secara keseluruhan daftar kasus yang diberikan telah terpenuhi.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan perancangan, implementasi, dan pengujian yang telah dilakukan, dapat disimpulkan bahwa:

1. Katalog *achievement* berhasil dirancang dengan baik di Directus, mencakup dua level (parent dan child) yang memenuhi kebutuhan alur sistem. Struktur koleksi dan field yang dibuat memungkinkan pengelolaan data *achievement* secara fleksibel, termasuk validasi progress dan reward. Hal ini menjawab rumusan masalah pertama tentang perancangan katalog yang sesuai dengan kebutuhan sistem.
2. Sistem *achievement* pada backend Game Portal berjalan sesuai spesifikasi fungsional, mencakup pembuatan katalog, pembaruan progress, dan mekanisme klaim *reward*.
3. Mekanisme validasi dan penanganan error efektif mengelola skenario gagal seperti request tidak valid, kode *achievement* tidak terdaftar, duplikasi data, dan pencapaian melebihi batas maksimum, dengan rollback otomatis saat reward gagal didistribusikan untuk menjaga integritas data.

6.2 Saran

Untuk pengembangan dan pemeliharaan jangka panjang, penulis mengajukan beberapa rekomendasi sebagai berikut:

1. Terapkan *caching* pada *endpoint* katalog untuk mengurangi beban dan meningkatkan kecepatan respon API.
2. Tambahkan variasi *reward* (misalnya badge, item virtual) untuk memperkaya pengalaman pengguna dan meningkatkan motivasi.
3. Kembangkan modul *monitoring* dan *logging* terpusat agar tim dapat memantau performa sistem dan mendeteksi anomali secara cepat.

DAFTAR PUSTAKA

1. Hostinger. 2021.Apa Itu Headless CMS? Ini Penjelasan, Fungsi, dan Contohnya. [ONLINE] Tersedia di: https://www.hostinger.com/id/tutorial/headless-cms. [Diakses 4 Juni 2025].
2. Jetorbit. 2022. Pengertian Headless CMS, Fungsi, dan Contoh Penggunaannya. [ONLINE] Tersedia di: https://www.jetorbit.com/blog/pengertian-headless-cms-fungsi-dan-contoh-penggunaannya/. [Diakses 4 Juni 2025].
3. Directus. 2025. Headless CMS | Directus Docs. [ONLINE] Tersedia di: https://docs.directus.io/use-cases/headless-cms/introduction. [Diakses 4 Juni 2025].
4. Vuexy Digital Agency. 2023. Memaksimalkan Manajemen Konten dengan Directus. [ONLINE] Tersedia di: https://vuexy.id/blog/memaksimalkan-manajemen-konten-dengan-directus. [Diakses 4 Juni 2025].
5. Microsoft Azure. 2025. PlayFab | Microsoft Azure. [ONLINE] Tersedia di: https://azure.microsoft.com/id-id/products/playfab/. [Diakses 4 Juni 2025].
6. Wikipedia. 2025. JavaScript. [ONLINE] Tersedia di: https://en.wikipedia.org/wiki/JavaScript. [Diakses 4 Juni 2025].
7. Wikipedia. 2025. ECMAScript. [ONLINE] Tersedia di: https://en.wikipedia.org/wiki/ECMAScript. [Diakses 4 Juni 2025].
8. Amazon Web Services. 2024. Apa Itu API? [ONLINE] Tersedia di: https://aws.amazon.com/id/what-is/api/. [Diakses 4 Juni 2025].
9. Showwcase. 2023. *Memulai Dengan Unity*. [ONLINE] Tersedia di: https://www.showwcase.com/article/18014/memulai-dengan-unity. [Diakses 4 Juni 2025]

BIODATA PENULIS

Nama	:	Adyuta Prajahita Murdianto
Tempat, Tanggal Lahir	:	Medan, 08 Agustus 2004
Jenis Kelamin	:	Laki-Laki
Agama	:	Islam
Status	:	Belum Menikah
Alamat	:	Jl. Krukah Selatan VIB NO. 16Q, Surabaya
Telepon	:	082112570732
Email	:	adyuta123@gmail.com

PENDIDIKAN FORMAL

2022 – sekarang	:	S1 Teknik Informatika ITS
2019 – 2022	:	SMA Negeri 16 Surabaya
2016 – 2019	:	SMP Negeri 1 Surabaya
2012 – 2016	:	SD Negeri Ketabang 1 Surabaya
2010 – 2012	:	SD Harapan 1 Medan

KEMAMPUAN

- *Programming* (Typescript, Golang, PHP, Python)
- *Database Management* (MySQL, PostgreSQL)
- *Software* Perkantoran (Microsoft Word, Excel, PowerPoint)
- Bahasa (Indonesia, Inggris(A2))

AKADEMIS

Kuliah	: Departemen Informatika – Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember Surabaya
Angkatan	: 2022
Semester	: 6 (Enam)
IPK	: 3.5

[Halaman ini sengaja dikosongkan]