

**INISIALISASI *DATA PERSISTENCE* PADA APLIKASI  
PERMAINAN “NOKIDNAP” SEBAGAI *GAME PROGRESSION*  
MENGUNAKAN *JSON DATA PARSING***

**SKRIPSI**

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah  
Sarjana Ilmu Komputer

**AQSHAL AURELLIO ATHAILLAH**

**201401087**



**PROGRAM STUDI S-1 ILMU KOMPUTER  
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI  
UNIVERSITAS SUMATERA UTARA  
MEDAN  
2024**

**INISIALISASI *DATA PERSISTENCE* PADA APLIKASI  
PERMAINAN “NOKIDNAP” SEBAGAI *GAME PROGRESSION*  
MENGUNAKAN *JSON DATA PARSING***

**SKRIPSI**

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah  
Sarjana Ilmu Komputer

**AQSHAL AURELLIO ATHAILLAH**

**201401087**



**PROGRAM STUDI S-1 ILMU KOMPUTER  
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI  
UNIVERSITAS SUMATERA UTARA  
MEDAN  
2024**

## LEMBAR PERSETUJUAN

**Judul** : INISIALISASI *DATA PERSISTENCE* PADA  
APLIKASI PERMAINAN "NOKIDNAP" SEBAGAI  
*GAME PROGRESSION* MENGGUNAKAN JSON  
*DATA PARSING*

**Kategori** : SKRIPSI

**Nama** : AQSHAL AURELLIO ATHAILLAH

**Nomor Induk Mahasiswa** : 201401087

**Program Studi** : SARJANA (S-1) ILMU KOMPUTER

**Fakultas** : ILMU KOMPUTER DAN TEKNOLOGI INFORMASI  
UNIVERSITAS SUMATERA UTARA

**Komisi Pembimbing** :

**Dosen Pembimbing I**   
Pauzi Ibrahim Nainggolan, S.Komp.,  
M.Sc  
NIP. 198809142020011001

**Dosen Pembimbing II**   
Anandhini Medianty Nababan S. Kom.,  
M. T.  
NIP. 199304132021022001

Diketahui/Disetujui oleh  
Program Studi S-1 Ilmu Komputer



Dr. Amalia S.T., M.T.  
NIP. 197812212014042001

## LEMBAR PERNYATAAN

Yang bertanda tangan dibawah ini:

Nama : Aqshal Aurellio Athaillah

NIM : 201401087

Jurusan : Ilmu Komputer

Fakultas : Ilmu Komputer dan Teknologi Informasi

Menyatakan dengan ini sesungguhnya bahwa skripsi berjudul “Inisialisasi *Data Persistence* pada Aplikasi Permainan “NoKidnap” sebagai *Game Progression* Menggunakan *JSON Data Parsing*” adalah betul-betul karya saya sendiri, bukan plagiat dan tidak dibuatkan oleh orang lain. Skripsi ini saya hasilkan melalui proses penelitian, bimbingan dan diskusi. Semua kutipan yang diperoleh telah disertai identitas sumbernya dengan cara yang sebagaimana lazimnya dalam penulisan karya ilmiah. Apabila di kemudian hari terbukti bahwa pernyataan ini tidak benar, maka saya bersedia menerima sanksi akademik berupa pencabutan skripsi dan gelar yang diperoleh dari skripsi tersebut.

Medan, 12 November 2024

Yang memberi pernyataan



Aqshal Aurellio Athaillah

NIM: 201401087



## ABSTRAK

Perkembangan teknologi yang pesat telah menyebabkan *video game* menjadi suatu media hiburan yang sangat populer di kalangan anak-anak. Popularitas *video game* dapat digunakan sebagai alat bantu pembelajaran dengan cara menerapkan konsep gamifikasi ke dalam *video game* agar proses belajar dapat menjadi lebih menyenangkan. Salah satu aplikasi yang menerapkan konsep gamifikasi tersebut adalah aplikasi permainan edukatif “NoKidnap” yang memiliki fokus terhadap pencegahan penculikan terhadap anak. Dalam permainan ini, pemain mengendalikan karakter bernama Leafy sebagai karakter utama menuju tujuan yang telah ditentukan sembari mengumpulkan buku catatan yang tersebar di dalam level. Aplikasi ini memiliki berbagai sistem, tetapi sistem penyimpanan kemajuan permainan belum ada di dalam aplikasi. Sistem penyimpanan kemajuan tersebut merupakan salah satu fitur yang ketersediaannya meningkatkan kualitas dari permainan menurut *video game heuristics*. Penelitian ini telah berhasil mengimplementasikan sistem *data persistence* terhadap aplikasi permainan “NoKidnap” guna meningkatkan kualitas aplikasi serta meningkatkan *user experience* pemain yang menggunakan aplikasi tersebut. Format *file* yang digunakan untuk mengimplementasikan sistem penyimpanan adalah JSON *data persistence*, dimana *file* JSON yang diciptakan oleh sistem disimpan di dalam game *files* secara lokal.

**Kata kunci :** “NoKidnap”, *Save System*, JSON *data persistence*, *game*

## **DATA PERSISTENCE INITIALIZATION IN “NOKIDNAP” GAME APPLICATION FOR GAME PROGRESSION USING JSON DATA PARSING**

### **ABSTRACT**

Rapid growth of technology has caused video games to become one of the most popular entertainment media for kids. The popularity that video games has can be used to help education by implementing gamification into video games to make learning more fun. One application that implements gamification is “NoKidnap” that focuses on preventing kidnapping. Inside the game, the player controls a character named Leafy as the main character towards a given goal while collecting books scattered across the level. This application has multiple systems, but a game progress saving system has not yet been implemented into the app. Game progress saving system is one of the systems that the availability of it boosts the quality of a given video game according to video game heuristics. This research has successfully implemented the game progression save system into the app “NoKidnap” to improve the quality of the app and enhance the user experience of the users that play the game. The file format that is used to implement the save system is JSON data persistence, where the file that is created by the system is saved within the game data locally

**Keywords :** “NoKidnap”, *Save System*, *JSON data persistence*, *game*

## KATA PENGANTAR

Puji syukur peneliti ucapkan kepada Allah SWT atas rahmat dan karunia-Nya peneliti dapat menyelesaikan skripsi yang berjudul **“Inisialisasi Data Persistence pada Aplikasi Permainan “NoKidnap” sebagai Game Progression Menggunakan JSON Data Parsing”**. Pada kesempatan ini peneliti mengucapkan rasa terima kasih kepada semua pihak yang ikut membantu dalam proses pembuatan skripsi ini, terutama kepada:

1. Tuhan Yang Maha Esa yang selalu melimpahkan kasih sukacita, berkat, dan rahmat-NYA kepada penulis, sehingga penulis dapat menyelesaikan skripsi ini.
2. Ayahanda Ivandi Darma Putra, Ibunda Sofianita dan seluruh keluarga yang selalu mendoakan, memberi perhatian serta kasih sayang kepada penulis.
3. Ibu Dr. Amalia S.T., M.T. selaku Ketua Program Studi S1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
4. Ibu Sri Melvani Hardi S.Kom., M.Kom selaku Sekretaris Program Studi S1 Ilmu Komputer Universitas Sumatera Utara.
5. Bapak Pauzi Ibrahim Nainggolan, S.Komp., M.Sc selaku dosen pembimbing I dan Ibu Anandhini Medianty Nababan S. Kom., M. T. selaku dosen pembimbing II yang telah memberikan bimbingan, kritik, dan saran kepada penulis dalam menyelesaikan skripsi ini.
6. Seluruh tenaga pengajar, *staff*, pegawai, serta seluruh Civitas Akademika di Fakultas Ilmu Komputer dan Teknologi Informasi USU.
7. Stambuk 2019 dan 2020, terutama Petrus Marcelino H. Tampubolon dan Venerio Uvandy, yang telah memberikan semangat, nasihat dan pengetahuan, serta sebagai teman diskusi.
8. Seluruh teman-teman yang tidak dapat saya sebutkan namanya satu persatu yang telah memberikan bantuan moral, nasihat, semangat, serta berbagi canda tawa kepada penulis.
9. Semua pihak yang terlibat langsung atau tidak langsung yang tidak dapat dituliskan satu per satu.

Semoga Allah SWT selalu memberikan kasih sayangnya kepada semua pihak yang telah berperan penting bagi penulis dalam menyelesaikan skripsi ini. semoga skripsi ini bermanfaat untuk pribadi maupun pembaca untuk mencerdaskan kehidupan bangsa.

Medan, 12 November 2024

Penulis





## DAFTAR ISI

LEMBAR PERSETUJUAN .....	i
LEMBAR PERNYATAAN.....	ii
ABSTRAK.....	iii
ABSTRACT.....	iv
KATA PENGANTAR .....	v
DAFTAR ISI.....	vii
DAFTAR TABEL.....	ix
DAFTAR GAMBAR .....	x
BAB I.....	1
1.1. Latar Belakang.....	1
1.2. Rumusan Masalah .....	3
1.3. Batasan Masalah .....	3
1.4. Tujuan Penelitian.....	3
1.5. Manfaat Penelitian.....	3
1.6. Metodologi Penelitian .....	4
1.7. Penelitian Relevan .....	4
BAB II.....	6
2.1 <i>Data Persistence</i> .....	6
2.2 <i>Game Progression</i> .....	6
2.3 JSON.....	6
2.4 NoKidnap .....	6
2.5 Serialisasi dan Deserialisasi.....	7
2.6 <i>User Experience</i> .....	7
2.7 <i>Save Point/Checkpoint</i> .....	7
2.8 <i>Video game heuristics</i> .....	8
2.9 <i>Game State</i> .....	8
2.10 <i>Quality Assurance</i> .....	9
BAB III .....	10
3.1 Metodologi.....	10

3.2 Analisis Sistem.....	11
3.3 Perancangan Sistem .....	13
3.4 Rencana Uji Coba dan Evaluasi.....	26
BAB IV .....	27
4.1 Hasil Implementasi Sistem <i>Data Persistence</i> .....	27
4.2 Hasil Uji Program .....	30
4.3 Hasil Eksperimen .....	41
BAB V .....	49
DAFTAR PUSTAKA .....	50



## DAFTAR TABEL

Tabel 4.1 Hasil pengujian sistem data persistence.....	30
Tabel 4.2 Pertanyaan yang diberikan kepada responden .....	41
Tabel 4.3 Jawaban Responden Terhadap Pertanyaan Ke-1 .....	42
Tabel 4.4 Jawaban Responden Terhadap Pertanyaan Ke-2 .....	43
Tabel 4.5 Jawaban Responden Terhadap Pertanyaan Ke-3 .....	43
Tabel 4.6 Jawaban Responden Terhadap Pertanyaan Ke-4 .....	44
Tabel 4.7 Jawaban Responden Terhadap Pertanyaan Ke-5 .....	45
Tabel 4.8 Jawaban Responden Terhadap Pertanyaan Ke-6 .....	46
Tabel 4.9 Tabel total skor untuk setiap pertanyaan kuisioner.....	46



## DAFTAR GAMBAR

Gambar 3.1 Metodologi Penelitian .....	10
Gambar 3.2 Flowchart Rancangan Sistem .....	13
Gambar 3.3 Tampilan inspector objek manager .....	13
Gambar 3.4 Tampilan script GameData .....	15
Gambar 3.5 Tampilan script SerializableDictionary .....	16
Gambar 3.6 Insialisasi DataPersistenceManager .....	17
Gambar 3.7 Interface IDataPersistence .....	18
Gambar 3.8 Inisialisasi FileDataHandler .....	18
Gambar 3.9 Awake dan Start method DataPersistenceManager .....	19
Gambar 3.10 Method FindAllDataPersistenceObjects .....	19
Gambar 3.11 Method NewGame, SaveGame, LoadGame pada DataPersistenceManager .....	20
Gambar 3.12 Method RestartGame pada DataPersistenceManager .....	20
Gambar 3.13 Method Save pada FileDataHandler .....	21
Gambar 3.14 Method load pada FileDataHandler .....	21
Gambar 3.15 Method Reset pada FileDataHandler .....	22
Gambar 3.16 Implementasi pada HealthManager .....	22
Gambar 3.17 Implementasi pada EnergyScript .....	23
Gambar 3.18 Implementasi pada CoinManager .....	23
Gambar 3.19 Implementasi pada TimeManager .....	24
Gambar 3.20 Implementasi pada ContentBookManager .....	24
Gambar 3.21 Implementasi pada ThirdPersonController .....	25
Gambar 3.22 Implementasi pada CoinSound .....	25
Gambar 3.23 Method GenerateGuid pada CoinSound .....	26
Gambar 4.1 Tampilan pemilihan level .....	27
Gambar 4.2 Tampilan UI dalam level .....	27
Gambar 4.3 Tampilan catatan awal .....	28
Gambar 4.4 Tampilan catatan setelah mendapat beberapa buku .....	28
Gambar 4.5 Tampilan isi file JSON .....	29
Gambar 4.6 Tampilan isi file JSON encrypted .....	29

Gambar 4.7 Tampilan folder penyimpanan lokal aplikasi setelah reinstall .....	32
Gambar 4.8 Tampilan awal level permainan .....	32
Gambar 4.9 Tampilan folder penyimpanan dengan file berhasil tercipta.....	33
Gambar 4.10 Isi file JSON saat pertama diciptakan .....	33
Gambar 4.11 Tampilan aplikasi dengan variabel yang telah diubah .....	34
Gambar 4.12 Tampilan pause menu .....	34
Gambar 4.13 Isi file JSON yang telah berubah .....	35
Gambar 4.14 Tampilan aplikasi ketika kembali memasuki level .....	36
Gambar 4.15 Pintu rumah leafy sebagai akhir dari level .....	37
Gambar 4.16 Isi file JSON setelah mencapai akhir level .....	37
Gambar 4.17 Tampilan permainan sebelum ditutup paksa.....	38
Gambar 4.18 Force stop aplikasi untuk menguji sistem .....	39
Gambar 4.19 Tampilan permainan setelah dibuka kembali.....	39
Gambar 4.20 Tampilan penyimpanan aplikasi .....	40
Gambar 4.21 Tampilan pengaturan di menu utama dengan tombol reset .....	40
Gambar 4.22 Tampilan isi penyimpanan lokal setelah aplikasi di reset.....	41
Gambar 4.23 Garis Kontinum Analisis Minimum Score Index (MSI).....	48



# BAB I

## PENDAHULUAN

### 1.1. Latar Belakang

Pada era perkembangan teknologi saat ini, kita memiliki kebebasan untuk memilih antara berbagai macam konten digital yang dapat memberikan akses terhadap informasi yang kita inginkan. Konten digital tersebut mencakup media sosial, berita terkini, dan konten hiburan. Sejak terjadinya era pandemi COVID-19, konten hiburan yang tersedia secara digital telah berkembang secara pesat. Salah satu bagian dari hiburan digital yang banyak diminati oleh pengguna adalah *video game* (Renu, 2021).

*Video game* secara umum sering dikenal sebagai suatu sarana hiburan, tetapi *video game* juga bisa digunakan sebagai suatu sarana pembelajaran dengan cara menerapkan gamifikasi terhadap materi-materi dengan konteks non-*game* yang ingin disajikan kepada pengguna (Deterding et al., 2011), sehingga proses pembelajaran dapat terasa lebih menyenangkan ketika diikuti oleh semua kalangan.

Dari semua kalangan umur, anak-anak lebih cenderung memilih untuk menghabiskan waktunya bermain *video game* dari pada bermain diluar (Tatli, 2018). Hal ini disebabkan oleh pesatnya perkembangan teknologi di beberapa tahun terakhir yang mengakibatkan anak-anak lebih tertarik untuk bermain *video game* dibandingkan dengan permainan tradisional, seperti permainan balok dan mobil-mobilan. Sehingga dalam konteks edukasi terhadap anak, konsep gamifikasi dapat diterapkan untuk membantu proses pembelajaran.

Gamifikasi adalah suatu metode yang menggunakan elemen *game design*, *game mechanics*, dan *game thinking* dalam suatu lingkungan non-*game* untuk memotivasi partisipan (Al-Azawi et al., 2016). Gamifikasi yang bersifat edukasional merupakan suatu cara mengajar yang memerlukan pelajar untuk berpartisipasi di dalam kompetisi dengan peraturan yang sudah ditetapkan (Sheldon, 2020). Dalam mewujudkan kontribusi pengajar di dalam permainan dengan peraturan yang sudah ditetapkan, diperlukan sebuah aplikasi yang menyediakan materi pembelajaran yang interaktif tanpa memerlukan ketersediaan pengajar secara langsung.

Aplikasi permainan “NoKidnap” merupakan suatu aplikasi pembelajaran mengenai pengetahuan umum tentang keselamatan anak dengan fokus untuk mencegah penculikan saat berinteraksi dengan dunia luar, dimana konsep gamifikasi telah

diaplikasikan sehingga sistem pembelajaran memiliki unsur-unsur *video game* seperti skor, kondisi menang, kondisi kalah, dan *game progression*.

*Game progression* adalah salah satu struktur dasar permainan dimana pemain harus menyelesaikan beberapa urutan tantangan yang telah disediakan oleh *developer* untuk dapat menyelesaikan permainan (Juul, 2002). Pada aplikasi permainan “NoKidnap”, pemain harus mengambil buku sebagai *collectible* untuk dapat menyelesaikan suatu level. Diperlukan sistem *data persistence* agar pemain dapat menyimpan *game progression* mereka.

*Data persistence* adalah suatu sifat *data* dimana *data* akan tetap ada walaupun aplikasi yang menciptakan *data* tersebut telah ditutup. Agar *data* bisa tetap ada, *data* tersebut harus disimpan kedalam sebuah *non-volatile storage*, sebuah tipe penyimpanan yang dapat menyimpan informasi dalam jangka waktu panjang.

Metode yang akan digunakan untuk mengimplementasikan *data persistence* kedalam aplikasi permainan “NoKidnap” adalah metode *data parsing*. *data parsing* memiliki beberapa teknik berbeda yang dapat digunakan, seperti XML *data parsing* dan JSON *data parsing*.

Teknik parsing tersebut masing-masing memiliki kelebihan dan kekurangannya tersendiri. Kelebihan teknik XML parsing adalah teknik ini dapat menyimpan komentar dan metadata sehingga *data* yang dimiliki lebih lengkap, sedangkan untuk teknik JSON parsing memiliki sintaks yang lebih mudah dibaca dan ditulis oleh program dan menggunakan ruang penyimpanan yang lebih kecil (Mazur, 2023).

Pada aplikasi permainan “NoKidnap”, *game progression* yang dimaksud adalah pemain harus melakukan eksplorasi di dalam level yang sudah disediakan untuk mencari buku yang tersebar di dalam level. Saat pemain menemukan dan mengambil suatu buku, aplikasi akan menampilkan pertanyaan mengenai pengetahuan umum tentang keselamatan. Pemain akan mendapatkan skor tergantung jawaban yang diberikan kepada pertanyaan dan penjelasan tentang materi tersebut akan ditambahkan kedalam catatan pengguna yang tersedia di dalam aplikasi. *Data* tersebut tidak memerlukan metadata yang kompleks, sehingga teknik JSON *data parsing* lebih sesuai untuk diimplementasikan dalam aplikasi permainan tersebut.

Fitur *data persistence* belum ada di dalam aplikasi permainan edukatif “NoKidnap”, sehingga pengguna yang melakukan pembelajaran melalui aplikasi tersebut masih harus mengulang sesi pembelajaran mereka setiap kali membuka sesi

permainan baru. Perlu di inisialisasikan suatu sistem *data persistence* untuk membantu *game progression* agar pengalaman bermain pengguna dapat ditingkatkan dan aplikasi permainan “NoKidnap” lebih efektif dalam memberikan materi pembelajaran.

## 1.2. Rumusan Masalah

Tidak adanya implementasi *data persistence* berbasis *load system* menggunakan algoritma JSON *data parsing* dalam fitur *game state progression* pada aplikasi permainan “NoKidnap”.

## 1.3. Batasan Masalah

Beberapa batasan masalah dalam penelitian ini sebagai berikut.

1. Algoritma *data persistence* diciptakan menggunakan JSON *parsing* di dalam Unity Engine.
2. *File* yang mengandung data *game progression* pemain akan disimpan secara lokal
3. Referensi *data* yang digunakan diambil dari jurnal dan buku yang berkaitan dengan JSON *parsing*, *data persistence*, *game progression*, dan *game user experience*.
4. Penyusunan program hanya ditujukan untuk anak-anak usia 7-11 tahun.
5. Program dirancang dengan menggunakan bahasa C#.
6. Program yang dirancang berbasis *mobile* Android.

## 1.4. Tujuan Penelitian

Penelitian ini bertujuan untuk menginisialisasi fitur *data persistence* dalam aplikasi permainan edukatif “NoKidnap” guna meningkatkan kualitas *user experience* bagi pengguna serta memenuhi kebutuhan dari *video game heuristics*. Hal ini mencakup upaya untuk meningkatkan efisiensi sistem materi pelajaran untuk pengguna dan menciptakan *save point* untuk menserialisasi berbagai *game state* yang tersedia pada aplikasi dan mampu untuk dideserialisasikan di kemudian waktu. Selain itu, penelitian ini juga bertujuan untuk memberikan kontribusi terhadap penelitian *data persistence* terhadap berbagai *game state* dalam konteks permainan edukatif.

## 1.5. Manfaat Penelitian

Adapun manfaat yang diharapkan dari penelitian ini yaitu sebagai berikut.

1. Meningkatkan kualitas aplikasi permainan edukatif “NoKidnap” dengan cara memenuhi salah satu poin dari *video game heuristics* mengenai *gameplay/game story* yaitu pemain harus mampu menyimpan permainan dalam berbagai *game state*



yang berbeda dan dapat dengan mudah menghidupkan atau mematikan permainan tersebut.

2. Meningkatkan *user experience* dari pengguna dengan cara memberikan pengguna opsi untuk menyimpan permainan pada *game state* yang telah ditentukan, sehingga pemain tidak merasa terpaksa dalam memainkan permainan.

## **1.6. Metodologi Penelitian**

### **1. Studi Pustaka**

Dalam penelitian ini, penulis menggunakan metode studi pustaka atau studi literatur untuk meninjau, dan mengumpulkan berbagai referensi dari buku-buku, jurnal, laporan-laporan dan tinjauan pustaka lainnya yang memiliki hubungan dengan penelitian yang akan dilakukan.

### **2. Analisis dan Perancangan Sistem**

Analisis sistem dilakukan menggunakan flowchart dan design thinking, serta perancangan sistem dilakukan menggunakan metode Game Development Life Cycle (GDLC).

### **3. Implementasi Sistem**

Implementasi dari sistem yang akan dilakukan dibangun sesuai dengan perancangan yang dibuat dengan bahasa pemrograman C# yang berbasis mobile menggunakan arsitektur Unity Engine.

### **4. Pengujian Sistem**

Sistem yang telah dibuat akan diuji dengan metode functional testing untuk melihat dan memastikan bahwa sistem tersebut berjalan dengan semestinya.

### **5. Dokumentasi**

Pada tahap ini, penelitian yang telah dilakukan, didokumentasikan mulai dari tahap analisa sampai kepada pengujian dalam bentuk skripsi.

## **1.7. Penelitian Relevan**

1. Berdasarkan hasil analisis penelitian yang telah dilakukan oleh Vaz Salles et al. (2009) dengan judul “*An evaluation of checkpoint recovery for massively multiplayer online games*” ditemukan bahwa teknik penyimpanan berbasis *checkpoint* untuk *main-memory* pada game *Massively Multiplayer Online* (MMO) dapat memberikan peningkatan waktu akses ke server yang berdampak baik terhadap permainan karena permainan MMO dimainkan oleh banyak orang secara bersamaan sehingga waktu akses merupakan faktor penting untuk UX pemain.

Perbedaan penelitian tersebut dengan penelitian ini adalah penyimpanan yang diusulkan dalam penelitian ini bersifat lokal.

2. Berdasarkan hasil penelitian yang dilakukan oleh Johanson et al. (2023) dengan judul “*If at First You Don't Succeed: Helping Players Make Progress in Games with Breaks and Checkpoints*” ditemukan bahwa *checkpoint* merupakan suatu metode yang efektif untuk meningkatkan performa pemain dalam permainan dan membantu pemain untuk mencapai kemajuan pada level tersebut. Perbedaan penelitian tersebut dengan penelitian ini adalah genre permainan yang digunakan pada penelitian tersebut adalah suatu permainan 2D *platformer* sedangkan pada penelitian ini menggunakan permainan edukatif “NoKidnap” yang bersifat *sandbox*, dengan keperluan *checkpoint* yang berbeda.





## **BAB II**

### **TINJAUAN PUSTAKA**

#### **2.1 Data Persistence**

*Data Persistence* adalah suatu sifat *data* dimana *data* tersebut tidak akan hilang walaupun aplikasi yang menciptakan *data* tersebut sudah ditutup (MongoDB, n.d.). *Data* harus disimpan di dalam memori *non-volatile* agar dapat memiliki sifat persisten. Ketika *data* bersifat persisten, maka *data* tersebut dapat diambil dan dibaca kembali di lain waktu ketika aplikasi tersebut kembali dibuka tanpa adanya kehilangan *data*.

#### **2.2 Game Progression**

Konsep *Game Progression* merupakan salah satu struktur dasar permainan dimana pemain harus menyelesaikan beberapa urutan tantangan yang telah disediakan oleh *developer* untuk dapat menyelesaikan permainan (Juul, 2002).

Permainan yang menggunakan *progression* sebagai struktur utamanya cenderung bersifat sinematik (Marques Maranhão et al., 2016), dikarenakan pemain akan mengalami sebuah cerita bersama avatar mereka yang ada di dalam permainan.

#### **2.3 JSON**

*Java Script Object Notation* (JSON) merupakan salah satu *data exchange format* yang menggunakan bahasa alami untuk menyimpan dan menyalurkan objek *data* yang memiliki nilai atribut dan *array*, yang sering digunakan dalam pertukaran *data* elektronik seperti dalam aplikasi web dan *server* (JSON, 2024).

Tipe *data* yang dapat digunakan di dalam sebuah *file* JSON adalah *number*, *string*, *boolean*, *array*, *object*, dan *null*. Sebuah *file* JSON akan mengabaikan *whitespace* yang terdapat di dalam *file*, kecuali *whitespace* tersebut berada di dalam *string*.

Penelitian ini menggunakan JSON sebagai metode pengimplementasian *data persistence* dikarenakan kemudahan optimasi penulisan oleh program dan kecilnya ukuran penyimpanan yang diperlukan dibandingkan dengan *data format* populer lainnya seperti XML (Zunke & Souza, 2014).

#### **2.4 NoKidnap**

“NoKidnap” adalah sebuah aplikasi permainan edukatif yang menawarkan pengalaman bermain yang unik dengan fokus pencegahan penculikan terhadap anak dengan target pemain anak usia 7-11 tahun. Pemain ditantang untuk mencapai tujuan dari misi tanpa tertangkap oleh karakter musuh yang telah dilengkapi dengan AI untuk mengejar pemain secara cerdas.

Pada permainan ini, terdapat beberapa level yang tersedia dimana masing-masing level memiliki buku catatan dan skor yang berbeda. Saat ini, pemain harus menyelesaikan seluruh level yang ada di dalam permainan dalam satu sesi bermain untuk membuka semua konten pembelajaran yang tersedia di dalam aplikasi karena aplikasi tidak memiliki sistem penyimpanan *data* pemain. Hal ini akan kembali seperti *game state* awal dimana pemain tidak memiliki skor dan belum membuka buku catatan apapun di dalam aplikasi ketika pemain keluar dari permainan.

## **2.5 Serialisasi dan Deserialisasi**

Serialisasi adalah proses mengubah *state* sebuah objek menjadi bentuk yang mampu untuk di disimpan dan dipindahkan, sedangkan Deserialisasi adalah proses kebalikan dari serialisasi, yaitu mengubah *state data* kembali menjadi objek yang mampu dibaca oleh program (Warren et al., 2023).

Pada penelitian ini, sistem *data persistence* akan melakukan proses serialisasi ketika pemain menyimpan permainan mereka yang akan mengubah objek C# yang digunakan oleh permainan kedalam bentuk JSON agar mampu disimpan. Saat pemain ingin melanjutkan permainan mereka, aplikasi akan melakukan proses deserialisasi yang akan mengubah *file* dalam bentuk JSON kembali menjadi objek C# awal yang kemudian digunakan untuk mengembalikan kemajuan pemain dalam NoKidnap.

## **2.6 User Experience**

*User experience* (UX) mencakup seluruh aspek tentang bagaimana pengguna berinteraksi dengan sebuah produk dan bersifat subjektif (Hassenzahl, 2018). UX yang ingin diaplikasikan oleh seorang *developer* kedalam aplikasinya mungkin berbeda dengan UX yang dialami oleh penggunanya dikarenakan perbedaan dari setiap individu.

Menurut Bernhaupt (2015), terdapat 4 tipe percobaan untuk mengevaluasi UX yaitu *user-oriented method*, *automated method*, *expert-oriented method*, dan *games-specific approaches*. Penelitian ini menggunakan *expert-oriented method* untuk melakukan evaluasi UX terhadap sistem yang dikembangkan yaitu dengan cara menerapkan konsep *video game heuristics*.

## **2.7 Save Point/Checkpoint**

*Save point* atau *checkpoint* merupakan poin-poin tertentu yang memungkinkan pemain untuk menyimpan kemajuan mereka dalam sebuah permainan. Semakin besar ukuran permainan, maka *save point* yang diperlukan permainan juga akan bertambah (Consalvo & Dutton, 2006).

Dalam penelitian ini, *save point* yang akan tersedia untuk pemain adalah ketika pemain menyelesaikan suatu level, ketika pemain kehabisan nyawa, dan ketika pemain keluar dari permainan menggunakan menu *pause*.

## **2.8 Video game heuristics**

Hochleitner et al. (2015) mengusulkan *video game heuristics* sebagai pedoman dalam membangun UX dalam permainan. Terdapat 2 bagian utama dalam heuristik ini yaitu *gameplay/game story* dan *visual interface*. Dalam bagian *gameplay/game story*, terdapat subbagian tentang *control* dimana salah satu poinnya adalah "pemain harus mampu menyimpan permainan dalam berbagai state yang berbeda (kecuali game arkade) dan juga mampu menghidupkan dan mematikan permainan dengan mudah".

Aplikasi permainan "NoKidnap" belum memenuhi poin heuristik yang telah disebutkan sehingga UX dalam aplikasi ini masih bersifat belum optimal dan dapat dikembangkan dengan cara menambahkan *data persistence* sehingga pemain dapat menyimpan kemajuan dalam permainan mereka.

## **2.9 Game State**

*Game state* adalah fase-fase berbeda dari suatu permainan yang diperlukan sehingga permainan dapat diselesaikan (Coregames, n.d.). Suatu permainan harus mampu melakukan perpindahan dari suatu *game state* ke *game state* lainnya agar pemain dapat mengalami kemajuan dalam proses menyelesaikan permainan.

Dalam aplikasi permainan "NoKidnap", terdapat *game state* dari awal pemain membuka permainan hingga pemain keluar dari permainan. Saat pemain membuka permainan, aplikasi masuk kedalam *state* menu utama dimana pemain bisa memilih untuk mengakses layar pemilihan level atau mengakses pengaturan.

Setelah pemain masuk kedalam suatu level, *game state* aplikasi berubah dari *state* menu utama menjadi *gameplay state*, dimana pemain mendapatkan kendali penuh terhadap karakter yang ada di dalam aplikasi dengan tujuan untuk menyelesaikan level tersebut.

Ketika pemain menyelesaikan suatu level, *game state* akan berubah dari *gameplay state* menjadi *level end state*, dimana aplikasi akan menyimpan performa pemain pada level tersebut dan diserialisasikan kedalam *file* JSON. Pada *state* ini pemain dapat memilih untuk mengulang level yang barusan dimainkannya untuk kembali ke *gameplay state* atau kembali ke menu utama untuk kembali ke *state* menu utama.



## 2.10 Quality Assurance

*Quality assurance* atau QA secara umum dapat didefinisikan sebagai sebuah proses untuk menentukan apakah sebuah produk atau layanan sudah memenuhi persyaratan yang dibutuhkan. Dalam konteks pengembangan *game*, QA adalah proses yang dilakukan untuk menemukan ketidakkonsistenan, *glitch*, atau *bug* dalam sebuah *game*.

Proses menjalankan QA terhadap suatu permainan pada umumnya memiliki beberapa tahapan, yaitu :

1. Identifikasi Masalah : Saat aplikasi tidak menjalankan perintah pengujian coba sesuai dengan apa yang seharusnya terjadi, atau ketika aplikasi berhenti tiba-tiba, pengujian harus mencatat apa yang terjadi agar masalah dapat diidentifikasi.
2. Pelaporan Masalah : Pengujian coba kemudian melaporkan temuannya kepada *developer* agar masalah yang telah ditemukan di dalam program dapat di perbaiki.
3. Analisis Masalah : Pada tahap ini, *developer* aplikasi menganalisis program untuk mencari sumber dari masalah yang telah dilaporkan.
4. Konfirmasi : Ketika *developer* telah menyelesaikan perbaikan terhadap aplikasi, pengujian harus memeriksa kembali untuk memastikan kesalahan tersebut tidak terjadi kembali

Terdapat beberapa metode yang dapat dilakukan agar dapat mengidentifikasi masalah yang mungkin ada di dalam sebuah *game*, dimana metode QA yang sering digunakan adalah *black box testing* dan *white box testing*. Pengujian *black box* berfokus terhadap fungsionalitas *game*, seperti tampilan antar muka, tampilan menu dalam permainan dan fungsi tombol yang ada diuji tanpa perlu memahami kodingan internalnya terlebih dahulu. Pengujian *white box* berfokus pada struktur internal dan koding dari aplikasi yang diuji untuk memverifikasi kegunaan input output dari sistem permainan, oleh sebab itu pengujian yang melakukan pengujian secara *white box* perlu memahami kode sumber dari aplikasi agar dapat melakukan pengujian.

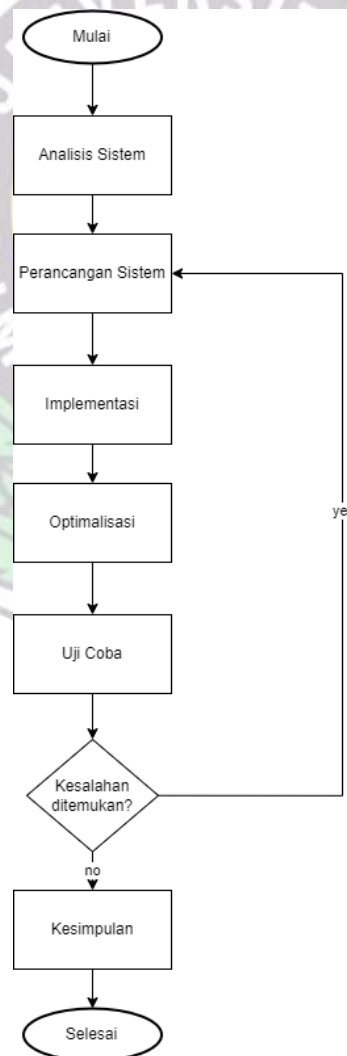
## BAB III

### ANALISIS DAN PERANCANGAN

Bab ini akan menjelaskan analisis yang telah dilakukan seperti metodologi dan tahap perancangan sistem *data persistence* pada aplikasi permainan “NoKidnap”.

#### 3.1 Metodologi

Dalam penelitian ini, metode yang digunakan adalah metode penelitian eksperimental dimana metode pengembangan sistem permainan *Game Development Life Cycle* (GDLC) telah diterapkan. Metodologi ini memungkinkan sistem *data persistence* untuk di implementasikan, dioptimalisasikan, dan dilakukan *testing* pada aplikasi permainan edukatif “NoKidnap”



*Gambar 3.1 Metodologi Penelitian*



### 3.2 Analisis Sistem

Tahap analisis sistem merupakan suatu tahapan penting dalam melakukan penelitian untuk membantu alur perencanaan rancangan suatu sistem. Pada tahap ini, terdapat dua tipe analisa yang perlu dilakukan terhadap sistem, yaitu analisis masalah dan analisis kebutuhan. Analisis masalah dilakukan guna memahami permasalahan yang ada pada aplikasi untuk di selesaikan menggunakan sistem yang dibuat. Analisis kebutuhan dilakukan agar peneliti mengetahui kebutuhan aplikasi yang akan disediakan oleh sistem.

#### 3.2.1 Analisis Masalah

Aplikasi permainan edukatif “NoKidnap” adalah permainan edukatif yang bertujuan untuk memberikan materi kepada pemainnya menggunakan aspek *video game*. Dalam game tersebut, pemain mengendalikan karakter bernama Leafy yang berperan sebagai *player character* yang bertujuan untuk pulang dengan selamat ke rumah sekaligus mengambil *pickup* buku yang tersedia didalam level untuk mendapatkan skor. Dalam versi aplikasi “NoKidnap” sekarang, sistem *data persistence* tidak tersedia yang menyebabkan seluruh kegiatan pemain seperti skor tertinggi dalam suatu level tidak dapat tersimpan sehingga *progress* pemain akan hilang ketika pemain keluar dari aplikasi. Kurangnya sistem penyimpanan *data* dalam aplikasi “NoKidnap” sangat menghambat kemampuan aplikasi dalam memberikan materi kepada pemainnya, dimana pemain tidak bisa berhenti dari suatu sesi pembelajaran untuk istirahat dikarenakan semua kemajuan yang telah dicapai akan hilang saat aplikasi ditutup.

#### 3.2.2 Analisis Kebutuhan

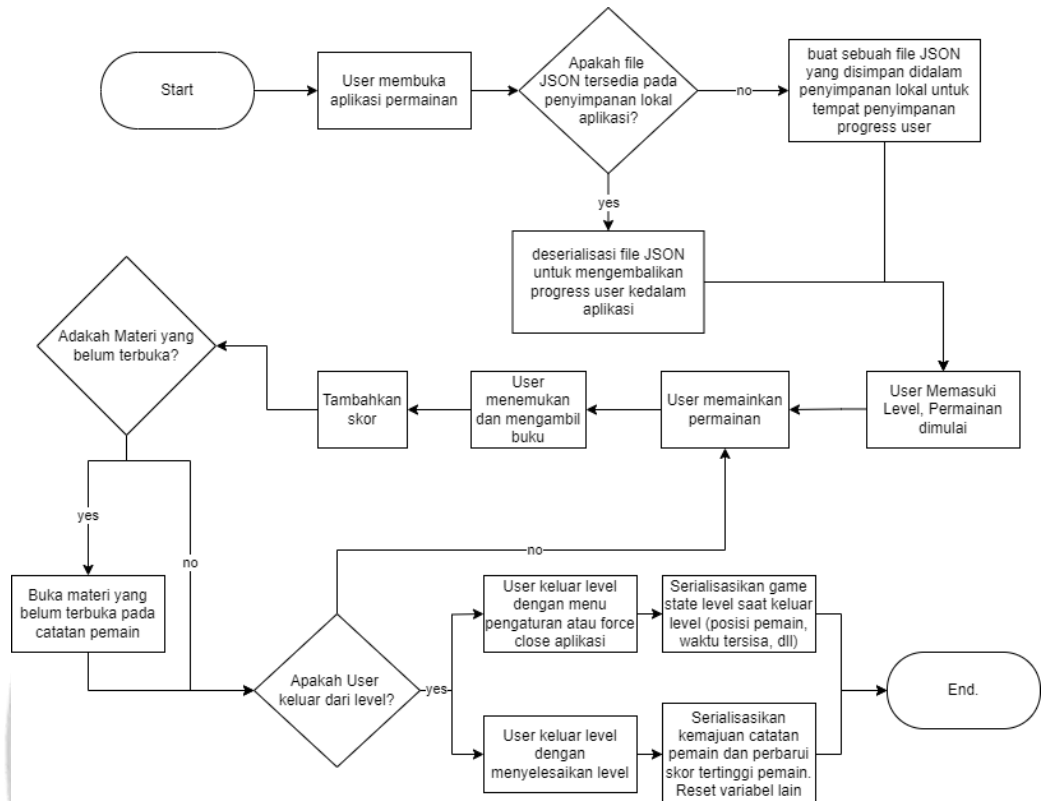
Agar efektifitas aplikasi “NoKidnap” untuk mencapai tujuannya yaitu sebagai aplikasi permainan interaktif untuk anak-anak yang dapat mengajarkan mereka tentang materi anti penculikan dapat ditingkatkan, perlu diimplementasikan suatu sistem *data persistence* yang akan membantu *developer* dalam mencapai tujuan mereka serta memungkinkan pemain untuk

menyerap materi didalam aplikasi dengan kecepatan pembelajaran yang nyaman. Fitur yang dibutuhkan oleh sistem *data persistence* adalah:

1. *Level Progression*; Ketika pemain sedang bermain suatu level, mereka dapat keluar dari aplikasi kapanpun dan status level tersebut seperti berapa *health point* mereka yang tersisa, berapa *pickup* buku yang telah mereka dapatkan di sesi tersebut, dan lokasi terakhir mereka sebelum keluar level akan tersimpan agar dapat dilanjutkan di lain waktu.
2. *Game Progression*; Materi yang telah dibuka dan dipelajari oleh pemain akan terus tersedia dalam permainan, sehingga pemain tidak dipaksa untuk membuka materi yang telah mereka dapatkan dalam sesi bermain sebelumnya, tetapi akan tetap ada jika pemain ingin mempelajari kembali materi tersebut.
3. *Robustness*; Dalam kondisi dimana terjadi suatu *system error* atau *user error*, sistem *data persistence* harus bersifat *robust*, dimana sistem tidak boleh mengalami eror yang menyebabkan pemain kehilangan kemajuan mereka dalam aplikasi.
4. *Maintainability*; Saat *developer* aplikasi “NoKidnap” ingin menambahkan level baru, sistem *data persistence* harus mampu diinisialisasikan dalam level baru tersebut dengan mudah.

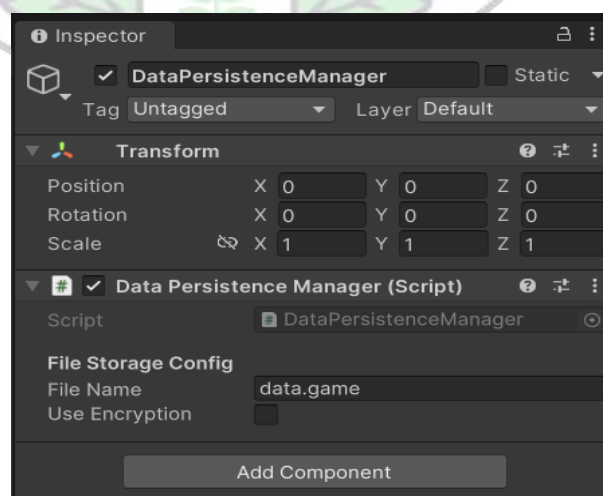
### 3.3 Perancangan Sistem

#### 3.3.1 Perancangan Arsitektur Sistem



Gambar 3.2 Flowchart Rancangan Sistem

Sistem *data persistence* yang akan diimplementasikan dalam aplikasi “NoKidnap” mencakup suatu objek dengan *script* “*DataPersistenceManager*” sebagai objek yang mengatur seluruh proses *save* dan *load data* pada sebuah scene yang ada di dalam permainan.



Gambar 3.3 Tampilan inspector objek manager

Saat pemain memulai permainan untuk pertama kalinya, objek *“DataPersistenceManager”* memanggil fungsi untuk menciptakan suatu *file* dengan nama yang bisa diatur pada field *“File Name”*. *File* ini merupakan *file* tempat penyimpanan *data* permainan yang menggunakan format JSON.

Apabila terdapat *data* yang telah tersimpan di dalam *file* JSON tersebut sebelumnya, objek *“DataPersistenceManager”* akan berkomunikasi dengan objek objek lain yang berada pada scene tersebut untuk mengatur proses *data persistence*. Objek objek UI dan *game state* dari permainan yang berkomunikasi dengan *“DataPersistenceManager”* adalah:

1. *Health system* yang mengatur banyaknya kesalahan yang boleh pemain lakukan sebelum permainan memunculkan layar *game over*
2. *Energy system* yang mengecek energi pemain sehingga karakter pemain bisa melakukan *sprint*
3. *Points system* yang menghitung berapa banyak buku yang dijawab dengan benar oleh pemain dan memberikan poin
4. *Randomly generated book placement system*, dimana pada suatu level yang sama, buku yang muncul untuk diambil oleh pemain akan terletak di lokasi yang berbeda setiap kali level tersebut dimainkan
5. *Player’s notebook*, berisikan materi yang telah didapatkan oleh pemain dan akan selalu tersedia untuk pemain untuk dibaca
6. *Player controller system*, yang memiliki variabel *Vector3* *“playerPosition”* untuk mengendalikan lokasi karakter pemain

Setelah pemain menyelesaikan sesi permainannya dan keluar dari level, sistem akan menangkap *game state* sebelum user keluar dan akan menyimpannya didalam *file* JSON yang telah disediakan. Apabila user memaksa aplikasi untuk berhenti (*force close*), sistem juga akan menangkap *game state* permainan agar user tidak kehilangan kemajuan walaupun aplikasi dipaksa berhenti.

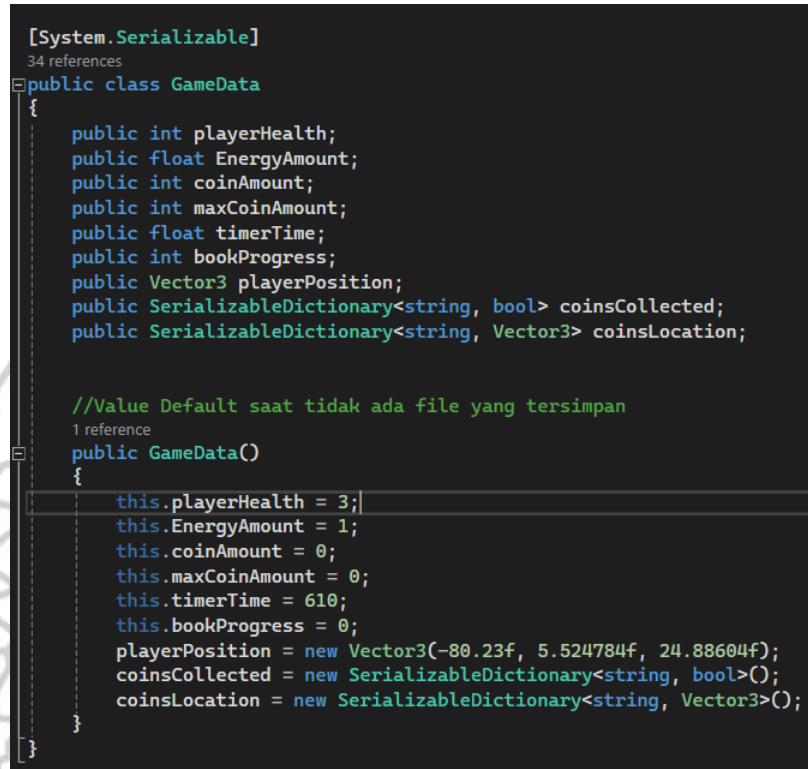
### 3.3.2 Perancangan Teknis Sistem

Sub bab ini akan menjelaskan secara detail bagaimana sistem *data persistence* akan berinteraksi dengan sistem lain yang ingin dipersistenkan pada aplikasi *“NoKidnap”*. Penjelasan akan mencakup baris kode dan tampilan

inspektor unity untuk memberikan gambaran yang jelas mengenai pengimplementasian sistem *data persistence* pada aplikasi “NoKidnap”

### 3.3.2.1 Perancangan *script* untuk sistem *data persistence*

Perancangan *script* dimulai dengan menciptakan *script* “*GameData*” yang berisi semua *data* yang ingin dipersistenkan



```
[System.Serializable]
34 references
public class GameData
{
    public int playerHealth;
    public float EnergyAmount;
    public int coinAmount;
    public int maxCoinAmount;
    public float timerTime;
    public int bookProgress;
    public Vector3 playerPosition;
    public SerializableDictionary<string, bool> coinsCollected;
    public SerializableDictionary<string, Vector3> coinsLocation;

    //Value Default saat tidak ada file yang tersimpan
    1 reference
    public GameData()
    {
        this.playerHealth = 3;
        this.EnergyAmount = 1;
        this.coinAmount = 0;
        this.maxCoinAmount = 0;
        this.timerTime = 610;
        this.bookProgress = 0;
        playerPosition = new Vector3(-80.23f, 5.524784f, 24.88604f);
        coinsCollected = new SerializableDictionary<string, bool>();
        coinsLocation = new SerializableDictionary<string, Vector3>();
    }
}
```

Gambar 3.4 Tampilan *script* *GameData*

Pada *script* “*GameData*”, terdapat variabel variabel yang akan diinisialisasi didalam *file* JSON untuk penyimpanan yaitu :

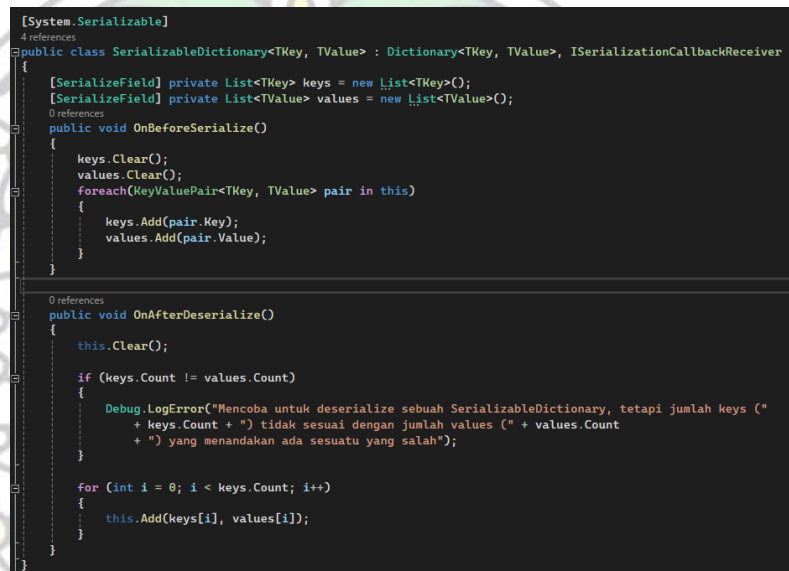
1. Integer “*playerHealth*” : Nyawa pemain dalam level
2. Float “*EnergyAmount*” : Energi pemain dalam level
3. Integer “*coinAmount*” : Jumlah skor pemain dalam level
4. Integer “*maxCoinAmount*”: Jumlah skor terbanyak pemain dalam stuuau level
5. Float “*timerTime*” : Sisa waktu pemain dalam level
6. Integer “*bookProgress*” : Kemajuan pemain dalam membuka konten buku catatan
7. Vector3 “*playerPosition*” : Koordinat lokasi pemain di dalam level dunia game



8. *SerializableDictionary* “*coinsCollected*” : *Data* buku dalam level mana yang telah di ambil”
9. *SerializableDictionary* “*coinsLocation*” : *Data* posisi buku dalam level

Kemudian variabel yang telah diinisialisasi tersebut diberi nilai *default* yang dapat diubah sesuai keperluan aplikasi sebagai nilai awal yang digunakan saat membuat *file* JSON baru.

Didalam daftar variabel yang telah disajikan, terdapat tipe *data* “*SerializeableDictionary*” yang merupakan suatu *script* agar tipe *data* dictionary mampu untuk dipersistenkan di dalam sebuah *file* JSON.



```
[System.Serializable]
4 references
public class SerializableDictionary<TKey, TValue> : Dictionary<TKey, TValue>, ISerializationCallbackReceiver
{
    [SerializeField] private List<TKey> keys = new List<TKey>();
    [SerializeField] private List<TValue> values = new List<TValue>();
    0 references
    public void OnBeforeSerialize()
    {
        keys.Clear();
        values.Clear();
        foreach (KeyValuePair<TKey, TValue> pair in this)
        {
            keys.Add(pair.Key);
            values.Add(pair.Value);
        }
    }

    0 references
    public void OnAfterDeserialize()
    {
        this.Clear();

        if (keys.Count != values.Count)
        {
            Debug.LogError("Mencoba untuk deserialize sebuah SerializableDictionary, tetapi jumlah keys ("
                + keys.Count + ") tidak sesuai dengan jumlah values (" + values.Count
                + ") yang menandakan ada sesuatu yang salah");
        }

        for (int i = 0; i < keys.Count; i++)
        {
            this.Add(keys[i], values[i]);
        }
    }
}
```

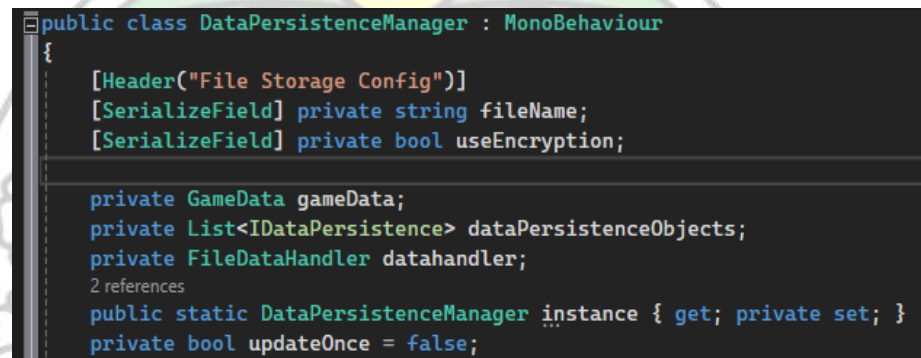
Gambar 3.5 Tampilan script *SerializableDictionary*

*Script* ini memerlukan inputan berupa 2 tipe *data* lain saat inisialisasi untuk digunakan sebagai *Tkey* dan *Tvalue* dari dictionary tersebut. *Script* ini menggunakan *inheritence* dari “*Dictionary*” yang digunakan untuk mengubah *data* yang diterima menjadi tipe *data* dictionary menggunakan list dan “*ISerializationCallbackReceiver*” yang merupakan suatu interface dengan *method* “*OnBeforeSerialize*” dimana *method* ini akan dipanggil sesaat sebelum serialisasi dan “*OnAfterDeserialize*” dimana *method* ini akan di panggil sesaat setelah deserialisasi.

Pada *method* “*OnBeforeSerialize*”, list *keys* dan *values* di *clear* untuk menghapus seluruh nilai yang tersimpan pada pemanggilan *method*

sebelumnya, kemudian setiap pasangan *key* dan *value* yang ada akan disimpan di dalam list yang telah dibuat. Pada *method* “*OnAfterDeserialize*”, *dictionary* di clear untuk memastikan isi *dictionary* kosong kemudian pasangan *key* dan *value* dari *list* sebelumnya akan dimasukkan ke dalam *dictionary* tersebut. Setelah *script* “*GameData*” selesai dibuat, selanjutnya *script* “*DataPersistenceManager*” sebagai inti dari sistem *data persistence* akan dirancang.

*Script* “*DataPersistenceManager*” ini berperan sebagai *script* yang mengatur berjalannya seluruh sistem *data persistence*. *Data* dari *script* lain harus berkomunikasi dengan *script* manager yang ada di dalam *scene* agar dapat dipersistenkan di dalam *file* JSON.

A screenshot of a code editor showing the C# code for the DataPersistenceManager class. The code is as follows:

```
public class DataPersistenceManager : MonoBehaviour
{
    [Header("File Storage Config")]
    [SerializeField] private string fileName;
    [SerializeField] private bool useEncryption;

    private GameData gameData;
    private List<IDataPersistence> dataPersistenceObjects;
    private FileDataHandler dataHandler;
    2 references
    public static DataPersistenceManager instance { get; private set; }
    private bool updateOnce = false;
```

Gambar 3.6 Insialisasi *DataPersistenceManager*

Didalam *script* “*DataPersistenceManager*”, terdapat variabel *fileName* dan *useEncryption* yang bisa digunakan untuk mengatur nama *file* JSON yang akan dibuat dan apakah *file* tersebut akan dienkripsi atau tidak di dalam inspektor unity. *Script* ini juga memanggil sebuah instansi dari *script* lain yaitu “*GameData*” yang telah dijelaskan, kemudian “*IDataPersistence*” sebagai *interface* yang digunakan oleh *script* lain agar dapat menyimpan *data* dan “*FileDataHandler*” sebagai *script* yang mengatur penyimpanan *file* JSON kedalam *storage* device yang detailnya akan dijelaskan.

```

16 references
public interface IDataPersistence
{
    9 references
    void LoadData(GameData data);
    9 references
    void SaveData(ref GameData data);
    9 references
    void RestartData(GameData data);
}

```

Gambar 3.7 Interface IDataPersistence

Interface “*IDataPersistence*” memiliki 3 *method*, yaitu *LoadData*, *SaveData*, dan *RestartData* yang bisa digunakan oleh *script* lain yang meng *inherit* interface ini untuk terhubung ke sistem *data persistence*.

```

3 references
public class FileDataHandler
{
    private string dataDirPath = "";
    private string dataFileName = "";
    private bool useEncryption = false;
    private readonly string encryptionCodeword = "geosatu";

    1 reference
    public FileDataHandler(string dataDirPath, string dataFileName, bool useEncryption)
    {
        this.dataDirPath = dataDirPath;
        this.dataFileName = dataFileName;
        this.useEncryption = useEncryption;
    }
}

```

Gambar 3.8 Inisialisasi FileDataHandler

Pada inisialisasi *script FileDataHandler*, terdapat string “*dataDirPath*” yaitu direktori yang akan digunakan sebagai tempat penyimpanan *file JSON*, “*dataFileName*” sebagai nama *file JSON*, dan boolean “*useEncryption*” untuk mengatur apakah isi *file JSON* akan dienkripsi menggunakan *XOR encryption*.

*Script “FileDataHandler”* dapat dipanggil dengan memberikan argumen untuk “*dataDirPath*”, “*dataFileName*”, dan “*useEncryption*” untuk melakukan penyimpanan *data*.

```

private void Awake()
{
    if (instance != null)
    {
        Debug.LogError("Lebih dari satu DataPersistenceManager di scene ini.");
    }
    instance = this;
}

Unity Message | 0 references
private void Start()
{
    this.datahandler = new FileDataHandler(Application.persistentDataPath, fileName, useEncryption);
    this.dataPersistenceObjects = FindAllDataPersistenceObjects();
    LoadGame();
}

```

Gambar 3.9 Awake dan Start method DataPersistenceManager

Pada script “DataPersistenceManager”, terdapat method Awake dan Start yang dipanggil saat script ini di inisialisasi. Pada method Awake, script “DataPersistenceManager” memastikan bahwa hanya ada satu script tersebut pada sebuah scene untuk mencegah error. Pada method Start, script “FileDataHandler” dipanggil sebagai “datahandler” dimana argumen yang diberikan untuk “dataDirPath” merupakan Application.persistentDataPath yaitu perintah yang menyediakan direktori untuk tempat penyimpanan file JSON sesuai dengan device yang menjalankan aplikasi, kemudian argumen “fileName” dan “useEncryption” akan diisi dengan variabel yang telah dibuat sebelumnya. Setelah direktori, nama, dan pengaturan enkripsi file JSON diinisialisasi, method FindAllDataPersistenceObjects dipanggil untuk menemukan seluruh objek di dalam scene yang menginherit kelas “MonoBehaviour” dan interface “IDataPersistence” agar dapat berkomunikasi dengan “DataPersistenceManager”.

```

7 references
private List<IDataPersistence> FindAllDataPersistenceObjects()
{
    IEnumerable<IDataPersistence> dataPersistenceObjects = FindObjectsOfType<MonoBehaviour>(true)
        .OfType<IDataPersistence>();

    return new List<IDataPersistence>(dataPersistenceObjects);
}

```

Gambar 3.10 Method FindAllDataPersistenceObjects



```

1 reference
public void NewGame()
{
    this.gameData = new GameData();
}

3 references
public void SaveGame()
{
    //pass data to other scripts
    foreach (IDataPersistence dataPersistenceObj in dataPersistenceObjects)
    {
        dataPersistenceObj.SaveData(ref gameData);
    }

    // todo - save data using data handler
    datahandler.Save(gameData);
}

2 references
public void LoadGame()
{
    //todo - load data
    this.gameData = datahandler.Load();

    //jika tidak ada data yang tersedia, kembali ke value default
    if (this.gameData == null)
    {
        Debug.Log("Tidak ada data ditemukan. Memulai permainan baru.");
        NewGame();
    }

    //push loaded data to other scripts that need it
    foreach (IDataPersistence dataPersistenceObj in dataPersistenceObjects)
    {
        dataPersistenceObj.LoadData(gameData);
    }
}

```

Gambar 3.11 Method *NewGame*, *SaveGame*, *LoadGame* pada *DataPersistenceManager*

```

public void RestartGame()
{
    this.dataPersistenceObjects = FindAllDataPersistenceObjects();
    //pass data to other scripts
    foreach (IDataPersistence dataPersistenceObj in dataPersistenceObjects)
    {
        dataPersistenceObj.RestartData(gameData);
    }

    // todo - save data using data handler
    datahandler.Save(gameData);
}

```

Gambar 3.12 Method *RestartGame* pada *DataPersistenceManager*

Script “*DataPersistenceManager*” memiliki 4 *method* yaitu *NewGame*, *SaveGame*, *LoadGame*, dan *RestartGame* yang dapat dipanggil sesuai kebutuhan di dalam aplikasi. *NewGame* akan menginisialisasi *GameData* baru, *SaveGame* akan menyimpan *data* yang ada di dalam permainan ke dalam *file JSON*, *LoadGame* akan mengambil

data yang ada di dalam file JSON untuk dimasukkan ke dalam permainan, dan *RestartGame* akan mengulang level di dalam permainan.

Method pada “*DataPersistenceManager*” tersebut memanggil method yang berada di dalam script “*FileDataHandler*” untuk memasukkan data ke dalam file JSON. Di dalam script “*FileDataHandler*”, terdapat method *Save*, *Load*, dan *Reset*.

```
2 references
public void Save(GameData data)
{
    string fullPath = Path.Combine(dataDirPath, dataFileName);
    try
    {
        //buat direktori file untuk tempat penyimpanan file jika belum ada
        Directory.CreateDirectory(Path.GetDirectoryName(fullPath));

        //Serialize C# game data menjadi JSON
        string dataToStore = JsonUtility.ToJson(data, true);

        //optionally encrypt data
        if (useEncryption)
        {
            dataToStore = EncryptDecrypt(dataToStore);
        }

        //Masukkan data yang diserialisasi kedalam file
        using (FileStream stream = new FileStream(fullPath, FileMode.Create))
        {
            using (StreamWriter writer = new StreamWriter(stream))
            {
                writer.Write(dataToStore);
            }
        }
    }
    catch (Exception e)
    {
        Debug.LogError("Error saat mencoba untuk menyimpan data dalam file: " + fullPath + "\n" + e);
    }
}
```

Gambar 3.13 Method Save pada FileDataHandler

```
1 reference
public GameData Load()
{
    string fullPath = Path.Combine(dataDirPath, dataFileName);
    GameData loadedData = null;
    if (File.Exists(fullPath))
    {
        try
        {
            //Load data yang telah diserialisasi dari file
            string dataToLoad = "";
            using (FileStream stream = new FileStream(fullPath, FileMode.Open))
            {
                using (StreamReader reader = new StreamReader(stream))
                {
                    dataToLoad = reader.ReadToEnd();
                }
            }

            //optionally decrypt data
            if (useEncryption)
            {
                dataToLoad = EncryptDecrypt(dataToLoad);
            }

            //Deserialize data dari JSON menjadi C# object
            loadedData = JsonUtility.FromJson<GameData>(dataToLoad);
        }
        catch (Exception e)
        {
            Debug.LogError("Error saat mencoba memuat data dari file: " + fullPath + "\n" + e);
        }
    }
    return loadedData;
}
```

Gambar 3.14 Method load pada FileDataHandler

```

public void Reset()
{
    string fullPath = Path.Combine(dataDirPath, dataFileName);
    try
    {
        Directory.Delete(Path.GetDirectoryName(fullPath), true);
    }
    catch (Exception e)
    {
        Debug.LogError("Error saat mencoba untuk mereset data dalam file: " + fullPath + "\n" + e);
    }
}

```

Gambar 3.15 Method Reset pada FileDataHandler

Method Save dan Load tersebut di panggil di dalam method SaveGame dan LoadGame pada script “DataPersistenceManager” untuk menyimpan data kedalam file JSON. Setiap script yang membangun sistem data persistence telah dioptimalisasi agar setiap script memegang suatu aspek kunci dari sistem, sehingga rancangan sistem mudah dimengerti. Setelah inti dari sistem data persistence selesai dirancang, sistem dapat mulai diimplementasikan ke dalam script script yang mengatur jalannya permainan.

### 3.3.2.2 Perancangan implementasi back-end untuk script permainan

Sistem data persistence akan diimplementasikan kedalam script permainan sebagai berikut:

1. “HealthManager” yang mengatur nyawa pemain

```

2 references
public void LoadData(GameData data)
{
    this.playerHealth = data.playerHealth;
}

2 references
public void SaveData(ref GameData data)
{
    data.playerHealth = this.playerHealth;
}

2 references
public void RestartData(GameData data)
{
    data.playerHealth = 3;
}

```

Gambar 3.16 Implementasi pada HealthManager

2. “EnergyScript” yang mengatur energi pemain

```
public void LoadData(GameData data)
{
    this.EnergyAmount = data.EnergyAmount;
}

2 references
public void SaveData(ref GameData data)
{
    EnergyAmount = EnergyHUD.fillAmount;
    data.EnergyAmount = this.EnergyAmount;
}

2 references
public void RestartData(GameData data)
{
    data.EnergyAmount = 1;
}
```

Gambar 3.17 Implementasi pada EnergyScript

3. “CoinManager” yang mengatur jumlah koin yang dimiliki pemain

```
2 references
public void LoadData(GameData data)
{
    this.coinAmount = data.coinAmount;
}

2 references
public void SaveData(ref GameData data)
{
    data.coinAmount = this.coinAmount;
    if (this.coinAmount > data.maxCoinAmount)
    {
        data.maxCoinAmount = this.coinAmount;
    }
}

2 references
public void RestartData(GameData data)
{
    if (this.coinAmount > data.maxCoinAmount)
    {
        data.maxCoinAmount = this.coinAmount;
    }
    data.coinAmount = 0;
}
```

Gambar 3.18 Implementasi pada CoinManager



4. “*TimeManager*” yang mengatur waktu berjalannya permainan

```
2 references
public void LoadData(GameData data)
{
    this.timerTime = data.timerTime;
}

2 references
public void SaveData(ref GameData data)
{
    data.timerTime = this.timerTime;
}

2 references
public void RestartData(GameData data)
{
    data.timerTime = 610;
}
```

*Gambar 3.19 Implementasi pada TimeManager*

5. “*ContentBookManager*” yang mengatur kemajuan pemain dalam membuka isi catatan

```
2 references
public void LoadData(GameData data)
{
    this.bookProgress = data.bookProgress;
}

2 references
public void SaveData(ref GameData data)
{
    data.bookProgress = this.bookProgress;
}

2 references
public void RestartData(GameData data)
{
    data.bookProgress = this.bookProgress;
}
```

*Gambar 3.20 Implementasi pada ContentBookManager*

6. “*ThirdPersonController*” yang mengatur seluruh kontrol pemain terhadap karakter utama, termasuk posisi karakter

```
2 references
public void LoadData(GameData data)
{
    this.transform.position = data.playerPosition;
}

2 references
public void SaveData(ref GameData data)
{
    data.playerPosition = this.transform.position;
}

2 references
public void RestartData(GameData data)
{
    data.playerPosition = new Vector3(-80.23f, 5.524784f, 24.88604f);
}
```

Gambar 3.21 Implementasi pada *ThirdPersonController*

7. “*CoinSound*” yang terdapat pada seluruh *pickup* buku untuk mengatur posisi *spawn* dari buku tersebut sekaligus memeriksa apakah suatu buku sudah pernah diambil oleh pemain

```
public void LoadData(GameData data)
{
    data.coinsCollected.TryGetValue(id, out collected);
    if (collected)
    {
        gameObject.SetActive(false);
    }
    if (data.coinsLocation.ContainsKey(id))
    {
        data.coinsLocation.TryGetValue(id, out pos);
        transform.position = pos;
    }
}

2 references
public void SaveData(ref GameData data)
{
    if (data.coinsCollected.ContainsKey(id))
    {
        data.coinsCollected.Remove(id);
        data.coinsLocation.Remove(id);
    }
    data.coinsCollected.Add(id, collected);
    data.coinsLocation.Add(id, transform.position);
}

2 references
public void RestartData(GameData data)
{
    if (data.coinsCollected.ContainsKey(id))
    {
        data.coinsCollected.Remove(id);
        data.coinsLocation.Remove(id);
    }
}
```

Gambar 3.22 Implementasi pada *CoinSound*

Pada *Script* ini juga diimplementasikan sebuah *method* untuk mengoptimalkan penyimpanan setiap buku dengan memberi objek buku sebuah ID acak yang akan digunakan oleh sistem *data persistence* supaya buku dapat dibedakan antara satu sama lain.

```
[SerializeField] private string id;
[ContextMenu("Generate guid for id")]
0 references
private void GenerateGuid()
{
    id = System.Guid.NewGuid().ToString();
}
```

Gambar 3.23 Method *GenerateGuid* pada *CoinSound*

### 3.4 Rencana Uji Coba dan Evaluasi

Rencana uji coba dan evaluasi dari sistem *data persistence* akan mencakup dua tahapan, yaitu diawali dengan pengujian aplikasi menggunakan *functional testing* untuk memastikan bahwa sistem *data persistence* telah berhasil diinisialisasikan kedalam aplikasi dan telah berjalan dengan semestinya, kemudian akan dilakukan survei terhadap *play-tester* yang mencoba aplikasi setelah sistem *data persistence* diinisialisasikan kedalam aplikasi permainan “NoKidnap” untuk mendapatkan data mengenai *user experience* aplikasi.

*Functional testing* yang dilakukan akan menggunakan metode *white box testing*, dimana metode ini berfokus pada struktur internal dan koding dari aplikasi yang diuji untuk memverifikasi kegunaan input output dari sistem permainan yang akan diimplementasikan.

Metode survei yang akan disajikan kepada *play-tester* untuk mendapatkan data subjektif pemain mengenai *user experience* permainan setelah sistem *data persistence* diinisialisasikan adalah skala likert, yang merupakan skala penelitian untuk mengukur sikap dan pendapat. Skala ini digunakan untuk mengetahui tingkat persetujuan responden terhadap suatu pertanyaan. Dengan skala ini, peneliti dapat melihat dengan akurat pendapat *play-tester* mengenai sistem *data persistence*. Hal ini memberikan pemahaman yang lebih baik tentang pengaruh sistem *data persistence* terhadap *user experience* di dalam aplikasi permainan edukatif “NoKidnap”

## BAB IV

### IMPLEMENTASI DAN PENGUJIAN SISTEM

#### 4.1 Hasil Implementasi Sistem *Data Persistence*

Berikut merupakan hasil dari implementasi sistem sesuai dengan rancangan yang telah disajikan sebelumnya pada bab 3:



*Gambar 4.1 Tampilan pemilihan level*

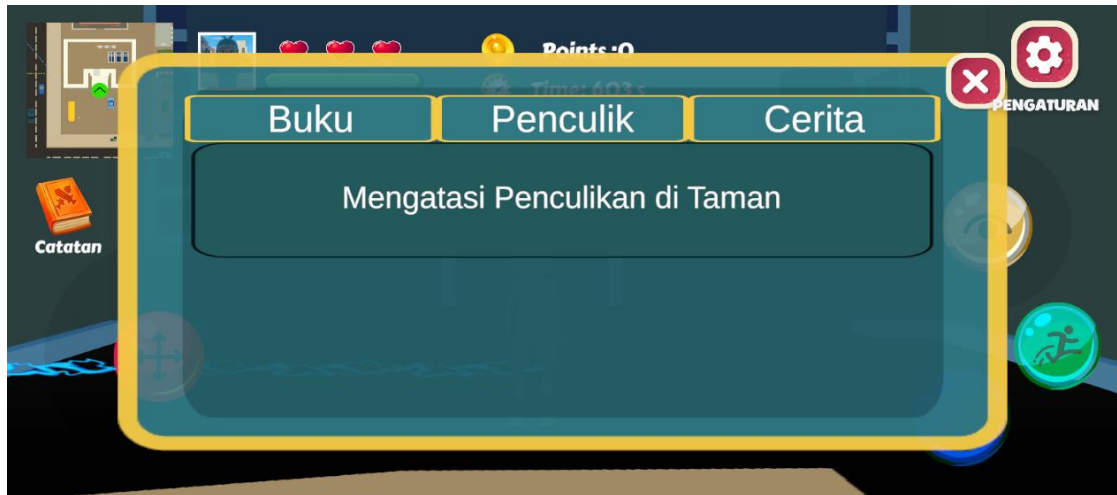
Gambar 4.1 merupakan tampilan pilih level saat skor tertinggi pemain pada level taman adalah 30 dengan 5 dari 7 catatan yang terbuka.



*Gambar 4.2 Tampilan UI dalam level*

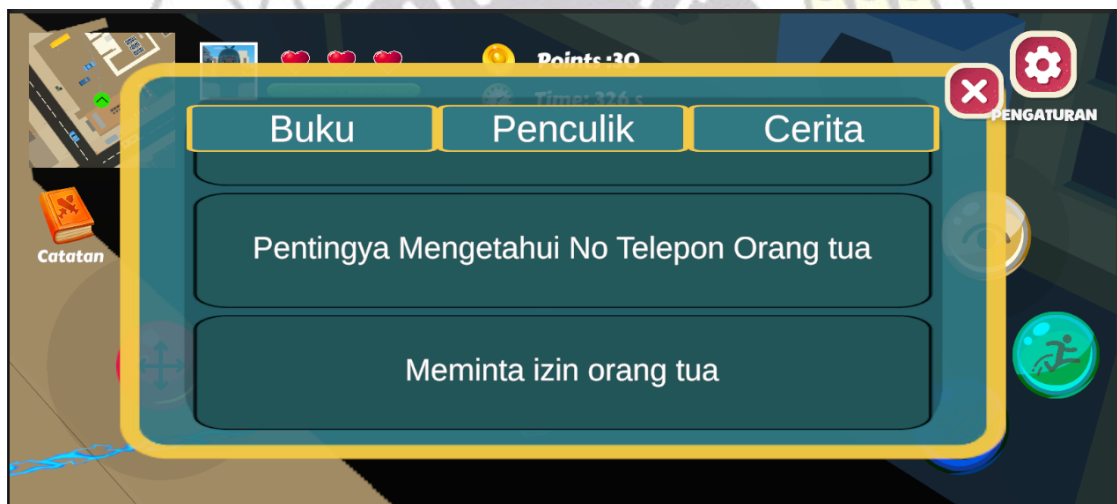
Gambar 4.2 adalah tampilan di dalam level permainan dimana sistem poin, waktu, nyawa, energi, catatan, lokasi pemain, dan lokasi *pickup* buku telah persisten.





*Gambar 4.3 Tampilan catatan awal*

Gambar 4.3 merupakan tampilan catatan pemain pada awal permainan, dimana pemain hanya memiliki 1 buah catatan yang terbuka.



*Gambar 4.4 Tampilan catatan setelah mendapat beberapa buku*

Gambar 4.4 menunjukkan tampilan catatan pemain setelah pemain mendapatkan beberapa buku, dimana isi catatan pemain akan bertambah sesuai dengan jumlah buku yang telah didapatkan



## 4.2 Hasil Uji Program

Berikut adalah hasil pengujian *functional testing* yang telah dilakukan terhadap sistem *data persistence* yang telah diimplementasikan ke dalam aplikasi “NoKidnap” :

Tabel 4.1 Hasil pengujian sistem *data persistence*

Detail Test Case	Langkah Pengujian	Hasil yang Diharapkan	Hasil yang Terjadi	Status
Menciptakan <i>file data persistence</i> JSON saat user memasuki permainan untuk pertama kalinya	1. Aplikasi dibuka untuk pertama kalinya 2. Masuk ke dalam level permainan melalui Main Menu 3. Memastikan terciptanya <i>file</i> JSON pada game <i>files</i>	<i>File data persistence</i> tercipta di dalam game <i>files</i>	Berhasil menciptakan <i>file data persistence</i> di dalam game <i>files</i>	Success
Serialisasi <i>file data persistence</i> JSON saat user keluar dari level melalui pause menu	1. Membuka aplikasi 2. Masuk ke dalam level permainan melalui main menu 3. Keluar dari level melalui pause menu untuk kembali ke main menu	Variabel permainan yang mengalami perubahan diserialisasi ke dalam <i>file data persistence</i>	Berhasil menserialisasikan perubahan variabel kedalam <i>file data persistence</i>	Success
Deserialisasi <i>file data persistence</i> JSON saat user	1. Pastikan <i>file data persistence</i> ada dalam game <i>files</i> 2. Membuka aplikasi	Permainan menggunakan nilai yang ada di dalam <i>file data persistence</i>	Berhasil menggunakan nilai yang ada di dalam <i>file data persistence</i>	Success

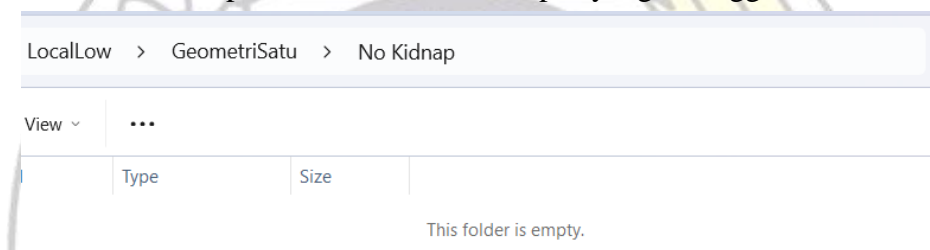
memasuki level permainan	3. Masuk ke dalam level permainan melalui main menu	sebagai nilai awal permainan		
Serialisasi variabel skor tertinggi dan kemajuan catatan saat mencapai akhir level dan tetapkan variabel lain ke nilai default	1.Membuka aplikasi 2. Masuk ke dalam level permainan melalui main menu 3. Selesaikan level	Variabel maxCoinAmount dan bookProgress akan di serialisasikan ke dalam <i>file data persistence</i> , dan variabel lainnya akan di tetapkan ke nilai default	Berhasil menserialisasikan Variabel maxCoinAmount dan bookProgress ke dalam <i>file data persistence</i> serta menetapkan variabel lain ke nilai default	Success
Serialisasi <i>file data persistence</i> JSON saat aplikasi di paksa tutup oleh user untuk menghindari kehilangan data	1.Membuka aplikasi 2. Masuk ke dalam level permainan melalui main menu 3. Tutup paksa aplikasi	Variabel permainan yang mengalami perubahan di serialisasikan ke dalam <i>file data persistence</i>	Berhasil menserialisasikan perubahan variabel kedalam <i>file data persistence</i>	Success
Menghapus <i>file data persistence</i> JSON untuk mereset aplikasi	1.Membuka aplikasi 2.Buka menu pengaturan pada main menu 3. Tekan tombol reset	<i>File data persistence</i> yang berada didalam game <i>files</i> terhapus agar permainan dapat direset	Berhasil menghapus <i>file data persistence</i> di dalam game <i>files</i>	Success



Berdasarkan hasil pengujian sistem, dapat disimpulkan bahwa sistem *data persistence* menggunakan JSON *data parsing* telah berhasil di inialisasikan ke dalam aplikasi “NoKidnap”. Berikut merupakan penjelasan dari setiap pengujian yang telah dilakukan pada tabel 4.1 disertai gambar sebagai visualisasi pengujian sistem.

### 1. Menciptakan *file data persistence* JSON saat user memasuki permainan untuk pertama kalinya

Pada pengujian ini, perlu di pastikan bahwa *file* JSON yang berisikan variabel-variabel permainan yang akan di persistenkan berhasil diciptakan saat pemain memasuki aplikasi untuk pertama kalinya. Untuk mensimulasikan pembukaan permainan untuk pertama kalinya, penguji menghapus dan menginstal ulang permainan serta memastikan bahwa tidak ada *data* dari permainan sebelum dihapus yang tertinggal.



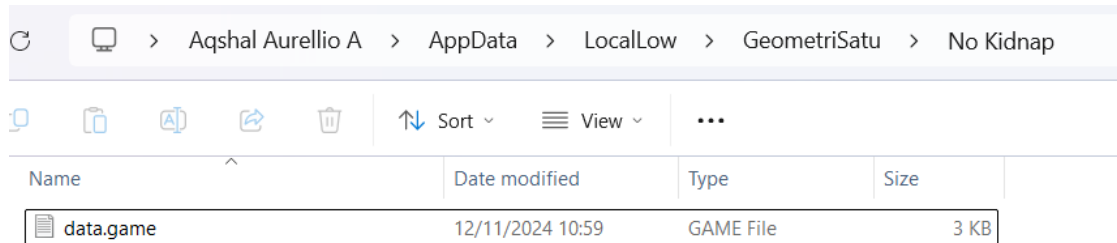
Gambar 4.7 Tampilan folder penyimpanan lokal aplikasi setelah instal

Setelah *data* dari permainan telah dipastikan sudah hilang, sistem *data persistence* yang bertugas untuk menciptakan *file* JSON di awal permainan dapat diuji dengan cara memasuki level permainan kemudian memeriksa kembali folder penyimpanan sehingga sistem dapat diverifikasi telah berjalan dengan baik.



Gambar 4.8 Tampilan awal level permainan

Setelah memasuki permainan, folder penyimpanan lokal diperiksa kembali untuk melihat apakah *file* JSON telah berhasil tercipta sebagai inti dari sistem *data persistence*. Berikut tampilan folder penyimpanan setelah memasuki permainan :



Gambar 4.9 Tampilan folder penyimpanan dengan file berhasil tercipta

*File* JSON telah berhasil tercipta saat permainan dibuka untuk pertama kalinya dan tampilan isi dari *file* tersebut sesuai dengan tampilan yang ada pada gambar 4.5 sehingga fitur sistem ini telah berhasil berjalan.

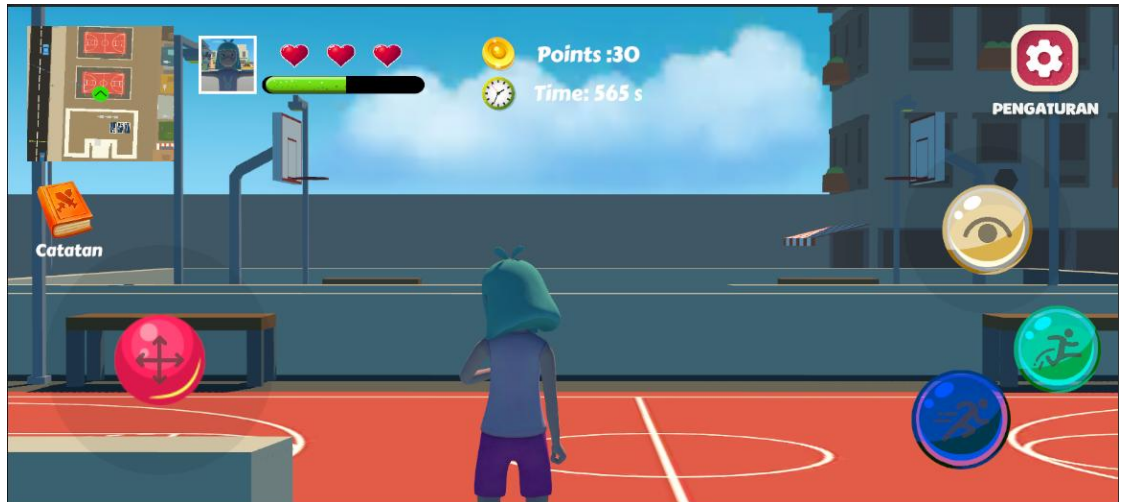
## 2. Serialisasi *file data persistence* JSON saat user keluar dari level melalui pause menu

Pada pengujian ini, sistem akan diuji untuk memastikan bahwa *file* JSON yang telah diciptakan akan diperbaharui saat pemain keluar dari permainan melalui pause menu sesuai dengan *game state* permainan sebelum pemain keluar. Untuk *test case* ini, isi *file* JSON akan diperiksa saat *file* diciptakan, kemudian penguji melakukan aktifitas di dalam permainan untuk mengubah variabel permainan dan melakukan penyimpanan melalui pause menu permainan, setelah itu isi dari *file* JSON akan diperiksa kembali untuk memastikan *data* permainan telah berhasil diserialisasi.

```
{
  "playerHealth": 3,
  "EnergyAmount": 1.0,
  "coinAmount": 0,
  "maxCoinAmount": 0,
  "timerTime": 610,
  "bookProgress": 0,
  "playerPosition": {
    "x": -80.2300033569336,
    "y": 5.524783611297607,
    "z": 24.886043548583986
  },
  "coinscollected": {
    "keys": [
      "afc3c5bf-795b-4451-ab3b-9c6d60cb2349",
      "6cd814dd-eb40-435a-aa98-e7820123549c",
      "ab976fd7-09c2-4629-a27c-ef50291696b4",
      "6a97b0d5-8b78-4d17-9507-1c3309a5c0f6",
      "842acd16-8235-45c0-b7e9-0f7326c590ec",
      "e118347f-09f2-44f4-bc9d-732720157252"
    ],
    "values": [
      false,
      false,
      false,
      false,
      false,
      false
    ]
  }
}
```

Gambar 4.10 Isi file JSON saat pertama diciptakan

Gambar 4.10 menunjukkan variabel awal yang ada saat *file* JSON diciptakan. Kemudian penguji memainkan permainan dan mengubah beberapa variabel agar serialisasi *data* dapat diperiksa.



Gambar 4.11 Tampilan aplikasi dengan variabel yang telah diubah

Pada gambar 4.11, dapat terlihat bahwa variabel Points, Time, dan stamina bar telah berubah. Penguji kemudian keluar melalui pause menu untuk memverifikasi apakah serialisasi *data* telah berhasil dilakukan.



Gambar 4.12 Tampilan pause menu

```
{
  "playerHealth": 3,
  "EnergyAmount": 0.5457692123,
  "coinAmount": 30,
  "maxCoinAmount": 30,
  "timerTime": 565.1446533203125,
  "bookProgress": 3,
  "playerPosition": {
    "x": -80.93196868896485,
    "y": 5.6999993324279789,
    "z": 49.43383026123047
  },
  "coinsCollected": {
    "keys": [
      "afc3c5bf-795b-4451-ab3b-9c6d60cb2349",
      "6cd814dd-eb40-435a-aa98-e7820123549c",
      "ab976fd7-09c2-4629-a27c-ef50291696b4",
      "6a97b0d5-8b78-4d17-9507-1c3309a5c0f6",
      "842acd16-8235-45c0-b7e9-0f7326c590ec",
      "e118347f-09f2-44f4-bc9d-732720157252"
    ],
    "values": [
      true,
      false,
      true
    ]
  }
}
```

Gambar 4.13 Isi file JSON yang telah berubah

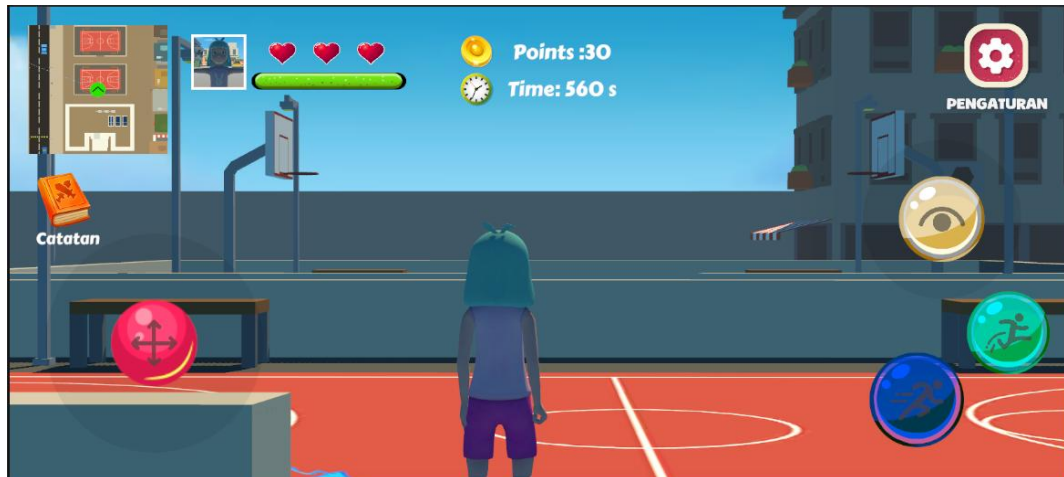
Pada gambar 4.13, dapat dilihat bahwa variabel didalam permainan ketika penguji keluar melalui pause menu sesuai dengan isi yang telah tersimpan di dalam *file* JSON, sehingga serialisasi variabel dalam aplikasi ke dalam *file* JSON telah berhasil dilakukan.

### 3. Deserialisasi *file data persistence* JSON saat user memasuki level permainan

Pada pengujian ini, sistem akan diuji untuk memastikan bahwa *data* yang telah tersimpan sebelumnya di dalam *file* JSON akan dideserialisasi kembali ke dalam permainan ketika pemain membuka aplikasi kembali. Pengujian dilakukan dengan cara melihat isi *file* JSON sebelum memasuki aplikasi, kemudian memasuki level dan memastikan variabel yang ditampilkan sesuai dengan variabel di dalam *file* JSON.

Dalam pengujian ini, isi *file* JSON awal dapat dilihat pada gambar 4.13 sebagai kondisi awal, kemudian penguji memasuki level untuk memastikan berjalannya sistem deserialisasi ke dalam aplikasi.





Gambar 4.14 Tampilan aplikasi ketika kembali memasuki level

pada gambar 4.14, dapat dilihat variabel telah berhasil di deserialisasi kembali ke dalam permainan. Jumlah points dan lokasi pemain sesuai dengan kondisi permainan pada sesi sebelumnya, waktu permainan terlihat telah berkurang dan stamina pemain telah bertambah dikarenakan kedua sistem ini mulai berjalan kembali disaat level dimasuki kembali. Proses deserialisasi *data* dari *file* JSON kembali ke dalam aplikasi telah berhasil dilakukan.

#### 4. Serialisasi variabel skor tertinggi dan kemajuan catatan saat mencapai akhir level dan tetapkan variabel lain ke nilai default

Pada pengujian ini, penguji akan memainkan aplikasi hingga mencapai akhir dari level untuk memastikan bahwa variabel skor tertinggi dan kemajuan catatan pemain tersimpan sebagai *game progression* dari permainan, tetapi variabel lainnya akan ditetapkan kembali menjadi nilai awal karena level tersebut telah selesai pada sesi sebelumnya. Penguji akan melihat isi *file* JSON sebelum pengujian, memainkan level hingga sampai pada tujuan akhir dari level, kemudian melihat kembali isi *file* JSON untuk memastikan perubahan dari isi *file* sesuai dengan yang diharapkan.

Isi dari *file* JSON di awal pengujian sesuai dengan isi pada gambar 4.13, dimana *maxCoinAmount* sebagai variabel skor tertinggi adalah 30 dan *bookProgress* sebagai kemajuan catatan pemain adalah 3. Penguji kemudian memainkan permainan hingga mencapai tujuan akhir level untuk menyelesaikan level tersebut.



Gambar 4.15 Pintu rumah leafy sebagai akhir dari level

Setelah mencapai akhir dari level, isi dari *file* JSON akan dilihat kembali untuk memastikan bahwa skor tertinggi dan kemajuan buku catatan pemain tetap bersifat persisten.

```
{
  "playerHealth": 3,
  "EnergyAmount": 1.0,
  "coinAmount": 0,
  "maxCoinAmount": 30,
  "timerTime": 610.0,
  "bookProgress": 3,
  "playerPosition": {
    "x": -80.2300033569336,
    "y": 5.524784088134766,
    "z": 24.88603973388672
  },
  "coinsCollected": {
    "keys": [],
    "values": []
  },
  "coinsLocation": {
    "keys": [],
    "values": []
  }
}
```

Gambar 4.16 Isi file JSON setelah mencapai akhir level

Pada gambar 4.16, dapat terlihat seluruh variabel kecuali `maxCoinAmount` dan `bookProgress` telah kembali menjadi nilai awal. Kedua variabel tersebut telah berhasil dipersistenkan setelah pengujian menyelesaikan level.

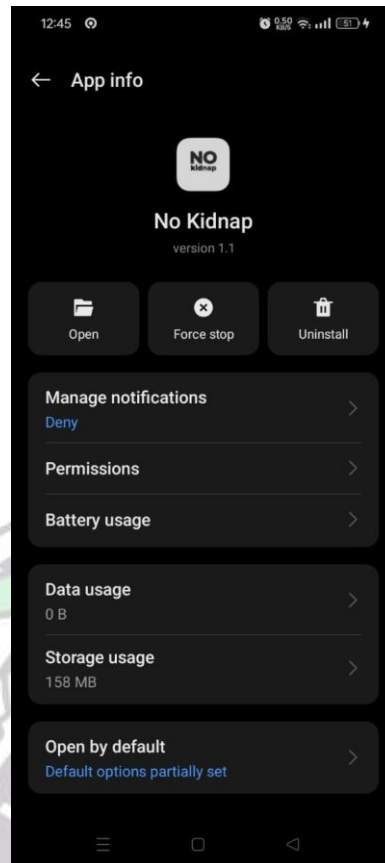
5. Serialisasi *file data persistence* JSON saat aplikasi di paksa tutup oleh user untuk menghindari kehilangan *data*

Pada pengujian ini, aplikasi akan ditutup paksa untuk mensimulasikan pengguna yang ingin keluar dari permainan tanpa mengikuti prosedur yang disediakan aplikasi, atau ketika terjadi eror yang menyebabkan aplikasi dihentikan paksa untuk memastikan bahwa sistem *data persistence* tetap dapat menserialisasikan *data* permainan saat aplikasi dihentikan. Penguji akan masuk dan memainkan permainan, kemudian menutup paksa aplikasi untuk melihat apakah variabel yang telah berubah saat bermain berhasil diserialisasikan ke dalam *file* JSON.



Gambar 4.17 Tampilan permainan sebelum ditutup paksa

Gambar 4.17 menampilkan *game state* permainan sebelum diberhentikan secara paksa. Untuk menguji apakah sistem *data persistence* tetap mampu menserialisasikan variabel di dalam permainan saat di tutup paksa, maka aplikasi akan di *force close* melalui setting pada android seperti yang bisa dilihat pada gambar 4.18.



*Gambar 4.18 Force stop aplikasi untuk menguji sistem*

Setelah dipaksa berhenti, aplikasi kemudian dibuka kembali untuk melihat apakah *game state* dapat tersimpan oleh sistem *data persistence*. Berikut merupakan tampilan permainan setelah dibuka kembali.



*Gambar 4.19 Tampilan permainan setelah dibuka kembali*

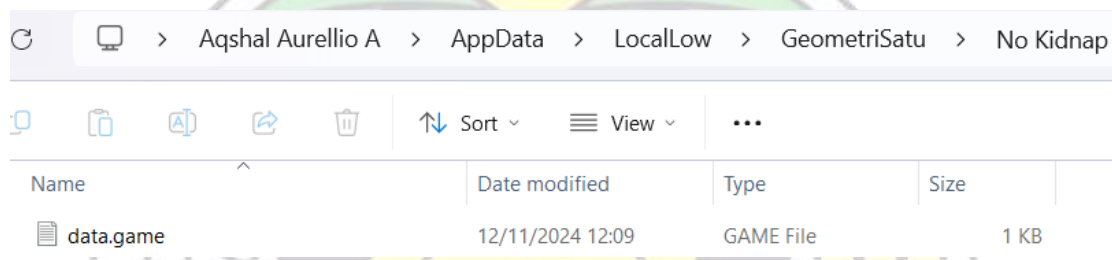
*Game state* dari aplikasi berhasil ditangkap dan diserialisasikan walaupun aplikasi di paksa berhenti oleh sistem. Pengujian *data persistence* ketika aplikasi dipaksa berhenti telah berhasil dilakukan



## 6. Menghapus *file data persistence* JSON untuk mereset aplikasi

Pada pengujian ini, fungsi untuk melakukan *reset* terhadap aplikasi akan diperiksa untuk memastikan fungsi tersebut berjalan dengan baik. *File* JSON akan dihapus menggunakan tombol *reset* yang ada pada pengaturan di menu utama. Fitur ini diciptakan agar user memiliki kemampuan untuk mengulang kemajuan aplikasi tanpa harus menginstal ulang keseluruhan aplikasi.

Pengujian dilakukan dengan cara memastikan *file* JSON ada pada penyimpanan lokal aplikasi, masuk kedalam permainan dan menekan tombol reset, kemudian memastikan bahwa *file* JSON telah terhapus dari penyimpanan lokal aplikasi.



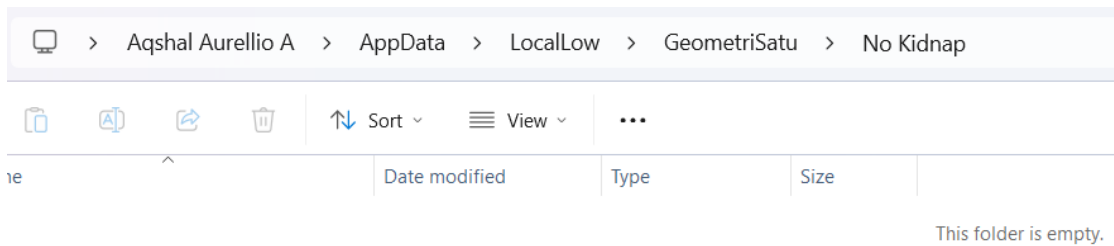
Gambar 4.20 Tampilan penyimpanan aplikasi

Pada gambar 4.20, dapat dilihat bahwa *file* penyimpanan JSON masih ada di dalam penyimpanan lokal aplikasi.



Gambar 4.21 Tampilan pengaturan di menu utama dengan tombol reset

Tombol reset yang berada di panel pengaturan pada menu utama kemudian ditekan untuk menghapus *file* JSON yang ada pada penyimpanan lokal.



*Gambar 4.22 Tampilan isi penyimpanan lokal setelah aplikasi di reset*

Pada gambar 4.22, dapat terlihat bahwa tombol *reset* telah berhasil menghapus *file* JSON yang akan menyebabkan kelakuan aplikasi sama seperti saat pertama kali di insialisasi. *File* JSON akan diciptakan kembali saat user memasuki kembali permainan dengan masing masing variabel permainan ditetapkan menjadi nilai awalnya.

#### 4.3 Hasil Eksperimen

Peneliti telah melakukan eksperimen terhadap 30 orang sampel *play-tester*, dimana setiap *play-tester* diberikan akses untuk memainkan aplikasi “NoKidnap” menggunakan perangkat *android* masing-masing. Hal ini dilakukan untuk memastikan bahwa sistem dapat berjalan dengan benar di berbagai macam perangkat. Beberapa perangkat yang digunakan dalam *playtest* ini merupakan *Samsung Galaxy A14*, *Redmi Note 13*, dan *Oppo A78*, dimana sistem berjalan dengan baik di seluruh perangkat yang digunakan. Data dari *play-tester* dikumpulkan menggunakan *Google Form*. *Play-tester* yang berpartisipasi diminta untuk memainkan level pertama dalam permainan dan membuka semua catatan yang tersedia di dalam level tersebut. Setelah *play-tester* menyelesaikan instruksi tersebut, *play-tester* kemudian diminta untuk mengisi kuisisioner yang telah disediakan. Data yang dikumpulkan melalui kuisisioner digunakan untuk menganalisis efektifitas sistem *data persistence* terhadap *user experience* dari aplikasi permainan. Adapapun pertanyaan yang diberikan bisa dilihat pada tabel berikut.

*Tabel 4.2 Pertanyaan yang diberikan kepada responden*

No	Pertanyaan
1	Bagaimana pendapat Anda mengenai pengalaman pengguna didalam game ini?
2	Apakah menurut anda save system mudah untuk digunakan?

3	Apakah save system memudahkan anda untuk kembali ke permainan tanpa kehilangan kemajuan?
4	Apakah save system membantu anda untuk tetap termotivasi untuk melanjutkan bermain?
5	Apakah pengalaman anda dengan sistem catatan dalam permainan mudah untuk dimengerti?
6	Bagaimana pendapat anda tentang ketersediaan save system untuk membantu proses pembelajaran dalam game ini?

Dari pertanyaan-pertanyaan yang telah disajikan terhadap *play-tester* di dalam kuisisioner, didapat hasil sebagai berikut:

*Tabel 4.3 Jawaban Responden Terhadap Pertanyaan Ke-1*

Tanggapan	Bobot Skor	Frekuensi	Jumlah Skor	Persentasi
Sangat Baik	5	13	65	43.3%
Baik	4	14	56	46.7%
Cukup	3	3	9	10%
Kurang Baik	2	-	-	-
Tidak Baik	1	-	-	-
Total		30	130	100%

Pada pertanyaan pertama mengenai *user experience*, 13 responden (43.3%) memberikan masukan bahwa *user experience* aplikasi sangat baik. Mayoritas responden yaitu 14 responden (46.7%) memberikan masukan bahwa *user experience* aplikasi sudah baik, dan 3 responden (10%) memberi masukan bahwa *user experience* aplikasi cukup. Tidak ada responden yang memberi masukan bahwa *user experience* aplikasi kurang baik atau tidak baik. Dari hasil yang didapat, mayoritas responden memiliki pengalaman positif terhadap *user experience* di dalam aplikasi permainan “NoKidnap” setelah sistem *data persistence* berhasil diimplementasikan.

*Tabel 4.4 Jawaban Responden Terhadap Pertanyaan Ke-2*

Tanggapan	Bobot Skor	Frekuensi	Jumlah Skor	Persentasi
Sangat mudah untuk digunakan	5	12	60	40%
Mudah untuk digunakan	4	12	48	40%
Cukup mudah untuk digunakan	3	6	18	20%
Sulit untuk digunakan	2	-	-	-
Sangat sulit untuk digunakan	1	-	-	-
Total		30	126	100%

Pertanyaan kedua menunjukkan bahwa 12 orang responden (40%) dapat menggunakan *save system* dengan sangat mudah. 12 orang responden lainnya (40%) memberikan masukan bahwa *save system* mudah untuk digunakan. Terdapat 6 responden (20%) yang menganggap *save system* yang telah diimplementasikan cukup mudah untuk digunakan. Tidak ada responden yang kesulitan dalam menggunakan *save system* yang telah diimplementasikan, sehingga dapat disimpulkan bahwa sistem bersifat aksesibel.

*Tabel 4.5 Jawaban Responden Terhadap Pertanyaan Ke-3*

Tanggapan	Bobot Skor	Frekuensi	Jumlah Skor	Persentasi
Sangat mudah	5	12	60	40%



Mudah	4	12	48	40%
Cukup mudah	3	6	18	20%
Sulit	2	-	-	-
Sangat sulit	1	-	-	-
Total		30	126	100%

Pertanyaan ketiga memastikan bahwa *play-tester* dapat melanjutkan permainan mereka tanpa kehilangan kemajuan. 12 responden (40%) melaporkan bahwa sistem sangat memudahkan mereka untuk melanjutkan permainan mereka. 12 responden (40%) melaporkan sistem memudahkan mereka untuk melanjutkan permainan. Terdapat 6 orang responden yang melaporkan bahwa sistem cukup memudahkan mereka untuk melanjutkan permainan. Tidak ada responden yang melaporkan kesulitan dalam melanjutkan permainan mereka, sehingga dapat disimpulkan bahwa sistem telah berhasil mempersistensikan data dari *play-tester* agar dapat dibuka di kemudian waktu.

*Tabel 4.6 Jawaban Responden Terhadap Pertanyaan Ke-4*

Tanggapan	Bobot Skor	Frekuensi	Jumlah Skor	Persentasi
Sangat termotivasi	5	7	35	23.3%
Termotivasi	4	16	64	53.3%
Netral	3	7	21	23.3%
Kurang termotivasi	2	-	-	-
Tidak termotivasi	1	-	-	-
Total		30	120	100%

Pada pertanyaan keempat, dipastikan bahwa *save system* membantu motivasi pemain untuk melanjutkan permainan. Mayoritas responden yaitu 16 responden

(53.3%) melaporkan bahwa *save system* membantu motivasi mereka untuk melanjutkan permainan. 7 orang responden (23.3%) melaporkan bahwa *save system* sangat membantu motivasi mereka untuk melanjutkan permainan, sedangkan 7 responden (23.3%) melaporkan bahwa *save system* tidak mempengaruhi motivasi mereka untuk melanjutkan permainan. Dari pertanyaan ini dapat disimpulkan bahwa *save system* membantu motivasi pemain untuk terus melanjutkan dan menyelesaikan permainan.

*Tabel 4.7 Jawaban Responden Terhadap Pertanyaan Ke-5*

Tanggapan	Bobot Skor	Frekuensi	Jumlah Skor	Persentasi
Sangat mudah dimengerti	5	9	45	30%
Mudah dimengerti	4	16	64	53.3%
Cukup mudah dimengerti	3	5	15	16.7%
Sulit dimengerti	2	-	-	-
Sangat sulit dimengerti	1	-	-	-
Total		30	124	100%

Pertanyaan kelima memastikan bahwa sistem catatan didalam aplikasi sebagai *game progression* dapat dimengerti oleh *play-tester*. 9 responden (30%) menganggap sistem catatan sangat mudah dimengerti. Mayoritas responden yaitu 16 responden (53.3%) menganggap sistem catatan mudah dimengerti, dan 5 responden (16.7%) menganggap sistem catatan cukup mudah dimengerti. Dapat ditarik kesimpulan bahwa sistem catatan sebagai *game progression* dalam aplikasi “NoKidnap” dapat dengan mudah dimengerti oleh mayoritas *play-tester*.

*Tabel 4.8 Jawaban Responden Terhadap Pertanyaan Ke-6*

Tanggapan	Bobot Skor	Frekuensi	Jumlah Skor	Persentasi
Sangat membantu	5	16	80	53.3%
Cukup membantu	4	11	44	36.7%
Netral	3	3	9	10%
Tidak membantu	2	-	-	-
Sangat tidak membantu	1	-	-	-
Total		30	133	100%

Pada pertanyaan keenam, pendapat responden mengenai pengaruh *save system* terhadap proses pembelajaran aplikasi “NoKidnap” didapatkan. Mayoritas responden yaitu 16 responden (53.3%) memberi masukan yaitu *save system* sangat membantu proses pembelajaran di dalam aplikasi “NoKidnap. 11 orang responden (36.7%) menganggap sistem cukup membantu proses pembelajaran, dan 3 responden (10%) berpendapat netral terhadap pengaruh *save system* kepada proses pembelajaran. Dari data yang tersedia, dapat disimpulkan bahwa *save system* pada aplikasi “NoKidnap” berpengaruh positif terhadap proses pembelajaran pengguna di dalam aplikasi.

Berdasarkan jawaban yang telah didapatkan dari setiap pertanyaan dalam kuisioner, dapat dihitung total skor dalam sebuah tabel rekapitulasi skor sebagai berikut:

*Tabel 4.9 Tabel total skor untuk setiap pertanyaan kuisioner*

No	Pertanyaan	Skor
1.	Bagaimana pendapat Anda mengenai pengalaman pengguna didalam game ini?	130
2.	Apakah menurut anda <i>save system</i> mudah untuk digunakan?	126

3.	Apakah save system memudahkan anda untuk kembali ke permainan tanpa kehilangan kemajuan?	126
4.	Apakah save system membantu anda untuk tetap termotivasi untuk melanjutkan bermain?	120
5.	Apakah pengalaman anda dengan sistem catatan dalam permainan mudah untuk dimengerti?	124
6.	Bagaimana pendapat anda tentang ketersediaan save system untuk membantu proses pembelajaran dalam game ini?	133
Total		759

Setelah mendapat skor total dari setiap pertanyaan dalam kuisioner, selanjutnya dilakukan analisis indeks minimum untuk mendapatkan nilai variabel dari pengalaman bermain dan reaksi terhadap perilaku musuh dalam *game*. Nilai ini diperoleh dengan cara memetakan total skor kedalam sebuah garis kontinum yang terdiri dari 5 rentang nilai yaitu Sangat Rendah (SR), Rendah (R), Sedang (S), Tinggi (T), dan Sangat Tinggi (ST). Analisis menggunakan metode ini dapat dihitung sebagai berikut:

$$\text{Rentang Nilai} = \text{nilai skor} \times \text{jumlah item pertanyaan} \times \text{jumlah responden}$$

Maka diperoleh:

$$\text{Sangat Rendah (SR)} = 1 \times 6 \times 30 = 180$$

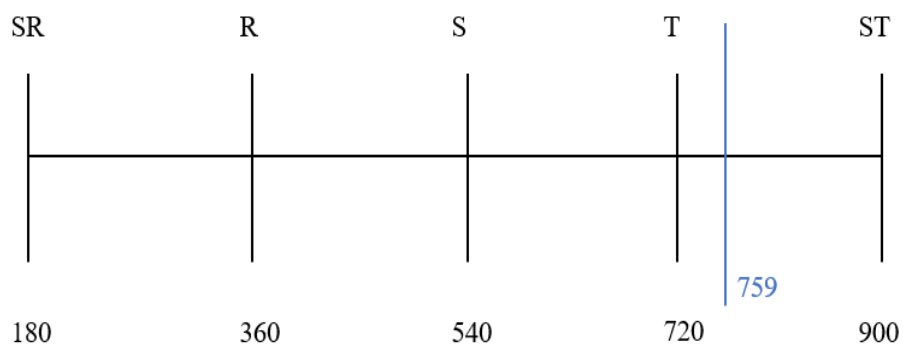
$$\text{Rendah (R)} = 2 \times 6 \times 30 = 360$$

$$\text{Sedang (S)} = 3 \times 6 \times 30 = 540$$

$$\text{Tinggi (T)} = 4 \times 6 \times 30 = 720$$

$$\text{Sangat Tinggi (ST)} = 5 \times 6 \times 30 = 900$$





*Gambar 4.23 Garis Kontinum Analisis Minimum Score Index (MSI)*

Berdasarkan hasil analisis diatas, dapat disimpulkan bahwa nilai dari implementasi sistem *data persistence* pada aplikasi “NoKidnap” sebesar 759 berada diantara rentang nilai tinggi dan sangat tinggi. Dari hasil ini dapat dipastikan bahwa secara keseluruhan, respons dari responden kuisisioner terhadap pengimplementasian *save system* terhadap permainan edukatif “NoKidnap” menghasilkan pengalaman bermain yang sangat baik dan positif.

## BAB V

### KESIMPULAN DAN SARAN

#### 5.1 Kesimpulan

Berdasarkan hasil penelitian dan inisialisasi sistem yang telah dilakukan, kesimpulan yang dapat ditarik, yaitu:

1. Implementasi sistem *data persistence* ke dalam aplikasi “NoKidnap” efektif dalam memungkinkan pengguna untuk menyimpan kemajuan permainan mereka agar dapat dilanjutkan di lain waktu. Hal ini meningkatkan *user experience* dari pengguna serta meningkatkan kualitas dari aplikasi “NoKidnap” karena telah memenuhi suatu poin dari *video game heuristics*.
2. Sistem *data persistence* yang telah diimplementasikan sangat mudah untuk digunakan pada level baru saat *developer* aplikasi “NoKidnap” melakukan pembaruan. Hal ini karena inti dari sistem *data persistence* yang perlu diimplementasikan ke dalam sebuah level baru merupakan satu objek yaitu “*DataPersistenceManager*”.
3. Tipe *data* JSON merupakan tipe *data* yang efektif untuk mengimplementasikan *data persistence* ke dalam *video game*. Tipe *data* ini mampu menyimpan variabel – variabel yang sering digunakan di dalam *video game* dengan ukuran yang kecil serta mampu dienkripsikan apabila diperlukan oleh aplikasi.

#### 5.2 Saran

Saran yang dapat peneliti berikan setelah melakukan penelitian ini adalah:

1. Peneliti sebaiknya mempertimbangkan kapan waktu terbaik bagi pengguna aplikasi untuk menyimpan kemajuan dalam permainan sehingga sistem *data persistence* dapat ditingkatkan.
2. Peneliti dapat meningkatkan tampilan grafis dari sistem *data persistence* sehingga interaksi pengguna terhadap sistem terasa lebih menarik.
3. Peneliti dapat menghubungkan sistem *data persistence* ke database online untuk meningkatkan kualitas dari sistem tersebut.

## DAFTAR PUSTAKA

- Al-Azawi, R., Al-Faliti, F., & Al-Blushi, M. (2016). Educational Gamification Vs. Game Based Learning: Comparative Study. *International Journal of Innovation, Management and Technology*, 131–136. <https://doi.org/10.18178/ijimt.2016.7.4.659>
- Bernhaupt, R. (2015). *User Experience Evaluation Methods in the Games Development Life Cycle* (pp. 1–8). [https://doi.org/10.1007/978-3-319-15985-0\\_1](https://doi.org/10.1007/978-3-319-15985-0_1)
- Consalvo, M., & Dutton, N. (2006). Game analysis: Developing a methodological toolkit for the qualitative study of games. *Game Studies*, 6. [https://gamestudies.org/0601/articles/consalvo\\_Dutton](https://gamestudies.org/0601/articles/consalvo_Dutton)
- Coregames. (n.d.). *Game States*. Retrieved May 7, 2024, from <https://learn.coregames.com/lessons/game-states/>
- Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). *From Game Design Elements to Gamefulness: Defining “Gamification.”* ACM.
- Hassenzahl, M. (2018). The Thing and I: Understanding the Relationship Between User and Product. In *Funology 2* (pp. 301–313). [https://doi.org/10.1007/978-3-319-68213-6\\_19](https://doi.org/10.1007/978-3-319-68213-6_19)
- Hochleitner, C., Hochleitner, W., Graf, C., & Tscheligi, M. (2015). *A Heuristic Framework for Evaluating User Experience in Games* (pp. 187–206). [https://doi.org/10.1007/978-3-319-15985-0\\_9](https://doi.org/10.1007/978-3-319-15985-0_9)
- Johanson, C., Piller, B., Gutwin, C., & Mandryk, R. L. (2023). If at First You Don’t Succeed: Helping Players Make Progress in Games with Breaks and Checkpoints. *Proceedings of the ACM on Human-Computer Interaction*, 7. <https://doi.org/10.1145/3611033>
- JSON. (2024, May 5). In Wikipedia. <https://en.wikipedia.org/w/index.php?title=JSON&oldid=1222403288>
- Juul, J. (2002). *The Open and the Closed: Games of Emergence and Games of Progression*.
- Marques Maranhão, D., de Oliveira da Rocha Franco, A., & Gilvan Rodrigues Maia, J. (2016). *Assessing Emergence and Progression in Games*. <https://www.researchgate.net/publication/315737783>
- Mazur, A. (2023, April 19). *XML vs JSON: Understanding the Fundamentals of Data Formats*. <https://apix-drive.com/en/blog/useful/xml-vs-json>

- MongoDB. (n.d.). *What Is Data Persistence?* Retrieved May 6, 2024, from <https://www.mongodb.com/resources/basics/databases/data-persistence>
- Renu, N. (2021). Technological advancement in the era of COVID-19. In *SAGE Open Medicine* (Vol. 9). SAGE Publications Ltd. <https://doi.org/10.1177/20503121211000912>
- Sheldon, L. (2020). *The Multiplayer Classroom: Designing Coursework as a Game*. <https://doi.org/10.1201/9780429285035>
- Tatli, Z. (2018). Traditional and Digital Game Preferences of Children: A CHAID Analysis on Middle School Students. *Contemporary Educational Technology*, 9, 90–110. <https://doi.org/10.30935/cedtech/6213>
- Vaz Salles, M., Cao, T., Sowell, B., Demers, A., Gehrke, J., Koch, C., & White, W. (2009). An evaluation of checkpoint recovery for massively multiplayer online games. *Proc. VLDB Endow.*, 2(1), 1258–1269. <https://doi.org/10.14778/1687627.1687769>
- Warren, G., Sherer, T., Dykstra, T., Schonning, N., Wenzer, M., & Jones, M. (2023, October 25). *Serialization*. Microsoft. <https://learn.microsoft.com/en-us/dotnet/standard/serialization/>
- Zunke, S., & Souza, V. D. ' (2014). JSON vs XML: A Comparative Performance Analysis of Data Exchange Formats. *IJCSN International Journal of Computer Science and Network*, 3(4). [www.IJCSN.org](http://www.IJCSN.org)