BAB IV

PERANCANGAN

4.1. Analisis Sistem Terdahulu

Sistem terdahulu masih mengandalkan desain manual untuk membuat aset dalam *game.* Dengan penerapan algoritma GAN dan VAE, sistem yang dikembangkan memungkinkan pembuatan aset virtual secara otomatis, lebih cepat dan lebih adaptif terhadap interaksi dalam *game* Ren'Py. Sistem terdahulu merupakan game visual novel yang hanya mengandalkan aset statis yang dibuat secara manual menggunakan *software* desain grafis seperti *Photoshop* atau *clip studio paint.* Proses pembuatan aset sangat memakan waktu dan tidak fleksibel untuk menghasilkan variasi karakter atau latar belakang baru.

Selain itu, tidak ada integrasi teknologi kecerdasaan buatan (AI) yang dapat menghasilkan aset visual secara otomatis. Hal ini menghambat pengembangan konten yang dinamis dan memperpanjang waktu produksi *game* yand dapat dilihat dari table 4.1.

Aspek	Sistem Terdahulu	Sistem yang Dikembangkan
Metode Generasi Aset	Manual oleh desainer *game*	Otomatis Menggunakan AI (GAN & VAE)
Kualitas Gambar	Bergantung pada keterampilan artist	Bergantung pada LoRa(Dataset Gambar) model
Waktu Pembuatan Aset	40–90 jam/aset	5–60 menit (≤1 jam)/aset
Interaktivitas	Statis, Dibutuhkan biaya tambahan jika mau perubahan	Dinamis (sesuai keinginan)
Kompatibilitas dengan Ren'Py	Hanya gambar statis yang dibuat manual	Gambar dinamis yang dihasilkan oleh AI
Biaya	Tinggi	Rendah

Tabel 4.1. Perbandingan Sistem Terdahulu dan Sistem yang Dikembangkan

4.2. Spesifikasi Kebutuhan Sistem Baru

Spesifikasi kebutuhan sistem baru dalam mengembangkan game Ren'Py dengan penerapan algorithma GAN dan VAE pada fitur asset virtual dibagi menjadi dua bagian, yaitu spesifikasi perangkat lunak (*software*) dan perangkat keras (*hardware*). Adapun penjelasan detail spesifikasi tersebut, sebagai berikut

4.2.1. Spesifikasi Perangkat Lunak (*Software*)

Spesifikasi perangkat lunak (*Software*) adalah deskripsi terperinci tentang fungsionalitas yang harus di miliki oleh perangkat lunak serta fitur-fitur yang harus disediakan untuk mendukung sistem yang baru. Adapun detail spesifikasi perangkat lunak tersebut, sebagai Tabel 4.2. berikut.

Perangkat	Keterangan
Sistem Operasi	Windows 10/11, Linux (Ubuntu) atau macOS
Bahasa Pemrograman	*Python* (Untuk AI), Ren'Py (Untuk pengembangan *game*)
Framework AI	Tensor*Flow*, *Pytorch* (Untuk pelatihan model GAN dan VAE)
Library Pendukung	*Open*CV, NumPy, *Scikit-Learn*, Matplotlib
Game Engine	Ren'Py 8.0
Database	SQ*Lite* atau JSON (Untuk penyimpanan aset dan konfigurasi)
Editor Kode	Visual Studio *Code*, Py*Charm*
Software Desain	*Adobe Photoshop*, GIMP

Tabel 4.2. Spesifikasi Kebutuhan Perangkat Lunak

4.2.2. Spesifikasi Perangkat Keras (*Hardware*)

Terdapat beberapa perangkat keras yang berperan penting dalam penelitian ini, yaitu prosesor, penyimpanan, dan memory. Adapun spesifikasi detail kebutuhan perangkat keras tercakup pada tabel 4.3. di bawah ini:

Perangkat Keras	Keterangan
Prosesor	Intel Core i7 / AMD Ryzen 7
RAM	16
GPU (Kartu Grafis)	NVIDIA RTX 3060
SSD (Penyimpanan)	512 GB SSD
Monitor	Resolusi Full HD
Mouse & Keyboard	Perangkat *input* standar untuk pengembangan dan pengujian *game*
Tablet Grafis (Opsional)	Digunakan untuk pembuatan aset virtual yang lebih detail
Headset & Mikrofon	Untuk pengujian audio dan dialog dalam *game*
Koneksi Internet	Stabil dengan kecepatan tinggi untuk mengunduh dataset dan *library* AI
UPS (*Uninterruptible Power Supply*)	Untuk menghindari kehilangan data saat terjadi pemadaman listrik mendadak

Tabel 4.3 Spesifikasi Kebutuhan Perangkat Keras

4.2.3. Spesifikasi Kebutuhan *Input*

Dalam membuat aplikasi diperlukan untuk memastikan kinerja aplikasi berjalan secara optimal, maka diperlukan sebuah proses *input*. Berikut adalah beberapa proses yang dibutuhkan dalam pengembangan *game* Ren'Py seperti table 4.4. di bawah ini:

Kebutuhan *Input*	Keterangan
Gambar Aset Virtual	Dataset gambar yang akan digunakan untuk pelatihan model GAN dan VAE

Parameter Model AI	Konfigurasi parameter seperti jumlah epoch, *batch size* dan *learning rate*
Data latih dan uji	Kumpulan data yang dipisahkan untuk pelatihan dan pengujian model
Skrip Ren'Py	Kode *skrip* untuk mengintegrasikan aset ke dalam *game* interaktif
Preferensi Pengguna	*Input* dari pengguna terkait pemilihan aset virtual dalam *game*

Tabel 4.4 Spesifikasi Kebutuhan *Input*

4.2.4. Spesifikasi Kebutuhan *Output*

Dalam membuat aplikasi diperlukan untuk memastikan kinerja aplikasi berjalan secara optimal, maka diperlukan sebuah proses *output*. Berikut adalah beberapa proses yang dibutuhkan dalam pengembangan *game* Ren'Py seperti table 4.5. di bawah ini:

Kebutuhan *Output*	**Keterangan**
Gambar aset virtual	Hasil generasi dari model GAN dan VAE dalam format PNG/JPG
Model Karakter	Karakter *game* yang dihasilkan secara otomatis berdasarkan dataset
Elemen UI	Komponen antarmuka pengguna dalam format grafis untuk Ren'Py
Laporan Evaluasi Model	Data mengenai performa model GAN dan VAE, termasuk akurasi *loss function*
Script Game	Kode *skrip* Ren'Py yang mengintegrasikan aset virtual ke dalam *game*

Tabel 4.5 Spesifikasi Kebutuhan *Output*

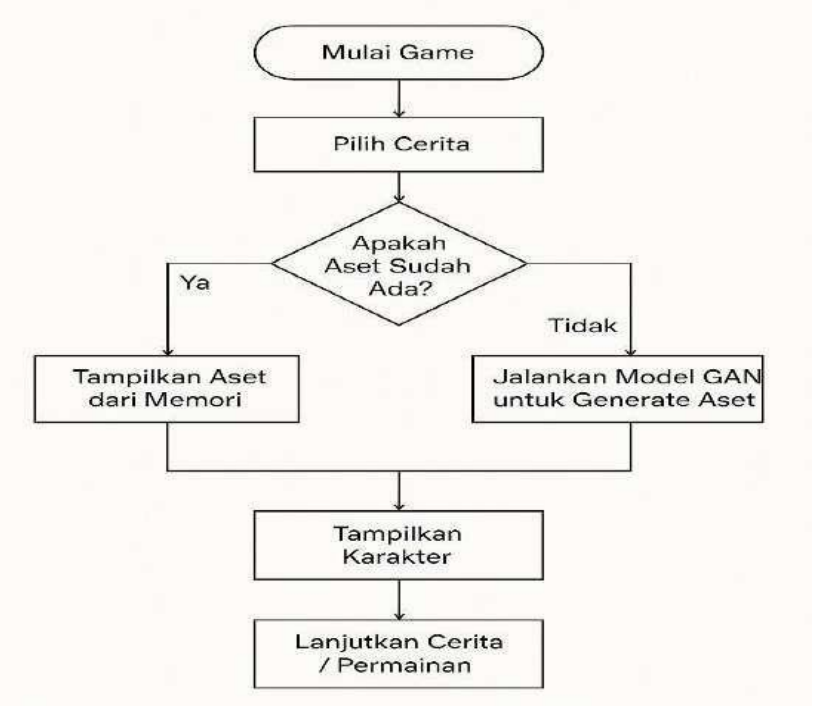
4.3. Perancangan Sistem

Tujuan ini untuk mengidentifikasi kebutuhan sistem agar dapat merancang aplikasi secara komprehensif. Proses perancangan ini mencakup penguraian secara rinci mengenai tahapan-tahapan, desain aplikasi, hingga perancangan metode pengujian. Dalam perancangan sistem, digunakan berbagai macam diagram, seperti diagram *flowchart, use case, activity* diagram, dan desain antarmuka pengguna. Ini

memiliki fungsi untuk mendukung pemahaman terhadap setiap proses.

4.3.1. *Use Case* Diagram

Berikut adalah penjelasan dari *use case* diagram proses data temuan dalam penelitian ini, adalah sebagai gamber 4.1 berikut:

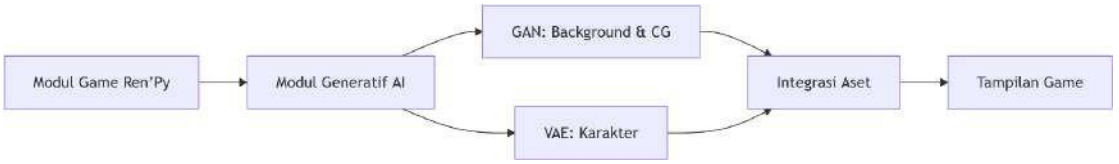


Gambar 4.1 *Use Case* Diagram Visual Game Ren'Py

Penjelasan langkah-langkah:

1. Mulai Game > Pengguna membuka dan memulai permainan
2. Pilih Cerita / Skenario > Pemain memilih jalur cerita yang ingin dimainkan.
3. Cek Aset > Sistem mengecek apakah aset virtual untuk karakter atau latar sudah tersedia.
4. Jika belum ada, sistem menjalankan model GAN/VAE untuk menghasilkan aset baru berdasarkan parameter atau random latent vector.
5. Jika aset sudah tersedia, langsung gunakan dari penyimpanan (cache atau direktori lokal).
6. Tampilkan Cerita > Game menampilkan dialog, visual dan pilihan kepada pemain.
7. Simpan Progress > Setiap pilihan dan kemajuan permainan disimpan otomatis
8. Selesai/Keluar > Game dapat diselesaikan

4.3.2. Desain Arsitektur Sistem



Gambar 4.2 Desain Sistem

Arsitektur sistem terdiri dari tiga komponen utama, adalah sebagai berikut:

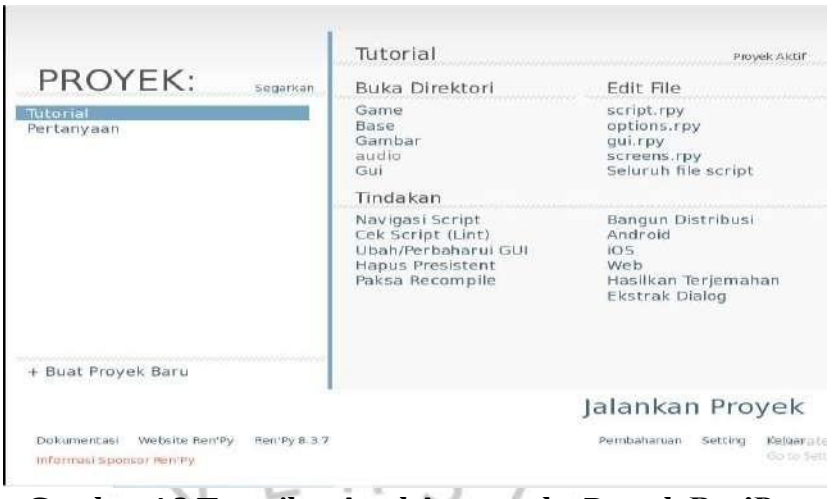
1. Modul Game: Dibuat dengan Ren'Py untuk mengelola alur cerita dan interaksi pengguna.
2. Modul Generatif: Berisi model GAN dan VAE yang dilatih sebelumnya untuk menghasilkan aset virtual.
3. Modul Integrasi: Menghubungkan hasil output model GAN dan VAE dengan sistem Ren'Py.

4.3.3. Perancangan Basis Data Sederhana

Walaupun Ren'Py tidak berbasis *database SQL*, data seperti progress permainan dan preferensi pengguna di simpan dalam file. *Save*. Namun jika dibutuhkan sistem pendukung:

1. Tabel User: id_user, nama, progess
2. Tabel Aset: id_aset, jenis, sumber (GAN dan VAE).
3. Tabel Pilihan Cerita: id_karakter, id_skenario, pilihan_user.

4.3.4. Perancangan Tampilan (GUI)



Gambar 4.3 Tampilan Awal Antarmuka Proyek Ren'Py

Gambar 4.4 Tampilan Halaman Utama Game Ren'Py

Gambar 4.5 Tampilan Game Ren'Py

1. Halaman Utama: Mulai Game, Lanjutkan, Opsi.
2. Tampilan Game: Background, Karakter, Teks dan Pilihan.
3. Tampilan Hasil Aset: Karakter hasil GAN dan VAE tampil dinamis.

4.3.5. Desain Model GAN dan VAE

- **Desain Model GAN (*Generative Adversarial Network*)**

```
def __init__(self):
    super().__init__()
    self.main = nn.Sequential(
        nn.ConvTranspose2d(100, 512, 4, 1, 0), # Input: noise 100d
        nn.BatchNorm2d(512),
        nn.ReLU(True),
        nn.ConvTranspose2d(512, 256, 4, 2, 1),
        ... # Layer output: 3x512x512
    )
```

Gambar 4.6. Kode 1 – *Generator* GAN

Kode ini mendefinisikan Generator untuk Generative Adversarial Network (GAN) yang dirancang untuk menciptakan gambar baru dari noise acak.

Proses:

1. ConvTranspose2d: "Memperbesar" noise menjadi gambar
2. BatchNorm: Menjaga konsistensi gaya
3. ReLU: Menambahkan detail kompleks
4. Tanh: Menyesuaikan warna ke range yang dapat ditampilkan

- **Discriminator**

```
class Discriminator(nn.Module):
    def __init__(self):
        super().__init__()
        self.main = nn.Sequential(
            nn.Conv2d(3, 64, 4, 2, 1, bias=False),  # Input: gambar 512x512
            nn.LeakyReLU(0.2, inplace=True),  # Deteksi pola tersembunyi
            nn.Conv2d(64, 128, 4, 2, 1, bias=False),  # Downsampling
            ...
            nn.Sigmoid()  # Output: probabilitas [0,1] (asli/palsu)
        )
```

Gambar 4.6. Kode 2 –*Discriminator*

Kode di atas digunakan untuk membangun *Discriminator*, yaitu jaringan yang bertugas untuk membedakan gambar asli dan hasil buatan generator, agar generator dapat terus berjalan. Proses:

1. Conv2d: "Memperkecil" gambar untuk ekstraksi fitur
2. LeakyReLU: Mendeteksi anomali/kejanggalan
3. Sigmoid: Memberikan skor keaslian (0.9 = asli, 0.1 = palsu)

- **Desain Model VAE (*Variational Autoencoder*)**

```
class Encoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(3, 32, 4, 2, 1),  # Kompresi gambar
            nn.ReLU(),
            ...
        )
        self.fc_mean = nn.Linear(256*4*4, latent_dim)  # Vektor mean (μ)
        self.fc_logvar = nn.Linear(256*4*4, latent_dim)  # Vektor variansi (log σ²)
```

Gambar 4.7. *Kode* 3 – *Encoder*

Kode ini membentuk bagian Encoder VAE, yang berfungsi untuk mengkodekan gambar menjadi representasi laten (*mean* dan *logvar*) yang nantinya akan digunakan untuk membentuk gambar baru. Representasi laten (*mean* dan *logvar*) adalah dua komponen penting yang merepresentasikan distribusi probabilistik dari data asli ke dalam ruang laten (*latent space*).

Mean (μ): Nilai rata-rata dari distribusi di ruang laten untuk Menentukan pusat distribusi untuk tiap input data.

Logvar ($\log(\sigma^2)$): logaritma dari variansi (*dispersion*) untuk Menentukan seberapa menyebar distribusinya (variansi) – Menunjukkan ketidakpastian

- **Decoder**

```
class Decoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.deconv = nn.Sequential(
            nn.ConvTranspose2d(256, 128, 4, 2, 1),  # Bangun gambar dari vektor
            nn.ReLU(),
            ...
            nn.Sigmoid()  # Output: [0,1] (pixel values)
        )

reconstruction_loss = F.mse_loss(output, input)  # Akurasi rekonstruksi
kl_divergence = -0.5 * torch.sum(1 + logvar - mean.pow(2) - logvar.exp()
i
total_loss = reconstruction_loss + 0.001 * kl_divergence  # Total loss
```

Gambar 4.8. Kode 4 – Decoder

Kode ini membentuk bagian Decoder VAE dan Lost Function VAE, yang bertugas mengubah representasi laten menjadi gambar utuh kembali, seperti karakter atau laten. Representasi laten adalah bentuk data tersembunyi yang dihasilkan oleh model pembelajaran mesin (seperti VAE atau autoencoder) setelah data asli (misalnya gambar) diproses oleh encoder. Proses:

1. Ambil vektor $z = \mu + \sigma * \varepsilon$ (ε = noise kecil)
2. ConvTranspose2d: "Memperbesar" vektor menjadi gambar
3. Sigmoid: Menjamin nilai pixel antara 0-1
4. reconstruction_loss: Kesalahan antara gambar input vs output
5. kl_divergence: "Hukuman" jika distribusi menyimpang dari normal
6. 0.001: Parameter penyeimbang (ditemukan di-tune)

4.3.6. Integrasi Model GAN dan VAE

Integrasi model GAN dan VAE dalam pengembangan game "*EcoAct: All Heroes Start Small*" dilakukan untuk menghasilkan aset visual karakter dan latar secara dinamis, sehingga menghadirkan pengalaman bermain yang lebih variatif dan imersif. Proses ini dibagi menjadi beberapa tahap utama:

- **Training dan Penyimpanan Model**

Model GAN dan VAE dilatih menggunakan dataset:

a. *Character Art Dataset* (Gambar Karakter)

b. *Background Art Dataset* (Gambar latar sekolah, hutan dan taman)

c. CG *Art Dataset* (*Cutscene* khusus)

Setelah pelatihan selesai, model disimpan dalam format .h5 (Keras Model Save) atau .pb (*Protobuf TensorFlow*) dan dipersiapkan untuk digunakan kembali (*interface*) saat *runtime* atau *pre-generation*.

- **Proses Interface (Pembuatan Aset)**

  Model yang telah dilatih digunakan untuk menghasilkan aset baru:

```
# GAN (StyleGAN2-ADA)
model_gan = StyleGAN2(
    resolution=512,
    latent_size=256,
    mapping_layers=8
)

# VAE
model_vae = VAE(
    input_dim=(128,128,3),
    latent_dim=256,
    encoder_layers=4
)
```

Gambar 4.8 Kode 5 – Pengaturan Model GAN dan VAE

Aset ini kemudian dipindahkan ke folder *images*/ pada proyek Ren'Py untuk ditampilkan dalam game.

- **Integrasi ke Script Ren'Py**

Dalam script. rpy, gambar hasil dari GAN/VAE dapat dimunculkan seperti biasa Dengan cara ini, pemain bisa melihat hasil dari proses AI secara langsung di dalam permainan:

```
label scene_hutan:
    # Tampilkan latar GAN
    scene bg_hutan_ai with fade

    # Tampilkan karakter VAE
    show ara senyum at center:
        zoom 0.8

    ara "Lihat! Sampah di hutan ini bisa kita daur ulang."
```

Gambar 4.9 Kode 6 – Contoh Integrasi Gambar ke Ren'py

- **Struktur Folder**

```
EcoAct_Game/
├── game/
│   ├── images/          # Aset GAN/VAE
│   ├── script.rpy       # Skrip utama
│   └── options.rpy
```

Gambar 4.10 – Struktur Folder

- **Penyederhanaan untuk proses *Deploy***

Karena Ren'Py tidak mendukung langsung pemrosesan Python lanjutan (seperti *TensorFlow*), proses *generating image* dilakukan sebelum *runtime*, lalu file hasilnya dipakai dalam Ren'Py. Namun, untuk keperluan pengembangan lebih lanjut, integrasi melalui Ren'Py- Python *Bridge* atau *eksternal tools* seperti *Flask* dapat dipertimbangkan.

4.4. Analisis Perbandingan Waktu Penyelesaian Karya Seni

Data diambil dari empat sumber Reddit, Waktu distandardisasi dalam jam (waktu kerja aktif, tidak termasuk istirahat/umpan balik klien):

- **Seniman Manusia**

  Laporan mandiri kualitatif dari seniman profesional, semi-profesional, dan hobi. Sumber:

  1. *https://www.reddit.com/r/DigitalPainting/comments/1bgeh7n/how_long_does_it_take_you_professional_artist_to/*
  2. *https://www.reddit.com/r/ArtistLounge/comments/12wybki/on_average_how_long_does_it_take_to_do_a/*
  3. *https://www.reddit.com/r/ArtistLounge/comments/stn7cp/how_long_does_it_take_you_to_complete_an/*

Sumber	Seniman	Jenis Karya Seni	Jam	Kutipan/Data Kunci
1	Kirbish88	Ilustrasi spot (mis., barang magis)	4–6	"Barang biasanya memakan waktu sekitar 4–6 jam."
1	Kirbish88	Potret karakter (seluruh tubuh)	8–12	"Potret karakter lebih seperti 8–12 jam."
1	Kirbish88	Ilustrasi halaman penuh	16–24	"Ilustrasi halaman penuh lebih seperti 16–24 jam."
1	MajorasKitten	Lukisan digital	4	"Saya dapat menyelesaikan lukisan dalam 4 jam."
1	MajorasKitten	Tradisional (ukuran A3)	8	"Untuk karya ukuran A3, saya membutuhkan dua hari, sekitar 8 jam maksimal."
1	MSMarenco	Sketsa warna hewan	1	"Saya dapat melukis sketsa warna hewan dalam 1 jam."

Sumber	Seniman	Jenis Karya Seni	Jam	Kutipan/Data Kunci
1	Wurlawyrm	Karya seni setengah jadi	6–30	"Antara 6 dan 30 jam untuk apa pun yang setengah jadi."
1	PippaBeeFree	Lukisan minyak (kanvas)	3–9	"Lukisan minyak memakan waktu antara 3 dan 9 jam untuk menyelesaikan satu kanvas."
1	filkearney	Adegan RPG (detail)	30–60	"Adegan interaktif satu halaman memakan waktu 30–60 jam."
1	collegeswant mesobad	Lukisan skala besar (36x24")	90	"Lukisan skala besar terbaru saya memakan waktu sekitar 90 jam."
2	FonsCordi	Karya digital	1–36	"Yang tercepat yang pernah saya lakukan adalah 1 jam, yang terlama 36 jam."
2	Artboggler	Karya yang dirender penuh	2–4	"Biasanya 2–4 jam untuk karya yang dirender penuh."
3	sketch_matt	Karakter (seluruh tubuh)	8–20	"8 hingga 20 jam untuk karakter seluruh tubuh."
3	sketch_matt	Ilustrasi penuh dengan latar belakang	20–40	"20–40 jam untuk ilustrasi penuh dengan latar belakang."
3	vexnir_art	Ilustrasi dengan latar belakang	4–10	"Ilustrasi saya memakan waktu 4 hingga 10 jam rata-rata."
3	FrozenEggPuck	Karakter detail + latar belakang	50–70	"Ilustrasi karakter dengan latar belakang memakan waktu 50–70 jam."
3	loudribs	Poster (seni vektor)	70+	"Setiap poster background memakan waktu setidaknya 70+ jam."
3	Deleted User 1	Karya seni penuh	6–7	"Karya seni penuh memakan waktu 6–7 jam."

Sumber	Seniman	Jenis Karya Seni	Jam	Kutipan/Data Kunci
3	Deteled User 2	Beragam	10–24	"10–24 jam. Latar belakang yang rumit memakan waktu lebih lama."

Tabel 4.6. Waktu Penyelesaian Seniman Manusia

Observasi Seniman Manusia:

a. Variabilitas luas: Waktu berkisar dari 1 jam (sketsa sederhana) hingga 90 jam (lukisan besar).

b. Kompleksitas menentukan waktu: Latar belakang, tingkat detail, dan medium (digital vs. tradisional) berdampak signifikan pada durasi.

c. Norma profesional: Sebagian besar profesional melaporkan 8–40 jam untuk karya yang dipoles.

- **Seniman AI**

Hasil jajak pendapat kuantitatif dari pengguna Stable Diffusion. Sumber: *https://www.reddit.com/r/StableDiffusion/comments/14l8lsu/on_average_how_much_time_are_you_taking_to/*

Kategori Waktu	Suara	% dari Total
< 15 menit	29	20,1%
15–30 menit	20	13,9%
30–60 menit	25	17,4%
1–2 jam	19	13,2%
2–3 jam	12	8,3%
> 3 jam	39	27,1%

Tabel 4.7: Waktu Penyelesaian Seniman AI

Observasi Seniman AI:

a. Mayoritas di bawah 1 jam**:** 51,4% pengguna menyelesaikan karya seni dalam ≤60 menit**.**

b. 27,1% melebihi 3 jam**:** Menunjukkan proyek kompleks (mis., penyempurnaan berulang, *inpainting, pembuatan karakter khusus*).

c. Waktu median**:** 30–60 menit (kategori paling sering).

- **Ringkasan Perbandingan**

Metrik	Seniman Manusia	Seniman AI
Rentang Khas	4–70 jam	15 menit–3+ jam
Waktu Rata-Rata	20–40 jam (kompleks)	30–60 menit (khas)
Pendorong Kecepatan	Keterampilan, medium, detail	Rekayasa *prompt*, GPU
Waktu Minimum	1 jam (sketsa sederhana)	<15 menit
Waktu Maksimum	90 jam (kanvas besar)	>3 jam (iteratif)

Tabel 4.8. Ringkasan Perbandingan

Dari Tabel ini dapat disimpulkan:

1. Perbedaan orde magnitudo: Seniman AI menyelesaikan pekerjaan 10 sampai 50× lebih cepat daripada seniman manusia untuk keluaran yang sebanding (mis., seni karakter: manusia 8–20 jam vs AI ≤1 jam).
2. Kendala manusia: Perbesaran detail, umpan balik klien, dan medium fisik memperpanjang waktu yang dibutuhkan manusia untuk membuat seni.
3. Efisiensi AI: Kecepatan berasal dari *rendering* otomatis, meskipun kasus >3 jam menyiratkan penyuntingan lanjutan.
4. Catatan metodologis: Waktu AI tidak termasuk eksperimen *prompt*; waktu manusia tidak termasuk periode tidak aktif (mis., menunggu umpan balik).