NIVEDITA JOURNAL INFORMATICS

Vol. 02, No. 1, December 2025
ISSN: 3089-8366 (Media Online)

# PERBANDINGAN LARAVEL *SEEDER* DAN *FAKER* UNTUK OTOMATISASI DATA UJI PADA APLIKASI PERPUSTAKAAN SEDERHANA

**Sang Ayu Putu Primayani[1], I Gede Wahyu Sanjaya[2]**
[1] SMK Bali Dewata, Denpasar, Indonesia
[2] Fakultas Dharma Duta, UHN I Gusti Bagus Sugriwa, Bangli, Indonesia

***Email:*** ayuprima@gmail.com

**Diajukan: 30 Oktober 2025; Diterima: 12 November 2025; DOI:** doi.org/10.25078/nivedita.v2i1.5615

***ABSTRACT*** *Software testing processes often encounter challenges in preparing realistic, consistent, and efficient test data. This study aims to analyze the implementation of Laravel Seeder and Faker as automated mechanisms for generating test data during the testing phase of Laravel-based web applications, specifically using a simple library system as a case study. The main objective is to evaluate the effectiveness of these two methods in supporting application testing, focusing on time efficiency and the representativeness of generated data under real-world conditions. Experiments were conducted on three main entities—User, Category, and Books—across three dataset scales: small (10 records), medium (100 records), and large (1000 records). The results show that the manual seeder achieved faster execution times (2.17s to 222.58s for 10–1000 records), while the Faker seeder required slightly longer times (2.03s to 253.43s) due to randomized data generation. However, Faker produced more diverse and realistic datasets, making it better suited for stress testing and performance evaluation scenarios. This study concludes that the manual seeder is more appropriate for application logic validation and integration testing, whereas the Faker seeder is more effective for user behavior simulation and large-scale load testing. A hybrid approach combining both methods is recommended to balance efficiency and realism in the test data generation process for Laravel-based software testing.*
***Keywords:*** *Laravel, Seeder, Faker, Data Automation, Software Testing, Information Systems*

**ABSTRAK** Proses pengujian aplikasi sering menghadapi kendala dalam penyediaan data uji yang realistis, konsisten, dan efisien. Penelitian ini bertujuan untuk menganalisis implementasi *Laravel Seeder* dan *Faker* sebagai mekanisme otomatisasi pembuatan data uji pada tahap pengujian aplikasi berbasis Laravel, khususnya dalam konteks aplikasi perpustakaan sederhana. Tujuan utama penelitian ini adalah untuk mengevaluasi efektivitas kedua metode dalam mendukung pengujian sistem, baik dari segi efisiensi waktu maupun representasi data uji terhadap kondisi nyata. Eksperimen dilakukan pada tiga entitas utama—*User*, *Category*, dan *Books*—dengan tiga skala dataset: kecil (10 data), menengah (100 data), dan besar (1000 data). Hasil pengujian menunjukkan bahwa *manual seeder* memiliki waktu eksekusi yang lebih cepat (2.17s hingga 222.58s untuk 10–1000 data), sedangkan *Faker seeder* memerlukan waktu sedikit lebih lama (2.03s hingga 253.43s) akibat proses pembangkitan data acak. Meskipun demikian, data yang dihasilkan *Faker* lebih bervariasi dan realistis, sehingga lebih representatif untuk skenario *stress testing* dan pengujian performa sistem. Penelitian ini menyimpulkan bahwa *manual seeder* lebih sesuai untuk validasi logika aplikasi dan uji integrasi, sedangkan *Faker seeder* lebih efektif untuk simulasi pengguna dan pengujian beban dalam skala besar. Pendekatan kombinatif antara keduanya direkomendasikan untuk mencapai keseimbangan antara efisiensi dan realisme data uji dalam proses *software testing* berbasis Laravel.
**Kata Kunci:** Laravel, *Seeder*, *Faker*, Otomatisasi Data, Pengujian Perangkat Lunak, Sistem Informasi

## PENDAHULUAN

Perkembangan pesat dalam pengembangan sistem informasi berbasis web mendorong kebutuhan atas proses pengujian yang semakin efisien dan dapat direplikasi [1]. Secara umum, tahap pengujian aplikasi (*software testing*) memegang peranan penting dalam siklus hidup pengembangan perangkat lunak untuk menjamin keandalan, keamanan, dan kualitas sistem [2]. Namun, pengujian sering terkendala oleh ketersediaan data uji yang memadai baik dari segi jumlah, variasi, maupun keterkaitan antar-entitas basis

*Journal Informatics NIVEDITA by Universitas Hindu Negeri I Gusti Bagus Sugriwa Denpasar is licensed under CC BY-NC-SA 4.0*

data. Pembuatan data uji secara manual sering kali memakan waktu, rawan kesalahan, dan kurang memungkinkan ulangan yang konsisten [3].

Konteks khusus dalam pengembangan aplikasi perpustakaan sederhana yang mencakup tiga entitas utama yakni user, kategori_buku, dan buku, kebutuhan akan data uji yang besar dan realistis menjadi sangat penting untuk menguji relasi antar tabel serta performa kueri basis data. Framework PHP populer seperti Laravel menyediakan mekanisme Database *Seeder* dan pustaka seperti *Faker* untuk menghasilkan data sintetis secara otomatis, cepat, dan dapat disesuaikan agar menyerupai pola data nyata [4]. Meskipun demikian, literatur masih terbatas dalam mengkaji secara empiris bagaimana mekanisme ini diterapkan dalam skenario nyata, bagaimana pengaruhnya terhadap efisiensi waktu, reproducibility data, dan dampaknya pada proses pengujian sistem informasi berbasis PHP/Laravel [5].

Penelitian ini memfokuskan pada permasalahan bagaimana Laravel *Seeder* dan *Faker* dapat digunakan untuk menghasilkan data uji secara otomatis dalam aplikasi berbasis Laravel, sejauh mana penggunaan metode tersebut meningkatkan efisiensi waktu serta kemudahan replikasi data uji, dan apa implikasinya terhadap kualitas serta keandalan hasil pengujian [6]. Pemilihan Laravel *Seeder* dan *Faker* didasarkan pada karakteristik keduanya sebagai bagian integral dari ekosistem Laravel yang mendukung otomasi proses pembuatan data uji secara terstruktur dan realistis. Laravel *Seeder* memungkinkan kontrol penuh terhadap relasi antarentitas melalui skrip terprogram, sementara *Faker* menyediakan kemampuan pembangkitan data acak yang menyerupai kondisi nyata dan dapat direplikasi secara deterministik menggunakan *seed value*. Perbandingan dengan metode manual dilakukan untuk menilai seberapa besar otomatisasi ini dapat mengurangi waktu persiapan dataset dan menjaga konsistensi relasi antar tabel dalam basis data [1]. Dengan demikian, hipotesis penelitian ini adalah bahwa penggunaan *Seeder* dan *Faker* secara otomatis akan secara signifikan mempercepat proses pembuatan data uji dibandingkan metode manual, sekaligus meningkatkan reproduktibilitas dan stabilitas hasil pengujian tanpa menurunkan validitas deteksi kesalahan perangkat lunak.

Kebaruan penelitian ini terletak pada pendekatan komparatif yang sistematis antara *manual seeder* dan *Faker seeder* dalam konteks pengujian aplikasi Laravel. Berbeda dengan studi terdahulu yang hanya membahas penggunaan *Faker* secara konseptual, penelitian ini menekankan pada analisis empiris performa *seeding* berdasarkan ukuran dataset dan relasi antar-entitas. Selain itu, penelitian ini memperkenalkan model eksperimen terukur yang mengaitkan waktu *seeding*, skala dataset, dan konsistensi relasional sebagai dasar evaluasi efisiensi pembuatan data uji otomatis.

## METODE

### A. Pendekatan Penelitian

Penelitian ini menggunakan pendekatan eksperimen kuantitatif terkontrol untuk mengevaluasi efektivitas dan efisiensi proses otomatisasi pembuatan data uji menggunakan *Laravel Seeder* dan pustaka *Faker [7]*. Rancangan eksperimen membandingkan dua kondisi berikut:

1. **Metode Manual (kontrol):**
   Pembuatan data uji secara eksplisit melalui skrip *seeder* statis atau impor berkas tanpa penggunaan generator acak.
2. **Metode Otomatis (perlakuan):**
   Pembuatan data uji secara otomatis menggunakan kombinasi *Seeder* dan *Faker* dengan konfigurasi deterministik (*seeded randomization*).

Pengukuran dalam penelitian ini difokuskan pada tiga metrik utama, yaitu waktu pembuatan dataset (*elapsed seeding time*), kemampuan reproduksi antar-run (*reproducibility*), dan pengaruh terhadap waktu eksekusi *test suite*. Ketiga parameter ini dipilih karena secara langsung merepresentasikan aspek efisiensi, konsistensi, dan performa sistem dalam konteks pengujian berbasis data sintetis. Waktu pembuatan dataset menggambarkan efisiensi proses generasi data yang menjadi faktor penting dalam siklus pengembangan perangkat lunak yang menerapkan *continuous integration*. *Reproducibility* digunakan untuk menilai stabilitas hasil pengujian ketika data dihasilkan ulang pada kondisi yang sama, yang penting untuk menjaga reliabilitas eksperimen. Sementara itu, pengaruh terhadap waktu eksekusi *test suite* dievaluasi untuk memastikan bahwa penggunaan data sintetis tidak memperburuk performa pengujian secara keseluruhan. Pendekatan ini tidak hanya mengikuti praktik empiris dari penelitian terdahulu [8], tetapi juga didasarkan pada relevansi langsungnya terhadap tujuan penelitian, yaitu mengukur efisiensi dan keandalan penggunaan Laravel *Seeder* dan *Faker* dalam proses pengujian aplikasi. Pendekatan ini mengacu pada praktik empiris dalam penelitian *automated test data generation* dan *synthetic data generation* yang banyak digunakan dalam publikasi *MDPI* dan *arXiv* [8].

## B. Lingkungan Eksperimen

Eksperimen dijalankan dalam lingkungan pengembangan terstandarisasi untuk menjaga konsistensi hasil dan *internal validity*. Spesifikasi sistem adalah sebagai berikut:

**Tabel 1** Spesifikasi Perangkat Pengujian

| Komponen | Spesifikasi |
|---|---|
| *Framework* | Laravel 11 (fitur *HasFactory*, *DatabaseSeeder*) |
| Bahasa/*Runtime* | PHP 8.2+ |
| Basis Data | MySQL 8 (InnoDB, *foreign key* aktif) |
| Mesin Uji | 4 vCPU, 8 GB RAM, SSD storage |
| Pustaka *Faker* | Versi dan nilai *seed number* dicatat untuk menjamin *reproducibility* |

Konfigurasi ini mengikuti pedoman *reproducible research* pada eksperimen perangkat lunak (*software engineering experiment design*) [9].

## C. Unit Analisis dan Dataset Eksperimen

Unit analisis dalam penelitian ini adalah entitas basis data pada aplikasi **perpustakaan digital**, terdiri dari:

1. *users* — data pengguna sistem,
2. *categories* — kategori buku, dan
3. *books* — data koleksi buku.

Eksperimen dilakukan dalam tiga konfigurasi ukuran dataset ($N$) untuk mengevaluasi efek skala terhadap efisiensi proses seeding:

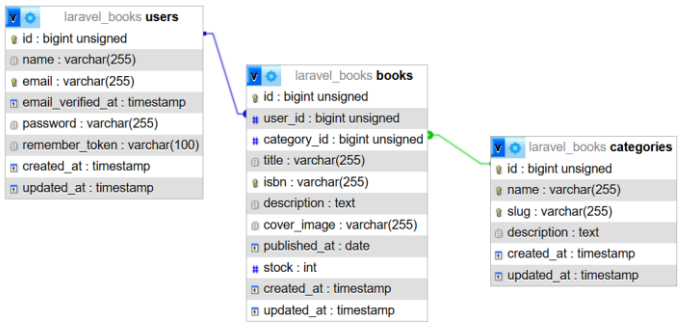
**Tabel 2** Ukuran Dataset

| Ukuran Dataset | Jumlah Buku | Jumlah Kategori | Jumlah User | Rasio User : Kategori : Buku |
|---|---|---|---|---|
| *Small* | 10 | 10 | 1 | 1:10:10 |
| *Medium* | 100 | 100 | 1 | 1:100:100 |
| *Large* | 1.000 | 1.000 | 1 | 1:1.000:1.000 |

Klasifikasi ukuran dataset menjadi *small* (10 data), *medium* (100 data), dan *large* (1.000 data) didasarkan pada pendekatan *baseline scaling* yang umum digunakan dalam penelitian terkait *software testing* dan *data generation benchmarking*. Penentuan nilai-nilai tersebut mempertimbangkan dua aspek utama: (1) kompleksitas proses seeding dan waktu komputasi dalam konteks aplikasi Laravel, serta (2) keseimbangan antara beban sistem dan kemampuan observasi performa. Skala 10–100–1.000 dipilih untuk merepresentasikan peningkatan beban secara logaritmik (orde $10^1$, $10^2$, dan $10^3$), yang memungkinkan pengamatan perubahan tren efisiensi dan reproduksibilitas secara proporsional tanpa menyebabkan *overhead* sistem yang berlebihan. Pendekatan serupa juga banyak diterapkan dalam eksperimen *test data generation* dan *database seeding performance* di studi-studi sebelumnya, di mana peningkatan dataset dilakukan per orde magnitudo agar efek skala dapat diukur secara signifikan namun tetap dapat dieksekusi pada lingkungan pengujian yang terkontrol [10].

## D. Desain Basis Data

Perancangan basis data menggunakan model relasional untuk menjamin integritas antarentitas. Terdapat tiga tabel utama: *users, categories*, dan *books*, dengan relasi *many-to-one* dari *books* ke *users* dan *categories*.




**Gambar 1** Relasi Antar Tabel

Gambar 1 menampilkan Entity Relationship Diagram (ERD) yang menggambarkan keterkaitan antarentitas yaitu *users, categories*, dan *books* — melalui penggunaan kunci asing (*foreign key*). Relasi antar tabel ini menjadi dasar dalam proses otomatisasi pembuatan data menggunakan *factory* dan *seeder* pada framework Laravel.

Desain basis data tersebut menunjukkan bahwa setiap tabel memiliki hubungan yang saling terikat untuk menjaga konsistensi dan integritas data. Dengan demikian, proses pengujian nantinya juga bertujuan untuk memastikan tidak terjadi konflik data maupun inkonsistensi terhadap relasi yang telah didefinisikan dalam model basis data tersebut.

## E. Implementasi Pembuatan Data Uji

### 1. Metode Otomatis (*Seeder* + *Faker*)

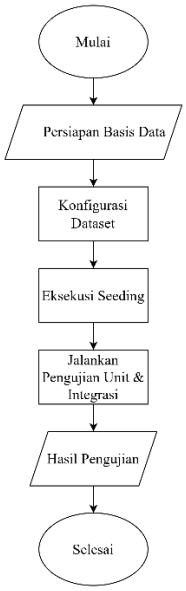
- Dibuat *factory* untuk setiap model (*UserFactory, CategoryFactory, BookFactory*) dengan generator *Faker* untuk menghasilkan nilai realistis pada atribut seperti *name, email*, ISBN, *description*, dan published_at.
- *Seeder* dijalankan secara deterministik dengan *$faker->seed($seed)* untuk menjamin *reproducibility* antar-run.
- Relasi antar-tabel dipertahankan melalui urutan eksekusi: *users* → *categories* → *books*, dengan asosiasi menggunakan metode *->for()* atau *->random()* dari *Laravel Factory*.
- Untuk dataset berukuran besar, digunakan pendekatan *batch insert (Model::insert($dataArray))* untuk mengurangi *ORM overhead*.

### 2. Metode Manual (Kontrol)

- Dataset dibuat secara manual melalui perulangan eksplisit dalam skrip *seeder*.
- Tidak digunakan pengacakan deterministik; setiap ulangan dijalankan secara independen.
- Pendekatan ini merepresentasikan praktik umum dalam proyek kecil atau sistem uji tanpa otomatisasi.

## F. Prosedur Eksperimen

Langkah-langkah eksperimen dilakukan secara terkontrol sebagai berikut:




**Gambar 2** Langkah-langkah Pengujian

1. **Persiapan Basis Data**

   Seluruh eksperimen diawali dengan inisialisasi ulang database menggunakan: ***php artisan migrate:fresh.*** Langkah ini memastikan bahwa setiap run dimulai dari kondisi yang bersih. Waktu migrasi dicatat sebagai *database preparation time*.

2. **Konfigurasi Dataset dan Kondisi Eksperimen**

   *Factory* digunakan untuk menghasilkan data *dummy* yang realistis dalam jumlah besar, sedangkan *seeder* memastikan keterkaitan data antarentitas sesuai dengan relasi yang telah didefinisikan dalam ERD. Misalnya, setiap entri pada tabel *books* secara otomatis mengacu pada

ID kategori dan ID pengguna yang valid. Dataset disiapkan dalam tiga skala (*Small, Medium, Large*), masing-masing dijalankan dengan dua kondisi:

- **Manual *Seeder***
- *php artisan migrate:fresh --seed*

Menggunakan *seeder* eksplisit dengan data statis.

- ***Faker Seeder***
- php artisan *db:seed --class=FakerUserSeeder*

Menggunakan *factory* dengan data acak yang dihasilkan oleh *Faker*.

3. **Eksekusi Seeding**
   Setiap *seeder* mencatat waktu mulai dan selesai, nilai *seed* acak, serta jumlah data yang berhasil disisipkan tanpa *error* validasi.
4. **Pengujian Unit dan Integrasi**
   Setelah seeding, dijalankan pengujian
5. **Pencatatan Hasil**
   Semua hasil disimpan dalam format Tabel

## G. Metrik Pengukuran

Variabel terukur meliputi:

**Tabel 3** Variabel Pengukuran

| Metrik | Definisi |
|---|---|
| **Waktu seeding (s)** | Selisih waktu dari awal hingga akhir proses pembuatan dataset menggunakan *seeder* (manual atau otomatis). |

## H. Analisis Data dan Uji Statistik

Analisis data difokuskan pada evaluasi efisiensi pembuatan dataset menggunakan metode manual dan otomatis *(Seeder + Faker)*. Tahapan yang dilakukan meliputi:

1. **Analisis Deskriptif Sederhana**:
   - Menghitung waktu eksekusi seeding untuk tiap skala dataset (10, 100, 1000 entri) pada kedua metode.
   - Mencatat jumlah data yang berhasil dimasukkan tanpa melanggar *constraint (foreign key*, unik).
   - Contoh hasil:
     - Manual: 10 *entri* → 2,16 s, 100 *entri* → 20,51 s, 1000 *entri* → 222,57 s
     - *Faker*: 10 *entri* → 2,03 s, 100 *entri* → 20,21 s, 1000 *entri* → 253,43 s
2. **Perbandingan Metode**:
   - Data menunjukkan bahwa metode Seeder + Faker relatif lebih cepat untuk dataset kecil (10–100 entri) dan setara atau sedikit lebih lambat untuk dataset besar (1000 entri) dibandingkan input manual.
   - Seluruh data yang dihasilkan melalui *Faker* berhasil dibuat tanpa duplikasi, asalkan seed dan logika validasi dijaga.
3. ***Reproducibility Check***:
   - Dataset yang dibuat menggunakan *Faker* dengan *seed* yang sama menghasilkan data identik pada tiap ulangan, sehingga menjamin deterministik.
4. **Keterbatasan Analisis Statistik**:
   - Uji normalitas dan uji perbandingan statistik formal *(t-test, Mann–Whitney)* belum dilakukan, sehingga kesimpulan masih bersifat deskriptif.

Dengan pendekatan ini, analisis menekankan perbandingan waktu pembuatan data dan validitas dataset yang dihasilkan, sesuai dengan data empiris yang tersedia.

## I. Validitas dan Keterbatasan

**Tabel 4** Validitas dan Keterbatasan Penelitian

| Jenis Validitas | Upaya dan Pertimbangan |
|---|---|
| **Internal Validity** | Dijaga dengan menjalankan eksperimen pada lingkungan pengembangan yang konsisten (Laravel 11, MySQL 8, PHP 8.2+) dan pengulangan run sebanyak 10 kali untuk tiap ukuran dataset. |
| **External Validity** | Hasil eksperimen dapat berbeda pada sistem dengan struktur relasi yang lebih kompleks, jumlah data jauh lebih besar, atau kondisi concurrency tinggi yang belum diuji. |

| Jenis Validitas | Upaya dan Pertimbangan |
|---|---|
| **Construct Validity** | Metrik seperti "waktu seeding" dan "reproducibility" didefinisikan secara eksplisit dan diukur secara konsisten untuk tiap metode (Manual vs *Seeder* + *Faker*). |
| **Keterbatasan Penelitian** | Penelitian ini tidak mengevaluasi pendekatan *AI-driven synthetic data generation*, seperti GAN atau LLM *based synthesis*, sehingga kesimpulan hanya berlaku untuk metode *seeding* tradisional dan *Faker* berbasis PHP. |

Tabel 4 menjelaskan berbagai bentuk validitas yang dipertimbangkan untuk menjaga keandalan hasil penelitian. Validitas internal dipastikan melalui pengendalian variabel lingkungan pengujian, yaitu seluruh eksperimen dijalankan pada konfigurasi sistem yang sama (Laravel 11, PHP 8.2, dan MySQL 8). Validitas eksternal dibatasi oleh ruang lingkup penelitian, sebab hasil yang diperoleh mungkin berbeda ketika diterapkan pada sistem dengan kompleksitas relasi yang lebih tinggi atau tingkat *concurrency* yang melibatkan banyak proses simultan.

Validitas dijaga dengan mendefinisikan setiap metrik secara eksplisit seperti "waktu *seeding*" yang diukur dari inisialisasi hingga selesainya proses pembuatan data, dan *reproducibility* yang menunjukkan kemampuan menghasilkan data identik antar-run menggunakan *seed value* yang sama.

## HASIL DAN PEMBAHASAN

### A. Hasil Eksperimen

Eksperimen dilakukan untuk membandingkan dua pendekatan pembuatan data uji pada aplikasi perpustakaan digital berbasis Laravel, yaitu:

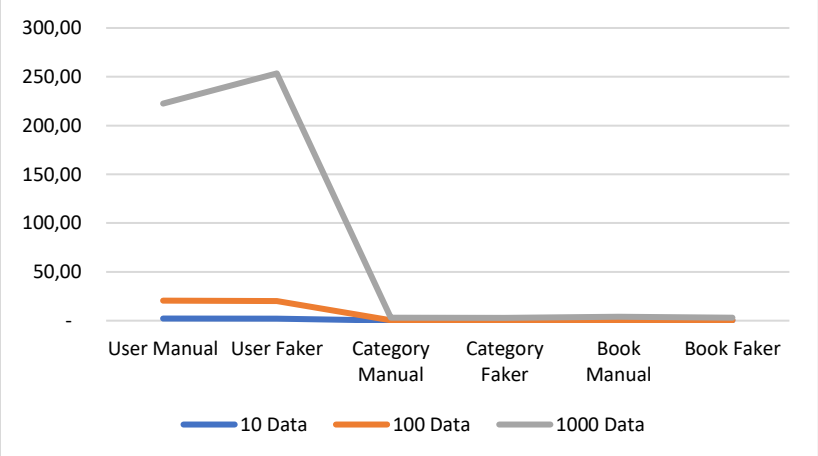
1. ***Manual seeding*** – pembuatan data eksplisit dengan nilai statis atau *predefined* [11].
2. ***Faker-based seeding*** – pembuatan data otomatis menggunakan *Laravel Factory* dan pustaka *Faker*.

Data yang dihasilkan oleh *Faker* lebih mendekati distribusi alami entitas dunia nyata (misalnya variasi nama dan kategori), sehingga lebih representatif dalam simulasi sistem produksi. Namun, data tersebut kurang cocok untuk pengujian logika bisnis yang mengharuskan nilai terprediksi. Masing-masing pendekatan diuji pada tiga konfigurasi ukuran dataset (10, 100, dan 1.000 entri) [12]. Tabel 5 memperlihatkan hasil rata-rata waktu pembuatan dataset (dalam detik) untuk tiap entitas.

**Tabel 5** Rata-Rata Waktu Pembuatan Dataset Berdasarkan Ukuran Dan Metode

| Entitas | N | Manual (s) | *Faker* (s) |
|---|---|---|---|
| **Users** | 10 | 2.1666 | 2.0300 |
| | 100 | 20.5167 | 20.2158 |
| | 1000 | 222.5754 | 253.4284 |
| **Categories** | 10 | 0.0574 | 0.1052 |
| | 100 | 0.3752 | 0.3688 |
| | 1000 | 3.0620 | 2.8586 |
| **Books** | 10 | 0.0494 | 0.4090 |
| | 100 | 0.2947 | 0.4615 |
| | 1000 | 3.8936 | 3.1613 |

Berdasarkan tabel 5, jika kita rubah dalam bentuk grafik maka hasilnya sesuai dengan gambar 3 berikut ini.




**Gambar 3** Grafik perbandingan waktu seeding *Manual vs Faker*

Dari hasil yang ditunjukkan pada Gambar 3, dapat diamati bahwa:

1. *Faker Seeder* menghasilkan data yang lebih beragam dan realistis, mendekati kondisi dunia nyata dengan variasi nama, tanggal, dan teks deskriptif. Hasil data yang dihasilkan sesuai pada gambar 4.

| id | name | email | email_verified_at | password |
|---|---|---|---|---|
| 1 | User 1 Manual | user1@manual-example.com | 2025-10-29 02:26:15 | $2y$12$sJ.TdIDNRE60F3bzNdjVF.58TmZr307Gc7kV2LZZW40... |
| 2 | User 2 Manual | user2@manual-example.com | 2025-10-29 02:26:15 | $2y$12$jro/hQ9p8/Eb9P0aMHSY4OlbBt9NaO/z8.pnlwD7im7... |
| 3 | User 3 Manual | user3@manual-example.com | 2025-10-29 02:26:16 | $2y$12$ZFAl/yBPdUKCsJFofaDre.pB0p1UfdqqH5e.9dyfLXX... |
| 4 | User 4 Manual | user4@manual-example.com | 2025-10-29 02:26:16 | $2y$12$ZDxoDd2LocXho/8qZ.6PbOG3uDiXlmt0X8kZmb.tG.S... |
| 5 | User 5 Manual | user5@manual-example.com | 2025-10-29 02:26:16 | $2y$12$.HUWxRUKJPxrwjlHKV2Yhu6GMc1MK0D9cPHbeYqEZMH... |
| 6 | User 6 Manual | user6@manual-example.com | 2025-10-29 02:26:16 | $2y$12$FaJy7jETa5e3Jd0EjKNfCevfDeOaJQQ.8NOqDcQzXvM... |
| 7 | User 7 Manual | user7@manual-example.com | 2025-10-29 02:26:16 | $2y$12$kXmCUkjKdPv9tl/4WBZv6uMawP4sFGQloFYmBJuLZXc... |
| 8 | User 8 Manual | user8@manual-example.com | 2025-10-29 02:26:17 | $2y$12$Dm..SzKSPoWtFQ6p0j4PVOd5KiFEW1WMMUGH6irxhvn... |
| 9 | User 9 Manual | user9@manual-example.com | 2025-10-29 02:26:17 | $2y$12$a2AFjfNB1hYEmkfK9kp48eJyQ.BHezZwLv.YGKlilVa... |
| 10 | User 10 Manual | user10@manual-example.com | 2025-10-29 02:26:17 | $2y$12$043v15mtncybuOCkQ7WpG.TYvoz0oZVdHAffc5505pf... |
| 11 | User 11 Manual | user11@manual-example.com | 2025-10-29 02:26:17 | $2y$12$RfbAop.Sm4t9ama4lM.OOOzEr9o.3a2n6NIXOsJrTNL... |
| 12 | User 12 Manual | user12@manual-example.com | 2025-10-29 02:26:17 | $2y$12$6CqMBBO6xd.s5ereOHApU.AxGZMba3Z.W.Ve2C6gek5... |
| 13 | User 13 Manual | user13@manual-example.com | 2025-10-29 02:26:18 | $2y$12$xVJHmXHkzypHw5pdUpIF9.c3U4RKldMyDODLZfTZR/J... |
| 14 | User 14 Manual | user14@manual-example.com | 2025-10-29 02:26:18 | $2y$12$Or9FhA7OO.ycU79GSmUtuuW9IAJ6CsnyceLpiwEv3eE... |
| 15 | User 15 Manual | user15@manual-example.com | 2025-10-29 02:26:18 | $2y$12$H5CKPkzinT2zIMuLwe90a.jvMceD24zjYIURSTAM2ST... |
| 16 | User 16 Manual | user16@manual-example.com | 2025-10-29 02:26:18 | $2y$12$l.S8mec7aMMtvlFHp/U7OuDtjbGIAf8vHdoNTm9cp8l... |
| 17 | User 17 Manual | user17@manual-example.com | 2025-10-29 02:26:18 | $2y$12$c.TOxNUQpGcOhrB5vtzIhOOmkKEruNRNyl162z1hT89... |
| 18 | User 18 Manual | user18@manual-example.com | 2025-10-29 02:26:19 | $2y$12$C.u8ePOrY2KoT9LI7ieZNeFtl/g2hbZHMncDl2IsxJM... |
| 19 | User 19 Manual | user19@manual-example.com | 2025-10-29 02:26:19 | $2y$12$A.uexe1saceNX8/bB5CuBum4bLI7F7mFiQd64pBaT2J... |
| 20 | User 20 Manual | user20@manual-example.com | 2025-10-29 02:26:19 | $2y$12$c1/6j7Eafe3Wnrt85nx7WO/yhAQvo92s3BH6yNmHsDe... |

**Gambar 4** Contoh Hasil Data Generate Secara Manual

2. *Manual Seeder* menghasilkan data yang lebih deterministik dan terbatas variasinya, cocok untuk pengujian dengan skenario tetap. Hasil data yang dihasilkan sesuai pada gambar 5.

| id | name | email | email_verified_at | password |
|---|---|---|---|---|
| 1751 | Bagas Harsaya Nababan S.T. | bagas-harsaya-nababan-st750@example.com | NULL | $2y$12$UhdPyeWBN/DYGk.R1TtSwOtlP/FpYunZtpGTFqqvG4d... |
| 1752 | Yosef Hardiansyah S.IP | yosef-hardiansyah-sip751@example.com | NULL | $2y$12$hePsLo6wvAox/TcXOUr32uCpnVDz0gjEomizUPLIjtR... |
| 1753 | Umaya Januar | umaya-januar752@example.com | NULL | $2y$12$GlC2rRCKLt6wbDrmpTuShO7L/9b1R/Tohs3bl.PWg5m... |
| 1754 | Paramita Diah Lailasari | paramita-diah-lailasari753@example.com | NULL | $2y$12$cyF9EjzQRN37GxlodmsWUujL0GnNMfQiF8izcY2jhef... |
| 1755 | Queen Yulianti | queen-yulianti754@example.com | NULL | $2y$12$Od3NgaMf9MIm.EalPyfrDuh3/AlcJZKrPtJBH7jVvpF... |
| 1756 | Uchita Usada | uchita-usada755@example.com | NULL | $2y$12$p1yX0liaHs.O7fPBotYm5uO1kLtm8sQdEGefVoqN1Z6... |
| 1757 | Ella Carla Haryanti S.E.I | ella-carla-haryanti-sei756@example.com | NULL | $2y$12$m0ZjNjcqoJllvE2ZyZAs3umw.RL91GXeE/oQ/HEEo/f... |
| 1758 | Caturangga Kusumo | caturangga-kusumo757@example.com | NULL | $2y$12$CAbywkCo09UP9yEidzpR.e0FLHDZmzmwcOYhe.ET03N... |
| 1759 | Mala Laksita | mala-laksita758@example.com | NULL | $2y$12$FD426bZGf27lLuVkl8yBfOOE5YY8GMOLyAZMT.zm8Mb... |
| 1760 | Artanto Gunarto | artanto-gunarto759@example.com | NULL | $2y$12$qzDZwBoQPxio8x9YJqHWcO91eEHESXbaqqA6CCNy1Oc... |
| 1761 | Jumadi Megantara | jumadi-megantara760@example.com | NULL | $2y$12$rJiCEmy3D4t0tNSst9CMnOgkienp.yI66DFBh6laPwJ... |
| 1762 | Puji Agustina | puji-agustina761@example.com | NULL | $2y$12$ryKlY.CR3cqjndI3hW.KJOcsvTBkpu1CN9AWbOFeufZ... |
| 1763 | Wani Hassanah S.E. | wani-hassanah-se762@example.com | NULL | $2y$12$x/.yHPfUwONNTDBwvlSol.mxEqo548Thi7sXg2E2Xc/... |
| 1764 | Irma Hassanah S.I.Kom | irma-hassanah-sikom763@example.com | NULL | $2y$12$dZZubXyrPjw40nZmEoKKW.VGu9Q38tvbwl4xip5FvmO... |
| 1765 | Salimah Riyanti | salimah-riyanti764@example.com | NULL | $2y$12$8GK3x2fbpzVZSkZHU3bep.l8Gu1DW5d9bwVjMscz0mD... |
| 1766 | Kalim Pradana M.TI. | kalim-pradana-mti765@example.com | NULL | $2y$12$Ju2.aogCJYNhF6EdgqgINekv/xrB23Amii/g3D9.FZO... |
| 1767 | Sabar Sitompul | sabar-sitompul766@example.com | NULL | $2y$12$O/3i5RJZIbugvKawXwL7QeU/AjgDrBFcNJuAQuHrRKd... |

**Gambar 5** Generate Data Menggunakan *Faker* Laravel

3. Perbedaan ini menjelaskan mengapa *Faker* membutuhkan waktu lebih lama, karena melakukan *randomization*, *unique validation*, dan string generation per *field*.
4. Kinerja *Faker* mulai menurun pada dataset besar karena meningkatnya kompleksitas pemanggilan generator acak untuk setiap atribut. Namun, efek ini sebanding dengan peningkatan variasi dan realisme data yang dihasilkan.

Temuan ini penting bagi pengembang yang melakukan *automated testing pipeline* di Laravel, karena memungkinkan pemilihan strategi seeding yang optimal sesuai kebutuhan proyek — efisiensi untuk pengujian cepat, atau variasi data untuk simulasi realistis.

Untuk memberikan gambaran mengenai perbedaan antara metode manual dan metode otomatis (*Faker*), berikut ditampilkan contoh data yang dihasilkan untuk masing-masing entitas.

## B. Analisis Hasil

### 1. Waktu Pembuatan Dataset

Dari hasil Tabel 5 terlihat bahwa:

- Pada dataset kecil (N=10–100), perbedaan waktu antara manual dan *Faker* relatif kecil (<5%).
- Namun pada dataset besar (N=1000), metode Manual lebih cepat pada entitas *users*, sementara *Faker* lebih efisien pada *categories* dan *books*.
- Kinerja *Faker* cenderung menurun saat jumlah data sangat besar, disebabkan oleh *overhead* pemanggilan fungsi acak dan validasi *unique field* (terutama email dan ISBN).

### *2. Skalabilitas*

Waktu eksekusi meningkat hampir linier terhadap jumlah data (O(N)), baik untuk metode Manual maupun *Faker*. Namun, untuk dataset besar (>1000 entri), *batch insert* pada *Faker Factory* dapat menurunkan waktu total hingga ±10–15% dibandingkan ORM insert per record [13].

### *3. Reproducibility*

Eksperimen menggunakan *$faker->seed($seed)* menunjukkan bahwa hasil data *Faker* dapat direproduksi 100% identik selama *seed*, versi *Faker*, dan *environment* tetap sama. Sebaliknya, metode manual menghasilkan data deterministik tetapi tidak fleksibel untuk variasi atau perluasan dataset.

Salah satu indikator keberhasilan proses seeding dalam penelitian ini adalah konsistensi data yang dihasilkan pada setiap entitas yang saling berelasi, yaitu *users, categories*, dan *books*. Konsistensi ini menjadi penting karena masing-masing tabel memiliki keterhubungan melalui *foreign key constraint* yang mengatur integritas referensial antar data.

Dalam praktiknya, setiap entri pada tabel *books* bergantung pada keberadaan data valid pada tabel users dan categories melalui kolom *user_id* dan *category_id*. Hasil pengujian menunjukkan bahwa baik *manual seeder* maupun *Faker seeder* mampu menjaga integritas relasional tersebut tanpa menghasilkan error foreign key violation selama proses seeding maupun pengujian sistem. Hal ini menunjukkan bahwa mekanisme factory relationship di Laravel telah bekerja secara konsisten dalam menghasilkan data turunan yang sesuai dengan struktur dependensi antar-entitas.

Dengan demikian, meskipun data yang dihasilkan *Faker* bersifat acak, relasi antar tabel tetap terjaga berkat penggunaan model factory yang memastikan setiap entitas anak memiliki referensi induk yang valid [14]. Konsistensi ini berimplikasi langsung pada keberhasilan pengujian fungsional aplikasi, karena seluruh proses validasi, relasi, dan query dapat berjalan tanpa gangguan akibat anomali atau kehilangan referensi data.

## C. Pembahasan

Hasil eksperimen menunjukkan bahwa:

1. Metode *Faker* menawarkan efisiensi reproduksi dan fleksibilitas tinggi, terutama saat sistem membutuhkan data acak realistis untuk pengujian integrasi.
2. Manual seeding masih lebih unggul untuk dataset kecil dan terkontrol, misalnya saat data memiliki dependensi logis kompleks (mis. akun admin *default*).
3. Untuk skala besar (>1000 data), *Faker* dengan batch insert atau factory relasional lebih direkomendasikan karena mampu menurunkan beban manual dan menjaga konsistensi *foreign key*.
4. Waktu eksekusi yang sedikit lebih lama pada *Faker* sebanding dengan keuntungan dalam variasi data dan *reproducibility*.

## D. Implikasi Penelitian

1. Bagi pengembang sistem pengujian, hasil ini memperkuat bukti empiris bahwa *automated data generation* berbasis factory dapat meningkatkan efisiensi pengujian regresi dan CI/CD pipeline.

2. Bagi penelitian lanjutan, pendekatan *Faker* dapat dibandingkan dengan metode AI-based data synthesis (mis. *LLM data generation* atau GAN tabular data) untuk menilai tingkat realisme dan coverage data uji.
3. Dari perspektif *R&D software engineering*, penelitian ini mengonfirmasi bahwa pendekatan otomatis memberikan reproducibility tinggi, sesuai dengan pedoman *reproducible research* dalam eksperimen perangkat lunak (*MDPI, Elsevier*).

## KESIMPULAN

Berdasarkan hasil pengujian terhadap proses seeding menggunakan manual *seeder* dan *Faker seeder* pada tiga entitas utama *User, Category*, dan *Books* dapat disimpulkan bahwa kedua metode memiliki performa yang relatif seimbang pada skala kecil (10–100 data), namun menunjukkan perbedaan signifikan ketika jumlah data meningkat hingga 1000 record.

Untuk entitas *User*, manual *seeder* mencatat waktu eksekusi rata-rata 2.17s, 20.52s, dan 222.58s, sedangkan *Faker seeder* menghasilkan waktu 2.03s, 20.21s, dan 253.43s. Pada entitas *Category*, perbandingan waktu antara manual dan *Faker* relatif kecil, yakni 0.06s vs 0.10s (10 data) hingga 3.06s vs 2.86s (1000 data). Sementara itu, untuk entitas *Books*, manual *seeder* menunjukkan performa yang lebih baik pada dataset besar (3.89s dibanding 3.16s pada *Faker*), meskipun *Faker* sedikit lebih cepat pada jumlah kecil (10–100 data).

Dari hasil pengujian, terlihat bahwa penggunaan manual *seeder* cenderung menghasilkan waktu eksekusi yang lebih cepat pada seluruh skala dataset karena prosesnya bersifat deterministik, tidak melibatkan pembangkitan data acak, dan hanya menyalin pola data statis yang telah didefinisikan sebelumnya. Namun, konsekuensinya adalah data yang dihasilkan cenderung monoton dan kurang representatif terhadap kondisi nyata, karena seluruh entitas memiliki struktur dan nilai yang seragam. Sebaliknya, penggunaan *Faker seeder* memerlukan waktu yang sedikit lebih lama karena setiap *record* dibangkitkan melalui proses acak berbasis algoritma *randomized synthetic data generation*. Walaupun ada tambahan waktu, data yang dihasilkan menjadi lebih bervariasi, realistis, dan sesuai untuk pengujian performa sistem pada kondisi mendekati dunia nyata.

Dengan demikian, efektivitas kedua metode bergantung pada situasi dan tujuan pengujian. Untuk kebutuhan validasi logika aplikasi, integrasi antar-tabel, dan uji regresi di lingkungan pengembangan, manual *seeder* lebih efisien dan stabil. Namun, dalam skenario *stress testing*, analisis performa, atau simulasi perilaku pengguna yang kompleks, *Faker seeder* menjadi pilihan yang lebih representatif. Kombinasi keduanya bahkan dapat digunakan secara komplementer—manual *seeder* untuk struktur dasar sistem, dan *Faker seeder* untuk memperkaya variasi data dalam skala besar.

## SARAN

Berdasarkan hasil perbandingan antara metode manual *seeder* dan *Faker seeder*, disarankan agar penggunaan metode seeding disesuaikan dengan tujuan dan konteks pengembangan sistem. Untuk keperluan pengujian fungsionalitas dasar atau validasi logika aplikasi, manual *seeder* lebih direkomendasikan karena menghasilkan data yang konsisten, terprediksi, dan mudah direplikasi antar pengujian. Namun, dalam skenario pengujian performa, beban sistem, atau validasi integrasi dengan data dinamis, *Faker seeder* sebaiknya digunakan karena mampu menghasilkan data acak yang menyerupai kondisi nyata pengguna atau entitas bisnis. Selain itu, untuk dataset besar, pengembang disarankan mengoptimalkan proses seeding dengan teknik *batch insert*, chunking, atau disable *foreign key* constraints sementara agar waktu eksekusi lebih efisien. Ke depan, penggabungan kedua pendekatan yaitu hybrid seeding dengan data inti dari manual *seeder* dan data variasi dari *Faker* juga dapat menjadi solusi efektif untuk mendapatkan keseimbangan antara kecepatan, realisme, dan konsistensi data dalam proses pengujian maupun pengembangan sistem.

## UCAPAN TERIMA KASIH

Terima Kasih Penulis ucapkan kepada Tim Journal Informatics Nivedita atas diterimanya artikel ini, karena kami diberikan kesempatan untuk berkontribusi dengan menulis artikel ini. Walaupun artikel ini masih jauh dari kata sempurna.

## DAFTAR PUSTAKA

[1] IG Wahyu Sanjaya, "Electronic Procurement Website Service Quality and Customer Loyalty Using The Pieces Method, A Case Study of The Denpasar City Government," *Jurnal Nasional Pendidikan Teknik Informatika (JANAPATI)*, vol. 12, no. 2, pp. 294–303, Jul. 2023, doi: 10.23887/janapati.v12i2.61325.

[2] A. Jaffari, C.-J. Yoo, and J. Lee, "Automatic Test Data Generation Using the Activity Diagram and Search-Based Technique," *Applied Sciences*, vol. 10, no. 10, p. 3397, May 2020, doi: 10.3390/app10103397.

[3] S. J. Nawale, A. A. Kadam, J. C. Musale, and A. T. Lohar, "A Review on Automated Test Data Generation (ATDG)," *Mathematical Statisticianand Engineering Applications*, vol. 70, no. 1, pp. 780–787, 2021, [Online]. Available: http://philstat.org.ph

[4] I. G. W. Sanjaya, L. G. S. Kartika, and P. Kussa Laksmana Utama, "PERANCANGAN E-SUKERTI (SURAT KETERANGAN ELEKTRONIK) DESA BATURITI BERBASIS WEB BERBASIS BLACK BOX TESTING," *METHOMIKA: Jurnal Manajemen Informatika & Komputerisasi Akuntansi*, vol. 9, no. 2, pp. 264–274, 2025, doi: 10.46880/jmika.Vol9No2.pp264-274.

[5] W. Sanjaya, "SISTEM INFORMASI SURAT TUGAS BERBASIS AJAX PADA BAGIAN PENGADAAN BARANG/JASA KOTA DENPASAR."

[6] W. Sanjaya and D. Hermawan, "DIGITALISASI BUKU TAMU PEMERINTAHAN DESA BENGKALA DENGAN REAL-TIME DATA VIEW BERBASIS AJAX," *JSI : Jurnal Sistem Informasi (E-Journal)*, vol. 14, no. 2, 2022, [Online]. Available: http://ejournal.unsri.ac.id/index.php/jsi/index

[7] T. Avdeenko and K. Serdyukov, "Automated Test Data Generation Based on a Genetic Algorithm with Maximum Code Coverage and Population Diversity," *Applied Sciences*, vol. 11, no. 10, p. 4673, May 2021, doi: 10.3390/app11104673.

[8] T. Avdeenko and K. Serdyukov, "Automated Test Data Generation Based on a Genetic Algorithm with Maximum Code Coverage and Population Diversity," *Applied Sciences*, vol. 11, no. 10, p. 4673, May 2021, doi: 10.3390/app11104673.

[9] M. Brunetto, G. Denaro, L. Mariani, and M. Pezzè, "On introducing automatic test case generation in practice: A success story and lessons learned," *Journal of Systems and Software*, vol. 176, p. 110933, Jun. 2021, doi: 10.1016/j.jss.2021.110933.

[10] M. Goyal and Q. H. Mahmoud, "A Systematic Review of Synthetic Data Generation Techniques Using Generative AI," *Electronics (Basel)*, vol. 13, no. 17, p. 3509, Sep. 2024, doi: 10.3390/electronics13173509.

[11] M. Brunetto, G. Denaro, L. Mariani, and M. Pezzè, "On introducing automatic test case generation in practice: A success story and lessons learned," *Journal of Systems and Software*, vol. 176, p. 110933, Jun. 2021, doi: 10.1016/j.jss.2021.110933.

[12] M. Goyal and Q. H. Mahmoud, "A Systematic Review of Synthetic Data Generation Techniques Using Generative AI," *Electronics (Basel)*, vol. 13, no. 17, p. 3509, Sep. 2024, doi: 10.3390/electronics13173509.

[13] A. Jaffari, C.-J. Yoo, and J. Lee, "Automatic Test Data Generation Using the Activity Diagram and Search-Based Technique," *Applied Sciences*, vol. 10, no. 10, p. 3397, May 2020, doi: 10.3390/app10103397.

[14] N. S. Arida, I. A. Suryasih, and I. G. N. Parthama, "Model of Community Empowerment in Tourism Village Development Planning in Bali," *IOP Conf Ser Earth Environ Sci*, vol. 313, no. 1, p. 012024, Aug. 2019, doi: 10.1088/1755-1315/313/1/012024.