

**RANCANG BANGUN TOOLS COMMAND LINE INTERFACE UNTUK
OTOMATISASI FACTORY DAN SEEDER BERDASARKAN MIGRATION
PADA FRAMEWORK LARAVEL**

TUGAS AKHIR

Untuk memenuhi sebagian persyaratan

Mencapai derajat Sarjana S-1

Program Studi Informatika



Disusun Oleh:

Muhammad Azwar Firmansyah

22106050051

**PROGRAM STUDI INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA
YOGYAKARTA
2026**

LEMBAR PENGESAHAN



KEMENTERIAN AGAMA
UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Marsda Adisucipto Telp. (0274) 540971 Fax. (0274) 519739 Yogyakarta 55281

PENGESAHAN TUGAS AKHIR

Nomor : B-2123/Un.02/DST/PP.00.9/08/2026

Tugas Akhir dengan judul : Rancang Bangun Tools Command Line Interface Untuk Otomatisasi Factory dan Seeder Berdasarkan Migration Pada Framework Laravel

yang dipersiapkan dan disusun oleh:

Nama : MUHAMMAD AZWAR FIRMANSYAH
Nomor Induk Mahasiswa : 22106050051
Telah diujikan pada : Rabu, 19 Agustus 2026
Nilai ujian Tugas Akhir : A/B

dinyatakan telah diterima oleh Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta

TIM UJIAN TUGAS AKHIR



Ketua Sidang

Ir. Muhammad Didik Rohmad Wahyudi, S.T., MT.
SIGNED

Valid ID: 6a8bcecf6efac



Penguji I

Muhammad Mustakim, S.T. M.T.
SIGNED

Valid ID: 6a8bc90c6c13



Penguji II

Muhammad Galih Wonoseto, M.T.
SIGNED

Valid ID: 6a8bebcc51a99



Yogyakarta, 19 Agustus 2026
UIN Sunan Kalijaga

Dekan Fakultas Sains dan Teknologi

Prof. Dr. Dra. Hj. Khurul Wardati, M.Si.
SIGNED

Valid ID: 6a8bf1556d817

LEMBAR PERNYATAAN KEASLIAN

LEMBAR PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama : Muhammad Azwar Firmansyah
NIM : 22106050051
Program Studi : Informatika
Fakultas : Sains dan Teknologi

Menyatakan dengan sesungguhnya, bahwa skripsi saya yang berjudul “ **Rancang Bangun Tools Command Line Interface Untuk Otomatisasi Factory dan Seeder Berdasarkan Migration Pada Framework Laravel**” merupakan penelitian saya sendiri, tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di suatu perguruan tinggi dan bukan plagiasi karya orang lain kecuali secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka. Sebagai salah satu syarat untuk memperoleh gelar Sarjana Program Studi Informatika pada Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Kalijaga.

Yogyakarta, 13 Agustus 2026

Yang membuat pernyataan,



Muhammad Azwar Firmansyah
22106050051

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

LEMBAR PERSETUJUAN

LEMBAR PERSETUJUAN SKRIPSI/TUGAS AKHIR

Hal : Persetujuan Skripsi / Tugas Akhir
Lamp : -

Kepada
Yth.
Dekan Fakultas Sains dan Teknologi
UIN Sunan Kalijaga
DI Yogyakarta

Assalamu'alaikum wr. wb.

Setelah membaca, meneliti, memberikan petunjuk dan mengoreksi serta mengadakan perbaikan seperlunya, maka saya selaku pembimbing berpendapat bahwa skripsi Saudara:

Nama : Muhammad Azwar Firmansyah
NIM : 22106050051
Judul Skripsi : Rancang Bangun Tools Command Line Interface Untuk Otomatisasi Factory dan Seeder Berdasarkan Migration Pada Framework Laravel

Sudah dapat diajukan kepada Program Studi Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu dalam Program Studi Informatika.

Dengan ini kami berharap agar skripsi / tugas akhir Saudari dapat segera dimunaqsyahkan. Atas perhatiannya saya ucapkan terima kasih.

Wassalamu'alaikum wr. wb.

Yogyakarta, 13 Agustus 2026
Pembimbing,



Muhammad Didik Rohmad Wahyudi, S.T., MT.
NIP. 19760812 200901 1 015

LEMBAR PEDOMAN PENGGUNAAN TUGAS AKHIR

Tugas Akhir ini tidak dipublikasikan, tetapi tersedia di perpustakaan dalam lingkungan UIN Sunan Kalijaga Yogyakarta, diperkenankan dipakai sebagai referensi kepustakaan, tetapi pengutipan harus seizin penyusun, dan harus menyebutkan sumbernya sesuai dengan kebiasaan ilmiah. Dokumen Tugas Akhir ini merupakan hak milik UIN Sunan Kalijaga Yogyakarta.



ABSTRAK

Pengembangan aplikasi Laravel sering menghadapi perbedaan antara struktur basis data yang terbentuk dari Migration dan ketersediaan data uji pada lingkungan development atau testing yang baru. Migration membentuk skema, sedangkan Factory dan Seeder tetap harus disusun dengan menerjemahkan tipe kolom, constraint, dan relasi secara manual. Penelitian ini bertujuan merancang dan membangun Laraseed, package Command Line Interface (CLI) Laravel untuk mengotomatisasi pembentukan Factory dan Seeder berdasarkan file Migration. Pengembangan sistem menggunakan Rapid Application Development (RAD). Laraseed menerapkan Software Reverse Engineering melalui analisis Abstract Syntax Tree (AST) untuk mengekstraksi metadata deklarasi Schema::create, kemudian mengirimkan metadata tersebut secara batch ke Gemini API guna menghasilkan kode Factory dan Seeder. Package menyediakan perintah php artisan make:laraseed, pemilihan tabel, jumlah data, jenis artefak, proteksi file yang telah ada, opsi --force, serta pembaruan DatabaseSeeder sesuai dependensi foreign key. Black Box Testing menunjukkan seluruh dua belas skenario BB-01 sampai BB-12 berstatus lulus pada Laravel 12.41.1 dan SQLite. Validasi artefak dan integrasi membuktikan bahwa Factory dan Seeder untuk barangs, stock_ins, dan stock_outs sesuai dengan metadata yang terbaca serta dapat dijalankan setelah prasyarat HasFactory dipenuhi. Namun, perubahan kolom melalui Schema::table belum tergabung sehingga UserSeeder masih mengalami ketidaksesuaian. Laraseed dapat mengurangi pekerjaan repetitif dalam penyiapan data uji, tetapi hasil generasi tetap perlu diperiksa terhadap struktur akhir basis data.

Kata Kunci: Laravel, Command Line Interface, Migration, Factory dan Seeder, Abstract Syntax Tree.

ABSTRACT

Laravel application development often faces differences between the database structure produced by Migrations and the availability of test data in a newly created development or testing environment. Migrations build the schema, whereas Factories and Seeders still have to be written by manually translating column types, constraints, and relationships. This study aims to design and build Laraseed, a Laravel Command Line Interface (CLI) package that automates the generation of Factories and Seeders from Migration files. The system was developed using Rapid Application Development (RAD). Laraseed applies Software Reverse Engineering through Abstract Syntax Tree (AST) analysis to extract metadata from Schema::create declarations and sends the metadata in batches to the Gemini API to generate Factory and Seeder code. The package provides the php artisan make:laraseed command, table selection, record count, artifact type selection, existing-file protection, the --force option, and DatabaseSeeder updates according to foreign-key dependencies. Black-Box Testing showed that all twelve scenarios, BB-01 through BB-12, passed on Laravel 12.41.1 and SQLite. Artifact and integration validation demonstrated that the Factories and Seeders for barang, stock_ins, and stock_outs conformed to the extracted metadata and could be executed after the HasFactory prerequisite was fulfilled. However, column changes defined through Schema::table have not yet been merged, causing UserSeeder to remain inconsistent. Laraseed can reduce repetitive work in test-data preparation, but the generated output must still be checked against the final database structure.

Keywords: *Laravel, Command Line Interface, Migration, Factory and Seeder, Abstract Syntax Tree.*

MOTTO

“One day I am gonna grow wings.”

Let Down-Radiohead



HALAMAN PERSEMBAHAN

Alhamdulillah Rabbil 'Alamin, Segala puji dan syukur saya panjatkan ke hadirat Allah Swt. atas limpahan rahmat, pertolongan, dan kemudahan-Nya sehingga Tugas Akhir ini dapat diselesaikan. Semoga karya sederhana ini dapat memberikan manfaat serta menjadi amal kebaikan. Dengan penuh kerendahan hati dan rasa kasih yang tulus, Tugas Akhir ini saya persembahkan kepada:

1. Bapak dan Ibu tercinta, yang senantiasa memberikan kasih sayang, doa, dukungan, dan pengorbanan tanpa henti. Setiap kemudahan yang saya peroleh tentu tidak terlepas dari doa tulus Bapak dan Ibu yang selalu mengiringi setiap langkah saya.
2. Mas-mas saya tersayang, yang telah menjadi teladan, memberikan semangat, serta turut mendukung perjalanan saya hingga sampai pada tahap ini. Terima kasih atas perhatian, nasihat, dan dorongan yang telah diberikan. Semoga karya ini dapat menjadi salah satu bentuk kebanggaan bagi kalian.
3. Almamater tercinta, UIN Sunan Kalijaga Yogyakarta, tempat saya menimba ilmu, memperoleh pengalaman, serta belajar untuk terus tumbuh dan mengembangkan diri.

Terima kasih atas segala dukungan, doa, dan kebaikan yang telah diberikan. Semoga Allah Swt. membalas setiap kebaikan dengan limpahan rahmat, keberkahan, dan kebahagiaan.

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

KATA PENGANTAR

Alhamdulillah Rabbil 'Alamin, segala puji dan syukur penulis panjatkan ke hadirat Allah Swt. atas limpahan rahmat dan karunia-Nya sehingga Tugas Akhir yang berjudul “Rancang Bangun Tools Command Line Interface untuk Otomatisasi Factory dan Seeder Berdasarkan Migration pada Framework Laravel” dapat diselesaikan dengan baik. Selawat serta salam semoga senantiasa tercurah kepada junjungan kita, Nabi Muhammad saw., beserta keluarga, sahabat, dan seluruh umatnya.

Penulis menyadari bahwa penyelesaian Tugas Akhir ini tidak terlepas dari doa, bimbingan, bantuan, dan dukungan berbagai pihak. Oleh karena itu, dengan segenap kerendahan hati, penulis menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Prof. Noorhaidi Hasan S.Ag., M.A., M.Phil., Ph.D. selaku Rektor UIN Sunan Kalijaga Yogyakarta.
2. Prof. Dr. Dra. Hj. Khurul Wardati, M.Si. selaku Dekan Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta.
3. Bapak Dr. Muhammad Mustakim, S.T. M.T. selaku Ketua Program Studi Informatika Fakultas Sains dan Teknologi UIN Sunan Kalijaga Yogyakarta.
4. Bapak Muhammad Didik Rohmad Wahyudi, S.T., MT., selaku dosen penasihat akademik penulis selama kuliah di UIN Sunan Kalijaga.
5. Bapak Muhammad Didik Rohmad Wahyudi, S.T., MT. selaku dosen pembimbing Tugas Akhir yang telah memberikan waktu, arahan, koreksi, dan berbagai masukan berharga selama proses penyusunan penelitian ini. Setiap bimbingan yang diberikan membantu penulis untuk memahami kekurangan, memperbaiki naskah, dan menyelesaikan Tugas Akhir ini dengan lebih baik.
6. Kedua orang tua penulis, Bapak dan Ibu tercinta, yang senantiasa mengiringi setiap langkah penulis dengan kasih sayang, pengorbanan, kesabaran, serta doa yang tidak pernah terputus. Segala pencapaian hingga tahap ini tidak terlepas dari ketulusan dan dukungan yang selalu Bapak dan Ibu berikan.
7. Teman-teman DJ Fullbas Santuy, yang telah kebersamai penulis sejak SMP. Terima kasih atas canda, tawa yang menghibur, dan persahabatan yang tetap bertahan.

8. Teman-teman Matdis dan Musang, yang telah menjadikan masa perkuliahan penuh dengan diskusi, perdebatan, candaan, dan berbagai kejadian di luar rencana. Terima kasih telah menjadi tempat bertukar pikiran sekaligus tempat beristirahat dari pikiran.
9. Seseorang yang pernah kebersamai banyak bagian dalam perjalanan penulis selama masa perkuliahan. Terima kasih atas segala bantuan, waktu, perhatian, kesabaran, dan dukungan yang telah diberikan, terutama ketika penulis berada dalam masa-masa yang tidak mudah. Kehadiran dan penguatan yang pernah diberikan menjadi bagian penting yang membantu penulis untuk bertahan, bertumbuh, dan sampai pada tahap ini. Semoga setiap kebaikan tersebut mendapatkan balasan yang terbaik.
10. Teman-teman Neroform (Informatika angkatan 2022), yang telah berjalan bersama dengan cerita, tantangan, dan perjuangannya masing-masing. Semoga setiap langkah setelah perkuliahan membawa kita menuju jalan terbaik.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki kekurangan dan jauh dari kesempurnaan. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan sebagai bahan perbaikan. Penulis berharap Tugas Akhir ini dapat memberikan manfaat bagi pembaca serta menjadi sumbangan pengetahuan bagi pengembangan penelitian selanjutnya.

STATE ISLAMIC UNIVERSITY
SUNAN KALIJAGA
YOGYAKARTA

DAFTAR ISI

LEMBAR PENGESAHAN	i
LEMBAR PERNYATAAN	ii
LEMBAR PERSETUJUAN	iii
LEMBAR PEDOMAN PENGGUNAAN TUGAS AKHIR.....	iv
ABSTRAK	v
<i>ABSTRACT</i>	vi
MOTTO	vii
HALAMAN PERSEMBAHAN	viii
KATA PENGANTAR.....	ix
DAFTAR ISI	xi
DAFTAR TABEL	xiv
DAFTAR GAMBAR	xv
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	5
1.3 Batasan Masalah.....	6
1.4 Tujuan Penelitian.....	8
1.5 Manfaat Penelitian	9
1.5.1 Manfaat Secara Teoritis.....	9
1.5.2 Manfaat Secara Praktis.....	9
BAB II TINJAUAN PUSTAKA DAN LANDASAN TEORI.....	10
2.1 Tinjauan Pustaka	10
2.1.1 Analisis Statis dan Otomatisasi Test Data	10
2.1.2 Implementasi LLM dalam Code Generation.....	12
2.1.3 Studi Komparasi Tools Scaffolding	14
2.1.4 Perbandingan Penelitian Terkait.....	16
2.2 Landasan Teori	18
2.2.1 Framework Laravel (Migration, Factory, dan Seeder).....	18
2.2.2 Software Reverse Engineering (SRE).....	20
2.2.3 Static Analysis dan Abstract Syntax Tree (AST)	23
2.2.4 Large Language Model (LLM) dan Gemini API	25
2.2.5 Command Line Interface (CLI)	27
2.2.6 Metode Rapid Application Development (RAD).....	29
2.2.7 Pengujian Fungsional dan Validasi Keluaran.....	32
BAB III METODE PENGEMBANGAN SISTEM	33
3.1 Jenis dan Rancangan Penelitian	33
3.2 Lokasi dan Waktu.....	33
3.3 Objek, Bahan, dan Alat	33
3.3.1 Objek Penelitian	34
3.3.2 Bahan Penelitian.....	34

3.3.3 Perangkat Keras	35
3.3.4 Perangkat Lunak.....	35
3.4 Metode Pengembangan	36
3.4.1 Tahapan Rapid Application Development (RAD)	37
3.5 Prosedur Penelitian.....	39
3.6 Metode Pengujian.....	40
3.6.1 Black Box Testing	40
3.6.2 Validasi Artefak dan Integrasi	40
3.6.3 Evaluasi Internal Siklus User Design.....	41
3.7 Teknik Pengumpulan Data	41
3.8 Teknik Analisis Data	42
3.9 Indikator Keberhasilan dan Batas Interpretasi	42
BAB IV HASIL DAN PEMBAHASAN	45
4.1 Requirements Planning	45
4.1.1 Analisis Masalah	45
4.1.2 Kebutuhan Fungsional	46
4.1.3 Kebutuhan Non-Fungsional	47
4.2 User Design.....	48
4.2.1 Use Case Diagram.....	48
4.2.2 Flowchart Sistem.....	49
4.2.3 Arsitektur Sistem Laraseed	50
4.2.4 Perancangan Parser AST	51
4.2.5 Perancangan Prompt Engineering	52
4.2.6 Desain Antarmuka CLI	52
4.2.7 Siklus Evaluasi dan Penyempurnaan Prototipe.....	52
4.3 Construction	57
4.3.1 Implementasi Command Line Interface.....	58
4.3.2 Implementasi Parser AST.....	59
4.3.3 Implementasi Ekstraksi Metadata Migration	59
4.3.4 Implementasi Integrasi Gemini API.....	60
4.3.5 Implementasi Generator Factory dan Seeder	62
4.3.6 Implementasi Manajemen API Key	64
4.4 Cutover.....	65
4.4.1 Metode Pengujian.....	65
4.4.2 Skenario Pengujian.....	67
4.4.3 Hasil Pengujian Fungsional	69
4.4.4 Validasi Kesesuaian Output Factory dan Seeder.....	78
4.4.5 Penerimaan Internal, Publikasi, dan Dokumentasi	83
4.5 Pembahasan.....	84
4.5.1 Interpretasi Hasil Pengujian	84

4.5.2 Efisiensi Pengembangan	85
4.5.3 Kesesuaian Struktural Output Factory dan Seeder.....	85
4.5.4 Kelebihan dan Keterbatasan Sistem.....	86
BAB V PENUTUP.....	89
5.1 Kesimpulan	89
5.2 Saran.....	90
DAFTAR PUSTAKA.....	91



DAFTAR TABEL

Tabel 2.1 Ringkasan studi analisis statis dan otomatisasi data uji	11
Tabel 2.2 Ringkasan studi LLM dan code generation	13
Tabel 2.3 Perbandingan pendekatan scaffolding	15
Tabel 2.4 Perbandingan penelitian terkait	17
Tabel 3.1 Objek dan data penelitian	34
Tabel 3.2 Perangkat keras yang digunakan	35
Tabel 3.3 Perangkat lunak yang digunakan	36
Tabel 3.4 Penerapan tahapan RAD pada pengembangan Laraseed	38
Tabel 3.5 Prosedur dan keluaran penelitian	39
Tabel 3.6 Keterlacakan tujuan, indikator, bukti, dan batas interpretasi	43
Tabel 4.1 Kebutuhan fungsional Laraseed	46
Tabel 4.2 Kebutuhan non-fungsional Laraseed	47
Tabel 4.3 Lingkungan pengujian Laraseed	65
Tabel 4.4 Skenario Black Box Testing Laraseed	68
Tabel 4.5 Hasil Black Box Testing Laraseed	69
Tabel 4.6 Validasi keluaran Factory dan Seeder	78
Tabel 4.7 Kelebihan Laraseed	86
Tabel 4.8 Keterbatasan Laraseed	87



DAFTAR GAMBAR

Gambar 3.1 Tahapan Rapid Application Development (RAD) pada pengembangan Laraseed	37
Gambar 4.1 Use Case Diagram Laraseed	49
Gambar 4.2 Flowchart proses Laraseed	50
Gambar 4.3 Arsitektur sistem Laraseed	51
Gambar 4.4 Statistik commit awal yang masih mencatat direktori vendor	54
Gambar 4.5 Perubahan .gitignore pada penyempurnaan distribusi prototipe	55
Gambar 4.6 Perbaikan namespace pada MigrationNodeVisitor	56
Gambar 4.7 Penambahan dokumentasi README Laraseed	57
Gambar 4.8 Tampilan eksekusi perintah Laraseed	58
Gambar 4.9 Migration stock_ins sebagai masukan parser	59
Gambar 4.10 Struktur metadata hasil ekstraksi Migration	60
Gambar 4.11 Implementasi pembacaan API key dan proses generateBatch	60
Gambar 4.12 Implementasi penyusunan prompt batch berdasarkan metadata	61
Gambar 4.13 Implementasi pemrosesan respons Gemini	62
Gambar 4.14 Factory StockIn yang dihasilkan Laraseed	63
Gambar 4.15 Seeder StockIn yang dihasilkan Laraseed	63
Gambar 4.16 Pembaruan DatabaseSeeder oleh Laraseed	64
Gambar 4.17 Informasi lingkungan dan commit proyek uji	66
Gambar 4.18 Informasi package Laraseed dari Composer	67
Gambar 4.19 Hasil BB-01 ketika GEMINI_API_KEY tidak tersedia	70
Gambar 4.20 Hasil BB-02 pada kombinasi opsi yang bertentangan	70
Gambar 4.21 Hasil BB-03 dan BB-04: batching serta penyaringan target	71
Gambar 4.22 Pemetaan tipe data pada BarangFactory	72
Gambar 4.23 Pemetaan foreign key pada StockInFactory	72
Gambar 4.24 Pemetaan foreign key pada StockOutFactory	73
Gambar 4.25 Urutan Seeder pada DatabaseSeeder.php	74
Gambar 4.26 Hasil BB-08 pada Model tanpa HasFactory	74
Gambar 4.27 Penerapan trait HasFactory pada Model Barang	75
Gambar 4.28 Hasil BB-09 setelah Model menerapkan HasFactory	76
Gambar 4.29 Hasil BB-10 ketika file target sudah tersedia	76
Gambar 4.30 Hasil BB-11 menggunakan opsi --force	77
Gambar 4.31 Hasil BB-12 untuk satu tabel target	77
Gambar 4.32 Jumlah record kumulatif setelah rangkaian pengujian	79
Gambar 4.33 Data pada tabel barangs bagian pertama	80
Gambar 4.34 Data pada tabel barangs bagian kedua	81
Gambar 4.35 Data pada tabel stock_ins	81
Gambar 4.36 Data pada tabel stock_outs	82
Gambar 4.37 Error users.username setelah hasil --force dijalankan	83
Gambar A.1 Tampilan repository Laraseed pada GitHub	I
Gambar A.2 Tampilan package Laraseed pada Packagist	II

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pengembangan aplikasi web modern memanfaatkan framework untuk menyediakan struktur, komponen, dan konvensi yang mengurangi pekerjaan teknis berulang. Laravel merupakan framework aplikasi web berbasis PHP yang menerapkan pola Model-View-Controller (MVC) dan menyediakan komponen seperti Eloquent ORM, routing, validasi, Migration, Factory, Seeder, serta Artisan Command Line Interface (CLI) dalam satu ekosistem pengembangan [1]. Ketersediaan komponen tersebut mendukung pembangunan layanan web berbasis API yang memerlukan pengelolaan skema basis data secara terstruktur [2].

Pada tahap development, pengembang menerjemahkan kebutuhan sistem menjadi Model, logika aplikasi, dan rancangan basis data. Model merepresentasikan entitas yang dikelola aplikasi, sedangkan Migration mendefinisikan perubahan skema, seperti pembuatan tabel, kolom, indeks, dan foreign key, dalam bentuk kode yang dapat dilacak. Setelah file Migration dibuat, perintah `php artisan migrate` dijalankan untuk menerapkan perubahan yang belum dieksekusi pada basis data yang terhubung. Dengan mekanisme ini, perubahan struktur dapat dibagikan bersama source code dan diterapkan secara berurutan pada lingkungan lain [3]. Dokumentasi Laravel [4] secara khusus menempatkan Migration sebagai version control bagi definisi skema basis data.

Kesamaan source code dan file Migration tidak serta-merta membuat data pada lingkungan development identik dengan data pada lingkungan tujuan. Pada

penggunaan umum, Migration merekonstruksi atau mengubah struktur basis data; perintah tersebut tidak otomatis menyalin seluruh record yang telah terkumpul pada basis data lokal, kecuali pengembang secara khusus menambahkan mekanisme pemindahan atau pengisian data[5]. Basis data lokal dapat berisi record hasil input manual dan rangkaian pengujian, sedangkan basis data pada lingkungan development atau testing yang baru dapat masih kosong, memiliki konfigurasi koneksi berbeda, atau mempunyai riwayat eksekusi Migration yang berbeda. Penelitian [6] mengenai CI/CD pada aplikasi basis data juga menunjukkan bahwa perubahan aplikasi dan basis data memerlukan proses integrasi serta deployment yang terkelola karena keduanya memiliki kondisi dan prasyarat tersendiri.

Perbedaan tersebut perlu ditempatkan secara tepat. Pada lingkungan production, data operasional yang sah tidak boleh digantikan begitu saja oleh data tiruan. Namun, pada lingkungan development atau testing yang baru dibangun, ketiadaan data dapat menghambat pemeriksaan fitur, relasi, validasi, dan alur transaksi. Dokumentasi Laravel [7] menjelaskan bahwa Model Factory dapat digunakan untuk mendefinisikan atribut bawaan serta menghasilkan variasi nilai menggunakan Faker, [8] sementara Seeder mengatur proses pengisian basis data dan urutan pemanggilan kelas pengisi data. Factory dan Seeder dengan demikian berfungsi sebagai mekanisme untuk membentuk ulang data uji pada basis data yang telah memiliki struktur.

Kebutuhan inisialisasi data merupakan bagian penting dalam pengujian aplikasi yang berinteraksi dengan basis data. Berdasarkan kajian terhadap 72 proyek open-source, Gobert [9] menemukan bahwa pengujian kode manipulasi

basis data masih sering kurang mendapat perhatian; persoalan yang dihimpun dari pengembang banyak berkaitan dengan lingkungan pengujian, pengelolaan basis data, serta inisialisasi dan pembersihan data antar pengujian. Jauh sebelumnya, McMinn [10] juga mengidentifikasi bahwa pemilihan data uji secara manual merupakan pekerjaan yang mahal, sulit, dan memerlukan tenaga besar. Kedua temuan tersebut menunjukkan bahwa penyiapan data uji bukan sekadar pekerjaan tambahan, tetapi bagian dari kesiapan lingkungan pengujian.

Dalam Laravel, pembentukan data uji tetap menuntut pengembang untuk menerjemahkan metadata Migration ke dalam kode Factory dan Seeder. Nama kolom perlu dipetakan ke atribut Model, tipe data perlu disesuaikan dengan generator nilai, modifier seperti nullable dan unique perlu diperhatikan, sedangkan foreign key menuntut pembentukan record induk sebelum record yang bergantung. Primayani dan Sanjaya [11] menyoroti tantangan penyediaan data uji yang memadai dan konsisten untuk kebutuhan pengujian. Apabila pemetaan tersebut dilakukan secara manual pada banyak tabel, pekerjaan menjadi repetitif, membutuhkan ketelitian, dan berisiko menghasilkan ketidaksesuaian antara skema dengan artefak data uji.

Masalah sinkronisasi semakin terlihat ketika skema berkembang. Penambahan kolom wajib, perubahan tipe data, atau penambahan foreign key pada Migration perlu diikuti dengan penyesuaian Factory, Seeder, dan urutan eksekusinya. Jika penyesuaian terlewat, kode pengisi data dapat gagal dijalankan atau menghasilkan record yang tidak mengikuti struktur kasus uji. Oleh karena itu, masalah yang diteliti bukan kecepatan eksekusi Seeder dan bukan pula jaminan

bahwa semua nilai yang dihasilkan bersifat realistis, melainkan beban prosedural untuk membentuk serta memelihara artefak Factory dan Seeder agar tetap selaras dengan metadata skema yang didukung.

Penelitian terdahulu memberikan dasar bagi otomatisasi tersebut, tetapi umumnya membahas komponennya secara terpisah. Primayani dan Sanjaya [11] membandingkan mekanisme penyediaan data uji menggunakan Seeder dan Faker. Habiby [12] menunjukkan pemanfaatan Abstract Syntax Tree (AST) untuk membaca struktur kode PHP tanpa mengeksekusinya. Nasir et al. [13] mengeksplorasi integrasi analisis kode statis dengan Large Language Model (LLM) dalam aktivitas penulisan kode. Dohmen et al. [14] menegaskan pentingnya metadata skema relasional, seperti tipe data dan constraint, sebagai konteks bagi sistem berbasis AI. Temuan tersebut mendukung penggunaan AST untuk mengekstraksi struktur yang deterministik dan LLM untuk membantu menyusun kode, tetapi belum dengan sendirinya membentuk alur otomatisasi Factory dan Seeder pada Laravel.

Pada tingkat tools, Blueprint [15] mendokumentasikan Laravel Blueprint sebagai tools yang dapat menghasilkan sejumlah komponen Laravel, termasuk Model, Migration, dan Factory, dari draft berformat YAML. Keberadaan Blueprint menunjukkan bahwa scaffolding komponen Laravel bukan hal yang sepenuhnya baru. Akan tetapi, sumber definisi dan fokus prosesnya berbeda: Blueprint menggunakan draft YAML sebagai masukan deklaratif, sedangkan penelitian ini memanfaatkan file Migration yang telah tersedia pada proyek sebagai sumber

utama metadata skema. Karena itu, kebaruan Laraseed tidak dinyatakan sebagai pengganti Blueprint atau generator aplikasi lengkap.

Berdasarkan penelitian dan alat yang ditinjau, gap penelitian didefinisikan sebagai belum ditemukannya alur khusus yang mengintegrasikan ekstraksi deklarasi `Schema::create` berbasis AST, penyusunan metadata kolom dan dependensi foreign key sebagai konteks LLM, serta penulisan Factory, Seeder, dan pemanggilan pada `DatabaseSeeder.php` melalui satu command Artisan tanpa file pemetaan skema khusus Laraseed. Pernyataan tersebut dibatasi pada sumber yang ditinjau, bukan klaim bahwa tidak pernah ada solusi lain. Laraseed juga tetap memiliki prasyarat berupa Model Eloquent dan trait `HasFactory`, serta belum menggabungkan perubahan skema yang dideklarasikan melalui `Schema::table`.

Penelitian ini karena itu berfokus pada rancang bangun dan validasi teknis Laraseed sebagai package CLI Laravel. Kontribusi teknisnya meliputi parser Migration berbasis AST, pembentukan metadata antar tabel, integrasi Gemini API untuk menyusun definisi Factory, pembuatan Seeder, proteksi file, dan pengaturan urutan pemanggilan Seeder. Keberhasilan sistem dinilai melalui Black Box Testing, pemeriksaan kesesuaian artefak terhadap Migration, eksekusi Seeder, dan pemeriksaan record pada proyek uji.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, penelitian difokuskan pada otomatisasi pembentukan data uji untuk lingkungan development dan testing melalui pemanfaatan metadata Migration. Rumusan masalah disusun agar setiap aspek perancangan, implementasi, dan evaluasi memiliki bukti yang dapat ditelusuri.

Adapun rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana merancang dan mengimplementasikan analisis statis berbasis Abstract Syntax Tree (AST) untuk mengekstraksi nama tabel, kolom, tipe data, modifier, dan foreign key dari deklarasi Schema::create pada Migration Laravel?
2. Bagaimana mengintegrasikan metadata hasil ekstraksi dengan Gemini API untuk menghasilkan kode Factory dan Seeder yang sesuai dengan nama, tipe, serta relasi kolom pada kasus uji?
3. Bagaimana membangun package Command Line Interface (CLI) yang mengotomatisasi pemilihan target, pembuatan Factory dan Seeder, proteksi file, serta pengaturan urutan pemanggilan Seeder pada DatabaseSeeder.php tanpa file pemetaan skema khusus Laraseed?
4. Bagaimana keberhasilan fungsional Laraseed serta kesesuaian artefak Factory dan Seeder yang dihasilkan berdasarkan Black Box Testing dan validasi artefak serta integrasi?

1.3 Batasan Masalah

Agar hasil penelitian dapat dinilai sesuai tujuan dan bukti yang tersedia, ruang lingkup penelitian dibatasi sebagai berikut:

1. Konteks penggunaan: Laraseed ditujukan untuk membantu penyiapan data uji pada lingkungan development atau testing. Sistem tidak dirancang untuk menyalin data operasional antar lingkungan dan tidak digunakan untuk menggantikan data production dengan data tiruan.

2. Lingkup framework: dependensi package dirancang untuk Laravel 10 sampai dengan 12, sedangkan bukti empiris pada penelitian ini diperoleh dari Laravel 12.41.1.
3. Sumber metadata skema: Laraseed membaca file PHP pada database/migrations dan mengekstraksi deklarasi Schema::create. Perubahan lanjutan melalui Schema::table belum digabungkan, sehingga Migration disebut sumber utama metadata, bukan representasi lengkap keadaan akhir basis data.
4. Teknik ekstraksi: metadata diperoleh melalui analisis statis berbasis AST tanpa menjalankan Migration dan tanpa melakukan introspeksi langsung ke basis data aktif.
5. Layanan generatif: code generation menggunakan Google Gemini API. Respons LLM diperlakukan sebagai kandidat kode yang tetap harus diperiksa dan dieksekusi sebelum digunakan.
6. Prasyarat proyek: pengguna menyediakan GEMINI_API_KEY pada .env, dependensi Composer, Model Eloquent untuk tabel target, dan trait HasFactory agar Seeder berbasis Model::factory() dapat dijalankan.
7. Artefak dan antarmuka: Laraseed menghasilkan file Factory dan Seeder, memperbarui pemanggilan Seeder pada DatabaseSeeder.php, dan dijalankan melalui command php artisan make:laraseed beserta opsi yang tersedia. Laraseed tidak menghasilkan Model, Controller, View, atau Resource.

8. Cakupan relasi dan evaluasi: pengurutan Seeder dibatasi pada foreign key yang terdefinisi eksplisit. Pengujian dilakukan pada satu proyek open-source, Laravel 12.41.1, SQLite, dan satu rangkaian keluaran. Hasil evaluasi hanya mendukung fungsi serta kesesuaian artefak pada kasus yang diuji dan tidak digunakan untuk menggeneralisasikan kompatibilitas seluruh proyek Laravel.

1.4 Tujuan Penelitian

Tujuan penelitian disusun selaras dengan rumusan masalah, batasan sistem, dan bukti evaluasi yang dikumpulkan.

Tujuan penelitian ini adalah sebagai berikut:

1. Merancang dan mengimplementasikan parser AST untuk mengekstraksi metadata deklarasi Schema::create pada Migration Laravel.
2. Mengintegrasikan metadata hasil ekstraksi dengan Gemini API untuk menghasilkan Factory dan Seeder yang dapat divalidasi terhadap nama, tipe, dan relasi kolom pada proyek uji.
3. Membangun package CLI yang menyediakan pemilihan target dan jumlah data, mode Factory atau Seeder, proteksi file, overwrite, serta pengaturan urutan Seeder pada DatabaseSeeder.php.
4. Mengevaluasi keberhasilan fungsi Laraseed serta kesesuaian artefak Factory dan Seeder yang dihasilkan melalui Black Box Testing dan validasi artefak serta integrasi.

1.5 Manfaat Penelitian

Hasil penelitian diharapkan memberikan manfaat teoritis dan praktis sesuai ruang lingkup serta batas kemampuan sistem.

1.5.1 Manfaat Secara Teoritis

Secara teoritis, penelitian ini memberikan contoh penerapan analisis statis berbasis Abstract Syntax Tree (AST) untuk mengekstraksi metadata skema sebagai konteks bagi LLM dalam code generation. Kontribusinya berada pada perancangan pipeline gabungan antara ekstraksi deterministik dan generasi kode probabilistik, serta pemisahan antara validasi fungsi command dan validasi kesesuaian artefak serta integrasi. Manfaat tersebut tidak digeneralisasi untuk seluruh pola Migration di luar cakupan pengujian.

1.5.2 Manfaat Secara Praktis

Secara praktis, Laraseed menyediakan alternatif untuk mengurangi langkah penulisan awal Factory dan Seeder pada proyek Laravel yang memenuhi prasyarat. Pengembang memperoleh satu command untuk memilih target, menghasilkan artefak, mengatur urutan Seeder, dan melindungi file yang telah tersedia. Hasil generate tetap memerlukan review serta pengujian integrasi, terutama ketika skema menggunakan pola yang belum didukung seperti Schema::table.

BAB V

PENUTUP

5.1 Kesimpulan

1. Penerapan Software Reverse Engineering melalui AST berhasil mengekstraksi metadata deklarasi Schema::create dan memetakan dependensi tabel tanpa melakukan koneksi langsung untuk introspeksi skema. Implementasi saat ini belum menggabungkan perubahan Schema::table.
2. Integrasi Gemini API melalui prompt berbasis metadata dan Batching Mode berhasil menghasilkan Factory dan Seeder untuk barangs, stock_ins, dan stock_outs. Validasi menunjukkan pemetaan teks, angka, foreign key, serta urutan Seeder sesuai struktur kasus uji.
3. Package CLI dapat dijalankan melalui php artisan make:laraseed dengan kontrol target, jumlah data, jenis artefak, skip existing file, overwrite, dan pengaturan urutan pemanggilan Seeder pada DatabaseSeeder.php. Model Eloquent dan trait HasFactory tetap menjadi prasyarat untuk menjalankan Seeder berbasis Model::factory().
4. Berdasarkan Black Box Testing, seluruh BB-01 sampai BB-12 berstatus Lulus pada Laravel 12.41.1 dan SQLite. Validasi artefak dan integrasi menunjukkan bahwa Factory serta Seeder untuk barangs, stock_ins, dan stock_outs sesuai dengan metadata yang terbaca dan dapat dijalankan setelah prasyarat HasFactory dipenuhi. Namun, UserSeeder belum berhasil pada perubahan kolom melalui Schema::table, sehingga hasil generate tetap harus diperiksa terhadap struktur akhir basis data.

5.2 Saran

1. Mengembangkan parser agar menggabungkan metadata `Schema::create` dengan perubahan `Schema::table` sesuai urutan Migration.
2. Menambahkan preflight check untuk keberadaan Model dan trait `HasFactory`, disertai pesan perbaikan yang spesifik sebelum Seeder dijalankan.
3. Membedakan tabel sistem melalui aturan eksplisit agar tabel domain tanpa Model tidak langsung dikategorikan sebagai Tabel Sistem.
4. Menambahkan validasi deterministik terhadap respons LLM, termasuk pemeriksaan kolom wajib, sintaks PHP, referensi Model, dan kesesuaian constraint.
5. Memperluas pengujian ke Laravel 10, 11, dan 12, beberapa driver basis data, serta beberapa proyek dengan pola Migration dan relasi yang lebih beragam.
6. Mengukur durasi generate, penggunaan token/API, tingkat keberhasilan output, dan waktu pembuatan manual sebagai pembanding kuantitatif.
7. Menerbitkan tag versi stabil serta menambahkan pengujian otomatis dan continuous integration untuk menjaga kompatibilitas package.

DAFTAR PUSTAKA

- [1] F. Sinlae, E. Irwanda, Z. Maulana, dan V. E. Syahputra, “Penggunaan Framework Laravel dalam Membangun Aplikasi Website Berbasis PHP”, doi: 10.38035/jsmd.v2i2.
- [2] F. P. E. Putra, R. W. Efendi, A. B. Tamam, dan W. A. Pramadi, “Tren dan Praktik Terbaik dalam Pengembangan Web Berbasis API : Kajian Literatur terhadap Framework Laravel dan React,” *Infomatek*, vol. 27, no. 1, hlm. 165–178, Jun 2025, doi: 10.23969/infomatek.v27i1.25122.
- [3] W. Setiawan Usman *dkk.*, “Implementasi framework Laravel dalam pengembangan website layanan administratif program studi,” *Jambura Journal of Informatics*, vol. 6, no. 2, hlm. 135–146, 2024, doi: 10.37905/jji.v4i2.27803.
- [4] Laravel, “Artisan Console | Laravel 13.x - The clean stack for Artisans and agents,” <https://laravel.com/docs/13.x/artisan>.
- [5] “Database: Migrations | Laravel 12.x - The clean stack for Artisans and agents,” https://laravel.com/docs/12.x/migrations?utm_source=chatgpt.com.
- [6] J. Fluri, F. Fornari, dan E. Pustulka, “On the importance of CI/CD practices for database applications,” *Journal of Software: Evolution and Process*, vol. 36, no. 12, Des 2024, doi: 10.1002/smr.2720.
- [7] “Eloquent: Factories | Laravel 12.x - The clean stack for Artisans and agents,” https://laravel.com/docs/12.x/eloquent-factories?utm_source=chatgpt.com.
- [8] “Database: Seeding | Laravel 12.x - The clean stack for Artisans and agents,” https://laravel.com/docs/12.x/seeding?utm_source=chatgpt.com.
- [9] M. Gobert, C. Nagy, H. Rocha, S. Demeyer, dan A. Cleve, “Best practices of testing database manipulation code,” *Inf. Syst.*, vol. 111, Jan 2023, doi: 10.1016/j.is.2022.102105.
- [10] P. McMinn, “Search-based software test data generation: A survey,” *Software Testing Verification and Reliability*, vol. 14, no. 2, hlm. 105–156, Jun 2004, doi: 10.1002/stvr.294.
- [11] S. A. Primayani dan I. G. W. Sanjaya, “Perbandingan Laravel Seeder dan Faker Untuk Otomatisasi Data Uji Pada Aplikasi Perpustakaan,” *Journal*

- Informatics Nivedita*, vol. 2, no. 1, hlm. 30–39, Des 2025, doi: 10.25078/nivedita.v2i1.5615.
- [12] M. E. Habiby, “Deteksi Kerentanan Kode PHP Menggunakan TF-IDF, AST Parsing, dan Random Forest,” *INTERNAL (Information System Journal)*, vol. 8, no. 1, hlm. 1–9, doi: 10.32627.
- [13] A. Nasir, K. Syariati, C. Suardi, D. Sundoro, J. S. Biantong, dan R. L. Lordianto, “Implementation of Static Code Analysis to Detect Vulnerabilities in Applications Developed with the Assistance of Large-Language Models (LLM),” *ENERGY: JURNAL ILMIAH ILMU-ILMU TEKNIK*, vol. 15, no. 2, hlm. 239–250, Nov 2025, doi: 10.51747/energy.v15i2.15210.
- [14] T. Döhmen, R. Geacu, M. Hulsebos, dan S. Schelter, “SchemaPile: A Large Collection of Relational Database Schemas,” *Ann. Entomol. Soc. Am.*, vol. 2, no. 3, Mei 2024, doi: 10.1145/3654975.
- [15] Blueprint, “Blueprint | Generating Components,” https://blueprint.laravelshift.com/docs/generating-components/?utm_source=chatgpt.com.
- [16] J. Jiang, F. Wang, J. Shen, S. Kim, dan S. Kim, “A Survey on Large Language Models for Code Generation,” Nov 2024, doi: 10.1145/3747588.
- [17] L. I. Rahman dan M. Z. C. Candra, “Pengembangan Bahasa Pemodelan dan Code Generation untuk Smart Contract,” *Jurnal Sistem dan Teknologi Informasi (JustIN)*, vol. 11, no. 2, hlm. 250, Jul 2023, doi: 10.26418/justin.v11i2.55077.
- [18] D. Tosi, “Studying the Quality of Source Code Generated by Different AI Generative Engines: An Empirical Evaluation,” *Future Internet*, vol. 16, no. 6, Jun 2024, doi: 10.3390/fi16060188.
- [19] I. A. Alfarisi, A. T. Priandika, dan A. S. Puspaningrum, “Penerapan Framework Laravel Pada Sistem Pelayanan Kesehatan (Studi Kasus: Klinik Berkah Medical Center),” *Jurnal Ilmiah Computer Science*, vol. 2, no. 1, hlm. 1–9, Jul 2023, doi: 10.58602/jics.v2i1.11.
- [20] M. Kusbianto *dkk.*, “Rekonstruksi Diagram Kelas UML Moodle Menggunakan Analisis Kode Statis,” 2026, doi: 10.55382/jurnalpustakadata.v6i2.1809.
- [21] nikic, “GitHub - nikic/PHP-Parser: A PHP parser written in PHP,” https://github.com/nikic/PHP-Parser?utm_source=chatgpt.com.

- [22] B. Firmanto, A. S. Aziz, dan J. Sesoca, “Tinjauan Perkembangan Kecerdasan Buatan Berbasis Arsitektur Transformer,” dalam *Advances in Neural Information Processing Systems*, Neural information processing systems foundation, 2020.
- [23] Gemini Team *dkk.*, “Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context,” Des 2024, [Daring]. Tersedia pada: <http://arxiv.org/abs/2403.05530>
- [24] M. Agarwal *dkk.*, “Project CLAI: Instrumenting the Command Line as a New Environment for AI Agents,” Jan 2020.
- [25] Laravel, “Package Development | Laravel 11.x - The clean stack for Artisans and agents,” <https://laravel.com/docs/11.x/packages>.
- [26] S. Mulyati, A. Herdiansah, R. Taufiq, D. Y. Prianggodo, dan S. Bukhori, “IMPLEMENTASI RAPID APPLICATION DEVELOPMENT (RAD) STUDI KASUS PENGEMBANGAN SISTEM INFORMASI SEKOLAH YAYASAN AL ABANIYAH,” *JIKA (Jurnal Informatika)*, vol. 8, no. 2, hlm. 156, Apr 2024, doi: 10.31000/jika.v8i2.10268.
- [27] E. Shandra Dewi, E. Ardy Mesia Putri, J. Tji Beng, dan F. Teknologi Informasi, “PERBANDINGAN ANTARA METODE WATERFALL DAN METODE RAD DALAM PEMBUATAN APLIKASI E-REKRUTMEN BERBASIS WEBSITE: STUDI KASUS PT XYZ COMPARISON BETWEEN THE WATERFALL METHOD AND THE RAD METHOD IN CREATING WEBSITE-BASED E-RECRUITMENT APPLICATIONS: A CASE STUDY OF PT XYZ,” *Journal of Information Technology and Computer Science (INTECOMS)*, vol. 7, no. 4, 2024.
- [28] pwg dan istqborg, “Certified Tester Foundation Level Syllabus v4.0.1 International Software Testing Qualifications Board,” 2024.
- [29] Y. Zhang *dkk.*, “Learning-based Widget Matching for Migrating GUI Test Cases,” dalam *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, New York, NY, USA: ACM, Feb 2024, hlm. 1–13. doi: 10.1145/3597503.3623322.