e-ISSN: 2460-870X | p-ISSN: 2085-1367

Evaluasi Performa Otomatisasi Skema Basis Data Dengan Model dan Migrasi Django Dalam Aplikasi Proyek Akhir

Optimizing Database Schema Automation with Django Models and Migrations in Final Project Applications

Gat[a,1], Wahyu Sindu Prasetya[b,2], Vellen Wibowo[c,3]
[a,b,c]STMIK Pontianak, Jl. Merdeka Barat No. 372, Pontianak, 78118, Indonesia
gat@stmikpontianak.ac.id[1], wahyusinduprasetya@stmikpontianak.ac.id[2], vellen.wibowo@stmikpontianak.ac.id[3]

ABSTRAK

Menjaga integritas, kinerja, dan skalabilitas aplikasi akademik memerlukan pengelolaan skema basis data yang baik. Otomatisasi skema berbasis rangka kerja masih jarang diuji secara menyeluruh untuk sistem yang memiliki entitas kompleks dan data besar. Tujuan dari penelitian ini adalah untuk mengevaluasi kemampuan model dan migrasi Django untuk mengotomatisasi skema basis data pada aplikasi manajemen proyek akhir yang terdiri dari enam entitas utama: Mahasiswa, Dosen, Proyek Akhir, Bimbingan, Evaluasi, dan Pemeriksa. Metode studi kasus digunakan dengan dua dataset yang saling terkait dengan 1.000 dan 10.000 entri masing-masing. Penelitian ini menggunakan pendekatan deskriptif-analitis dengan fase implementasi, pengujian, dan analisis. Pengujian melibatkan unit, fungsional, integrasi, dan kinerja menggunakan Django 5.1.2, MariaDB, dan stopwatch digital. Waktu membaca dataset dengan 1.000 entri adalah 0,00010 detik, waktu mengubah 0,00439 detik, dan waktu menghapus 0,00124 detik. Hasil menunjukkan bahwa model tetap konsisten, migrasi berjalan lancar, dan semua operasi CRUD dapat diselesaikan dengan waktu efektif rata-rata. Untuk dataset dengan 10.000 entri, waktu rata-rata adalah 0,00045 detik untuk operasi, 0,00013 detik untuk membaca, 0,04535 detik untuk mengubah, dan 0,00345 detik untuk menghapus. Singkatnya, Django dapat diterapkan pada aplikasi akademik berskala besar karena terbukti dapat mengotomatisasi skema basis data kompleks secara konsisten dan efisien.

Kata Kunci : Model Django, Migrasi, Pengujian, Object-Relational Mappinge

ABSTRACT

Effective management of database schemas is essential to ensure the scalability, performance, and integrity of academic applications. However, for systems with complex entities and large volumes of data, the use of framework-based schema automation remains relatively untested. Examiner, Guidance, Evaluation, Final Project, Lecturer, and Student are the six main entities that make up the final project management application, and the purpose of this study is to assess how well Django's models and migration tools can automate database schemas for this application. The case study methodology was used on two linked datasets with 1,000 and 10,000 entries, respectively. During the stages of installation, testing, and analysis, a descriptive-analytical approach was employed. Unit, functional, integration, and performance tests were conducted using MariaDB, Django 5.1.2, and a digital stopwatch. For the dataset with 1,000 entries, the read operation averaged 0.00010 seconds, the update operation 0.00439 seconds, and the delete operation 0.00124 seconds. The results demonstrate that the models remain consistent, migrations proceed smoothly, and all CRUD operations are completed with an effective average time. For the dataset containing 10,000 entries, the average operation times were 0.00045 seconds per operation, 0.00013 seconds for reads, 0.04535 seconds for updates, and 0.00345 seconds for deletions. In summary, Django can be effectively applied to large-scale academic applications, as it consistently and efficiently automates complex database schemas.

Keywords : Django Models, Migrations, Testing, Object-Relational Mappinge

Info Artikel :
Disubmit: *Direview:* *Diterima :*

DOI: https://www.doi.org/10.22303/csrid

1. PENDAHULUAN

Pengelolaan skema basis data sangat penting karena pengembangan aplikasi web memengaruhi skalabilitas, integritas, dan kinerja sistem [1][2]. Beberapa strategi untuk otomatisasi administrasi skema telah diusulkan untuk meningkatkan konsistensi, mengurangi kesalahan manual, dan mempercepat proses pengembangan [3]. Metode ini memungkinkan tim pengembang untuk bekerja lebih keras dan berkolaborasi dengan lebih baik. Namun, dalam praktiknya, menerapkan otomatisasi seringkali menghadapi masalah ketika diterapkan pada sistem yang memiliki hubungan entitas yang kompleks dan volume data yang besar, seperti aplikasi akademik yang terintegrasi. Studi sebelumnya hanya menguji ide otomatisasi dalam bentuk sederhana dan dalam skala uji coba kecil [4][5]. Akibatnya, tidak memenuhi kebutuhan sistem akademik yang memerlukan banyak entitas, integritas hubungan yang ketat, dan skala data yang besar.

Django memiliki banyak fitur praktis dan menjadikannya salah satu framework web berbasis Python yang paling disukai pengembang [6]. Salah satu fiturnya adalah Object-Relational Mapping (ORM), dan ada juga sistem migrasi bawaan yang sangat membantu pengelolaan basis data [7][8]. Analisis data dapat dilakukan dengan ORM hanya dengan menuliskan kode Python, tanpa perlu menulis SQL secara manual. Perintah SQL akan dihasilkan secara otomatis dari kode tersebut. Sementara itu, sistem migrasi Django menyederhanakan interaksi basis data sehingga memberikan kemudahan dalam pengelolaan perubahan, seperti mengecek atau mengubah nama kolom, menyesuaikan tipe data, hingga memvalidasi nama tabel, dengan cara yang lebih cepat, konsisten, dan minim kesalahan [9]-[11]. Selain itu, Django memiliki TestCase, yang meningkatkan pengujian integritas model dan validitas data tanpa mengorbankan data produksi [12][13]. Namun, tidak banyak penelitian yang secara sistematis mengevaluasi kinerja Django ORM dan migrasi ketika diterapkan pada aplikasi akademis dengan struktur data yang kompleks. Oleh karena itu masih perlu untuk mengevaluasi efektivitas klaim Django dalam konteks ini.

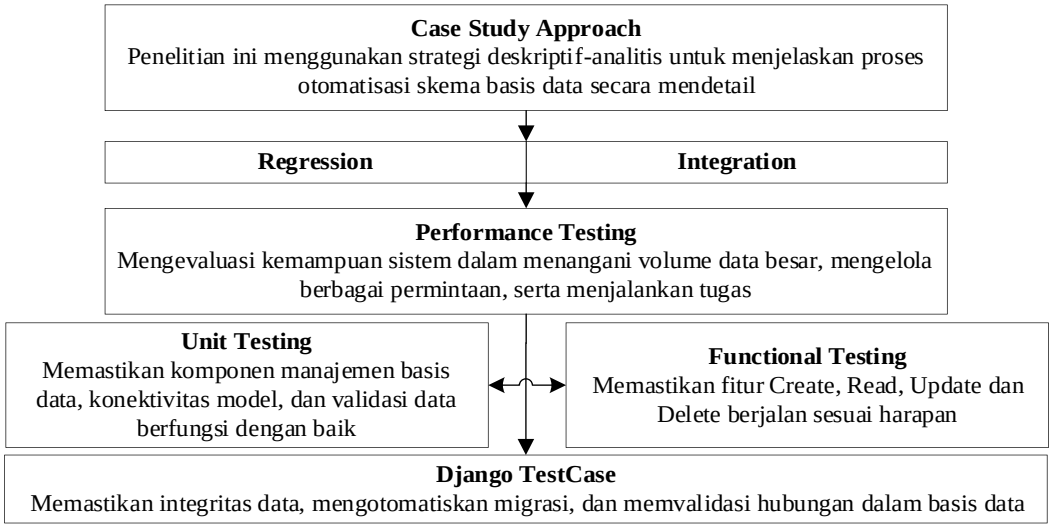
Penelitian ini difokuskan pada pertanyaan utama, yaitu sejauh mana model dan fitur migrasi pada Django mampu mengotomatisasi skema basis data, menjaga konsistensi hubungan antarentitas, serta meminimalkan kesalahan manual dalam aplikasi akademik berskala besar? Studi ini mengeksplorasi cara kerja otomatisasi skema basis data berbasis Django pada model akademik yang mencakup enam entitas penting, yaitu Students, Lecturers, FinalProjects, Guidance, Evaluation, dan Examiners. Penelitian ini dilakukan untuk menjawab kekurangan yang masih ditemukan pada studi-studi sebelumnya. Temuan yang ada dinilai belum cukup luas dan mendalam. Karena itu, evaluasi dalam penelitian ini difokuskan pada beberapa hal penting. Pertama, konsistensi relasi data saat sistem dijalankan. Kedua, sejauh mana efisiensi operasi CRUD dapat dipertahankan dalam kondisi nyata. Terakhir, bagaimana kinerja migrasi ketika sistem perlu memproses jumlah data yang sangat besar? Hasil yang diharapkan akan memberikan pemahaman yang lebih baik tentang kemampuan Django untuk mengelola skema basis data secara otomatis pada sistem perangkat lunak akademik yang kompleks.

2. METODE

Penelitian ini menggunakan pendekatan studi kasus pada pengembangan aplikasi manajemen proyek akhir sebagai representasi sistem akademik dengan banyak entitas dan relasi kompleks. Pemilihan studi kasus ini didasarkan pada kebutuhan nyata institusi akademik untuk mengelola data Students, Lecturers, FinalProjects, Guidance, Evaluation, dan Examiners secara terintegrasi dengan beban data besar. Pendekatan studi kasus dinilai tepat untuk mengevaluasi penerapan Django Model dan Migrasi dalam konteks nyata dengan kompleksitas yang relevan. Studi ini bersifat deskriptif-analitis dan berkonsentrasi pada implementasi, pengujian, dan analisis kinerja skema otomatisasi basis data. Untuk menjelaskan implementasi model Django dan migrasi, diagram alur proses digunakan untuk menunjukkan langkah-langkah yang harus dilakukan, mulai dari perancangan model (models.py), pembuatan file migrasi (makemigrations), penerapan migrasi (migrate), dan pengujian performa dan integritas. Dataset yang

e-ISSN: 2460-870X | p-ISSN: 2085-1367

digunakan terdiri dari 1.000 entri dan 10.000 entri, masing-masing berisi data terkait enam entitas utama: Students, Lecturers, FinalProjects, Guidance, Evaluation, dan Examiners. Data disusun secara bersamaan untuk menguji integritas hubungan antar model. Pengujian dilakukan dalam empat jenis, yaitu pengujian unit menggunakan Django TestCase untuk memverifikasi fungsi model secara individual, seperti pembuatan skema, konektivitas antar entitas, validasi batas, dan integritas data. Pengujian fungsional memastikan bahwa semua operasi utama sistem (CRUD) berjalan sesuai harapan pengguna. Pengujian integrasi, untuk memastikan bahwa seluruh komponen model berfungsi dengan benar saat digabungkan dalam satu sistem utuh. Pengujian kinerja, untuk mengukur waktu eksekusi rata-rata operasi CRUD pada kedua dataset sebagai parameter efisiensi skema. Alat yang digunakan termasuk framework pengujian Django (versi 5.1.2), basis data MariaDB, dan stopwatch digital pada level kode yang dapat merekam waktu eksekusi dalam detik. Untuk mengetahui seberapa efektif otomatisasi skema basis data menggunakan Model Django dan Migrasi untuk aplikasi akademik berskala besar, hasil pengujian dievaluasi secara kuantitatif. Kriteria evaluasi termasuk tingkat keberhasilan validasi model, integritas relasi data, keberhasilan migrasi tanpa error, dan waktu rata-rata eksekusi CRUD. Berikut ini adalah gambar 1 blok diagram proses penelitian.



Gambar 1. Blok Diagram Proses Penelitian

3. HASIL DAN PEMBAHASAN

Studi ini melihat otomatisasi skema basis data pada aplikasi manajemen proyek akhir yang menggunakan Model Django dan Migrasi. Implementasi dilakukan pada aplikasi web berbasis Django yang menggunakan MariaDB sebagai DBMS. Enam model utama, Student, Lecturer, FinalProject, Guidance, Evaluation, dan Examiner, dibuat menggunakan Django ORM dan berfungsi sebagai entitas penting dalam manajemen proyek akhir mahasiswa. Dengan sistem migrasi Django, aplikasi dapat secara otomatis membuat, mengubah, dan menyinkronkan skema basis data. Hasilnya, proses pengembangan, pemeliharaan, dan perluasan sistem bisa dilakukan dengan lebih efisien.

A. Persiapan Django Environment

Untuk memastikan pengembangan berjalan lancar, pengembang harus mempersiapkan lingkungan kerja Django. Ini mencakup konfigurasi proyek, memilih DBMS yang tepat (MariaDB), dan penyusunan struktur proyek dengan cara yang sistematis. Entitas seperti OneToOneField, ForeignKey, dan ManyToManyField dibuat sejak awal untuk memastikan fleksibilitas sistem, kemurnian skema, dan keterbacaan kode saat menangani hubungan kompleks di kemudian hari. Persiapan yang matang ini

menunjukkan peningkatan efisiensi dalam kolaborasi tim, mempercepat proses migrasi, dan mengurangi kemungkinan kesalahan sintaksis.

B. Defining Models

Pendefinisian model dalam Django menjadi inti otomatisasi skema karena model bertindak sebagai cetak biru bagi tabel-tabel basis data. Dalam sistem ini, masing-masing model merepresentasikan tahapan proses akademik secara spesifik, yaitu Student untuk menyimpan identitas mahasiswa dan status akademik; Lecturer untuk dosen pembimbing dan penguji; FinalProject untuk rincian proyek akhir; Guidance untuk log bimbingan; Evaluation untuk hasil penilaian; dan Examiner untuk tim penguji. Keenam model ini terhubung melalui relasi ForeignKey dan ManyToManyField, memastikan integritas antar data dan konsistensi operasional. Tabel 1 merangkum struktur detail masing-masing model.

Table 1. Model di Django

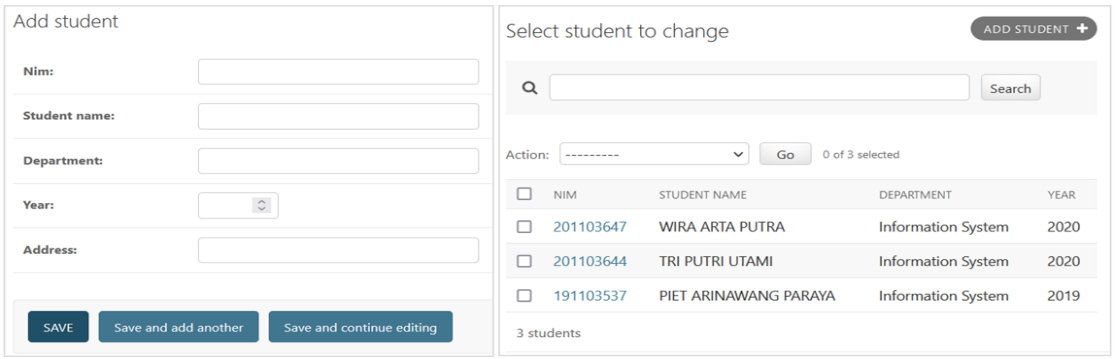
Model	Deskripsi	Field	Hubungan
Lecturer	Mewakili dosen dalam sistem.	id, lecturer_name, nidn, email, phone_number	Memiliki hubungan dengan model FinalProject sebagai pembimbing. Berelasi dengan Guidance dan Evaluation dalam membimbing dan mengevaluasi mahasiswa.
Student	Mewakili siswa dalam aplikasi.	nim, student_name, department, year, address	Berelasi dengan FinalProject sebagai pemilik proyek akhir. Berhubungan dengan Guidance untuk sesi bimbingan dengan dosen.
FinalProject	Merupakan proyek akhir yang dikerjakan oleh seorang mahasiswa di bawah bimbingan seorang Dosen.	title, description, start_date, end_date, status	ForeignKey ke Student dan Lecturer untuk menghubungkan proyek dengan mahasiswa dan dosen pembimbing. Terhubung dengan Evaluation untuk proses penilaian proyek.
Guidance	Mencatat sesi bimbingan yang berlangsung antara mahasiswa dan dosen terkait proyek tertentu.	student, lecturer, date, notes, status	ForeignKey ke Student dan Lecturer untuk menghubungkan sesi bimbingan dengan mahasiswa dan dosen pembimbing.
Evaluation	Merupakan evaluasi proyek akhir yang dilakukan oleh penguji.	final_project, examiner, date, comments, score	ForeignKey ke FinalProject dan Lecturer untuk menghubungkan evaluasi dengan proyek akhir dan dosen.
Examiner	Mewakili penguji yang mengevaluasi proyek akhir.	id, examiner_name, department, email, phone_number	Berpartisipasi dalam many-to-many relationship dengan FinalProject melalui Evaluation.

Sistem manajemen tugas akhir berbasis Django dirancang dengan model relasional, sebagaimana ditunjukkan pada Tabel 1. Sistem ini menggunakan enam model utama: Lecturer, Student, Final Project, Guidance, Evaluation, dan Examiner, yang saling terhubung melalui relasi ForeignKey dan Many-to-Many untuk menjaga integritas dan konsistensi data. Mahasiswa berperan sebagai pelaksana tugas akhir, dan dosen bertindak sebagai pembimbing dan penilai. Akhir proyek berfungsi sebagai entitas utama yang menghubungkan proses bimbingan melalui model bimbingan dan proses evaluasi melalui model evaluasi yang digunakan tim penguji. Dengan desain seperti ini, sistem menjadi lebih terintegrasi, tetapi tetap mengikuti prinsip konvensi atas konfigurasi Django.

C. Pengujian Model Dengan Django Admin

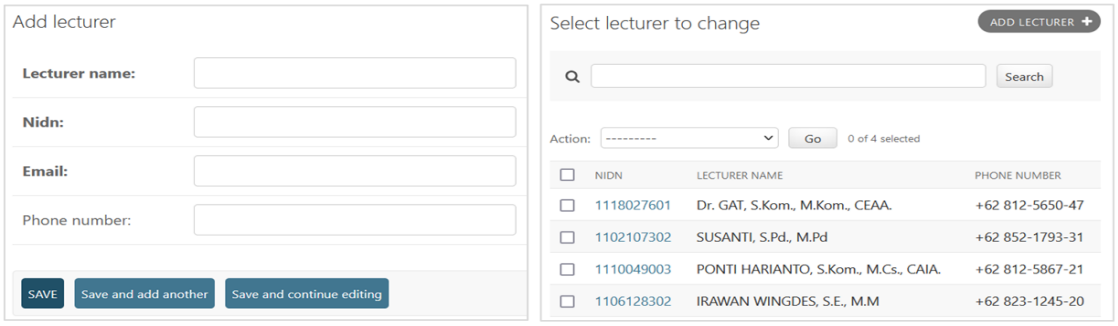
Untuk memastikan bahwa aplikasi backend berfungsi dengan baik, pengujian awal dilakukan melalui antarmuka Django Admin. Sebagai interface standar yang efisien dan fleksibel, Django Admin memungkinkan pengelolaan data dan verifikasi konfigurasi model secara langsung. Hasil pengujian menunjukkan bahwa semua model telah terdaftar, dapat diakses, dan berjalan sesuai spesifikasi, dengan integritas hubungan terjaga. Pengujian ini memastikan stabilitas performa aplikasi dengan menghindari inkonsistensi skema yang dapat mengganggu operasional sistem dan membantu mendeteksi kesalahan konfigurasi sejak dini.

e-ISSN: 2460-870X | p-ISSN: 2085-1367



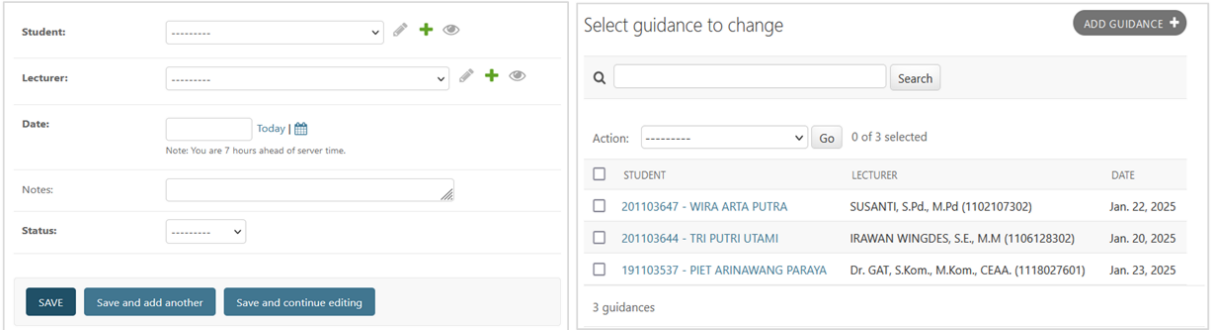
Gambar 2. Antarmuka Mahasiswa

Formulir pada Gambar 2 dirancang dengan tata letak yang terstruktur dan label yang jelas, sehingga memudahkan pengguna dalam menambah, menyimpan, dan mengedit data mahasiswa. Fitur pencarian yang terintegrasi meningkatkan efisiensi dalam pengelolaan basis data mahasiswa, terutama pada skala yang besar. Antarmuka ini disusun secara intuitif untuk mendukung administrator dalam mengelola data mahasiswa dengan lebih efektif.



Gambar 3. Antarmuka Dosen

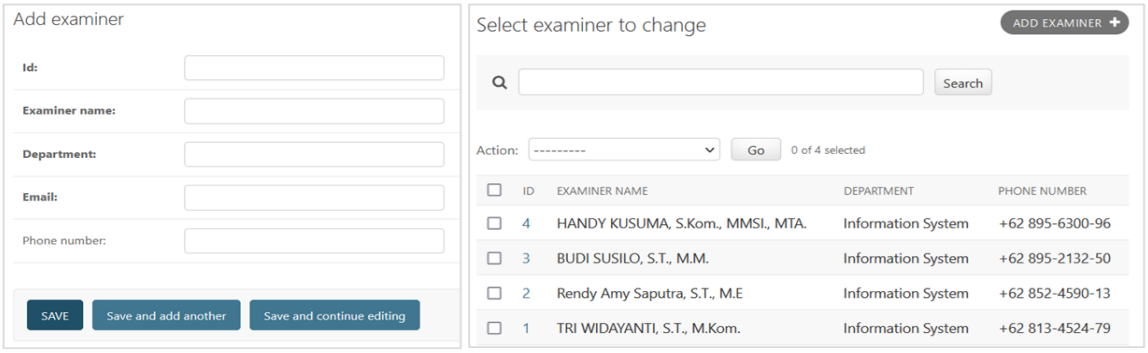
Seperti yang ditunjukkan pada Gambar 3, antarmuka administrasi dosen dirancang dengan cara yang memudahkan pengguna untuk menambahkan, mencari, dan mengubah data dosen. Sistem ini meningkatkan keandalan dan efisiensi dengan memperbaiki proses validasi, memastikan bahwa input seragam, dan menyediakan mekanisme konfirmasi pengguna.



Gambar 4. Antarmuka Bimbingan

Antarmuka untuk mengelola bimbingan dirancang secara fungsional dan mudah digunakan untuk mempermudah penambahan, pencarian, dan manajemen sesi bimbingan antara dosen dan siswa, seperti

yang ditunjukkan pada Gambar 4. Memantau bimbingan akademik menjadi lebih efektif dengan penjadwalan yang akurat, sistem validasi yang kuat, dan konsistensi status yang terjaga.



Gambar 5. Antarmuka Penguji

Gambar 5 menunjukkan antarmuka yang memfasilitasi pengelolaan data Examiner dengan fitur pencarian yang memudahkan navigasi dalam basis data yang luas. Desain yang terstruktur, didukung oleh sistem validasi dan mekanisme penanganan kesalahan yang efektif, memberi pengguna kontrol yang lebih besar atas perubahan dan penghapusan catatan.

D. Pengujian Unit

Pengujian unit dilakukan pada enam model utama (Student, Lecturer, FinalProject, Guidance, Evaluation, dan Examiner) untuk memverifikasi bahwa validasi data, relasi antar model, serta manajemen basis data berjalan sesuai rancangan. Tujuan utama uji ini adalah untuk menjamin integritas skema pada tingkat entitas, menemukan kesalahan konfigurasi sejak awal, dan mengevaluasi tanggapan aplikasi terhadap input yang tidak valid. Metode ini menggunakan Django TestCase. Ada 30 skenario pengujian untuk validasi tipe data, integritas ForeignKey, dan perilaku model terhadap data yang tidak memenuhi persyaratan. Hasil pengujian menunjukkan bahwa semua dari 30 skenario berhasil dijalankan tanpa error, dengan total waktu eksekusi 0,115 detik. Sistem menghasilkan output sebagai berikut:

```
Found 30 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
----------------------------------------------------------
Ran 30 tests in 0.115s
OK
Destroying test database for alias 'default'...
```

Dari output tersebut, Django telah menjalankan 30 unit test dengan sukses tanpa error dalam waktu 0.115 detik. Keberhasilan ini menunjukkan bahwa semua model didefinisikan dengan benar, bahwa hubungan antar entitas konsisten, dan bahwa tidak ada data yang salah yang gagal divalidasi. Hasil ini mendukung temuan penelitian sebelumnya [12][13] yang menunjukkan bahwa Django TestCase berfungsi dengan baik untuk menguji integritas skema dan hubungan pada sistem berbasis ORM. Namun demikian, pengujian unit dalam penelitian ini masih terbatas pada validasi kolom sederhana dan relasi dasar (one-to-one dan one-to-many). Situasi yang lebih kompleks, seperti hubungan many-to-many, kendala lintas tabel, serta penanganan konflik migrasi dalam pengembangan kolaboratif, belum menjadi fokus dalam penelitian ini. Skema yang lebih ketat bahkan sering memicu masalah migrasi dan penurunan kinerja, sebagaimana ditunjukkan dalam penelitian [5]. Oleh karena itu, pengujian unit ini menunjukkan bahwa model stabil dan dapat diterapkan pada sistem berskala kecil hingga menengah. Namun, untuk mendukung bukti empiris sebagai dasar pengambilan keputusan teknologi, evaluasi lanjutan pada skenario kompleks dan perbandingan dengan teknologi lain diperlukan. Dibawah ini adalah Table 2 poin penting dari pengujian unit untuk semua model.

e-ISSN: 2460-870X | p-ISSN: 2085-1367

Table 2. Poin Penting Dari Pengujian Unit Untuk Semua Model

Model	Poin Penting
Student	Pengujian pada model Student berhasil memvalidasi kolom-kolom penting seperti nim, student_name, department, dan year, memastikan bahwa setiap atribut berfungsi sesuai ketentuan.
Lecturer	Memverifikasi pembuatan objek Dosen dengan nidn, email, dan phone_number opsional yang valid. Uji validasi email dan phone_number berhasil lulus. Representasi string diuji dan dikonfirmasi sesuai harapan.
FinalProject	Pengujian pada model FinalProject berhasil memvalidasi field penting seperti title, description, start_date, end_date, dan status. Pengujian juga memastikan bahwa nilai start_date dan end_date mematuhi batasan logis, di mana start_date tidak berada di masa mendatang.
Guidance	Konfirmasi pembuatan objek Guidance dengan hubungan yang valid dengan Student dan Lecturer. Memastikan validasi tanggal berfungsi (bimbingan tidak dapat dijadwalkan di masa lalu).
Evaluation	Pengujian pada model Evaluation berhasil memvalidasi field score, date, serta keterkaitan dengan model FinalProject dan Lecturer. Validasi memastikan bahwa nilai score berada dalam rentang yang sesuai dan logis.
Examiner	Pembuatan objek Examiner yang telah terverifikasi mencakup fields seperti ID, examiner_name, department, dan email. Proses validasi untuk email dan nomor telepon telah berhasil dilakukan. Selain itu, representasi string telah diuji dan dikonfirmasi sesuai dengan format yang diharapkan.

Hasil pengujian unit (Tabel 2) menunjukkan bahwa seluruh model diuji untuk validasi hubungan antar entitas, penerapan batas logis, validasi field, dan representasi string dengan benar. Hal ini menunjukkan bahwa sistem memiliki rencana yang solid dan mampu menjaga pengelolaan data pada tingkat entitas yang konsisten dan akurat. Keberhasilan ini meningkatkan stabilitas aplikasi dan mengurangi kesalahan saat pengembangan. Namun, penelitian tambahan diperlukan untuk mengetahui apakah skenario ini efektif. Ini karena pengujian hanya memeriksa relasi dasar, bukan skenario many-to-many atau validasi batas lintas tabel yang lebih kompleks.

E. Pengujian Fungsional

Pengujian fungsional bertujuan untuk memastikan bahwa alur proses dan interaksi antar model sesuai dengan kebutuhan aplikasi, bukan pada detail implementasi. Setiap fitur diuji dengan skenario yang mewakili penggunaan nyata. Ini memastikan bahwa konfigurasi kolom, hubungan, dan integrasi data berjalan seperti yang diharapkan. Sistem telah secara konsisten memenuhi spesifikasi fungsional dasar, seperti yang ditunjukkan oleh keberhasilan semua kasus uji (Tabel 3). Namun, skenario beban tinggi atau konflik migrasi dalam tim kolaboratif belum dimasukkan dalam pengujian ini, yang merupakan komponen penting dari evaluasi menyeluruh aplikasi skala besar.

Table 3. Poin Utama Hasil Pengujian Fungsional Untuk Semua Model

Model	Kasus Uji	Tujuan	Pengamatan Utama
Lecturer	Validasi Fields	Memastikan lecturer_name, nidn, email, and phone_number tervalidasi.	Semua Fields yang diperlukan telah berhasil divalidasi.
Student	Validasi Data	Uji nim, student_name, department, dan year yang valid.	Semua pemeriksaan validasi data berhasil.
FinalProject	Manajemen Status dan Tanggal	Lakukan validasi terhadap status dan pastikan bahwa start_date serta end_date memiliki logika yang konsisten.	status dan ketergantungan tanggal ditangani seperti yang diharapkan.
Guidance	Interaksi Lecturer-Student	Periksa hubungan antara lecturer, student, dan guidance.	Semua pemetaan hubungan di antara model-model dipastikan berfungsi dengan baik.
Evaluation	Penilaian dan Ketergantungan FinalProject	Pastikan skor valid dan terhubung dengan finalproject yang sesuai.	Skor dihubungkan dengan benar dan disimpan dengan benar untuk evaluasi.
Examiner	Hubungan Examiner-Student	Validasi pemetaan examiner dengan beberapa mahasiswa dan finalproject.	Hubungan Many-to-Many dikelola dengan baik dan tautannya telah divalidasi.

Semua model beroperasi sesuai spesifikasi, dengan integritas data dan hubungan ForeignKey yang aman, seperti yang ditunjukkan oleh hasil pengujian fungsional (Tabel 3). Semua fitur utama sistem diuji

dengan sukses tanpa menemukan kesalahan, menegaskan kesiapan sistem untuk pengembangan atau penerapan di dunia nyata. Namun, pengujian hanya mencakup skenario fungsional dasar dan tidak memeriksa dinamika interaksi data dalam kondisi beban tinggi atau skenario kolaboratif yang kompleks, sehingga diperlukan pengujian tambahan untuk mengevaluasi stabilitas sistem dalam situasi seperti itu.

F. Pengujian Kinerja

Pengujian kinerja mengukur waktu eksekusi operasi CRUD (Create, Read, Update, Delete) pada dua skala dataset, 1.000 dan 10.000 entri, untuk mengukur efisiensi dan skalabilitas sistem. Simulasi ini menunjukkan kondisi beban nyata dan menggambarkan kemampuan Django ORM untuk mempertahankan kinerja saat volume data meningkat. Berikut adalah hasil pengujian kinerja untuk data sebanyak 1.000 entri:

```
Found 4 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
Create operation time: 0.0004317760467529297 seconds
.Delete operation time: 0.0012433528900146484 seconds
.Read operation time: 0.00010514259338378906 seconds
.Update operation time: 0.004390239715576172 seconds
----------------------------------------------------------

Ran 4 tests in 0.057s
OK
Destroying test database for alias 'default'...
```

Pengujian kinerja yang dilakukan menggunakan Django TestCase pada dataset 1.000 entri menunjukkan bahwa operasi Create berjalan dengan kecepatan yang sangat tinggi, rata-rata 0,00043 detik, yang menunjukkan bahwa ORM Django sangat efisien dalam penyimpanan data. Dengan kueri yang dioptimalkan, operasi Read menjadi yang tercepat berikutnya (0,00010 detik), sementara operasi Delete menjadi yang terlambat berikutnya (0,00124 detik) karena proses verifikasi integritas relasi ForeignKey. Operasi Update tercatat paling lambat (0,00439 detik) karena mencarinya, mengubahnya, dan menyimpannya kembali data. Pengujian kemudian dilanjutkan dengan jumlah data yang lebih besar, yaitu 10.000 entri, untuk menilai skalabilitas sistem secara lebih menyeluruh.

```
Found 4 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
Create operation time: 0.00044536590576171875 seconds
.Delete operation time: 0.0034513473510742188 seconds
.Read operation time: 0.0001342296600341797 seconds
.Update operation time: 0.04535698890686035 seconds
----------------------------------------------------------
Ran 4 tests in 0.499s
OK
Destroying test database for alias 'default'...
```

Setelah skala data diperbesar menjadi 10.000 entri, operasi Create dan Read tetap stabil dengan waktu eksekusi masing-masing 0,00045 detik dan 0,00013 detik, menunjukkan kemampuan basis data untuk menangani kueri dan penulisan dalam volume besar. Sebaliknya, operasi Update meningkat secara signifikan menjadi 0,04535 detik dan operasi Delete meningkat menjadi 0,00345 detik, yang menunjukkan beban yang lebih besar yang disebabkan oleh pengelolaan integritas data dan hubungan kompleks dalam volume besar. Pada dataset terbesar, operasi CRUD selesai dalam 0,04938 detik, dan pengujian rampung tanpa error dalam kurang dari 0,5 detik. Tabel 4 menunjukkan bahwa operasi modifikasi data (Update dan Delete) paling terpengaruh oleh pertambahan volume data, mengikuti tren pertumbuhan linier. Sebaliknya, operasi Create dan Read menunjukkan skalabilitas yang baik dengan kenaikan waktu yang paling sedikit. Tren ini menunjukkan bahwa Django ORM dapat mempertahankan efisiensi penulisan dan pembacaan data, tetapi pengelolaan pembaruan data massal membutuhkan

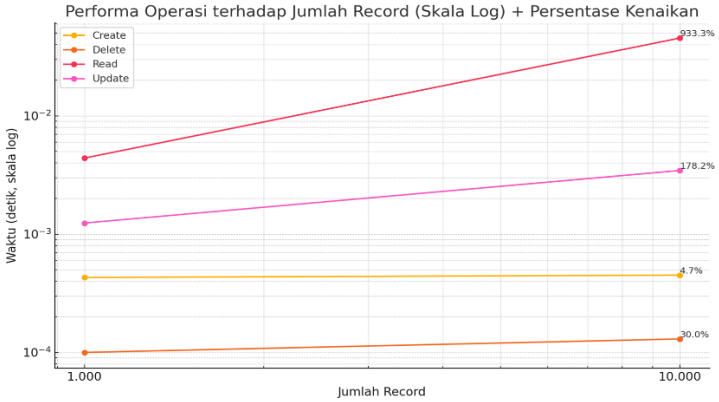
e-ISSN: 2460-870X | p-ISSN: 2085-1367

optimasi kinerja. Tabel 4 di bawah ini menyajikan perbandingan kinerja operasi Create, Read, Update, dan Delete pada dua skenario data 1.000 dan 10.000 entri dengan waktu eksekusi dalam satuan detik.

Table 4. Ringkasan Perbedaan Dalam Pengujian Kinerja

Operasi	1000 Record (Waktu dalam Detik)	10.000 Record (Waktu dalam Detik)	Perbedaan
Create	0.00043	0.00045	Peningkatan sedikit; dampak yang dapat diabaikan.
Delete	0.00010	0.00013	Peningkatan minimal; kueri yang efisien.
Read	0.00439	0.04536	Peningkatan signifikan; penskalaan linier.
Update	0.00124	0.00345	Peningkatan sedang; penskalaan yang dapat diterima

Tabel 4 menunjukkan bahwa operasi modifikasi data (Update dan Delete) paling terpengaruh oleh pertambahan volume data, mengikuti tren pertumbuhan linier. Sebaliknya, operasi Buat dan Baca menunjukkan skalabilitas yang baik dengan kenaikan waktu yang paling sedikit. Tren ini menunjukkan bahwa Django ORM dapat mempertahankan efisiensi penulisan dan pembacaan data, tetapi pengelolaan pembaruan data massal membutuhkan optimasi kinerja. Sebuah grafik line chart dengan skala lebar menunjukkan bahwa waktu eksekusi meningkat seiring bertambahnya jumlah data dari 1.000 ke 10.000. Hal ini menunjukkan bahwa evaluasi lebih lanjut diperlukan, terutama untuk operasi update pada jumlah data yang sangat besar dan aplikasi yang memiliki struktur basis data yang kompleks.



Gambar 6. Grafik Line Chart

Gambar 6 menunjukkan perbandingan waktu eksekusi untuk operasi Create, Delete, Read, dan Update pada skala data 1.000 dan 10.000 entri untuk memperjelas tren. Hasil pengujian pada operasi create menunjukkan performa yang stabil dan efisien; waktu penulisan data hanya meningkat sebesar 4,7%; Delete meningkat sebesar 30%, yang masih berada dalam batas wajar mengingat adanya verifikasi integritas relasional (ForeignKey); dan Read meningkat hingga 933,3%, menunjukkan kemungkinan bottleneck yang berasal dari kurangnya indeksasi atau optimasi kueri pada basis data. Update meningkat sebesar 178,2%, menunjukkan bahwa operasi modifikasi data mulai terjadi.

Hasil ini menunjukkan bahwa proses membaca dan mengupdate aplikasi berskala besar harus dioptimalkan khusus untuk menjaga kinerja stabil. Sebagai perbandingan, penelitian sebelumnya dalam International Journal of Research Publication and Reviews [8] melaporkan bahwa operasi Update pada data besar membutuhkan waktu lebih dari 0,1 detik, dan operasi Read menunjukkan keterbatasan saat data melebihi 5.000 entri. Hasil penelitian ini menunjukkan peningkatan efisiensi dan skalabilitas Django ORM yang digunakan, dengan waktu eksekusi yang lebih rendah dan kestabilan performa yang lebih baik. Hal ini mengindikasikan bahwa implementasi arsitektur model, indexing, dan struktur relasional

pada sistem ini telah dimanfaatkan secara optimal sehingga mampu bersaing atau melampaui hasil studi terdahulu.

G. Integration Testing

Tujuan pengujian integrasi adalah untuk memastikan interaksi model dan konsistensi data dalam sistem, terutama untuk relasi ForeignKey dan Many-to-Many. Fokus utama pengujian adalah validasi koneksi antar entitas, integritas data lintas model, dan dampak operasi berjenjang, seperti penghapusan dan pembaruan data pada model yang relevan.

1) Uji Model Hubungan Antara Student, FinalProject, and Lecturer

Pengujian ini memastikan bahwa setiap FinalProject memiliki referensi valid ke Student dan Lecturer melalui ForeignKey. Tidak ada FinalProject yang dapat dibuat tanpa relasi yang valid. Penghapusan Student atau Lecturer menyebabkan pembaruan atau penghapusan berjenjang yang konsisten tanpa meninggalkan data inkonsisten. Perubahan data pada Student atau Lecturer tercermin secara tepat pada FinalProject. Berikut adalah hasil pengujian hubungan antara Student, FinalProject, dan Lecturer:

```
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
----------------------------------------------------------
Ran 4 test in 0.011s
OK
Destroying test database for alias 'default'...
```

Menurut pengujian hubungan antara Student, FinalProject, dan Lecturer, konfigurasi relasi ForeignKey telah diterapkan dengan benar. Django secara otomatis menggunakan database uji yang berbeda sehingga pengujian dapat dilakukan dengan aman tanpa mengganggu data produksi. Sebagai bukti efisiensi tinggi model, validasi dilakukan dengan kecepatan luar biasa (0,011 detik). Sistem siap untuk pengembangan dan implementasi lebih lanjut, karena kesalahan konfigurasi tidak ditemukan. Meskipun hasil pengujian menunjukkan keberhasilan pada skenario dasar, cakupan pengujian belum mencakup hubungan many-to-many yang lebih kompleks, keterbatasan lintas tabel, dan skenario kolaboratif dengan risiko konflik migrasi. Selain itu, kontribusi praktis dari temuan ini dibatasi oleh ketiadaan perbandingan kuantitatif dengan framework atau metode manual lainnya. Oleh karena itu, sangat disarankan untuk melakukan pengujian tambahan dengan skenario dan data yang lebih kompleks untuk meningkatkan kebaruan dan validitas penelitian. Rangkuman hasil uji model hubungan antara student, finalproject, and lecturer disajikan pada Tabel 5.

Table 5. Hasil Model Hubungan Antara Student, FinalProject, dan Lecturer

Kasus Uji	Tujuan	Hasil yang Diharapkan
Student-FinalProject Relationship	Memverifikasi bahwa FinalProject terhubung dengan benar ke Student tertentu melalui foreign key.	Atribut FinalProject-student cocok dengan objek Student.
FinalProject- Lecturer Relationship	Memastikan bahwa FinalProject terhubung dengan Lecturer yang sesuai.	Atribut FinalProject-Lecturer cocok dengan objek Lecturer.
Cascade Delete (Student)	Menguji perilaku penghapusan berjenjang: ketika Student dihapus, FinalProject yang terkait juga turut dihapus.	FinalProject yang terkait dengan Student yang dihapus sudah tidak lagi tersedia dalam basis data.
Set Null on Lecturer Delete	Verifikasi bahwa penghapusan Lecturer akan mengakibatkan field guidance pada FinalProject menjadi nol.	Field FinalProject- Lecturer menjadi null, namun record FinalProject tetap tersimpan dalam tabel.

Hasil Tabel 5 menunjukkan bahwa mekanisme cascading dan SET_NULL berfungsi dengan baik. Ini dimaksudkan untuk menjaga integritas data saat menghapus entitas yang relevan. Setiap proyek terakhir

e-ISSN: 2460-870X | p-ISSN: 2085-1367

dihubungkan secara valid ke siswa dan pengajar, yang memastikan keterkaitan data yang konsisten dan dapat diandalkan.

2) Pengujian Hubungan Antara FinalProject, Guidance, dan Evaluation:

Tujuan dari pengujian ini adalah untuk memastikan integritas dan kelengkapan hubungan ForeignKey yang ada antara model FinalProject, Guidance, dan Evaluation. Setiap entri pada Guidance harus memiliki referensi yang sah ke FinalProject, dan entri pada Evaluasi harus terkait dengan proyek akhir yang sebenarnya. Pengujian memastikan bahwa perubahan atau penghapusan data pada satu model secara akurat mencerminkan model terkait tanpa menimbulkan inkonsistensi. Temuan pengujian ini menegaskan bahwa sistem mampu mempertahankan keterkaitan antar entitas secara konsisten saat terjadi perubahan data, yang penting untuk menjaga keutuhan data dan kelancaran proses manajemen proyek akhir. Temuan dari pengujian ini disajikan pada bagian berikut:

```
Found 2 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
----------------------------------------------------------
Ran 2 tests in 0.020s
OK
Destroying test database for alias 'default'..
```

Django berhasil menjalankan dua pengujian pada hubungan antara FinalProject, Guidance, dan Evaluation tanpa error dengan waktu eksekusi yang singkat, menunjukkan beban pengujian yang ringan dan stabilitas kode dalam skenario tersebut. Tabel 6 menggambarkan hasil pengujian yang menunjukkan bahwa penerapan relasi antar model memenuhi persyaratan dan aturan database. Saat entitas utama dihapus, integritas data dijaga oleh mekanisme delete cascading dan SET_NULL.

Table 6. Hasil Hubungan Model antara FinalProject, Guidance, dan Evaluation

Kasus Uji	**Deskripsi**	**Hasil yang Diharapkan**
-Student Relationship	Menguji keterhubungan antara objek Guidance dan objek Student dengan tepat.	Kolom Student di Guidance harus sesuai dengan objek Student.
Guidance-Lecturer Relationship	Menguji hubungan antara objek Guidance dan objek Lecturer untuk memastikan keduanya terhubung dengan benar.	Kolom Lecturer di Guidance harus sesuai dengan objek Lecturer.
Evaluation-FinalProject Relationship	Memastikan bahwa objek Evaluation terhubung dengan objek FinalProject dengan tepat.	Kolom FinalProject dalam Evaluation harus sesuai dengan objek FinalProject.
Evaluation-Lecturer Relationship	Menguji apakah objek Evaluation terhubung dengan objek Lecture secara tepat.	Kolom pemeriksaan dalam Evaluation harus selaras dengan objek tempat Lecturer tersebut bernaung.

Tabel 6 menunjukkan bahwa hubungan antara FinalProject, Guidance, dan Evaluation telah diimplementasikan dengan baik, memastikan integritas data dan penanganan yang andal. Ketiga model ini saling terhubung sesuai dengan batasan yang ditetapkan. Selain itu, mekanisme penghapusan berjenjang dan SET_NULL memastikan konsistensi database, bahkan ketika entitas utama dihapus.

3) Test Multiple Examiners and Many-to-Many Relationship:

Pengujian selanjutnya menekankan hubungan Many-to-Many, terutama bagi Examiners yang dapat berhubungan dengan banyak FinalProject. Fokus utama dari pengujian ini adalah pengesahan integritas hubungan, kemampuan sistem untuk memperbarui dan menghapus data terkait tanpa menimbulkan inkonsistensi, dan skalabilitas manajemen hubungan dalam volume data yang lebih besar. Berikut adalah hasil pengujiannya:

```
Found 1 test(s).
```

```
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
----------------------------------------------------------
Ran 1 test in 0.013s
OK
Destroying test database for alias 'default'...
```

Hasil pengujian menunjukkan bahwa Django ORM dapat dengan baik mengelola hubungan banyak-ke-banyak, menjaga kerahasiaan data tanpa kesalahan, dan berjalan dengan lancar. Namun, pengujian ini tidak mencakup situasi kompleks seperti keterbatasan lintas tabel, konflik migrasi skema dalam tim kolaboratif, atau efek beban data besar terhadap kinerja produksi sebenarnya. Berikut adalah tabel yang merangkum poin utama dari hasil pengujian multiple Examiners serta hubungan Many-to-Many antara FinalProject dan Examiners (Tabel 7):

Table 7. Uji Multiple Examiners dan Hubungan Many-To-Many

Skenario Uji Coba	Deskripsi	Hasil yang Diharapkan	Hasil dari Tes
Multiple Examiners for FinalProject	Memastikan FinalProject dapat memiliki Multiple Examiners.	FinalProject harus terhubung dengan beberapa Examiner, yang menegaskan adanya hubungan many-to-many.	Multiple Examiners telah dikaitkan dengan FinalProject secara tepat.
Examiners Assigned to Multiple Projects	Memastikan bahwa Examiners dapat ditugaskan ke lebih dari satu FinalProject.	Seorang Examiner harus dapat mengevaluasi beberapa instance FinalProject, yang menegaskan hubungan terbalik.	Examiners telah ditugaskan dengan benar ke multiple FinalProjects.
Remove Examiners from FinalProject	Memastikan bahwa Examiners dapat dihapus dari suatu FinalProject.	Setelah dihapus, Examiners tidak boleh lagi terhubung dengan FinalProject.	Examiners telah berhasil dihapus dari FinalProject.
Delete Examiners	Menguji penghapusan Examiners untuk memastikan keterpisahan dari FinalProject.	Saat Examiners dihapus, hubungan yang terkait harus dihapus dengan benar, tanpa meninggalkan rekaman tanpa referensi.	Penghapusan Examiners juga menghapus asosiasinya dengan FinalProject.
Delete FinalProject	Menguji penghapusan FinalProject untuk memastikan keterpisahannya dari Examiners.	Menghapus FinalProject juga akan menghapus asosiasinya dengan Examiners.	Penghapusan FinalProject juga menghapus hubungannya dengan semua Examiners yang terhubung.

Tabel 7 menunjukkan bahwa pengujian hubungan many-to-many antara FinalProject dan Examiners telah berhasil divalidasi. Operasi CRUD berjalan dengan baik, yang mencakup pengelolaan asosiasi, penghapusan, dan peniadaan hubungan dengan benar. Meskipun pengujian fungsional ini menunjukkan kestabilan, uji beban dan kompetisi harus dilakukan untuk memastikan keandalan dengan data dalam kondisi nyata dan akses simultan yang lebih tinggi.

H. Pengujian Regresi

Pengujian regresi dilakukan untuk memastikan bahwa perubahan kode tidak mengganggu fungsi yang sudah ada pada model siswa, pengajar, proyek akhir, bimbingan, evaluasi, dan penilaian. Dua tes berhasil tanpa error dalam waktu singkat (0,015 detik), menunjukkan bahwa model tetap stabil dan konsisten setelah pembaruan. Baik representasi string maupun definisi kolom model Evaluasi tidak mengalami masalah, yang menunjukkan bahwa aplikasi siap untuk pengembangan lanjutan. Berikut adalah temuan dari pengujian regresi:

```
Found 2 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
----------------------------------------------------------
Ran 2 tests in 0.015s
OK
```

e-ISSN: 2460-870X | p-ISSN: 2085-1367


```
Destroying test database for alias 'default'...
```

Pengujian regresi menunjukkan bahwa sistem tetap stabil dan dapat diandalkan meski ada perubahan pada kode, model, atau skema basis data. Data, fungsi, dan kinerjanya juga tetap terjaga dengan baik. Hasil ini menandakan bahwa pembaruan tidak mengganggu fitur yang sudah ada, sehingga sistem siap digunakan dan dikembangkan lebih lanjut. Berikut adalah Tabel 8 yang merangkum poin utama dari hasil Pengujian Regresi:

Table 8. Hasil Uji Regresi

Poin Utama	Deskripsi	Hasil Uji
Data Integrity	Memastikan tidak ada kerusakan atau kehilangan data saat model atau skema basis data mengalami perubahan.	Semua catatan data tetap konsisten tanpa kerusakan.
Correct Relationships	Memverifikasi bahwa hubungan antar model, seperti Student, Lecturer, dan FinalProject, tetap utuh.	Foreign key dan hubungan antar model tetap valid.
CRUD Operations	Memastikan bahwa operasi Create, Read, Update, dan Delete tetap berfungsi dengan baik setelah modifikasi.	Operasi Create, Read, Update, dan Delete tetap berfungsi dengan baik setelah perubahan.
Validation Rules	Memastikan bahwa validasi field model, seperti keunikan dan panjang maksimum, tetap berfungsi dengan baik setelah diperbarui.	Aturan validasi, seperti keunikan dan batas panjang field, berfungsi sesuai harapan.
Error Handling	Memastikan penanganan kesalahan yang tepat untuk operasi tidak valid, seperti menghapus atau memperbarui entri yang tidak ada.	Meningkatkan penanganan kesalahan dan pengecualian saat diperlukan.
Model Data Consistency	Memastikan bahwa field model dan data terkait tetap konsisten setelah perubahan.	Nilai field model tetap valid dan konsisten.
Functional Continuity	Memastikan semua fungsionalitas, termasuk pengambilan dan tampilan data model, tetap berfungsi dengan baik sesuai harapan.	Fungsionalitas sistem tetap utuh setelah pembaruan.
Foreign Key Integrity	Memastikan hubungan foreign key tetap terjaga agar tidak ada catatan yang terisolasi.	Foreign key berfungsi dengan baik tanpa meninggalkan catatan yang terisolasi.
Data Persistence	Memastikan keakuratan penyimpanan data dalam operasi Create, Update, dan Delete.	Data tersimpan dengan baik dan dapat diambil sesuai harapan.

Seperti yang ditunjukkan oleh ringkasan hasil pengujian regresi di Tabel 8, sistem dapat dengan sukses mempertahankan integritas data, fungsionalitas, dan konsistensi pasca-modifikasi. Perubahan pada kode atau model dan skema basis data menunjukkan bahwa sistem masih andal dan memenuhi harapan. Sebagai bukti keamanan sistem, tidak ada masalah yang signifikan.

4. KESIMPULAN

Otomatisasi skema basis data melalui model dan migrasi di Django terbukti menjadi pendekatan yang efisien dalam menangani kompleksitas aplikasi proyek akhir. Kombinasi Django ORM, sistem migrasi, dan pengujian otomatis memungkinkan basis data tetap konsisten, optimal, dan skalabel. Praktik desain model yang baik membantu pengembang fokus pada fungsionalitas utama tanpa terbebani oleh pengelolaan data secara manual, sekaligus mempercepat proses pengembangan dan meningkatkan keandalan sistem.Terbukti bahwa model dan migrasi Django untuk otomatisasi skema basis data adalah metode yang efektif dan dapat diandalkan untuk mengelola aplikasi proyek akhir. Operasi CRUD dapat dilakukan dengan waktu eksekusi yang sangat cepat, seperti yang ditunjukkan oleh pengujian kinerja. Operasi Create membutuhkan 0,00043 detik untuk 1.000 entri, sementara operasi Read hanya membutuhkan 0,00010 detik. Operasi Create tetap efisien dalam 0,00045 detik dan operasi Read dalam 0,00013 detik, bahkan ketika jumlah data meningkat menjadi 10.000 entri. Namun, operasi Update meningkatkan waktunya secara signifikan dari 0,00439 detik menjadi 0,04535 detik, yang merupakan peningkatan yang signifikan dari 0,00439 detik menjadi 0,04535 detik. Model utama seperti Student, FinalProject, Lecturer, Guidance, Evaluation, dan Examiner diimplementasikan dengan baik, menjaga data tetap aman dan hubungan kompleks antar entitas tanpa kesalahan, seperti yang ditunjukkan oleh

pengujian unit dan fungsional. Pengujian regresi juga membuktikan stabilitas sistem dengan waktu hanya 0,015 detik, yang menunjukkan bahwa perubahan kode tidak merusak fungsi yang sudah ada.

Meskipun demikian, cakupan penelitian ini masih terbatas pada pengujian skenario kompleks seperti pengelolaan migrasi di lingkungan kolaboratif, keterbatasan lintas tabel, dan hubungan many-to-many yang lebih kompleks. Selain itu, kekuatan temuan sebagai dasar pengambilan keputusan dibatasi oleh ketiadaan perbandingan kuantitatif dengan struktur atau metode manual lainnya. Sebuah penelitian lebih lanjut disarankan untuk menyelidiki pengelolaan migrasi dan konflik skema dalam tim pengembang, pengoptimalan kueri update dan delete, dan kinerja Django ORM pada volume data yang lebih besar dan kondisi nyata. Untuk aplikasi skala produksi, aspek keamanan, ketahanan kesalahan, dan dukungan multibahasa harus menjadi fokus penelitian. Dengan demikian, Django terbukti sebagai platform yang kuat dan skalabel untuk pengembangan sistem manajemen proyek akhir, selama pengoptimalan lanjutan diterapkan untuk menangani skenario skala besar dan kompleks.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Sekolah Tinggi Manajemen Informatika dan Komputer (STMIK) Pontianak atas dukungan finansial yang memungkinkan penelitian ini terlaksana. Penghargaan yang tulus juga disampaikan kepada rekan-rekan dosen yang telah berbagi masukan berharga serta memberikan dukungan selama proses penulisan. Tidak lupa, penulis berterima kasih kepada para reviewer atas bimbingan dan arahannya, yang sangat membantu dalam menyempurnakan tulisan ini. Semoga karya ini dapat memberikan manfaat bagi banyak orang, baik saat ini maupun di masa yang akan datang.

REFERENSI

[1] Lakkimsetty, N. R. S. C. G. (2023). Database optimization strategies: Enhancing performance and scalability. International Journal of Computer Science and Mobile Computing, 12(11), 69-89.

[2] Uzzaman, A., Jim, M. M. I., Nishat, N., & Nahar, J. (2024). Optimizing SQL databases for big data workloads: techniques and best practices. Academic Journal on Business Administration, Innovation & Sustainability, 4(3), 15-29.

[3] Ramu, V. B. (2023). Optimizing Database Performance: Strategies for Efficient Query Execution and Resource Utilization. International Journal of Computer Trends and Technology, 71(7), 15-21.

[4] Thakur, P., & Jadon, P. (2023, May). Django: Developing web using python. In 2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE) (pp. 303-306). IEEE.

[5] Kumar, M., & Nandal, D. R. (2024). Python's Role in Accelerating Web Application Development with Django. Int. Res. J. Adv. Eng. Manag, 2(06), 2092-2105.

[6] Dhadil, S., Srivastava, A., Shinde, V., Walhekar, V., Patil, A., Ganeshpurkar, A., ... & Kulkarni, R. (2024). Django Unleashed: A Deep Dive into the Features and Advantages of the Django Framework. RGUHS Journal of Pharmaceutical Sciences, 14(3).

[7] Perez, B. M. C., Ramos, M. J. R., Mora, J. J. H., Arenas, M. C., & Hernandez, M. M. J. S. (2021). Experience of using the Django ORM for Migration Redesign and Implementation of a Database of MySQL Data to Postgre SQL. International Journal of Science and Research (IJSR), 10(11), 96-98.

[8] Pandey, R., Shrivastava, V., Pandey, A., & Tiwari, A. K. (2025). Building scalable web application with Python Django. International Journal of Research Publication and Reviews, 6(5), 6658-6659.

e-ISSN: 2460-870X | p-ISSN: 2085-1367

[9] Chen, S., Ahmmed, S., Lal, K., & Deming, C. (2020). Django web development framework: Powering the modern web. American Journal of Trade and Policy, 7(3), 99-106.

[10] Ewing, T., Redfern, J., Alexander, A., Medina, R., & Birath, E. (2021, March). Django as a Mission Planning Tool Interface for the CYGNSS Mission. In 2021 IEEE Aerospace Conference (50100) (pp. 1-6). IEEE.

[11] Pang, R. (2024, July). Analysis of Python web development applications based on the Django framework. In Third International Conference on Electronic Information Engineering, Big Data, and Computer Technology (EIBDCT 2024) (Vol. 13181, pp. 1580-1584). SPIE.

[12] Gobert, M., Nagy, C., Rocha, H., Demeyer, S., & Cleve, A. (2023). Best practices of testing database manipulation code. Information systems, 111, 102105.

[13] Strich, J., Schneider, F., Nikishina, I., & Biemann, C. (2024, August). On Improving Repository-Level Code QA for Large Language Models. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop) (pp. 209-244).