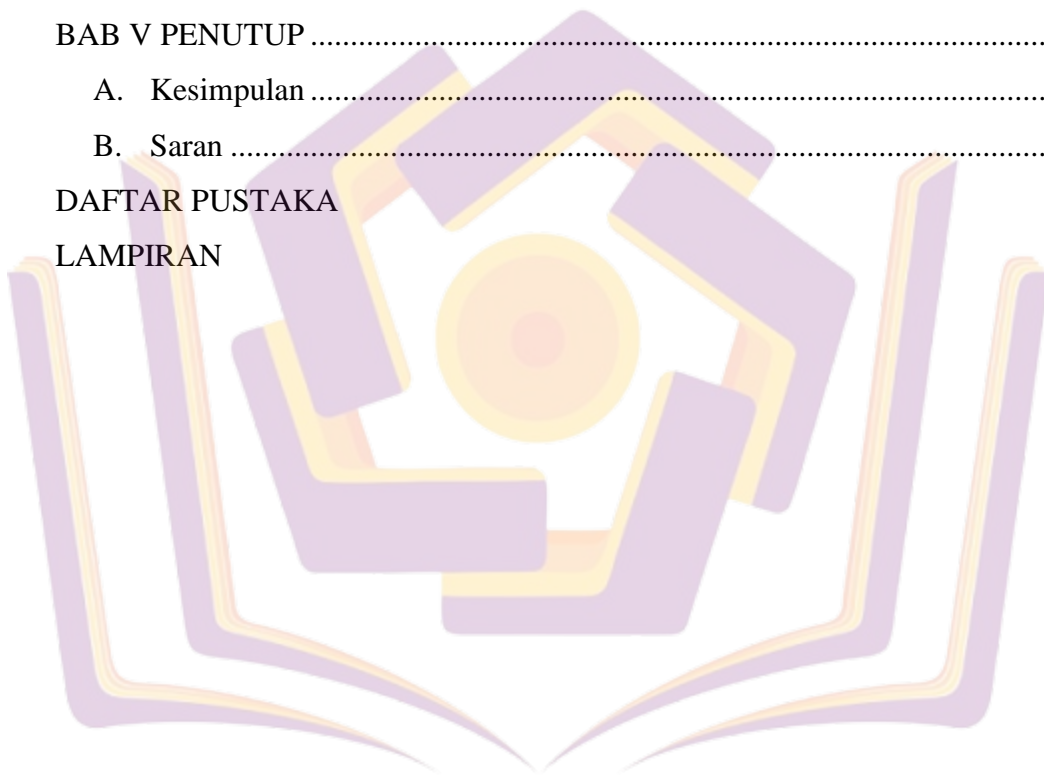


DAFTAR ISI

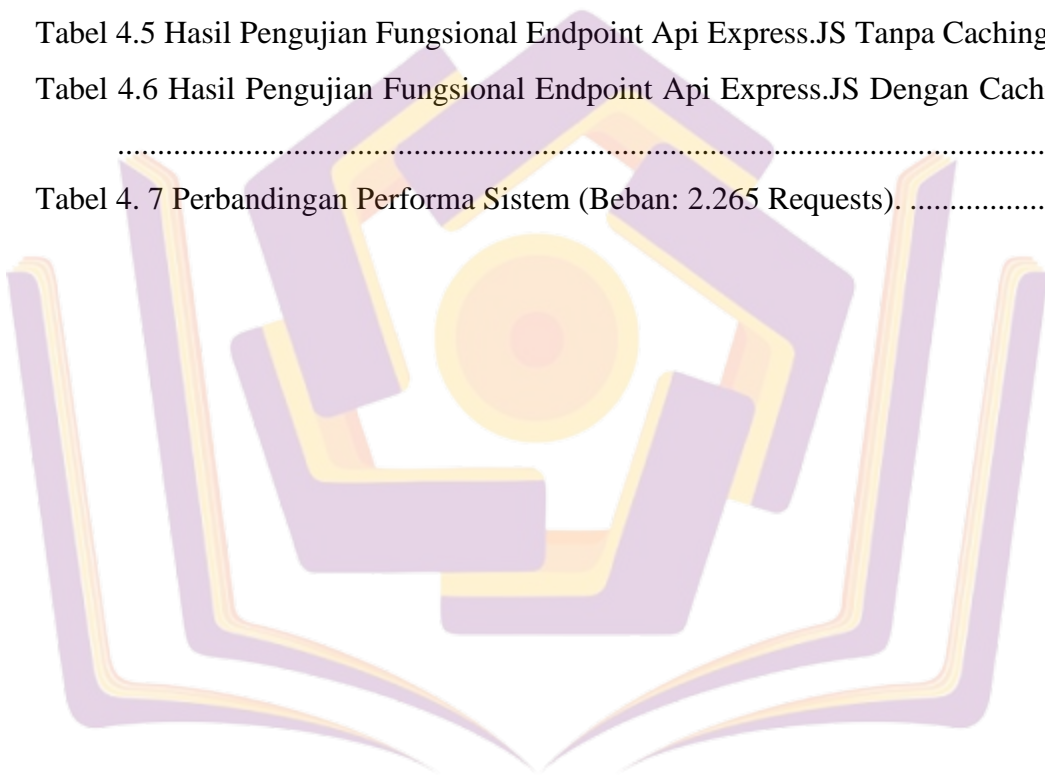
HALAMAN SAMPUL.....	i
HALAMAN JUDUL	ii
HALAMAN PERSETUJUAN	iii
HALAMAN PENGESAHAN	iv
HALAMAN PERNYATAAN KEASLIAN	v
HALAMAN PERSEMBAHAN	vi
HALAMAN MOTTO.....	viii
KATA PENGANTAR.....	ix
DAFTAR ISI	xi
DAFTAR TABEL	xiii
DAFTAR GAMBAR.....	xiv
DAFTAR ISTILAH.....	xv
DAFTAR LAMPIRAN	xvii
INTISARI	xviii
<i>ABSTRACT</i>	xix
BAB I PENDAHULUAN.....	1
A. Latar Belakang Masalah	1
B. Rumusan Masalah.....	5
C. Batasan Masalah	5
D. Tujuan Penelitian	6
E. Manfaat Penelitian	7
BAB II TINJAUAN PUSTAKA	9
A. Landasan Teori.....	9
B. Penelitian Sebelumnya.....	42
BAB III METODE PENELITIAN	46
A. Tempat dan Waktu Penelitian.....	46
B. Metode Pengumpulan Data.....	46
C. Alat dan Bahan Penelitian.....	49
D. Konsep Penelitian	51

BAB IV HASIL DAN PEMBAHASAN.....	58
A. Perancangan Spesifikasi API (<i>API Specification Design</i>).....	58
B. Perancangan Basis Data (<i>Database Design</i>).....	62
C. Perancangan Alur Interaksi (<i>Interaction Flow Design</i>).....	65
D. Pembangunan Layanan Prediksi FastAPI.....	69
E. Pengembangan Modul Integrasi Express.js.....	72
F. Pengujian Fungsional (<i>Functional Testing</i>).....	80
G. Pengujian Performa (<i>Performance Testing</i>).....	83
BAB V PENUTUP.....	88
A. Kesimpulan.....	88
B. Saran.....	89
DAFTAR PUSTAKA	
LAMPIRAN	



DAFTAR TABEL

Tabel 2.1 Penelitian Sebelumnya.....	45
Tabel 3.1 Dataset.....	48
Tabel 4.1 Detail spesifikasi api layanan prediksi FastApi.	59
Tabel 4.2 Spesifikasi Endpoint Layanan Integrasi (Express.js).....	60
Tabel 4.3 Spesifikasi Atribut Baru pada Tabel Mahasiswa.	65
Tabel 4.4 Hasil Pengujian Fungsional Layanan Prediksi.	81
Tabel 4.5 Hasil Pengujian Fungsional Endpoint Api Express.JS Tanpa Caching.	82
Tabel 4.6 Hasil Pengujian Fungsional Endpoint Api Express.JS Dengan Caching.	83
Tabel 4. 7 Perbandingan Performa Sistem (Beban: 2.265 Requests).	87



DAFTAR GAMBAR

Gambar 2.1 Pattern Kode Backend.....	10
Gambar 2.2 Felder-Silverman Learning Model.....	18
Gambar 2.3 Relasi One-to-one.....	21
Gambar 2.4 Relasi One-to-many.....	22
Gambar 2.5 Relasi Many-to-many.....	23
Gambar 2.6 Controller.....	31
Gambar 2.7 Middleware.....	32
Gambar 2.8 Teknik Caching.....	39
Gambar 3.1 Alur Penelitian.....	54
Gambar 4.1 Rancangan ERD Pengembangan Tabel Mahasiswa.....	63
Gambar 4.2 Perbedaan Tabel mhs Sebelum Dan Sesudah Dijalankan Migrasi. ...	64
Gambar 4.3 Sequence diagram alur tanpa caching.....	66
Gambar 4.4 Sequence Diagram Alur Dengan Caching (Stale-While-Revalidate).	68
Gambar 4.5 Algoritma Logika Prediksi pada Layanan FastAPI.....	70
Gambar 4.6 Kode Semu Algoritma Prediksi Gaya Belajar.....	71
Gambar 4.7 Diagram Alir Logika Controller Tanpa Caching.....	73
Gambar 4.8 Kode Semu Algoritma Endpoint Tanpa Caching.....	75
Gambar 4.9 Diagram Alir Logika Controller Dengan Caching.....	77
Gambar 4.10 Kode Semu Algoritma Endpoint Dengan Caching.....	79
Gambar 4.11 Ringkasan Beban (Load Summary) Skenario Tanpa Caching.....	84
Gambar 4.12 Distribusi Latensi Skenario Tanpa Caching.....	85
Gambar 4.13 Ringkasan Beban (Load Summary) Skenario Dengan Caching.....	86
Gambar 4.14 Distribusi Latensi Skenario Dengan Caching.....	86

DAFTAR ISTILAH

<i>API (Application Programming Interface):</i>	Antarmuka yang memungkinkan dua aplikasi perangkat lunak yang berbeda untuk saling berkomunikasi dan bertukar data.
<i>Backend:</i>	Bagian dari aplikasi web yang berjalan di sisi <i>server</i> , bertanggung jawab atas logika bisnis, pengelolaan basis data, dan <i>API</i> .
<i>Caching:</i>	Proses penyimpanan data sementara di lokasi penyimpanan cepat (seperti memori) agar permintaan data di masa mendatang dapat dilayani lebih cepat.
<i>Endpoint:</i>	Titik akhir spesifik (<i>URL</i>) dalam <i>API</i> di mana aplikasi klien dapat mengakses sumber daya atau fungsi tertentu.
<i>FSLSM (Felder-Silverman Learning Style Model):</i>	Model gaya belajar yang mengkategorikan preferensi belajar individu ke dalam empat dimensi: Aktif/Reflektif, Sensing/Intuitif, Visual/Verbal, dan Sekuensial/Global.
<i>JSON (JavaScript Object Notation):</i>	Format pertukaran data yang ringan, mudah dibaca manusia, dan mudah diuraikan oleh mesin.
<i>Latency:</i>	Waktu tunda yang dibutuhkan data untuk berpindah dari satu titik ke titik lain dalam jaringan.
<i>Microservices:</i>	Arsitektur pengembangan perangkat lunak di mana aplikasi disusun sebagai kumpulan layanan kecil yang independen dan saling berkomunikasi.
<i>REST (Representational State Transfer):</i>	Gaya arsitektur standar untuk sistem terdistribusi yang sering digunakan dalam pengembangan layanan web.
<i>Stale-While-Revalidate:</i>	Strategi caching di mana sistem menyajikan data lama (<i>stale</i>) dari <i>cache</i> secara instan kepada

pengguna, sambil melakukan pembaruan data di latar belakang.

Throughput:

Jumlah unit data atau permintaan yang dapat diproses oleh sistem dalam periode waktu tertentu.

TTL (Time-To-Live):

Batas waktu yang menentukan berapa lama data disimpan dalam *cache* sebelum dianggap kedaluwarsa.



DAFTAR LAMPIRAN

Lampiran 1. Kartu Bimbingan Skripsi.

Lampiran 2. Spesifikasi Api Layanan Prediksi FastApi.

Lampiran 3. Spesifikasi Endpoint Layanan Integrasi Dengan Caching (Express.js).

Lampiran 4. Spesifikasi Endpoint Layanan Integrasi Tanpa Caching (Express.js).

Lampiran 5. Kode Program Migrasi.

Lampiran 6. Implementasi Kode Program Untuk Layanan Prediksi FastApi.

Lampiran 7. Implementasi Kode Program Untuk Controller Express JS Tanpa Caching.

Lampiran 8. Implementasi Kode Program Untuk Controller Express JS Dengan Caching.

Lampiran 9. Hasil Pengujian Fungsional Layanan Prediksi FastApi.

Lampiran 10. Hasil Pengujian Fungsional Api Express Js Tanpa Caching

Lampiran 11. Hasil Pengujian Fungsional Api Expressjs Dengan Caching.