



TUGAS AKHIR - EF234801

PENGEMBANGAN MODUL PAYROLL: INTEGRASI *FINGERPRINT* DAN OTOMASI PENGHITUNGAN GAJI BERBASIS JABATAN DENGAN KINDERFIN

YUSUF HASAN NAZILA

NRP 5025211225

Dosen Pembimbing I

Adhatus Solichah Ahmadiyah, S.Kom, M.Sc.

NIP 198508262015042002

Dosen Pembimbing II

Dr. Kelly Rossa Sungkono, S.Kom., M.Kom

NIP 1994201912088

Program Studi S1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2025



TUGAS AKHIR - EF234801

**PENGEMBANGAN MODUL PAYROLL: INTEGRASI
FINGERPRINT DAN OTOMASI PENGHITUNGAN GAJI
BERBASIS JABATAN DENGAN KINDERFIN**

YUSUF HASAN NAZILA

NRP 5025211225

Dosen Pembimbing I

Adhatus Solichah Ahmadiyah, S.Kom, M.Sc.

NIP 198508262015042002

Dosen Pembimbing II

Dr. Kelly Rossa Sungkono, S.Kom., M.Kom

NIP 1994201912088

Program Studi S1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2025



FINAL PROJECT - EF234801

PAYROLL MODULE DEVELOPMENT: FINGERPRINT INTEGRATION AND AUTOMATED SALARY CALCULATION BASED ON POSITION USING KINDERFIN

Yusuf Hasan Nazila

NRP 5025211225

Advisor I

Adhatus Solichah Ahmadiyah, S.Kom, M.Sc.

NIP 198508262015042002

Advisor II

Dr. Kelly Rossa Sungkono, S.Kom., M.Kom

NIP 1994201912088

Study Program Bachelor of Informatics

Department of Informatics

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

Surabaya

2025

LEMBAR PENGESAHAN

PENGEMBANGAN MODUL PAYROLL: INTEGRASI FINGERPRINT DAN OTOMASI PENGHITUNGAN GAJI BERBASIS JABATAN DENGAN KINDERFIN

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Komputer pada
Program Studi S-1 Teknik Informatika
Departemen Teknik Informatika
Fakultas Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh : YUSUF HASAN NAZILA

NRP. 5025211225

Disetujui oleh Tim Penguji Tugas Akhir :

- | | |
|---|---------------|
| 1. Adhatus Solichah Ahmadiyah, S.Kom, M.Sc. | Pembimbing |
| 2. Dr. Kelly Rossa Sungkono, S.Kom., M.Kom | Ko-pembimbing |
| 3. Nama dan gelar penguji | Penguji |
| 4. Nama dan gelar penguji | Penguji |
| 5. Nama dan gelar penguji | Penguji |

SURABAYA

Mei, 2025

(Halaman ini sengaja dikosongkan)

APPROVAL SHEET

PAYROLL MODULE DEVELOPMENT: FINGERPRINT INTEGRATION AND AUTOMATED SALARY CALCULATION BASED ON POSITION USING KINDERFIN

TUGAS AKHIR

Submitted to fulfill one of the requirements
for obtaining a degree Bachelor of Computer Science at
Undergraduate Study Program of Informatics
Department of Informatics
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember

By : YUSUF HASAN NAZILA

NRP. 5025211225

Approved by Final Project Examiner Team:

- | | |
|---|---------------|
| 1. Adhatus Solichah Ahmadiyah, S.Kom, M.Sc. | Pembimbing |
| 2. Dr. Kelly Rossa Sungkono, S.Kom., M.Kom | Ko-pembimbing |
| 3. Nama dan gelar penguji | Examiner |
| 4. Nama dan gelar penguji | Examiner |
| 5. Nama dan gelar penguji | Examiner |

SURABAYA

June, 2025

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

Nama mahasiswa / NRP : Yusuf Hasan Nazila
Program studi : S-1 Teknik Informatika
Dosen Pembimbing I / NIP : Adhatus Solichah Ahmadiyah, S.Kom, M.Sc. /
198508262015042002
Dosen Pembimbing II / NIP : Dr. Kelly Rossa Sungkono, S.Kom., M.Kom /
1994201912088

ini menyatakan bahwa Tugas Akhir dengan judul “Pengembangan Modul *Payroll*: Integrasi *Fingerprint* dan Otomasi Perhitungan Gaji Berbasis Jabatan dengan KinderFin” adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, _____

Mahasiswa

Yusuf Hasan Nazila
NRP. 5025211225

Mengetahui
Dosen Pembimbing I

Dosen Pembimbing II

Adhatus Solichah Ahmadiyah, S.Kom,
M.Sc.
NIP. 198508262015042002

Dr. Kelly Rossa Sungkono, S.Kom., M.Kom
NIP. 1994201912088
(Halaman ini sengaja dikosongkan)

STATEMENT OF ORIGINALITY

The undersigned below:

Name of student / NRP : Yusuf Hasan Nazila
Program studi : S-1 Teknik Informatika
Advisor I / NIP : Adhatus Solichah Ahmadiyah, S.Kom, M.Sc. /
198508262015042002
Advisor II / NIP : Dr. Kelly Rossa Sungkono, S.Kom., M.Kom /
1994201912088

Hereby declare that Final Project with the title of " Payroll Module Development: Fingerprint Integration and Automated Salary Calculation Based on Position Using KinderFin " is the result of my own work, is original, and is written by following the rules of scientific writing.

If in the future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with the provisions that apply at Institut Teknologi Sepuluh Nopember.

Surabaya, _____

Student

Yusuf Hasan Nazila
NRP. 5025211225

Acknowledge
Advisor I

Advisor II

Adhatus Solichah Ahmadiyah, S.Kom,
M.Sc.
NIP. 198508262015042002

Dr. Kelly Rossa Sungkono, S.Kom., M.Kom
NIP. 1994201912088

(Halaman ini sengaja dikosongkan)

ABSTRAK

PENGEMBANGAN MODUL PAYROLL: INTEGRASI FINGERPRINT DAN OTOMASI PENGHITUNGAN GAJI BERBASIS JABATAN DENGAN KINDERFIN

Nama Mahasiswa / NRP : Yusuf Hasan Nazila / 5025211225
Departemen : Teknik Informatika FTEIC - ITS
Dosen Pembimbing 1 : Adhatus Solichah Ahmadiyah, S.Kom, M.Sc.
Dosen Pembimbing 2 : Dr. Kelly Rossa Sungkono, S.Kom., M.Kom.

Abstrak

Payroll adalah sistem krusial dalam pengelolaan keuangan organisasi yang berperan memastikan pembayaran gaji dilakukan secara tepat, akurat, dan efisien; mengingat proses manual sering kali menimbulkan risiko kesalahan perhitungan dan memakan waktu lebih lama, sebuah sistem *payroll* berbasis digital telah dikembangkan untuk mengunggah dan mengotomatisasi perhitungan gaji berdasarkan kehadiran, tunjangan, dan potongan. Penelitian ini bertujuan menganalisis dampak penerapan sistem *payroll* otomatis terhadap efisiensi dan akurasi proses penggajian, di mana sistem ini dirancang dengan modul pencatatan kehadiran terintegrasi teknologi *fingerprint* agar data kehadiran pegawai tercatat secara *real-time* dan akurat. Data tersebut kemudian diproses untuk menghitung gaji guru, meliputi komponen seperti tunjangan, insentif, potongan keterlambatan, dan pajak penghasilan, dengan model perhitungan yang diimplementasikan menggunakan algoritma untuk meminimalkan kesalahan serta memastikan transparansi data. Hasil pengujian menunjukkan bahwa sistem *payroll* berbasis digital mampu meningkatkan efisiensi waktu secara signifikan, yaitu dari 30 menit menjadi 5 menit setara sekitar 83,33% dibandingkan metode manual, sekaligus mengurangi kesalahan pencatatan gaji hingga 95%, dan transparansi data memungkinkan guru memverifikasi rincian gaji yang diterima secara langsung.

Kata kunci: *Payroll, Sistem Otomasi, Kehadiran Fingerprint, Efisiensi, Transparansi.*

(Halaman ini sengaja dikosongkan)

ABSTRACT

PAYROLL MODULE DEVELOPMENT: FINGERPRINT INTEGRATION AND AUTOMATED SALARY CALCULATION BASED ON POSITION USING KINDERFIN

Student Name / NRP	: Yusuf Hasan Nazila / 5025211225
Department	: Informatics FTEIC - ITS
Advisor	: Adhatus Solichah Ahmadiyah, S.Kom, M.Sc.
Co - Advisor	: Dr. Kelly Rossa Sungkono, S.Kom., M.Kom.

Abstract

Payroll is an essential system in organizational financial management, ensuring that salary payments are made accurately, precisely, and efficiently. Given that manual *payroll* processes often pose risks of calculation errors and require more time, a digital *payroll* system was developed to automate salary calculations based on attendance, allowances, and deductions. This study aims to analyze the impact of implementing an automated *payroll* system on the efficiency and accuracy of the *payroll* process. The system is designed with an attendance recording module integrated with *fingerprint* technology, allowing employee attendance data to be recorded in *real-time* and accurately. This data is then processed to calculate employee salaries, including components such as allowances, incentives, late deductions, and income tax, with the calculation model implemented using algorithms designed to minimize errors and ensure transparency in *payroll* data. Test results show that the digital *payroll* system can increase time efficiency significantly, from 30 minutes to 5 minutes, equivalent to approximately 83.33% compared to manual methods, while also reducing *payroll* recording errors by up to 95%, and its data transparency allows employees to directly verify the salary details they receive.

Keywords: *Payroll, Automation System, Fingerprint Attendance, Efficiency, Transparency.*

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Segala puji dan syukur penulis haturkan ke hadirat Allah SWT., atas berkat limpahan rahmat dan karunia-Nya, penulis dapat menyelesaikan penelitian tugas akhir yang berjudul “Pengembangan Modul *Payroll*: Integrasi *Fingerprint* dan Otomasi Perhitungan Gaji Berbasis Jabatan dengan KinderFin”. Dalam penyusunan tugas akhir ini, penulis tidak lepas dari berbagai pihak yang telah memberikan dukungan serta bantuannya, mulai dari awal hingga tugas akhir dapat terselesaikan dengan baik. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Allah SWT. yang memberikan petunjuk dan kemudahan sehingga penulis dapat menyelesaikan penelitian tugas akhir ini untuk mencapai gelar strata satu di Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember.
2. Orang tua, Nenek (Simak) dan saudara-saudara penulis yang memberikan dukungan penuh, baik secara fisik dan mental, serta memenuhi segala kebutuhan penulis selama studi di Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember.
3. Ibu Adhatus Solichah Ahmadiyah, S.Kom, M.Sc. dan Ibu Dr. Kelly Rossa Sungkono, S.Kom., M.Kom. sebagai dosen pembimbing yang telah memberikan dukungan berupa bimbingan, arahan, dan masukan berharga selama pengerjaan tugas akhir ini.
4. Pihak KB-TKIT AL-IHSAN Surabaya yang telah bersedia menjadi lokasi penelitian dan memberikan dukungan data, serta kebutuhan terkait sistem *fingerprint* dan informasi mengenai perhitungan gaji berbasis jabatan yang menjadi dasar pengembangan modul *payroll* ini.
5. Bapak Bagus Jati Santoso, S.Kom, Ph.D dan Bapak Dr. Yudhi Purwananto, S.Kom., M.Kom sebagai dosen wali yang senantiasa memberikan arahan, dukungan, dan memfasilitasi kebutuhan akademik penulis selama masa studi di Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember.
6. Dosen dan tenaga pendidik di Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember yang telah memberikan ilmu pengetahuan, wawasan, dan pengalaman yang sangat berarti selama masa studi.
7. Danar Sodik, Arkan Karindra, Farhan Dwi, Syomeron Ansell, Adrian Ismu, Faiz Haq, Faiz Fernanda, I Gusti Agung, Dafarel Fatih, Imanuel, Ken, Dimas Fadilah Akbar, Fathin M, Zien Zidan, Ariel Imata dan rekan-rekan mahasiswa angkatan 2021 Teknik Informatika, Institut Teknologi Sepuluh Nopember, lainnya yang telah memberikan dukungan dan semangat kepada penulis selama masa studi hingga saat ini.

Penulis berharap dengan tugas akhir ini dapat memberikan kontribusi yang bermanfaat terhadap pengembangan ilmu pengetahuan di bidang pengembangan aplikasi berbasis *website*. Penulis juga ingin menyampaikan permohonan maaf atas segala kekurangan dan kesalahan dalam dalam tugas akhir ini.

Surabaya 16 Juni 2025

Yusuf Hasan Nazila

(Halaman ini sengaja dikosongkan)

DAFTAR ISI

LEMBAR PENGESAHAN	i
APPROVAL SHEET	iii
PERNYATAAN ORISINALITAS	v
STATEMENT OF ORIGINALITY	vii
ABSTRAK	ix
ABSTRACT	xi
KATA PENGANTAR	xiii
DAFTAR ISI	xv
DAFTAR GAMBAR	xix
DAFTAR TABEL	xxiii
DAFTAR KODE SEMU	xxv
DAFTAR KODE SUMBER	xxvii
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah	2
1.4 Tujuan	2
1.5 Manfaat	3
BAB 2 TINJAUAN PUSTAKA	5
2.1 Penelitian Terkait	5
2.2 Aplikasi Sejenis	7
2.3 KinderFin	8
2.4 Dasar Teori	9
2.4.1 Sistem <i>Payroll</i>	9
2.4.2 <i>Fingerprint</i>	9
2.4.3 <i>Fingerprint System</i> (Sistem <i>Fingerprint</i>)	9
2.4.4 <i>Framework Pengembangan</i>	9
2.4.5 <i>User Interface (UI)</i> dan <i>User Experience (UX)</i>	10
2.4.6 Arsitektur Web	10
2.4.7 Keamanan Data	10
2.4.8 <i>REST API</i>	10
2.4.9 <i>Clean Architecture</i>	11

2.4.10	Interoperabilitas	11
BAB 3	METODE PENELITIAN	13
3.1	Analisis	13
3.1.1	Analisis Permasalahan	14
3.1.2	Analisis Proses Bisnis.....	14
3.1.3	Analisis Kebutuhan.....	15
3.1.4	Spesifikasi Kebutuhan Perangkat Lunak	19
3.2	Desain	30
3.2.1	Desain Arsitektur Sistem	30
3.2.2	Desain Antarmuka	32
3.2.3	Desain Algoritma Perhitungan Gaji	37
3.2.4	Desain Basis Data	41
3.3	Implementasi	54
3.3.1	Lingkungan Implementasi	54
3.3.2	Implementasi Antarmuka Pengguna.....	55
3.3.3	Implementasi Integrasi Aplikasi KinderFin dengan Mesin <i>Fingerprint</i> ..	66
3.3.4	Implementasi Aplikasi KinderFin	69
3.3.5	Implemetasi Integrasi <i>Frontend</i> dan <i>Backend</i>	104
3.4	Pengujian	105
3.4.1	Pengujian Fungsionalitas	105
3.4.2	Pengujian Interoperabilitas	105
3.5	Instalasi.....	108
3.5.1	Arsitektur Instalasi.....	108
3.5.2	Instalasi <i>Backend</i>	109
3.5.3	Instalasi <i>Frontend</i>	109
3.5.4	Konfigurasi Basis Data dan <i>Environment</i>	109
BAB 4	HASIL DAN PEMBAHASAN	111
4.1	Hasil Uji Coba Fungsional	111
4.1.1	Pengujian Oleh Administrator	111
4.1.2	Pengujian Oleh Bendahara	113
4.1.3	Pengujian Oleh Guru	125
4.2	Hasil Uji Coba Non Fungsional.....	127
4.3	Pembahasan Hasil Uji Coba Fungsional	129
4.3.1	Evaluasi Hasil Uji Coba Bendahara	130

4.3.2	Evaluasi Hasil Uji Coba Guru	131
4.4	Pembahasan Hasil Uji Coba Interoperabilitas	131
BAB 5	Kesimpulan dan Saran	133
5.1	Kesimpulan	133
5.2	Saran	134
	DAFTAR PUSTAKA	135
	LAMPIRAN – LAMPIRAN	137
	LAMPIRAN A	137
	LAMPIRAN B	139
	LAMPIRAN C	142
	LAMPIRAN D	145
	LAMPIRAN E	147
	LAMPIRAN F	150
	LAMPIRAN G	165
	BIODATA PENULIS	170

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

Gambar 3.1 Alur Diagram Pengembangan Modul Penggajian Aplikasi KinderFin 2.0	13
Gambar 3.2 Proses Bisnis Modul Penggajian Aplikasi KinderFin 2.0	14
Gambar 3.3 Diagram <i>Use Case</i> Modul Penggajian Aplikasi KinderFin 2.0.....	16
Gambar 3.4 Diagram Aktivitas Alur Proses Bisnis Modul Penggajian Aplikasi KinderFin 2.0	17
Gambar 3.5 Diagram Aktivitas Tambah Data Presensi.....	22
Gambar 3.6 Diagram Aktivitas Validasi Gaji	24
Gambar 3.7 Diagram Aktivitas Cetak Gaji	26
Gambar 3.8 Diagram Aktivitas Tambah Bonus	28
Gambar 3.9 Diagram Aktivitas Lihat Rincian Gaji	29
Gambar 3.10 Desain Antarmuka Halaman Utama Bendahara	32
Gambar 3.11 Desain Halaman Pengaturan Gaji	32
Gambar 3.12 Desain Antarmuka Halaman Fitur <i>Setting</i>	33
Gambar 3.13 Desain Antarmuka Halaman Rincian Gaji Pegawai Daftar Pegawai pada Modul Bendahara.....	33
Gambar 3.14 Desain Antarmuka Halaman Rincian Gaji Setiap Pengguna.....	34
Gambar 3.15 Desain Antarmuka Halaman Master Jabatan.....	34
Gambar 3.16 Desain Antarmuka Halaman Formulir Perubahan Keuangan	35
Gambar 3.17 Desain Antarmuka Halaman Log Riwayat Keuangan	35
Gambar 3.18 Desain Halaman Utama Pengajuan Perubahan Gaji	36
Gambar 3.19 Desain Antarmuka Detail Setiap Pengajuan Perubahan Gaji	36
Gambar 3.20 Desain Antarmuka Halaman Utama <i>Dashboard</i> Guru.....	36
Gambar 3.21 Desain Antarmuka Halaman Detail Gaji Guru	37
Gambar 3.22 Diagram <i>Physical Data Model</i> (PDM) atau Skema Basis Data Aplikasi KinderFin 2.0 Modul Penggajian.....	42
Gambar 3.23 Tampilan Tambah Perbarui Peran Pengguna dan Hapus Pengguna.....	55
Gambar 3.24 Tampilan Halaman Pengaturan Gaji Aktif	56
Gambar 3.25 Tampilan Awal Master Jabatan	56
Gambar 3.26 Tampilan Menambah Komponen Gaji Setiap Jabatan	57
Gambar 3.27 Detail Rincian Gaji Setiap Jabatan	57
Gambar 3.28 Tampilan Sunting Besaran Komponen Gaji Setiap Jabatan	57
Gambar 3.29 Tampilan Halaman Awal Potongan Gaji	58
Gambar 3.30 Tampilan Menambah Potongan Gaji	59
Gambar 3.31 Tampilan Halaman Utama Rekap Gaji Pegawai	59
Gambar 3.32 Tampilan Data Sedang Diproses	60
Gambar 3.33 Tampilan Data Selesai Diproses	60
Gambar 3.34 Contoh format pengisian data Presensi manual yang memuat tanggal, jam masuk, dan jam keluar	61
Gambar 3.35 Data Gaji belum Menyertakan Jam Pulang	61
Gambar 3.36 Data Gaji dengan Menyertakan Jam Pulang (Sudah Diperbarui)	61
Gambar 3.37 Hasil Gaji Untuk di Verifikasi	62
Gambar 3.38 Tampilan Saat Memproses Gaji untuk Interval Tertentu	62
Gambar 3.39 Tampilan Saat Jika Ingin Memasukkan Bonus	62

Gambar 3.40 Gambar Nota Pembayaran Untuk Setiap Pengguna	63
Gambar 3.41 Hasil seluruh Gaji dan Seluruh Rekap Bonus.....	63
Gambar 3.42 Tampilan Daftar Pengajuan Perubahan Gaji pada Peran Guru	64
Gambar 3.43 Tampilan Halaman Pengajuan Perubahan Gaji pada Peran Guru	64
Gambar 3.44 Tampilan Daftar Pengajuan Perubahan Gaji pada Peran Bendahara.....	65
Gambar 3.45 Tampilan Detail Gaji dan Seluruh Rekap Bonus Pribadi Guru	66
Gambar 3.46 Tampilan Log Aktivitas	66
Gambar 3.47 Alur Kerja <i>API process</i> dan <i>fetch</i>	68
Gambar 3.48 Contoh <i>File</i> dari Mesin Presensi <i>Fingerprint</i>	91
Gambar 3.49 Ilustrasi Integrasi <i>Frontend</i> dan <i>Backend</i> Aplikasi KinderFin 2.0	105
Gambar 3.50 <i>Create New Collection</i> pada Aplikasi <i>Postman</i>	106
Gambar 3.51 Proses untuk Login	106
Gambar 3.52 Contoh Balasan Sistem Jika Pengguna berhasil <i>login</i>	107
Gambar 3.53 Proses <i>input access_token</i> setelah login	107
Gambar 4.1 Tampilan Sebelum Data Presensi Masuk	128
Gambar 4.2 Proses Memasukkan Data Presensi Melalui Aplikasi <i>Postman</i> menggunakan <i>API process</i>	128
Gambar 4.3 Tampilan Setelah Data Presensi Masuk	129
Gambar 4.4 Pengujian Pengambilan Data Presensi Menggunakan <i>API fetch</i>	129
Gambar A.1 Mesin <i>Fingerprint</i> pada <i>server developer.fingerspot.io</i>	137
Gambar A.2 Mesin <i>Fingerprint</i> pada Lokasi pengetesan (TK Al Ihsan Surabaya)	138
Gambar A.3 Tampilan Perintah Tertunda di <i>Server developer.fingerspot.io</i>	138
Gambar B.1 Menu Awal Mesin <i>fingerprint</i>	139
Gambar B.2 Halaman Opsi pada mesin <i>fingerprint</i>	139
Gambar B.3 Halaman Jaringan pada mesin <i>fingerprint</i>	140
Gambar B.4 Halaman <i>WLAN</i> pada mesin <i>fingerprint</i>	140
Gambar B.5 Nama <i>Server</i> dan <i>Port</i> pada <i>developer.fingerspot.io</i>	141
Gambar C.1 Detail Gaji Harian dan Pokok Jabatan Guru	142
Gambar C.2 Detail Potongan Gaji Jabatan	142
Gambar C.3 Detail Gaji Harian dan Pokok Jabatan Bendahara	143
Gambar C.4 Bukti Detail Gaji Peran Guru	143
Gambar C.5 Bukti Detail Gaji Peran Bendahara	144
Gambar D.1 Bukti Administrator Berhasil Menambahkan Pengguna atau <i>Users</i> di Aplikasi KinderFin 2.0	145
Gambar D.2 Bukti Bendahara Melakukan Penarikan Data Presensi dari Aplikasi KinderFin 2.0	145
Gambar D.3 Bukti Bendahara Melakukan Unggah <i>File</i> Presensi di Aplikasi KinderFin 2.0	146
Gambar D.4 Bukti Guru Berhasil Mengajukan dan Dapat melihat Ajuan Perubahan Gaji di Aplikasi KinderFin 2.0	146
Gambar E.1 Pengisian Kuesioner Oleh Pengguna Uji Coba Peran Administrator	147
Gambar E.2 Pengisian Kuesioner Oleh Pengguna Uji Coba Peran Bendahara.....	148
Gambar E.3 Pengisian Kuesioner Oleh Pengguna Uji Coba Peran Guru	149
Gambar F.1 Diagram Aktivitas Menambah Pengguna.....	151
Gambar F.2 Diagram Aktivitas Edit dan Hapus Pengguna	152
Gambar F.3 Diagram Aktivitas Aktifkan Label Gaji	154

Gambar F.4 Diagram Aktivitas Tambah atau Ubah Gaji Harian	156
Gambar F.5 Diagram Aktivitas Pengaturan Potongan Gaji Harian.....	158
Gambar F.6 Diagram Aktivitas Setujui Perubahan Gaji	161
Gambar F.7 Diagram Aktivitas Penggunaan Lihat Log Aktivitas	162
Gambar F.8 Diagram Aktivitas Ajukan Perubahan Gaji	164

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

Tabel 2.1 Penelitian Terkait.....	5
Tabel 2.2 Perbandingan Aplikasi Talenta dengan Aplikasi Kindefin 2.0	7
Tabel 2.3 Perbedaan Fitur Perhitungan Gaji pada KinderFin 1.0 dengan KinderFin 2.0	8
Tabel 3.1 Penjelasan Proses Bisnis Modul Penggajian Aplikasi KinderFin 2.0	15
Tabel 3.2 Daftar Kebutuhan Administrator	18
Tabel 3.3 Daftar Kebutuhan Bendahara	18
Tabel 3.4 Daftar Kebutuhan Guru	19
Tabel 3.5 Kasus Penggunaan Tambah Data Presensi.....	20
Tabel 3.6 Kasus Penggunaan Validasi Gaji.....	23
Tabel 3.7 Kasus Penggunaan Cetak Gaji.....	24
Tabel 3.8 Kasus Penggunaan Tambah Bonus	27
Tabel 3.9 Kasus Penggunaan Lihat Rincian Gaji	29
Tabel 3.10 Struktur Tabel <i>activity_logs</i>	43
Tabel 3.11 Struktur Tabel <i>master_jabatan</i>	43
Tabel 3.12 Struktur Tabel <i>master_jabatan_pokok</i>	45
Tabel 3.13 Struktur Tabel <i>detail_salary</i>	47
Tabel 3.14 Struktur Tabel <i>pengajuan_perubahan_gaji</i>	50
Tabel 3.15 Struktur Tabel <i>pengaturan_gaji_aktif</i>	52
Tabel 3.16 Struktur Tabel <i>potongan_keterlambatan</i>	52
Tabel 3.17 Struktur Tabel <i>rekap_bonus</i>	53
Tabel 3.18 Spesifikasi Perangkat Keras Lingkungan Implementasi	54
Tabel 3.19 Spesifikasi Perangkat Lunak Lingkungan Implementasi	54
Tabel 3.20 Contoh Potongan Gaji	58
Tabel 4.1 Daftar Pengguna Ujicoba	111
Tabel 4.2 Hasil Uji <i>Use Case</i> Kasus Pengujian Menambah Pengguna	111
Tabel 4.3 Hasil Uji <i>Use Case</i> Kasus Pengujian Menambah Pengguna dengan Data Duplikat	112
Tabel 4.4 Hasil Uji <i>Use Case</i> Kasus Pengujian Edit Pengguna	112
Tabel 4.5 Hasil Uji <i>Use Case</i> Kasus Hapus Pengguna.....	113
Tabel 4.6 Hasil Uji <i>Use Case</i> Kasus Pengujian Aktifkan Label Gaji	114
Tabel 4.7 Hasil Uji <i>Use Case</i> Kasus Pengujian Aktifkan Label Gaji dengan Nama Tabel Duplikat	114
Tabel 4.8 Hasil Uji <i>Use Case</i> Kasus Pengujian Tambah Gaji Harian dan Pokok.....	115
Tabel 4.9 Hasil Uji <i>Use Case</i> Kasus Pengujian Perbarui Gaji Harian dan Pokok	115
Tabel 4.10 Hasil Uji <i>Use Case</i> Kasus Pengujian Hapus Gaji Harian dan Pokok.....	116
Tabel 4.11 Hasil Uji <i>Use Case</i> Kasus Pengujian Tambah Gaji Harian dan Pokok dengan Nama Jabatan sama dengan yang Tersedia di Sistem.....	116
Tabel 4.12 Hasil Uji <i>Use Case</i> Kasus Pengujian Tambah Potongan Gaji Harian.....	117
Tabel 4.13 Hasil Uji <i>Use Case</i> Kasus Pengujian Tambah Potongan Gaji Harian dengan detail Potongan Sudah Tersedia dalam Sistem (Duplikat)	117
Tabel 4.14 Hasil Uji <i>Use Case</i> Kasus Pengujian Hapus Potongan Gaji Harian	118
Tabel 4.15 Hasil Uji <i>Use Case</i> Kasus Pengujian Tambah Data Presensi dengan Mengambil Langsung Dari Server <i>developer.fingerspot.io</i>	119

Tabel 4.16 Hasil Uji <i>Use Case</i> Kasus Pengujian Tambah Data Presensi dengan Unggah <i>File</i>	119
Tabel 4.17 Hasil Uji <i>Use Case</i> Kasus Pengujian Tambah Data Presensi dengan Unggah <i>File</i> Terproses Sebelumnya	120
Tabel 4.18 Hasil Uji <i>Use Case</i> Kasus Pengujian Tambah Data Presensi dengan <i>Input Manual</i>	120
Tabel 4.19 Hasil Uji <i>Use Case</i> Kasus Pengujian Tambah Data Presensi dengan <i>Input Manual</i> dengan Tanggal Sama.....	121
Tabel 4.20 Hasil Uji <i>Use Case</i> Kasus Pengujian Tambah Data Presensi dengan Unggah <i>File</i> dengan Format Tidak Sesuai dengan Mesin <i>Fingerprint</i>	121
Tabel 4.21 Hasil Uji <i>Use Case</i> Kasus Pengujian Validasi Gaji.	122
Tabel 4.22 Hasil Uji <i>Use Case</i> Kasus Pengujian Validasi Gaji untuk Menghapus Detail Gaji.....	122
Tabel 4.23 Hasil Uji <i>Use Case</i> Kasus Pengujian Cetak Gaji.	123
Tabel 4.24 Hasil Uji <i>Use Case</i> Kasus Pengujian Setujui Perubahan Gaji	123
Tabel 4.25 Hasil Uji <i>Use Case</i> Kasus Pengujian Menolak Perubahan Gaji.....	124
Tabel 4.26 Hasil Uji <i>Use Case</i> Kasus Pengujian Tambah Bonus	124
Tabel 4.27 Hasil Uji <i>Use Case</i> Kasus Lihat Log Aktivitas	125
Tabel 4.28 Hasil Uji <i>Use Case</i> Kasus Pengujian Lihat Rincian Gaji.....	126
Tabel 4.29 Hasil Uji <i>Use Case</i> Kasus Pengujian Ajukan Perubahan Gaji	126
Tabel 4.30 Hasil Uji <i>Use Case</i> Kasus Pengujian Ajukan Perubahan Gaji dengan Data Tidak Lengkap	127
Tabel 4.31 Hasil Kuesioner Pengujian oleh Pengguna Uji Coba 1 sebagai Administrator....	130
Tabel 4.32 Hasil Kuesioner Pengujian oleh Pengguna Uji Coba 2 sebagai Bendahara	130
Tabel 4.33 Perbandingan Waktu Antara Rekap Kehdiran Manual dan KinderFin Terintegrasi <i>Fingerprint</i>	131
Tabel 4.34 Hasil Kuesioner Pengujian oleh Pengguna Uji Coba 2 sebagai Guru	131
Tabel F.1 Kasus Penggunaan Menambah Pengguna.....	150
Tabel F.2 Kasus Penggunaan Edit dan Hapus Pengguna	151
Tabel F.3 Kasus Penggunaan Aktifkan Label Gaji	153
Tabel F.4 Kasus Penggunaan Tambah atau Ubah Gaji Harian	155
Tabel F.5 Kasus Penggunaan Pengaturan Potongan Gaji Harian.....	157
Tabel F.6 Kasus Penggunaan Setujui Perubahan Gaji	159
Tabel F.7 Kasus Penggunaan Lihat Log Aktivitas	162
Tabel F.8 Kasus Penggunaan Ajukan Perubahan Gaji	163

DAFTAR KODE SEMU

Kode Semu 3.1 Algoritma Persiapan Data <i>Input</i>	37
Kode Semu 3.2 Pemrosesan Jam Masuk dan Deteksi Keterlambatan	39
Kode Semu 3.3 Pemrosesan Jam Pulang dan Deteksi Pulang Cepat	40
Kode Semu 3.4 Penghitungan Gaji <i>Final</i>	41

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

Kode Sumber 3.1 Struktur <i>Code</i> Data Presensi dari Sever <i>developer.fingerspot.io</i>	67
Kode Sumber 3.2 <i>Code processAttendanceData</i> Penerimaan Data Melalui <i>Input Manual</i>	67
Kode Sumber 3.3 <i>Code fetchAndProcessAttendance</i> yang secara aktif mengambil data dari <i>API Fingerspot</i>	68
Kode Sumber 3.4 <i>Code processFingerspotRecords</i> Pengambilan data langsung dari <i>API</i> <i>Fingerspot</i>	69
Kode Sumber 3.5 Contoh <i>Code Router</i>	70
Kode Sumber 3.6 <i>Code execute SavePengaturanGajiAktifCommand</i>	71
Kode Sumber 3.7 <i>Code execute DeletePengaturanGajiAktifCommand</i>	71
Kode Sumber 3.8 <i>Code execute GetPengaturanGajiAktifQuery</i>	71
Kode Sumber 3.9 <i>Code Model PengaturanGajiAktif</i>	72
Kode Sumber 3.10 <i>Code getAll Repository</i>	73
Kode Sumber 3.11 <i>Code save Repository</i>	73
Kode Sumber 3.12 <i>Code deleteByField Repository</i>	74
Kode Sumber 3.13 <i>Code get Controller</i>	74
Kode Sumber 3.14 <i>Code save Controller</i>	74
Kode Sumber 3.15 <i>Code delete Controller</i>	75
Kode Sumber 3.16 <i>Code createFull</i>	75
Kode Sumber 3.17 <i>Code getAll Master Jabatan</i>	76
Kode Sumber 3.18 <i>Code update Master Jabatan</i>	77
Kode Sumber 3.19 <i>Code delete Master Jabatan</i>	77
Kode Sumber 3.20 <i>Code Fungsi validateNonNegativeValues Master Jabatan</i>	78
Kode Sumber 3.21 <i>Code Sinkronisasi Data Antar Tabel</i>	78
Kode Sumber 3.22 <i>Code Model MasterJabatan</i>	79
Kode Sumber 3.23 <i>Code Model MasterJabatanPokok</i>	79
Kode Sumber 3.24 <i>Code Command Pattern Master Jabatan</i>	80
Kode Sumber 3.25 <i>Code query Pattern</i> untuk Pembacaan Data Master Jabatan.....	80
Kode Sumber 3.26 <i>Code Fungsi create Potongan Keterlambatan</i>	81
Kode Sumber 3.27 <i>Code Pola Command</i> Potongan Keterlambatan	81
Kode Sumber 3.28 <i>Code Fungsi delete Potongan Keterlambatan</i>	82
Kode Sumber 3.29 <i>Code Fungsi findAll dan findById</i> Potongan Keterlambatan.....	82
Kode Sumber 3.30 <i>Code Fungsi findBestPotongan</i> Potongan Keterlambatan	83
Kode Sumber 3.31 Definisi Entitas <i>Activity Log</i>	83
Kode Sumber 3.32 <i>Code Repository Activity Log</i>	84
Kode Sumber 3.33 Repotori dengan <i>Sequelize Activity Log</i>	84
Kode Sumber 3.34 <i>Code CreateLogCommand Activity Log</i>	85
Kode Sumber 3.35 <i>Code Controller Activity Log</i>	85
Kode Sumber 3.36 Migrasi Tabel <i>Activity Log</i>	86
Kode Sumber 3.37 <i>Code calculateSalary Salary Detail</i>	87
Kode Sumber 3.38 <i>Code getBestPotongan Salary Detail</i>	88
Kode Sumber 3.39 <i>Code saveOrUpdateSalaryDetail Salary Detail</i>	88
Kode Sumber 3.40 <i>Code getAllSalaryByTeacher Salary Detail</i>	89

Kode Sumber 3.41 <i>Code processFinalSalary Salary Detail</i>	89
Kode Sumber 3.42 <i>Code generateFinalPdf Salary Detail</i>	90
Kode Sumber 3.43 <i>Code readExcelFile</i> untuk Pembacaan <i>File Excel</i>	91
Kode Sumber 3.44 <i>Code Fungsi validateExcelSheet (upload.controller.ts)</i>	92
Kode Sumber 3.45 <i>Code generateFinalSalaryPDF</i>	92
Kode Sumber 3.46 <i>Code get-salary-detail.query.ts</i>	93
Kode Sumber 3.47 <i>Code getMySalaryDetail</i> untuk Pengambilan Detail Gaji Guru	94
Kode Sumber 3.48 <i>Code Fungsi Unggah dan Sinkronisasi Data Kehadiran</i>	94
Kode Sumber 3.49 <i>Code Fungsi Manajemen Gaji Guru (Input, Proses, Hapus)</i>	94
Kode Sumber 3.50 <i>Code execute CreateRekapBonusCommand</i> Rekap Bonus	95
Kode Sumber 3.51 <i>Code execute GetRekapBonusByNipQuery</i> Rekap Bonus	96
Kode Sumber 3.52 <i>Code createRekap</i> dan <i>getRekapByNip</i> Rekap Bonus	96
Kode Sumber 3.53 <i>Code create</i> dan <i>findByNip</i> Rekap Bonus	97
Kode Sumber 3.54 <i>Code RekapBonus Model Entity</i>	97
Kode Sumber 3.55 <i>Code create RekapBonusController</i>	97
Kode Sumber 3.56 <i>Code getByNip RekapBonusController</i>	98
Kode Sumber 3.57 <i>Code ajukan</i>	99
Kode Sumber 3.58 <i>Code getAll</i>	99
Kode Sumber 3.59 <i>Code updateStatus</i>	100
Kode Sumber 3.60 <i>Code updateStatus Repository</i>	101
Kode Sumber 3.61 <i>Code createPengajuan</i>	101
Kode Sumber 3.62 <i>Code ajukanPerubahan Service</i>	101
Kode Sumber 3.63 <i>Code getPengajuan Service</i>	102
Kode Sumber 3.64 <i>Code PengajuanPerubahanGaji Model Entity</i>	103
Kode Sumber 3.65 <i>Code Upload Middleware</i>	103
Kode Sumber 3.66 <i>Code Pengelolaan State Tampilan Modul Users</i>	103
Kode Sumber 3.67 <i>Code Pengambilan Data Awal</i>	104
Kode Sumber 3.68 <i>Code Fungsi Handler</i> untuk Aksi Pengguna	104
Kode Sumber 3.69 <i>Code Render Antarmuka Pengguna (JSX)</i>	104
Kode Sumber 3.70 <i>URL</i> untuk simulasi mengirimkan data presensi	107
Kode Sumber 3.71 Contoh Balasan Pesan Berhasil serta Data Presensi dari <i>Server developer.fingerspot.io</i>	108
Kode Sumber 3.72 <i>Request Body</i> Perintah Pengiriman Data Presensi	108
Kode Sumber 3.73 <i>URL Endpoint API</i> untuk Pengambilan, Pengolahan, dan Integrasi Data <i>Fingerspot</i>	108
Kode Sumber 3.74 <i>URL</i> pengambilan Data dari Sever <i>developer.fingerspot.io</i>	108
Kode Sumber 3.75 Perintah Instalasi <i>Backend</i>	109
Kode Sumber 3.76 Perintah membangun Program <i>Backend</i>	109
Kode Sumber 3.77 Perintah menjalankan Program <i>Backend</i>	109
Kode Sumber 3.78 Perintah Instalasi <i>Frontend</i>	109
Kode Sumber 3.79 Perintah membangun Program <i>Frontend</i>	109
Kode Sumber 3.80 Perintah menjalankan Program <i>Frontend</i>	109
Kode Sumber 3.81 <i>Environment Variables Backend</i>	110
Kode Sumber 3.82 <i>Environment Variables Frontent</i>	110
Kode Sumber G.1 Contoh <i>Code Pengelolaan State Lokal</i>	165
Kode Sumber G.2 <i>Code Penanganan Autentikasi dan Pengambilan Data Pengguna Awal</i> ..	165

Kode Sumber G.3 Contoh <i>Code</i> Pengambilan Data dari Server	166
Kode Sumber G.4 Contoh <i>Code</i> Navigasi Halaman	166
Kode Sumber G.5 <i>Code</i> Validasi dan Penghapusan Field Gaji	167
Kode Sumber G.6 <i>Code</i> Hapus Jabatan dan Tombol Aksi.....	167
Kode Sumber G.7 <i>Code</i> Utilitas Formatting Angka.....	167
Kode Sumber G.8 <i>Code</i> Fungsi Simpan dan Validasi Input Gaji	168
Kode Sumber G.9 <i>Code</i> Fungsi Pengelompokan Aturan Potongan Berdasarkan Jabatan	168
Kode Sumber G.10 <i>Code</i> Logika Kondisional Input Batas Menit	168
Kode Sumber G.11 <i>Code</i> Tampilan Tabel Log Aktivitas	169
Kode Sumber G.12 <i>Code</i> Fungsi Mekanisme Persetujuan atau Penolakan Pengajuan Gaji..	169
Kode Sumber G.13 <i>Code</i> Fungsi Pengunggahan <i>File</i> Gambar pada Formulir Pengajuan....	169

(Halaman Ini Sengaja Dikosongkan)

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Payroll merupakan sistem pengomunikasian yang dilakukan perusahaan secara rutin kepada pegawainya dalam jangka waktu tertentu. *Payroll* memiliki sistem yang berbeda beda berdasarkan kebijakan masing masing perusahaan. Besaran nominal *payroll* dihitung berdasarkan kinerja yang diberikan oleh individu pegawai ke perusahaan. Namun, tidak hanya kinerja yang berpengaruh, adapun yang dapat menjadi perhitungan suatu perusahaan untuk memberikan besaran nominal *payroll* tersebut yaitu kehadiran pegawai. Kehadiran dihitung mayoritas menggunakan sistem *fingerprint*. *Fingerprint* yaitu teknologi biometrik yang menggunakan pola sidik jari untuk mengidentifikasi seseorang. Biometrik tersebut dinilai sangat efektif untuk mengurangi tingkat kecurangan karena sidik jari atau *fingerprint* setiap individu berbeda.

Dalam sistem *payroll* kehadiran merupakan sistem yang dapat dikatakan krusial atau penting karena kehadiran juga dapat menentukan kinerja suatu individu. Suatu perusahaan dapat memiliki ratusan bahkan ribuan pegawai. Untuk memudahkan dalam penghitungan nominal besaran nominal yang diberikan kepada pegawai, sistem *fingerprint* memudahkan untuk mencatat kehadiran pegawai. Namun, sistem tersebut masih hanya dapat menyimpan dan mencetak kehadiran di *spreadsheet*. Tujuan dari sistem yang akan dirancang ini adalah untuk mengintegrasikan antara sistem *payroll* dan *fingerprint* agar memudahkan penghitungan nominal yang harus dikeluarkan setiap pegawai. Transparansi dalam suatu perusahaan terutama dalam lembaga pendidikan adalah kunci yang sangat perlu diperhatikan, terutama transparansi terhadap keuangan. Transparansi keuangan tidak hanya penting dalam kredibilitas suatu lembaga pendidikan dan reputasi terhadap perusahaan tersebut, tetapi berperan dalam hal pertanggungjawaban semua aktivitas keuangan terhadap pihak yang berwenang. Dalam hal ini, transparansi berarti menyampaikan laporan keuangan dan kejelasan terhadap keuangan dengan lengkap tanpa adanya manipulasi dan menyembunyikan data yang dapat membuat data keuangan tidak asli. Transparansi di lembaga pendidikan sangat risikan terhadap kecurangan, dengan itu diberikan pengawasan yang sangat ketat. Namun, keterbatasan sumber daya manusia untuk pengawasan merupakan masalah baru dalam hal ini. Maka diperlukan pengawasan yang berbasis teknologi dengan sistem yang otomatis dalam penghitungan gaji di setiap individu dengan akurat.

Pada saat ini, sistem untuk penggajian untuk guru sudah ada, namun dilakukan dengan manual dengan hanya *upload* atau unggah gambar bukti transfer yang sudah dilakukan oleh pihak administratif sebagai tanpa pembayaran. Pihak administratif memiliki akses langsung untuk *input* data dan penghitungan manual tanpa verifikasi dan tanpa rincian pembayaran gaji. Sistem saat ini sangat risikan untuk terjadinya manipulasi komponen gaji lainnya. Dari sisi guru yang menerima gaji, transparansi sangat perlu karena transparansi keuangan merupakan bukti yang dibayarkan untuk kewajiban guru sebagai pengajar Keuntungan dari otomasi ini untuk guru adalah untuk memantau dan memverifikasi rincian gaji yang mereka terima, termasuk komponen-komponen seperti kehadiran, tunjangan, dan insentif, sehingga mereka lebih yakin bahwa perhitungan gaji dilakukan secara akurat dan adil. Dari sisi administratif transparansi merupakan tanggung jawab yang diberikan pihak administratif untuk membayarkan hak kepada pengajar sebagai komponen utama dalam suatu lembaga pendidikan. Keuntungan dari pihak administratif yaitu memudahkan dengan hanya *input* di awal untuk penghitungan otomatis pada

setiap komponen pengajar, bendahara ataupun komponen di lembaga tersebut. Namun, ada pihak lain yang sangat berpengaruh dalam lembaga pendidikan terutama lembaga pendidikan swasta yaitu, pemilik yayasan lembaga pendidikan. Pihak pemilik yayasan sangat diuntungkan dengan adanya otomasi rincian pemilik dapat memantau secara langsung keuntungan pendapatan dan pembagian hasil.

Pengembangan pada tugas akhir ini akan dikembangkan untuk otomasi rincian pembayaran gaji untuk guru dari proyek sebelumnya bernama KinderFin. KinderFin adalah program untuk penggajian guru. Saat ini, *KinderFin* hanya dapat *Input Manual* dan *upload* bukti untuk pembayaran gaji dan *role* yang ada hanyalah guru. Tugas akhir ini, akan membuat fitur tambahan untuk *input role* dan penghitungan gaji diawal yang akan selanjutnya dilakukan otomasi pembayaran gaji yang diintegrasikan dengan sistem *fingerprint* untuk salah satu komponen gaji yaitu kehadiran. Fitur lainnya adalah pembagian *role* untuk pembagian yang lebih rinci seperti kepala sekolah, guru, bendahara serta komponen dalam lembaga tersebut.

1.2 Rumusan Masalah

1. Bagaimana mengintegrasikan sistem *Payroll* Aplikasi KinderFin dengan sistem *fingerprint*?
2. Bagaimana menghitung gaji secara otomatis berdasarkan jabatan, sehingga sesuai dengan aturan penggajian yang ditentukan oleh sekolah?
3. Bagaimana hasil evaluasi efisiensi waktu Bendahara dalam pengelolaan gaji antara penggunaan sistem *payroll* Aplikasi KinderFin versi lama dan Aplikasi KinderFin 2.0 yang terintegrasi dengan sistem *fingerprint*?
4. Bagaimana hasil evaluasi sistem *payroll* yang diintegrasikan dengan sistem *fingerprint*?

1.3 Batasan Masalah

1. Sistem hanya menggunakan data kehadiran yang dicatat melalui *fingerprint* sebagai dasar perhitungan *payroll*, sehingga kehadiran yang dicatat secara manual atau dengan metode lain tidak disertakan dalam perhitungan.
2. Ruang lingkup sistem ini terbatas pada penghitungan *payroll* berdasarkan kehadiran yang dicatat melalui *fingerprint* dan fleksibilitas kustomisasi item pembentuk gaji, seperti tunjangan, pinjaman, atau komponen lain yang dapat diatur sesuai kebutuhan sekolah.
3. Mesin *fingerprint* yang digunakan bermerek “*Fingerspot*” dengan varian “Revo WF-206BNC”.

1.4 Tujuan

1. Mengembangkan sistem *payroll* yang terintegrasi dengan sistem *fingerprint* untuk mempermudah perhitungan nominal berdasarkan data kehadiran pegawai.
2. Meminimalisasi kesalahan dalam penghitungan nominal pegawai melalui sistem otomatis yang memanfaatkan data biometrik kehadiran.
3. Meningkatkan efisiensi pengelolaan data pegawai dengan mengintegrasikan catatan kehadiran dan perhitungan nominal dalam satu sistem.

1.5 Manfaat

Manfaat yang diharapkan dari tugas akhir ini adalah transparansi dan efisiensi yang lebih baik bagi seluruh pihak yang terlibat. Bagi guru, sistem ini memungkinkan mereka untuk memantau dan memverifikasi rincian gaji secara transparan, termasuk komponen seperti kehadiran, tunjangan, dan insentif, sehingga memastikan perhitungan gaji dilakukan secara akurat dan adil. Dari sisi administratif, otomasi ini memudahkan proses *input* data, memungkinkan penghitungan komponen gaji secara otomatis, dan mengurangi risiko manipulasi data. Selain itu, pemilik yayasan dapat memantau keuntungan dan pembagian hasil secara langsung, sehingga dapat memastikan pengelolaan keuangan yang lebih transparan dan bertanggung jawab dalam lembaga pendidikan. Integrasi sistem *payroll* dengan *fingerprint* membantu mencegah kesalahan dan kecurangan dalam penghitungan nominal secara otomatis. Sistem dapat memastikan data kehadiran yang akurat dan tidak dapat dimanipulasi, sehingga penghitungan gaji berdasarkan Presensi menjadi lebih valid dan dapat dipertanggungjawabkan.

(Halaman ini sengaja dikosongkan)

BAB 2 TINJAUAN PUSTAKA

Pada ini berisi mengenai penelitian terkait yang berisi mengenai penelitian yang sebelumnya yang relevan dengan Tugas Akhir ini. Sub-bab lainnya yaitu tinjauan terhadap penelitian-penelitian sebelumnya yang relevan, analisis perbandingan dengan aplikasi sejenis, serta penjelasan mengenai pengembangan Aplikasi KinderFin 2.0 yang menjadi fokus utama dalam Tugas Akhir ini.

2.1 Penelitian Terkait

Dalam proses tugas akhir ini, informasi dari penelitian sebelumnya dikumpulkan untuk digunakan sebagai pembanding dan referensi saat melakukan pengujian, yang tersaji dalam Tabel 2.1. Sumber informasi juga diperoleh dari jurnal, buku, dan skripsi yang relevan dengan judul Tugas Akhir ini guna memperoleh dasar teoritis yang kuat.

Tabel 2.1 Penelitian Terkait

Judul	Hasil Penelitian
Sistem Presensi yang Terintegrasi dengan Proses Penggajian (Musdalifa, Rosmawati Tamin, Andi Emil Multazam, 2020)	Kesimpulan dari skripsi yang berjudul: "System Presensi <i>Fingerprint</i> Yang Terintegrasi Dengan Proses Penggajian" adalah dengan adanya Sistem ini dapat membantu dalam mengelola data Presensi serta data penggajian dan memberi kemudahan pada bagian keuangan untuk melakukan penggajian pada karyawan.
Pemodelan Proses Bisnis Penggajian yang Efektif melalui BPMN di PT. Bumi Sawindo Permai (Soekarno-Hatta, Vinda Daningrum, Ariyadi, 2023)	Penelitian di PT. Bumi Sawindo Permai menghasilkan pemodelan proses bisnis penggajian yang direpresentasikan melalui workflow dalam sistem ERP ASCEND, menggunakan Business Process Model and Notation (BPMN) untuk visualisasi. Proses ini melibatkan pengumpulan data melalui observasi langsung dan wawancara, yang kemudian dianalisis untuk mengidentifikasi ketidakselarasan dalam komponen proses penggajian, seperti penerimaan tenaga kerja dan pemberhentian. Hasil analisis ini mendorong perancangan <i>Standart Operational Procedure</i> (SOP) yang terintegrasi, bertujuan untuk menjadi standar baku dalam pengelolaan sistem pembayaran gaji karyawan. Dengan implementasi SOP yang baru, diharapkan efisiensi dan efektivitas dalam proses penggajian dapat meningkat, serta mengurangi bottleneck yang ada dalam sistem saat ini. Rekomendasi untuk perbaikan lebih lanjut juga disusun berdasarkan temuan dari penelitian ini, menekankan pentingnya penggunaan teknologi

Judul	Hasil Penelitian
	informasi dalam mendukung proses bisnis yang lebih sistematis dan terukur.
Pengembangan Sistem Penggajian Karyawan Berbasis Web untuk Meningkatkan Efisiensi dan Akurasi di CV. Mufidah Terminal Print (Chairunnisa, Rusdianto, Jonemaro, 2023).	Hasil penelitian menunjukkan bahwa sistem penggajian yang dikembangkan di CV. Mufidah Terminal Print berhasil mengintegrasikan mesin <i>fingerprint</i> untuk akurasi data kehadiran, sehingga mengurangi kesalahan <i>input</i> yang sering terjadi pada sistem konvensional. Proses perhitungan gaji yang sebelumnya memakan waktu hingga 3 hari kini dapat dilakukan secara otomatis dan lebih cepat. Pengujian sistem menunjukkan tingkat validitas 100% dalam pengujian unit, integrasi, dan validasi, serta kompatibilitas yang baik di berbagai browser dan perangkat. Dengan analisis kebutuhan yang mencakup 64 kebutuhan fungsional, sistem ini tidak hanya meningkatkan efisiensi dan akurasi, tetapi juga memudahkan pengelolaan data karyawan secara keseluruhan.
Sistem Informasi Penggajian Guru Honorer Berbasis Web pada SMK Hasanudin Kandangahur (Dekiki & Hasti, 2021).	Hasil penelitian menunjukkan bahwa penulis merancang sistem informasi penggajian guru honorer berbasis web di SMK Hasanudin Kandanghaur. Tujuan dari perancangan sistem ini adalah untuk memudahkan proses Presensi, penggajian, dan kasbon yang dilakukan oleh bendahara dan koordinator guru piket. Dengan sistem ini, diharapkan masalah yang ada pada kegiatan Presensi dan penggajian dapat diminimalisir. Implementasi sistem mencakup perangkat lunak dan perangkat keras yang diperlukan, seperti sistem operasi Microsoft Windows 7, Apache, MySQL, dan spesifikasi perangkat keras minimal seperti prosesor Intel Core2Duo atau AMD Athlon X2.
Pengembangan Aplikasi Berbasis Web untuk Pengelolaan Keuangan Kelompok Belajar dan Taman Kanak-Kanak (KinderFin) (Puspa & Putra, 2024)	Aplikasi KinderFin berhasil dikembangkan sesuai dengan spesifikasi dan kebutuhan yang telah dirancang untuk mendukung pengelolaan keuangan di kelompok belajar dan taman kanak-kanak. Implementasi sistem berbasis otorisasi pada aplikasi KinderFin memungkinkan pengguna mengakses fitur sesuai peran masing-masing serta meningkatkan keamanan dan kenyamanan dalam penggunaan aplikasi. Hasil pengujian menunjukkan bahwa aplikasi KinderFin telah memenuhi seluruh kriteria pengujian dan berfungsi sesuai harapan

Judul	Hasil Penelitian
Pengembangan <i>API</i> Aplikasi KinderFin: Sistem Monitoring Keuangan Kelompok Belajar dan Taman Kanak-kanak Berbasis Website (Akbar, 2024).	<p>sehingga operasional aplikasi berjalan dengan lancar tanpa kendala berarti bagi pengguna. Aplikasi KinderFin menjadikan pengelolaan keuangan di kelompok belajar dan taman kanak-kanak lebih terstruktur, efisien, dan transparan serta meningkatkan akurasi dan kemudahan akses data bagi pengguna.</p> <p>Proyek "Pengembangan <i>API</i> Aplikasi KinderFin: Sistem Monitoring Keuangan Kelompok Belajar dan Taman Kanak-kanak Berbasis Website" dirancang untuk mempermudah pengelolaan administrasi keuangan sekolah tingkat taman kanak-kanak di Surabaya. Aplikasi ini mendukung berbagai kebutuhan keuangan, seperti pembayaran PPDB, daftar ulang, SPP, biaya komite, ekstrakulikuler, hingga pencatatan gaji guru, dan pengeluaran sekolah. Aplikasi dibuat dengan menggunakan <i>framework Express.js</i> dan <i>database PostgreSQL</i> dengan menerapkan arsitektur clean modular sehingga tiap modul terpisah dengan baik dan mempermudah dalam pemeliharaan aplikasi. Aplikasi <i>backend</i> menggunakan <i>dockploy</i> agar dapat dijalankan dan diakses secara publik. Proyek ini menjadi solusi efektif untuk meningkatkan efisiensi keuangan sekolah.</p>

2.2 Aplikasi Sejenis

Terdapat aplikasi yang mirip seperti Aplikasi KinderFin 2.0 namun digunakan di berbagai aspek seperti di pabrik serta di tempat lainnya. Aplikasi Sejenis merupakan aplikasi yang memiliki fitur yang mirip seperti Aplikasi KinderFin 2.0. Pada Tabel 2.2 akan dijelaskan perbandingan fitur aplikasi yang serupa dengan Aplikasi KinderFin 2.0.

Tabel 2.2 Perbandingan Aplikasi Talenta dengan Aplikasi Kindefin 2.0

Kategori Fitur	Fitur Aplikasi pada Talenta	Keterangan Fitur (berdasarkan Talenta)	Apakah Fitur Ini Tersedia di KinderFin?
Manajemen Pengguna	Pengelolaan Data Pengguna	Modul untuk menambahkan, mengubah, atau mengelola data pengguna.	Ya
Manajemen Presensi	Integrasi Absensi <i>Fingerprint</i>	Dukungan untuk pencatatan presensi melalui mesin pemindai sidik jari.	Ya (melalui <i>Fingerspot</i>)
	Unggah Berkas Presensi	Kemampuan untuk mengunggah data presensi dari berkas eksternal.	Ya

Kategori Fitur	Fitur Aplikasi pada Talenta	Keterangan Fitur (berdasarkan Talenta)	Apakah Fitur Ini Tersedia di KinderFin?
	Unggah Presensi Manual	Memasukkan atau menyesuaikan data presensi secara manual.	Ya
Manajemen Gaji	Penghitungan Gaji Otomatis	Sistem melakukan perhitungan gaji karyawan secara otomatis.	Ya
	Pengaturan Komponen Gaji	Modul untuk mengatur komponen gaji harian, potongan, dan tunjangan lainnya.	Ya
	Validasi Gaji	Proses validasi data dan perhitungan gaji untuk memastikan keakuratan.	Ya
	Distribusi Slip Gaji	Pembuatan dan akses slip gaji karyawan.	Ya
	Pembayaran Gaji (<i>Payroll Disbursement</i>)	Proses transfer atau pencairan gaji kepada karyawan.	Tidak Ada
	Laporan Pengupahan	Pembuatan laporan Gaji	Ya
	Pengelolaan Beban/Pengeluaran	Kontrol biaya dan pengeluaran bisnis.	Tidak Ada
	Perhitungan Penghasilan Tidak Tetap	Kalkulasi untuk pembayaran tunjangan hari raya (THR), pesangon, dan gaji prorata.	Ya (Penerapan Pada Bonus)

2.3 KinderFin

Aplikasi KinderFin yang dikembangkan dalam kegiatan pengabdian masyarakat telah didaftarkan secara resmi pada Pangkalan Data Kekayaan Intelektual (PDKI) Direktorat Jenderal Kekayaan Intelektual Kementerian Hukum dan HAM (KinderFin, 2024). Aplikasi ini pada modul penggajian sebelumnya hanya memuat daftar pengguna serta nominal gaji yang diisikan secara manual oleh bendahara, sehingga belum memberikan transparansi kepada seluruh pengguna. Dalam pengembangannya, modul penggajian pada Aplikasi KinderFin 2.0 kini mendukung penyesuaian label nama gaji, nominal komponen gaji, serta potongan yang memudahkan proses perhitungan gaji secara otomatis. Perbandingan antara versi awal dan versi setelah pengembangan dijelaskan pada Tabel 2.3 Perbedaan Fitur .

Tabel 2.3 Perbedaan Fitur Perhitungan Gaji pada KinderFin 1.0 dengan KinderFin 2.0

KinderFin 1.0	KinderFin 2.0 (dikembangkan dalam Tugas Akhir)
Belum ada modul integrasi mesin <i>fingerprint</i> .	Sudah terdapat modul dan mesin <i>fingerprint</i> sudah terintegrasi dengan Aplikasi KinderFin (penghitungan manual)
Aplikasi hanya memiliki fitur berupa gaji akhir sehingga rawan kecurangan	Aplikasi dapat mengakomodasi rekap gaji harian sehingga dapat mengurangi kecurangan dan kesalahan.

KinderFin 1.0	KinderFin 2.0 (dikembangkan dalam Tugas Akhir)
Aplikasi Tidak memiliki log aktivitas yang dilakukan oleh Administrator dan Bendahara.	Aplikasi memiliki log aktivitas yang dilakukan oleh Administrator dan Bendahara.
Aplikasi tidak dilengkapi dengan pengajuan perubahan gaji.	Aplikasi dilengkapi dengan pengajuan perubahan gaji.

2.4 Dasar Teori

Bagian ini menyajikan berbagai dasar teori yang menjadi landasan dalam perancangan serta implementasi Tugas Akhir Pengembangan Aplikasi KinderFin 2.0. Teori-teori ini mencakup konsep-konsep kunci yang relevan dengan pengembangan Aplikasi *Website* dan Modul Penggajian.

2.4.1 Sistem *Payroll*

Payroll adalah proses pembayaran yang diberikan kepada pegawai sebagai imbalan atas pegawaian yang telah mereka lakukan. Secara lebih spesifik, gaji adalah bentuk kompensasi finansial yang merupakan hak setiap pegawai, diterima dalam bentuk uang dari pemberi kerja atau instansi. Pembayaran gaji ini diatur sesuai dengan kontrak kerja yang telah disepakati, baik antara pegawai dan pemberi kerja, maupun berdasarkan ketentuan hukum yang berlaku. Di dalamnya juga termasuk berbagai tunjangan, seperti tunjangan untuk pegawai dan keluarganya, yang merupakan bagian dari imbalan atas kontribusi pegawai dalam melaksanakan tugas-tugas yang telah diselesaikan atau akan dilakukan di masa depan (Chairunnisa., Rusdianto, D. S., & Jonemaro, E. M. A., 2023).

2.4.2 *Fingerprint*

Fingerprint adalah tanda dari suatu orang yang tidak dapat ditiru atau disamakan dengan orang lain. *Fingerprint* menjadi suatu pengenal dan tanda di berbagai aspek kehidupan seperti pendidikan, pegawaian dan bidang lainnya. *Fingerprint* merupakan salah satu ciri dari seseorang sehingga setiap orang pasti mempunyai sidik jari yang berbeda dengan orang yang lainnya dan tidak mungkin ada sistem sidik jari yang sama pada orang lain.(Widhiantoro, Suryadewi, Arindhita, 2013).

2.4.3 *Fingerprint System (Sistem Fingerprint)*

Menggunakan sistem absensi biometrik *fingerprint* mengurangi masalah yang timbul dari penggunaan sistem absensi manual. Memiliki sistem absensi *fingerprint* biometrik dapat mengurangi jumlah penipuan umum seperti manipulasi data dan keamanan absensi. *Fingerprint* adalah jenis pengenal biometrik yang menggunakan karakteristik fisik penduduk untuk mengidentifikasi mereka (Supriyadi et al., 2023). Sistem ini sering digunakan dalam berbagai aspek kehidupan seperti pegawaian dan pendidikan. Sistem ini menghubungkan antara alat *fingerprint* dengan basis data yang akan mengonversi dari bilangan biner ke dalam suatu *file* untuk memudahkan pembacaan oleh Administrator atau pengguna yang akan mengolah data tersebut.

2.4.4 *Framework Pengembangan*

Framework pengembangan *web full stack* menyediakan struktur dan alat untuk membangun aplikasi web baik pada sisi *frontend* maupun *backend*. Dalam pengembangan modul *payroll* yang terintegrasi dengan *fingerprint system* dan Aplikasi KinderFin 2.0,

framework seperti *Express.js* sebagai *backend* (*Express.js*, 2024) dan *Next.js* sebagai *frontend* (*Vercel Inc.*, 2025) memungkinkan pengembang untuk membuat aplikasi web yang dinamis, cepat, dan responsif. Selain itu, *framework full stack* memudahkan pengelolaan komunikasi antara *frontend* dan *backend* serta integrasi *API* untuk sinkronisasi data Presensi dan *payroll* (*Raza et al.*, 2019).

2.4.5 User Interface (UI) dan User Experience (UX)

Desain UI dan UX adalah aspek yang krusial dalam pengembangan sistem *payroll*. UI berkaitan dengan elemen visual aplikasi, sedangkan UX berfokus pada pengalaman pengguna secara keseluruhan saat menggunakan aplikasi. Dalam sistem *payroll*, desain UI yang baik harus menyajikan data kehadiran dan gaji dengan jelas, sehingga mudah dipahami oleh pengguna. UX yang baik akan memastikan bahwa pengguna, baik bendaraha maupun guru/pengasuh, dapat mengakses informasi dan melakukan pengaturan dengan mudah. *Next.js* adalah *framework* yang dapat digunakan dengan *React* untuk membangun aplikasi satu halaman, seluler, atau *server-render*. *Framework* ini memungkinkan pengembang untuk memanfaatkan *React* sebagai platform untuk pengembangan *frontend* (*Faruque*, 2022). *Node.js* adalah *runtime environment platform server-side* yang bersifat *open source* dan *cross-platform* yang menggunakan *Chrome V8 JavaScript engine*, sehingga *Node.js* dapat menjalankan *JavaScript* di luar *browser*. Dalam konteks *backend*, *Node.js* digunakan untuk mengembangkan *server* dan *database* yang bekerja di balik layar dalam suatu aplikasi, termasuk pengembangan metode *HTTP* (*Hypertext Transfer Protocol*) dan *API* (*Sauda & Barokah*, 2022).

2.4.6 Arsitektur Web

Arsitektur web merujuk pada cara komponen-komponen dalam aplikasi web saling berinteraksi. Dalam pengembangan modul *payroll*, arsitektur yang baik mencakup pemisahan antara *frontend*, *backend*, *database*, dan *Application Programming Interface API*. Hal ini memungkinkan sistem *payroll* untuk menangani banyak pengguna dan menyediakan informasi secara *real-time*, termasuk data kehadiran dan perhitungan gaji. Struktur yang modular ini juga memberikan keleluasaan dalam pengembangan, di mana *backend* menangani logika bisnis dan penyimpanan data, sementara *frontend* berfokus pada antarmuka pengguna (*Berners-Lee et al.*, 2001).

2.4.7 Keamanan Data

Keamanan data menjadi aspek penting dalam pengembangan sistem *payroll*, terutama karena data yang dikelola mencakup informasi pribadi karyawan dan data keuangan. Langkah-langkah untuk menjaga keamanan data dapat meliputi:

1. Enkripsi data: Melindungi data sensitif yang dikirim antara sistem *fingerprint*, *backend*, dan *frontend*.
2. Autentikasi dan otorisasi: Memastikan akses yang tepat hanya diberikan kepada pengguna yang berwenang.
3. *Backup* dan pemulihan data: Memastikan data dapat dipulihkan jika terjadi kehilangan akibat masalah teknis atau insiden lainnya (*Whitman & Mattord*, 2016).

2.4.8 REST API

Pengembangan fitur ini membutuhkan pengiriman data dan pengelolaan serta penerimaan data. Data tersebut membutuhkan sebuah jembatan untuk menghubungkan dan komunikasi. Sistem *fingerprint* akan mengirimkan data serta diterima *server* untuk diolah. REST *API* adalah

teknologi yang digunakan untuk menghubungkan suatu aplikasi dengan aplikasi lainnya melalui komunikasi *Representational State Transfer* (REST). REST termasuk dalam konstruksi perangkat lunak dan pada tahap implementasinya menggunakan komunikasi HTTP (Hermansyah dan Maryam, 2023).

2.4.9 *Clean Architecture*

Pada pengembangan suatu aplikasi, terdapat banyak sekali arsitektur untuk memudahkan pembuat aplikasi untuk menerapkan fitur serta logika bisnis untuk aplikasi tersebut. Pada penerapan pengembangan Aplikasi KinderFin memakai arsitektur yang sudah ada yaitu *Clean Architecture*. *Clean architecture* adalah pendekatan dalam rekayasa perangkat lunak yang membantu memisahkan aspek-aspek yang terkait dengan fungsionalitas yang bergantung pada platform dan yang tidak bergantung pada *platform*. *Clean architecture* menekankan pemisahan yang jelas antara berbagai lapisan aplikasi, memungkinkan kode yang lebih mudah dipelihara, diuji, dan dikembangkan ulang. Selain itu, *clean architecture* bersifat berpusat pada domain, yang memungkinkan untuk menjelaskan semua elemen dari domain tersebut (Resty *et al.*, 2024)

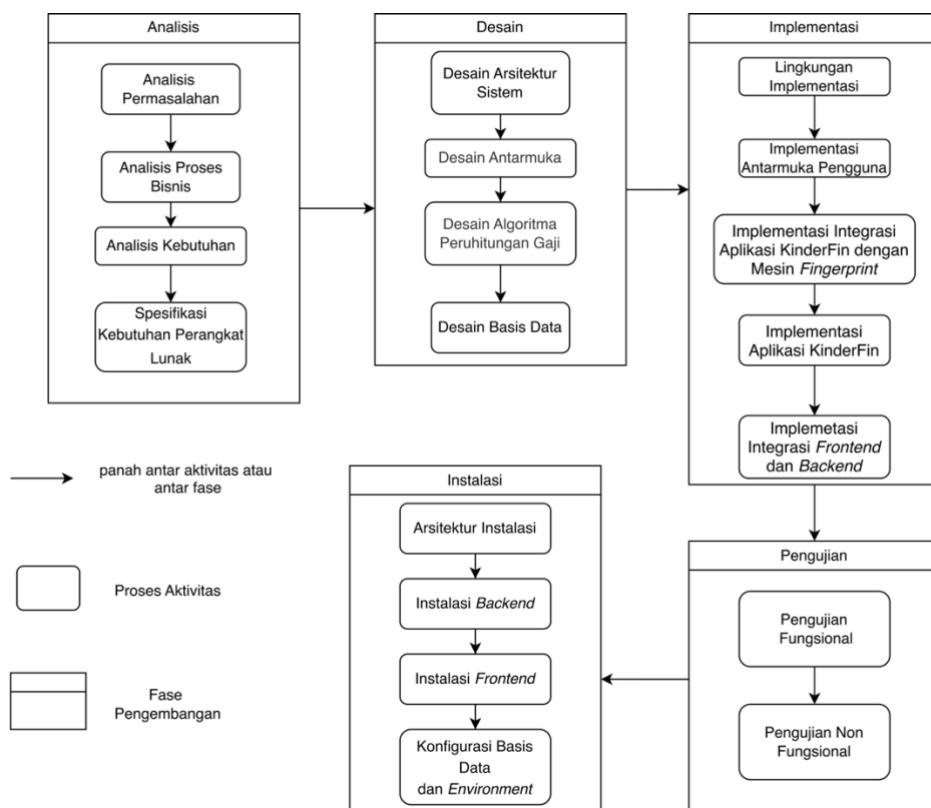
2.4.10 Interoperabilitas

Interoperabilitas merupakan kemampuan dua atau lebih sistem untuk bertukar informasi dan memanfaatkan informasi yang telah dipertukarkan dengan mengikuti standar umum yang telah disepakati (Kurniawan, 2021). Interoperabilitas pada Tugas Akhir ini bertujuan sebagai alat pengujian untuk menguji Aplikasi KinderFin bertukar informasi dengan mesin *fingerprint* melalui *server*.

(Halaman Ini Sengaja Dikosongkan)

BAB 3 METODE PENELITIAN

Pada tugas akhir ini, pengembangan aplikasi menggunakan metode SDLC *Waterfall*. Proses perencanaan sampai instalasi dapat dilihat di Gambar 3.1. Model *Waterfall* dikenal dengan pendekatannya yang sekuensial, di mana setiap fase pengembangan harus diselesaikan secara tuntas sebelum melanjutkan ke fase berikutnya. Fase pertama dimulai dari analisis mengenai kebutuhan serta permasalahan dan dijelaskan bagaimana proses bisnis itu berjalan. Selanjutnya ada desain, desain berisi mengenai rancangan apa saja di dalam pengembangan Aplikasi. Setelah adanya rancangan, dilanjutkan implementasi serta dilanjutkan pengujian setelah diimplementasikan. Terakhir adalah instalasi untuk mengetahui Aplikasi berjalan dengan normal.



Gambar 3.1 Alur Diagram Pengembangan Modul Penggajian Aplikasi KinderFin 2.0

3.1 Analisis

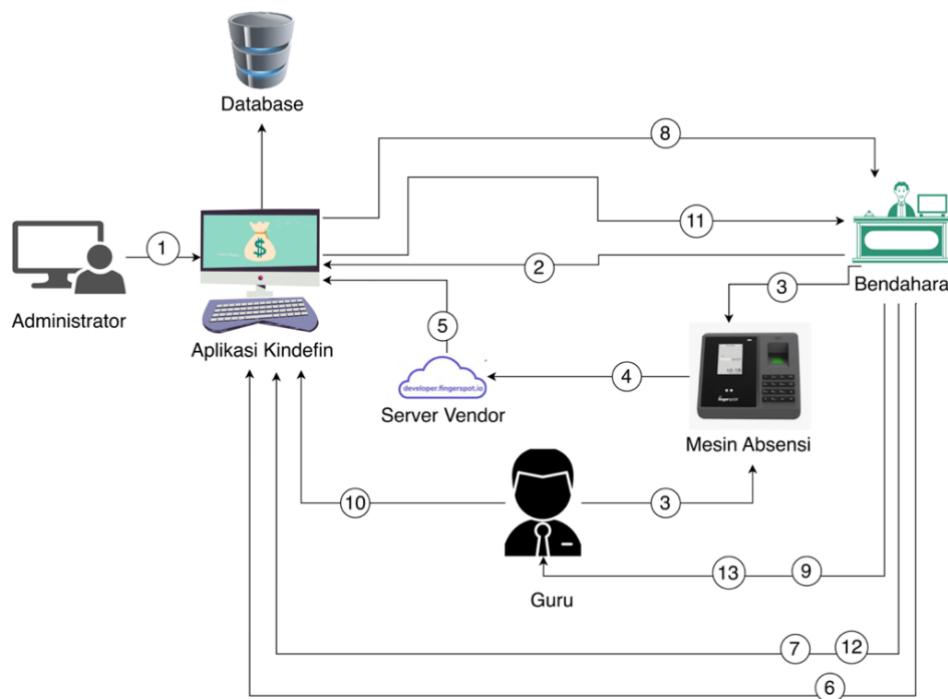
Pada sub-bab ini dibahas mengenai analisis permasalahan dari penggunaan Aplikasi KinderFin 1.0, analisis proses bisnis yang diusulkan dan analisis kebutuhan Aplikasi KinderFin 2.0 yang dikembangkan. Pada sub-bab 3.1.1 akan dijelaskan analisis permasalahan yang berisi tentang permasalahan yang terjadi pada Aplikasi KinderFin 1.0. Dilanjutkan pada sub-bab 3.1.2 dijelaskan analisis proses bisnis yang menguraikan bagaimana proses penggajian dapat diotomatisasi dengan melibatkan pihak ketiga (*fingerspot*) dan mekanisme validasi yang lebih efektif. Pada sub-bab terakhir yaitu 3.1.3 menguraikan analisis kebutuhan yang merinci fungsi-fungsi utama yang harus dimiliki aplikasi, berdasarkan hasil identifikasi masalah dan perbaikan proses bisnis. Keterkaitan antara ketiga sub-bab ini sangat penting, di mana pemahaman terhadap permasalahan yang ada menjadi landasan untuk merancang proses bisnis yang lebih efisien, yang pada akhirnya menentukan kebutuhan sistem yang harus dikembangkan.

3.1.1 Analisis Permasalahan

Aplikasi KinderFin 2.0 hadir untuk mentransformasi modul penggajian yang terbatas pada versi sebelumnya. Saat ini, Aplikasi KinderFin 1.0 hanya menyajikan nominal dan status pembayaran tanpa rincian gaji serta potongan, dengan proses manual yang rawan kesalahan dan kurang transparan karena tidak adanya integrasi data kehadiran dan digitalisasi pengajuan perubahan gaji. Untuk mengatasi ini, KinderFin 2.0 dirancang untuk menampilkan rincian gaji secara transparan, meliputi gaji pokok, tunjangan, potongan, dan bonus. Modul ini akan terintegrasi langsung dengan data *fingerprint* untuk perhitungan gaji otomatis berdasarkan kehadiran dan presensi, serta mampu mengimpor data kehadiran dari *file Excel* mesin *fingerprint*. Digitalisasi pengajuan perubahan gaji juga menjadi prioritas untuk mempercepat proses, memastikan dokumentasi, dan mempermudah pemantauan. Otomatisasi dan transparansi ini, Aplikasi KinderFin 2.0 diharapkan dapat meminimalkan kesalahan perhitungan dan meningkatkan kepercayaan pengguna terhadap sistem penggajian.

3.1.2 Analisis Proses Bisnis

Aplikasi KinderFin merupakan aplikasi yang dirancang untuk sekolah. Untuk memudahkan proses administratif sekolah. Pada modul penggajian, dikembangkan model penggajian secara otomatis agar semua guru dapat mengetahui detail gaji serta transparansi gaji setiap guru serta jajarannya. Gambar 3.2 merupakan Gambaran proses bisnis yang terjadi di modul penggajian Aplikasi KinderFin 2.0.



Gambar 3.2 Proses Bisnis Modul Penggajian Aplikasi KinderFin 2.0

Dalam pengembangan modul penggajian otomatis ini, Aplikasi KinderFin 2.0 berinteraksi dengan pihak ketiga (*fingerspot*), yaitu *vendor* mesin presensi, melalui *API* untuk mendapatkan data kehadiran langsung dari *server developer.fingerspot.io*. Proses dimulai ketika Administrator menambahkan pengguna ke sistem. Selanjutnya, Bendahara melakukan pengaturan gaji aktif, serta memasukkan nominal gaji harian dan pokok untuk setiap jabatan di tabel *master_jabatan* dan detail potongan gaji di tabel *potongan_keterlambatan*. Setelah

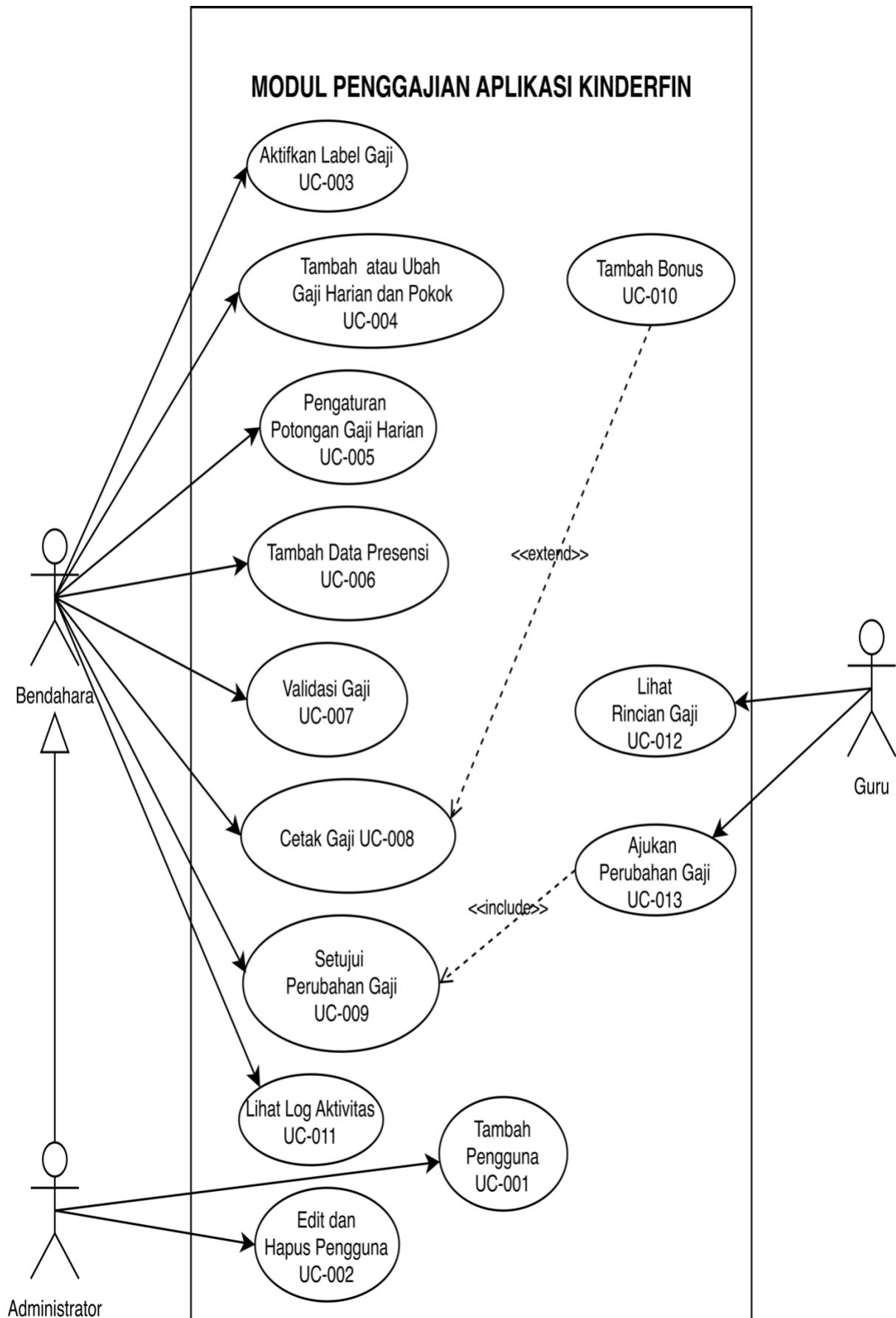
konfigurasi awal ini, semua pengguna dapat melakukan presensi seperti biasa, dengan data kehadiran yang akan otomatis dikirimkan ke aplikasi melalui *server* pihak ketiga (*fingerspot*). Apabila terjadi kendala teknis dalam pengiriman data presensi otomatis, Bendahara dapat mengunggah *file* presensi secara manual dari mesin ke aplikasi, dan jika pengunggahan *file* juga mengalami masalah, Bendahara dapat memasukkan data presensi secara manual melalui antarmuka aplikasi. Data ini kemudian akan menghasilkan detail gaji per hari berdasarkan jabatan masing-masing pengguna. Secara berkala, Bendahara bertanggung jawab untuk memvalidasi dan mendistribusikan hasil gaji kepada semua pengguna. Bagi pengguna yang ingin mengajukan perubahan gaji, fitur digitalisasi pengajuan perubahan gaji melalui Aplikasi KinderFin 2.0 telah disediakan, yang kemudian akan divalidasi oleh Bendahara. Setelah validasi bukti, Administrator atau Bendahara dapat melakukan pembaruan gaji pengguna. Detail nama tabel dan aktor yang terlibat dalam setiap tugas dapat dilihat pada Tabel 3.1.

Tabel 3.1 Penjelasan Proses Bisnis Modul Penggajian Aplikasi KinderFin 2.0

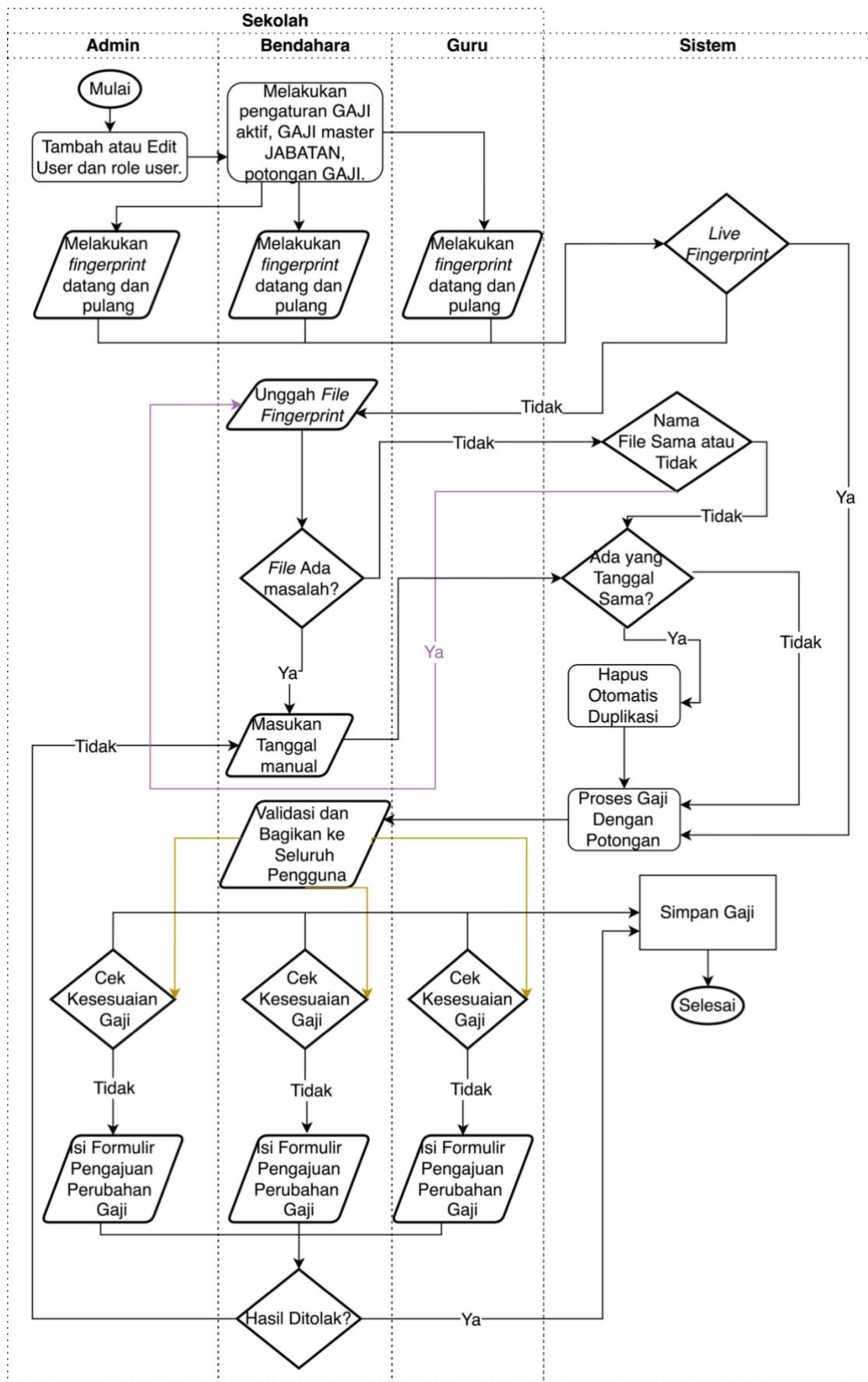
Nomor Alur Proses	Nama Alur Proses	Aktor
1	Tambah Pengguna	Administrator
2	Aktifkan label Gaji, Tambah atau ubah gaji harian, Pengaturan Potongan Gaji Harian	Bendahara
3	Presensi di Mesin <i>fingerprint</i>	Guru, Bendahara
4	Kirim data presensi ke <i>server</i> vendor (<i>fingerspot</i>)	Mesin Presensi
5	Teruskan data presensi dari vendor (<i>fingerspot</i>) ke Aplikasi KinderFin	Server Vendor (<i>fingerspot</i>)
6	Unggah <i>file</i> presensi	Bendahara
7	Unggah presensi manual	Bendahara
8	Validasi gaji	Bendahara
9	Terima gaji	Guru
10	Pengajuan perubahan gaji	Guru
11	Validasi pengajuan perubahan gaji	Bendahara
12	Unggah presensi manual perubahan gaji	Bendahara
13	Terima gaji setelah pengajuan perubahan gaji	Guru

3.1.3 Analisis Kebutuhan

Pada analisis bisnis, terdapat tiga aktor utama yang menggunakan aplikasi ini, yaitu Administrator, Bendahara, dan Guru. Ketiganya memiliki kebutuhan fungsional yang berbeda. Untuk memahami kebutuhan tersebut. Maka terdapat Diagram *Use Case* pada Gambar 3.3 menampilkan rancangan modul penggajian Aplikasi KinderFin 2.0 berdasarkan sub-bab 3.1.2 Analisis Proses Bisnis dan pengembangan kebutuhan fungsional utama. Fungsional utama, meliputi memasukkan nominal gaji di master jabatan, membuat potongan gaji, mengunggah data Presensi, memberikan informasi pembayaran gaji, Validasi Gaji, serta Setujui Perubahan Gaji. Terdapat hubungan (interaksi) antar aktor yaitu saat pengajuan perubahan gaji. Melengkapi gambaran fungsionalitas, Gambar 3.4 memberikan visualisasi mendalam alur kerja. Diagram ini menampilkan langkah-langkah dan urutan aktivitas antara ketiga aktor dalam modul penggajian Aplikasi KinderFin 2.0



Gambar 3.3 Diagram Use Case Modul Penggajian Aplikasi KinderFin 2.0



Gambar 3.4 Diagram Aktivitas Alur Proses Bisnis Modul Penggajian Aplikasi KinderFin 2.0

3.1.3.1 Daftar Kebutuhan Administrator

Administrator merupakan aktor yang bertanggung jawab mengenai hak pengguna. Tugas utama Administrator meliputi Tambah Pengguna, mengelola data pengguna yang sudah ada, serta mengatur hak akses agar setiap pengguna dapat menggunakan sistem sesuai dengan perannya. Administrator memastikan keamanan pengguna setiap hak pengguna dalam memakai Aplikasi tersebut. Pada Tabel 3.2 adalah daftar kebutuhan Administrator.

Tabel 3.2 Daftar Kebutuhan Administrator

Kebutuhan	Kode Use Case	Deskripsi
Tambah Pengguna	UC-001	Administrator dapat menambah, mengubah, dan menghapus data pengguna sesuai dengan peran dan jabatan dalam Aplikasi KinderFin 2.0
Edit dan Hapus Pengguna	UC-002	Administrator mengatur hak akses agar setiap pengguna dapat menggunakan fitur sesuai perannya.

3.1.3.2 Daftar Kebutuhan Bendahara

Bendahara merupakan aktor yang memiliki kebutuhan paling banyak dibandingkan semua aktor modul penggajian Aplikasi KinderFin 2.0. Setelah Kebutuhan Administrator terpenuhi, maka setelahnya kebutuhan bendahara. Kebutuhan bendahara tersebut meliputi, dapat melakukan aktivasi pada pengaturan gaji aktif, lalu memasukkan nominal gaji setiap jabatan atau *role* pengguna serta memasukkan potongan gaji setiap jabatan. Kebutuhan krusial bendahara yaitu, mengunggah *file* Presensi yang nantinya diproses otomatis ke aplikasi. Kebutuhan lainnya yaitu, memberikan validasi gaji serta menyetujui ajuan perubahan gaji. Pada Tabel 3.3 adalah daftar kebutuhan Bendahara.

Tabel 3.3 Daftar Kebutuhan Bendahara

Kebutuhan	Kode Use Case	Deskripsi
Aktifkan Label Gaji	UC-003	Bendahara dapat mengaktifkan jenis-jenis gaji yang akan digunakan dalam proses penggajian dengan memberikan label sesuai kebutuhan sedang berjalan agar proses penggajian dapat dilakukan.
Tambah atau Ubah Gaji Harian	UC-004	Bendahara memasukkan atau memperbarui nominal gaji pokok atau harian setiap jabatan
Pengaturan Potongan Gaji Harian	UC-005	Bendahara mengelola potongan-potongan gaji yang berlaku untuk setiap jabatan, untuk nantinya diolah otomatis dengan gaji harian oleh aplikasi.
Tambah Data Presensi	UC-006	Bendahara mengunggah data kehadiran guru yang diperoleh dari mesin <i>fingerprint</i> atau secara manual.
Validasi Gaji	UC-007	Bendahara memeriksa dan memastikan data gaji sudah benar sebelum proses pembayaran dilakukan.

Kebutuhan	Kode <i>Use Case</i>	Deskripsi
Cetak Gaji	UC-008	Bendahara menyediakan informasi terkait pembayaran dan memberikan bonus (opsional) gaji kepada guru.
Setujui Perubahan Gaji	UC-009	Bendahara memeriksa pengajuan perubahan gaji dari pengguna dan memberikan persetujuan jika data sudah valid.
Tambah Bonus	UC-010	Bendahara menambah bonus dengan rentang waktu tertentu kepada guru.
Lihat Log Aktivitas	UC-011	Aktor dapat melihat seluruh riwayat aktivitas mengenai sistem penggajian.

3.1.3.3 Daftar Kebutuhan Guru

Guru adalah aktor yang sangat mendapat manfaat dengan adanya pengembangan modul penggajian Aplikasi KinderFin 2.0 ini. Guru membutuhkan fitur untuk melakukan Presensi menggunakan mesin *fingerprint* agar kehadiran tercatat secara otomatis dan akurat. Selain itu, Guru juga perlu dapat Lihat Rincian Gaji secara transparan setiap periode penggajian. Jika terdapat ketidaksesuaian atau kebutuhan penyesuaian, Guru dapat Ajukan Perubahan Gaji yang akan divalidasi oleh Bendahara. Pada Tabel 3.4 adalah daftar kebutuhan Guru.

Tabel 3.4 Daftar Kebutuhan Guru

Kebutuhan	Kode <i>Use Case</i>	Deskripsi
Lihat Rincian Gaji	UC-012	Guru dapat Lihat Rincian Gaji yang sudah dihitung berdasarkan data Presensi dan potongan.
Ajukan Perubahan Gaji	UC-013	Guru dapat mengajukan permohonan perubahan gaji jika terdapat kesalahan atau penyesuaian yang diperlukan.

3.1.4 Spesifikasi Kebutuhan Perangkat Lunak

Dalam Proyek tugas akhir diperlukan penentuan metodologi yang tepat agar memudahkan pengembangan fitur penggajian serta rincian data gaji yang akan di *export* untuk mempermudah guru serta pengguna lainnya untuk melihat detail gaji. Mengacu pada sub-bab sebelumnya yaitu sub-bab 3.1.3.1, sub-bab 3.1.3.2, dan sub-bab 3.1.3.3 maka pada sub-bab 3.1.4 akan dijelaskan detail kasus penggunaan yang dibutuhkan oleh Aktor. Terdapat total 13 Kasus penggunaan dengan 3 Aktor, namun pada sub-bab 3.1.4 akan dijelaskan hanya bagian mengenai penggajian seperti Kasus Penggunaan Tambah Data Presensi, Kasus Penggunaan Validasi Gaji, Kasus Penggunaan Cetak Gaji, Kasus Penggunaan Tambah Bonus, Kasus Penggunaan Lihat Rincian Gaji. Untuk Kasus Penggunaan Edit dan Hapus Pengguna, Kasus Penggunaan Aktifkan Label Gaji, Kasus Penggunaan Tambah atau Ubah Gaji Harian dan Pokok, Kasus Penggunaan Pengaturan Potongan Gaji Harian, Kasus Penggunaan Setujui Perubahan Gaji, Kasus Penggunaan Lihat Log Aktivitas, dan Kasus Penggunaan Ajukan Perubahan Gaji dirincikan pada LAMPIRAN F.

3.1.4.1 Kasus Penggunaan Tambah Data Presensi

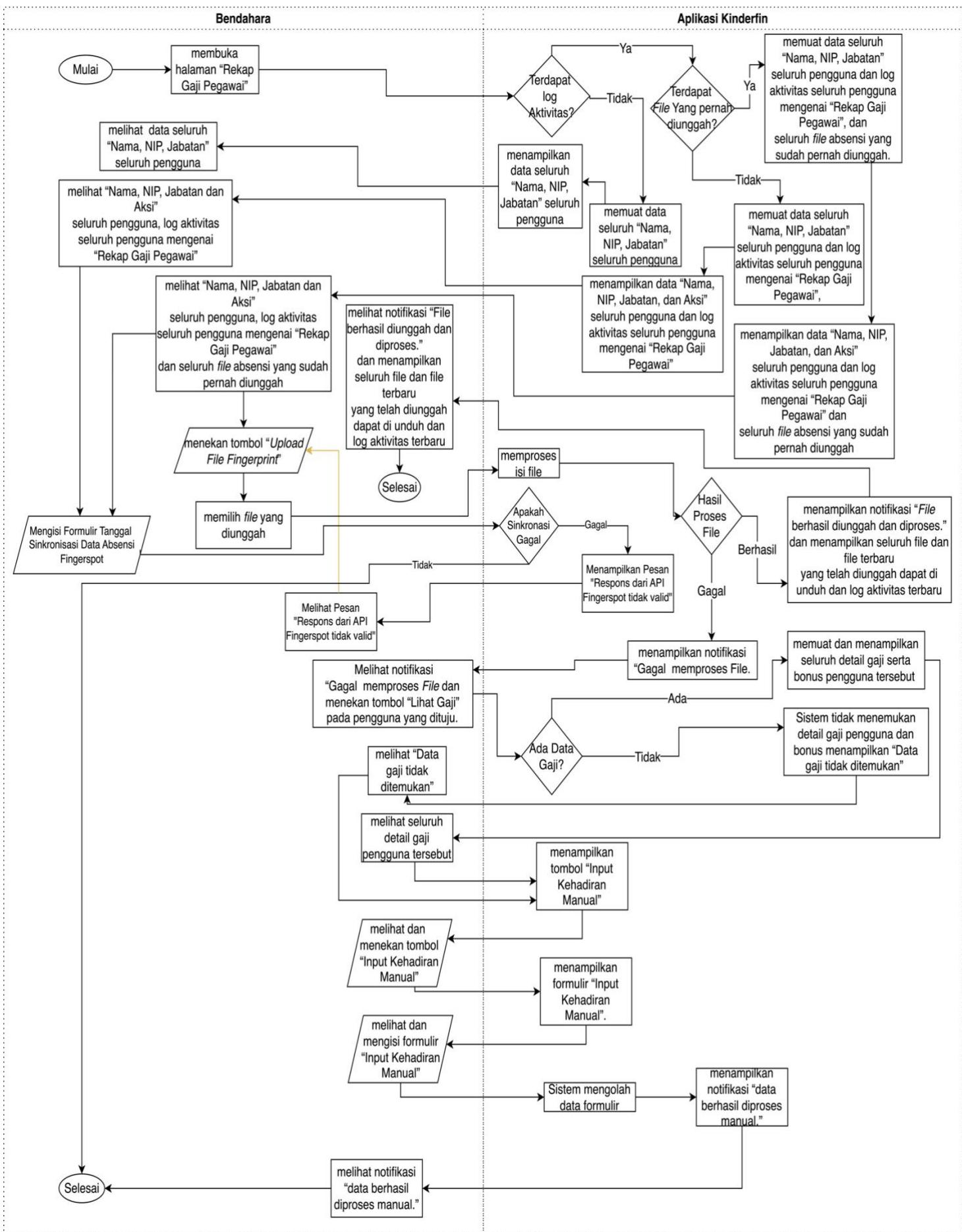
Kasus penggunaan ini terjadi jika pengiriman data dari mesin *fingerprint* ke Aplikasi KinderFin 2.0 gagal. Dalam kasus ini, proses pengelolaan data Presensi harus dilakukan secara

unggah *file* atau memasukkan data manual oleh aktor yang bertanggung jawab agar data kehadiran tetap tercatat dengan akurat dan lengkap. Untuk alurnya terdapat pada Tabel 3.5 dan divisualisasikan pada Gambar 3.5

Tabel 3.5 Kasus Penggunaan Tambah Data Presensi

Komponen	Deskripsi
Nama	Tambah Data Presensi
Nomor	UC-006
Deskripsi	Bendahara mengunggah data kehadiran guru yang diperoleh dari mesin <i>fingerprint</i> atau secara manual.
Aktor	Bendahara
Kondisi Awal	Sistem gagal menerima data Presensi secara otomatis dari mesin <i>fingerprint</i> , sehingga data Presensi tidak tercatat dalam Aplikasi KinderFin 2.0.
Kondisi Akhir	Data Presensi berhasil dimasukkan secara manual ke dalam sistem dan tercatat dengan benar, sehingga proses Presensi kehadiran dapat dilanjutkan.
Alur Normal	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Rekap Gaji Pegawai” 2. Sistem memuat data seluruh “Nama, NIP, Jabatan” seluruh pengguna dan log aktivitas seluruh pengguna mengenai “Rekap Gaji Pegawai”, dan seluruh <i>file</i> Presensi yang sudah pernah diunggah. 3. Sistem menampilkan data “Nama, NIP, Jabatan, dan Aksi” seluruh pengguna dan log aktivitas seluruh pengguna mengenai “Rekap Gaji Pegawai” dan seluruh <i>file</i> Presensi yang sudah pernah diunggah 4. Bendahara melihat “Nama, NIP, Jabatan dan Aksi” seluruh pengguna, log aktivitas seluruh pengguna mengenai “Rekap Gaji Pegawai” dan seluruh <i>file</i> Presensi yang sudah pernah diunggah. 5. Bendahara Mengisi Formulir Tanggal “Sinkronisasi Data Absensi <i>Fingerspot</i>”
Alur Alternatif	<ul style="list-style-type: none"> - Sinkronisasi Data Absensi <i>Fingerspot</i> gagal <ul style="list-style-type: none"> 5a. Bendahara menekan tombol “Upload File Fingerprint” 6a. Bendahara memilih <i>file</i> yang diunggah 7a. Sistem memproses isi <i>file</i> 8a. Sistem menampilkan notifikasi “File berhasil diunggah dan diproses.” 9a. Sistem menampilkan seluruh <i>file</i> dan <i>file</i> terbaru yang telah diunggah dapat di unduh dan log aktivitas terbaru 10a. Bendahara melihat notifikasi dan log aktivitas serta <i>file</i> yang diproses dapat di unduh. - Belum ada <i>file</i> yang diunggah

Komponen	Deskripsi
	<p>2b. Sistem memuat data seluruh “Nama, NIP, Jabatan” seluruh pengguna dan log aktivitas seluruh pengguna mengenai “Rekap Gaji Pegawai”</p> <p>3b. Sistem menampilkan data “Nama, NIP, Jabatan, dan Aksi” seluruh pengguna dan log aktivitas seluruh pengguna mengenai “Rekap Gaji Pegawai”.</p> <p>4b. Bendahara melihat “Nama, NIP, Jabatan dan Aksi” seluruh pengguna, log aktivitas pengguna mengenai “Rekap Gaji Pegawai”</p> <ul style="list-style-type: none"> - Belum ada log mengenai Rekap Gaji Pegawai 2c. Sistem memuat data seluruh “Nama, NIP, Jabatan” seluruh pengguna 3c. Sistem menampilkan data “Nama, NIP, Jabatan, dan Aksi” seluruh pengguna 4c. Bendahara melihat “Nama, NIP, Jabatan dan Aksi” seluruh pengguna <ul style="list-style-type: none"> - <i>File</i> gagal diunggah 8d. Sistem menampilkan notifikasi “Gagal memproses File. 11d. Bendahara melihat notifikasi “Gagal memproses <i>File</i> dan menekan tombol “Lihat Gaji” pada pengguna yang dituju. 12d. Sistem memuat dan menampilkan seluruh detail gaji serta bonus pengguna tersebut. 13d. Bendahara melihat seluruh detail gaji dan bonus pengguna tersebut 14d. Sistem menampilkan tombol “<i>Input</i> Kehadiran Manual” 15d. Bendahara melihat dan menekan tombol “<i>Input</i> Kehadiran Manual” 16d. Sistem menampilkan formulir “<i>Input</i> Kehadiran Manual”. 17d. Bendahara melihat dan mengisi formulir “<i>Input</i> Kehadiran Manual” 18d. Sistem mengolah data formulir 19d. Sistem menampilkan notifikasi “data berhasil diproses manual.” Dan data terbaru 20d. Bendahara melihat notifikasi “data berhasil diproses manual.” dan data terbaru. <ul style="list-style-type: none"> - Tidak ada Data Gaji. 12d. Sistem tidak menemukan detail gaji pengguna dan menampilkan “Data gaji tidak ditemukan” 13d. Bendahara melihat “Data gaji tidak ditemukan”



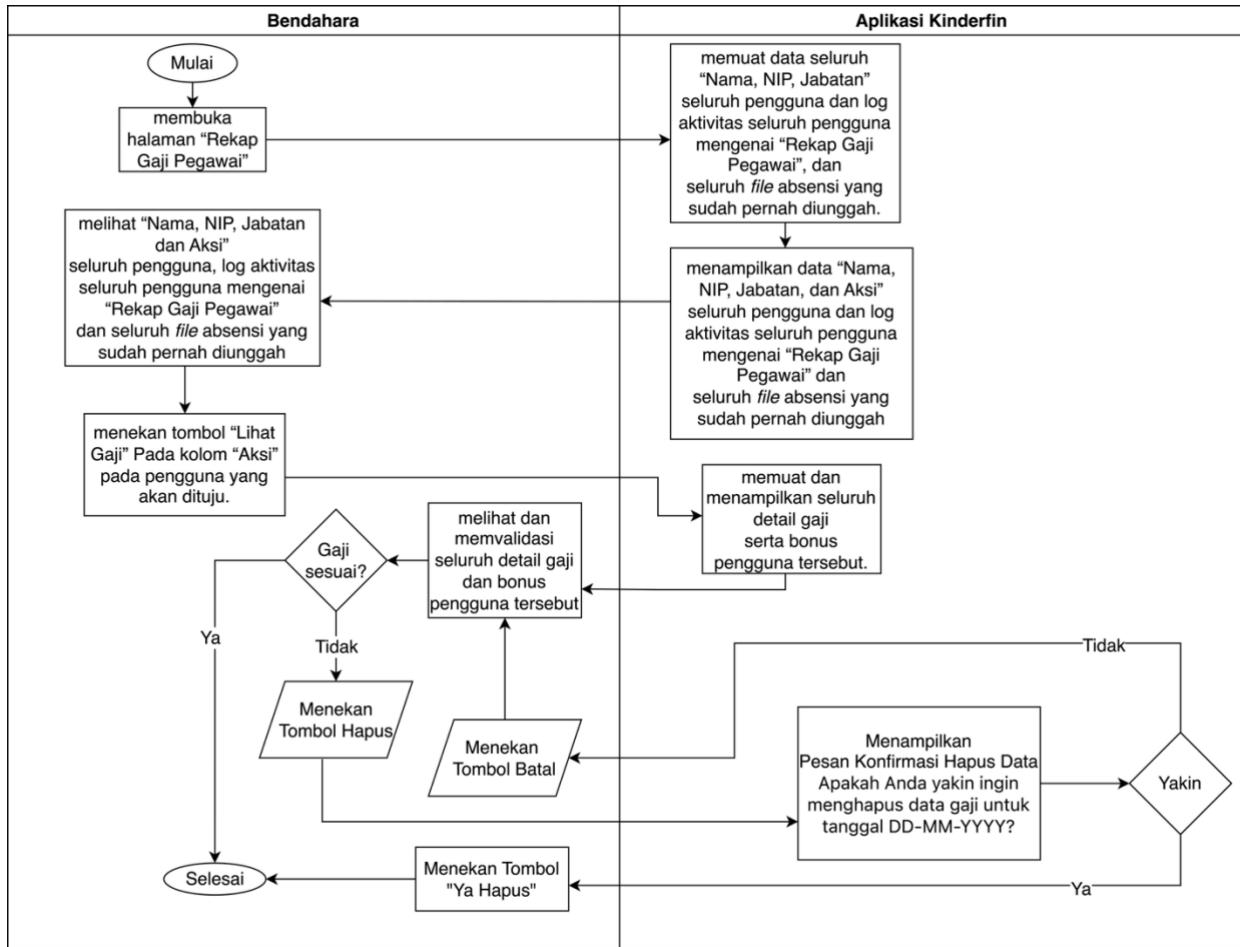
Gambar 3.5 Diagram Aktivitas Tambah Data Presensi

3.1.4.2 Kasus Penggunaan Validasi Gaji

Pada kasus memvalidasi gaji, aktor yaitu Bendahara melakukan tugas sebagai validator gaji atau memvalidasi apakah gaji tersebut sudah sesuai dengan data di lapangan atau tidak. Bendahara pada kasus ini hanya cek kesesuaian data. Tidak melakukan aktivitas lain selain itu. Alur akan dijelaskan dalam Tabel 3.6 dan akan divisualisasikan dalam Gambar 3.6

Tabel 3.6 Kasus Penggunaan Validasi Gaji

Komponen	Deskripsi
Nama	Validasi Gaji
Nomor	UC-007
Deskripsi	Bendahara memeriksa dan memastikan data gaji sudah benar sebelum proses pembayaran dilakukan.
Aktor	Bendahara
Kondisi Awal	Data gaji telah dihitung dan tersedia dalam sistem, namun belum divalidasi oleh Bendahara.
Kondisi Akhir	Data gaji telah divalidasi oleh Bendahara; jika sesuai, data siap untuk proses pembayaran; jika tidak sesuai, pengajuan revisi gaji dilakukan.
Alur Normal	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Rekap Gaji Pegawai” 2. Sistem memuat data seluruh “Nama, NIP, Jabatan” seluruh pengguna dan log aktivitas seluruh pengguna mengenai “Rekap Gaji Pegawai”, dan seluruh file Presensi yang sudah pernah diunggah. 3. Sistem menampilkan data “Nama, NIP, Jabatan, dan Aksi” seluruh pengguna dan log aktivitas seluruh pengguna mengenai “Rekap Gaji Pegawai” dan seluruh file Presensi yang sudah pernah diunggah 4. Bendahara melihat “Nama, NIP, Jabatan dan Aksi” seluruh pengguna, log aktivitas seluruh pengguna mengenai “Rekap Gaji Pegawai” dan seluruh file Presensi yang sudah pernah diunggah 5. Bendahara menekan tombol “Lihat Gaji” Pada kolom “Aksi” pada pengguna yang akan dituju. 6. Sistem memuat dan menampilkan seluruh detail gaji serta bonus pengguna tersebut. 7. Bendahara melihat dan memvalidasi seluruh detail gaji dan bonus pengguna tersebut
Alur Alternatif	<ul style="list-style-type: none"> - Terdapat Gaji Tidak sesuai 8a. Menekan Tombol Hapus 9a. Sistem menampilkan Pesan Konfirmasi Hapus Data “Apakah Anda yakin ingin menghapus data gaji untuk tanggal DD-MM-YYYY?” 10a. Bendahara Menekan Tombol "Ya Hapus"



Gambar 3.6 Diagram Aktivitas Validasi Gaji

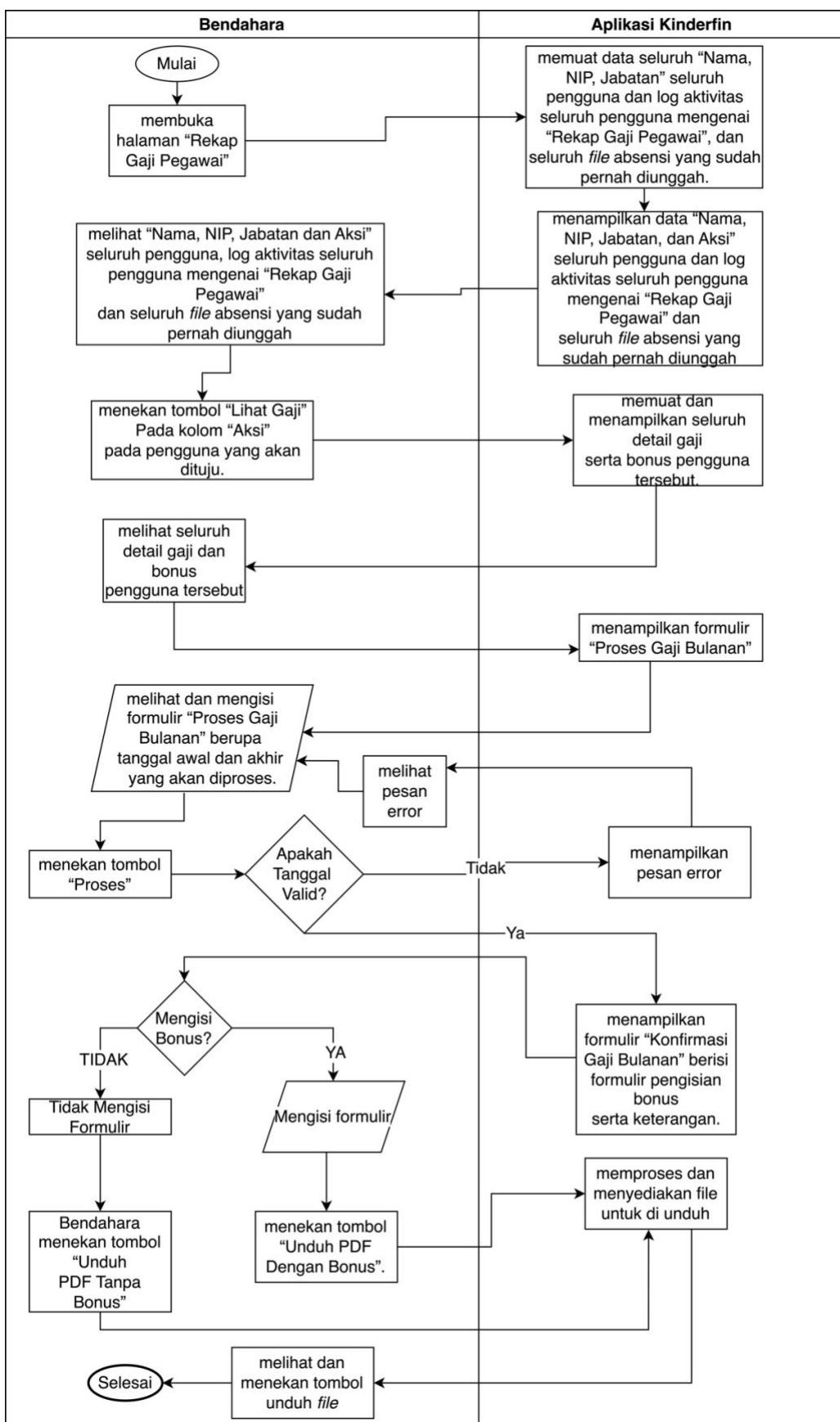
3.1.4.3 Kasus Penggunaan Cetak Gaji

Memberikan informasi gaji merupakan kasus penting pada aktor Bendahara. Informasi gaji merupakan objek utama pada Tugas Akhir ini. Di samping gaji utama, terdapat bonus yang termasuk dalam komponen gaji yang bersifat opsional karena tergantung kebijakan dari Aktor Bendahara dengan diskusi dengan aktor lain seperti Aktor Administrator. Informasi gaji ini dimulai dengan memasukkan tanggal yang akan diproses setiap halaman detail gaji setiap pengguna lalu terdapat bonus yang dapat diisi atau tidak. Lalu akan tercetak secara digital gaji serta detail waktu masuk serta keluar setiap pengguna tersebut. Alur akan dijelaskan dalam Tabel 3.7 dan akan divisualisasikan dalam Gambar 3.7.

Tabel 3.7 Kasus Penggunaan Cetak Gaji

Komponen	Deskripsi
Nama	Cetak Gaji
Nomor	UC-008
Deskripsi	Bendahara menyediakan informasi terkait pembayaran dan memberikan bonus (opsional) gaji kepada guru.
Aktor	Bendahara
Kondisi Awal	Data detail gaji seluruh pengguna sudah tersedia di Aplikasi KinderFin 2.0 namun belum di cek dan di cetak, dan belum diberikan bonus (opsional) serta belum diberikan ke guru oleh Bendahara

Komponen	Deskripsi
Kondisi Akhir	Data detail gaji seluruh pengguna sudah tersedia di Aplikasi KinderFin 2.0 dan sudah di cek dan di cetak, dan diberikan bonus (opsional) serta sudah diberikan ke guru oleh Bendahara.
Alur Normal	<p>1. Bendahara membuka halaman “Rekap Gaji Pegawai”.</p> <p>2. Sistem memuat data seluruh “Nama, NIP, Jabatan” seluruh pengguna dan log aktivitas seluruh pengguna mengenai “Rekap Gaji Pegawai”, dan seluruh <i>file</i> Presensi yang sudah pernah diunggah.</p> <p>3. Sistem menampilkan data “Nama, NIP, Jabatan, dan Aksi” seluruh pengguna dan log aktivitas seluruh pengguna mengenai “Rekap Gaji Pegawai” dan seluruh <i>file</i> Presensi yang sudah pernah diunggah</p> <p>4. Bendahara melihat “Nama, NIP, Jabatan dan Aksi” seluruh pengguna, log aktivitas seluruh pengguna mengenai “Rekap Gaji Pegawai” dan seluruh <i>file</i> Presensi yang sudah pernah diunggah.</p> <p>5. Bendahara menekan tombol “Lihat Gaji” Pada kolom “Aksi” pada pengguna yang akan dituju.</p> <p>6. Sistem memuat dan menampilkan seluruh detail gaji serta bonus pengguna tersebut.</p> <p>7. Bendahara melihat seluruh detail gaji dan bonus pengguna tersebut</p> <p>8. Sistem menampilkan formulir “Proses Gaji Bulanan”</p> <p>9. Bendahara melihat dan mengisi formulir “Proses Gaji Bulanan” berupa tanggal awal dan akhir yang akan diproses.</p> <p>10. Bendahara menekan tombol “Proses”</p> <p>11. Sistem menampilkan formulir "Konfirmasi Gaji Bulanan" berisi formulir pengisian bonus serta keterangan.</p> <p style="text-align: center;"><<extend>> UC-010 : Tambah Bonus.</p> <p>12. Bendahara mengisi formulir bonus dan keterangan</p> <p>13. Bendahara menekan tombol "Unduh PDF Dengan Bonus".</p> <p>14. Sistem memproses dan menyediakan <i>file</i> untuk di unduh</p> <p>15. Bendahara melihat dan menekan tombol unduh <i>file</i></p>
Alur Alternatif	<ul style="list-style-type: none"> - Tidak menyertakan Bonus 12b. Bendahara tidak mengisi formulir bonus 13b. Bendahara menekan tombol "Unduh PDF Tanpa Bonus" - Periode tanggal tidak valid 13c. Sistem Menampilkan pesan error 16c. Bendahara melihat pesan error



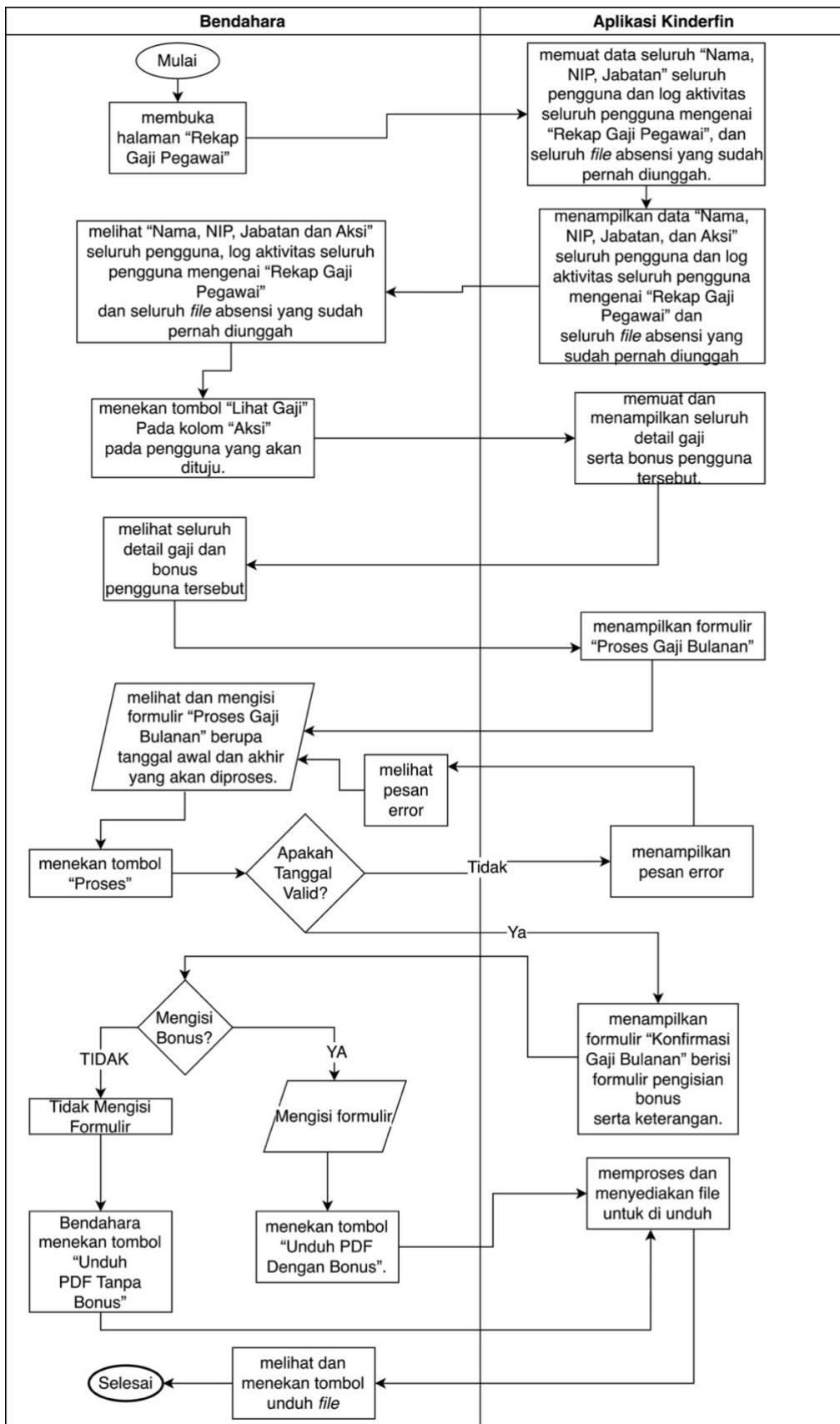
Gambar 3.7 Diagram Aktivitas Cetak Gaji

3.1.4.4 Kasus Penggunaan Tambah Bonus

Kasus Penggunaan Tambah Bonus, merupakan langkah tambahan atau *extend* dari subbab 3.1.4.4. Pengguna dapat menambahkan bonus saat ingin mencetak gaji. Jika akan ditambahkan bonus, bonus akan muncul di slip gaji serta tersimpan di basis data dan akan ditampilkan pada detail gaji pada pengguna tersebut saat membuka halaman detail gaji. Alur akan dijelaskan dalam Tabel 3.8 dan akan divisualisasikan dalam Gambar 3.8.

Tabel 3.8 Kasus Penggunaan Tambah Bonus

Komponen	Deskripsi
Nama	Tambah Bonus
Nomor	UC-010
Deskripsi	Bendahara memberikan bonus kepada guru berdasarkan kriteria tertentu dalam proses penggajian bulanan. Use case ini merupakan <i>extension</i> dari UC-008 (Cetak Gaji) yang diaktifasi ketika bendahara memilih untuk memberikan bonus.
Aktor	Bendahara
Kondisi Awal	Formulir "Konfirmasi Gaji Bulanan" sudah ditampilkan sistem. Data gaji guru untuk periode tertentu sudah diproses. Bendahara berada pada tahap konfirmasi sebelum mencetak slip gaji
Kondisi Akhir	Bonus berhasil ditambahkan ke dalam data gaji guru. Formulir bonus telah diisi dengan nominal dan keterangan yang valid. Data siap untuk dicetak dengan informasi bonus. Bonus sudah tersimpan di basis data dan dapat ditampilkan di Aplikasi KinderFin 2.0
Alur Normal	<ol style="list-style-type: none"> [Extension Point dari UC-008 langkah 11] Sistem menampilkan modal "Konfirmasi Gaji Bulanan" Sistem menampilkan formulir dengan <i>field</i>: <ul style="list-style-type: none"> "Uang Tambahan (Bonus)" - <i>input field</i> untuk nominal "Keterangan Bonus" - text area dengan placeholder "Masukkan keterangan bonus (opsional, max 500 karakter)" Bendahara mengisi <i>field</i> "Uang Tambahan (Bonus)" dengan nominal dalam angka Bendahara mengisi <i>field</i> "Keterangan Bonus" (opsional) Bendahara menekan tombol "Unduh PDF Dengan Bonus" Sistem memvalidasi <i>input</i> : <ul style="list-style-type: none"> - Nominal bonus berupa angka yang valid - Keterangan tidak melebihi 500 karakter Sistem menghitung total gaji + bonus Sistem memproses dan menyediakan <i>file</i> PDF dengan bonus. [Kembali ke UC-008 langkah 14-15] Bendahara melihat dan mengunduh <i>file</i> PDF
Kondisi Extension	<p>Use case ini akan diaktifasi ketika:</p> <ul style="list-style-type: none"> - Bendahara memilih opsi "Berikan Bonus" pada formulir konfirmasi gaji - Periode gaji yang diproses memenuhi kriteria pemberian bonus - Data gaji dasar guru sudah berhasil diproses



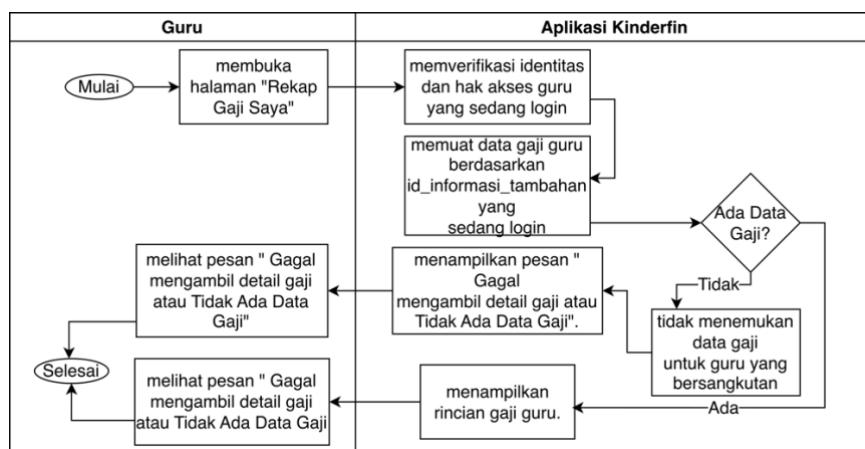
Gambar 3.8 Diagram Aktivitas Tambah Bonus

3.1.4.5 Kasus Penggunaan Lihat Rincian Gaji

Kasus Penggunaan Lihat Rincian Gaji yang dilakukan oleh aktor guru diperlukan karena untuk memastikan bahwa aktor tersebut memiliki hak untuk melihat detail gaji aktor tersebut. Kasus ini memberikan transparansi gaji kepada aktor namun hanya dapat mengakses gaji milik aktor tersebut. Kasus ini akan dijelaskan pada Tabel 3.9 dan divisualisasikan pada Gambar 3.9.

Tabel 3.9 Kasus Penggunaan Lihat Rincian Gaji

Komponen	Deskripsi
Nama	Lihat Rincian Gaji
Nomor	UC-012
Deskripsi	Guru dapat melihat detail informasi gaji pribadi guru tersebut untuk memastikan transparansi gaji. Sistem memastikan bahwa guru tersebut hanya dapat mengakses rincian gaji milik guru tersebut berdasarkan autentikasi dan autorisasi yang valid.
Aktor	Guru
Kondisi Awal	Data gaji guru sudah tersedia di sistem. Guru memiliki hak akses untuk melihat gaji pribadi
Kondisi Akhir	Guru berhasil melihat rincian gaji pribadi mereka. Informasi gaji ditampilkan secara lengkap dan akurat
Alur Normal	<ol style="list-style-type: none"> 1. Guru membuka halaman "Rekap Gaji Saya" 2. Sistem memverifikasi identitas dan hak akses guru yang sedang login 3. Sistem memuat data gaji guru berdasarkan id_informasi_tambahan yang sedang <i>login</i>. 4. Sistem menampilkan rincian gaji guru. 5. Guru melihat dan memverifikasi informasi gaji.
Alur Alternatif	<ul style="list-style-type: none"> - Data Gaji Tidak Ditemukan <ul style="list-style-type: none"> 3a. Sistem tidak menemukan data gaji untuk guru yang bersangkutan 4a. Sistem menampilkan pesan " Gagal mengambil detail gaji atau Tidak Ada Data Gaji". 5a. Guru melihat pesan " Gagal mengambil detail gaji atau Tidak Ada Data Gaji".



Gambar 3.9 Diagram Aktivitas Lihat Rincian Gaji

3.2 Desain

Dalam suatu aplikasi, desain merupakan langkah awal Pembangunan aplikasi tersebut. Desain merupakan dasar dari implementasi teknis yang akan memastikan aplikasi dapat berjalan sesuai dengan yang direncanakan. Pada tahap ini, berbagai aspek teknis yang terkait dengan sistem aplikasi dirancang secara rinci, dimulai dari Desain Arsitektur Sistem yang akan menggambarkan struktur dasar dari Aplikasi KinderFin 2.0. Selanjutnya, desain antarmuka pengguna dirancang untuk memastikan pengalaman pengguna yang optimal. Desain Algoritma Perhitungan Gaji menjadi bagian penting dalam sistem ini, di mana perhitungan yang akurat dan efisien akan dilakukan secara otomatis. Terakhir, Desain Basis Data dirancang untuk memastikan data yang dibutuhkan dapat disimpan dengan aman dan dapat diakses dengan mudah.

3.2.1 Desain Arsitektur Sistem

Desain arsitektur sistem yang dibangun dalam adalah “*Clean Architecture*”. Desain arsitektur ini memisahkan lapisan rata-rata menjadi empat lapisan dengan tugas dan tanggung jawab yang berbeda setiap lapisan. Lapisan tersebut berisi *application*, *domain*, *infrastructure*, *presentation*. Lapisan tersebut akan dijelaskan dalam sub-bab 3.2.1.1 sampai 3.2.1.4. Terdapat lapisan lain di sub-bab 3.2.1.5 untuk mendukung pembangunan Aplikasi KinderFin 2.0.

3.2.1.1 Lapisan Application

Lapisan *Application* merupakan lapisan yang bertanggung jawab pada logika utama bisnis dalam sistem. Pada lapisan ini pada *file command* penerapannya pada Aplikasi KinderFin 2.0 yaitu pada modul *Salary Detail* yaitu dengan menghitung gaji harian guru berdasarkan data presensi dan menerapkan berbagai jenis potongan keterlambatan. Fungsi ini dimulai dengan mengambil seluruh aturan potongan keterlambatan dari basis data tabel *potongan_keterlambatan*. Selanjutnya, sistem memproses data presensi baik dari *file CSV* atau dari *xls* atau *xlsx* maupun *Input Manual*. Lalu, untuk setiap data presensi, sistem mencari data guru berdasarkan NIP dan mengambil komponen gaji sesuai jabatan. Setelah itu, sistem menghitung menit keterlambatan dan pulang cepat, menerapkan potongan sesuai aturan yang berlaku. Terakhir, sistem menyimpan hasil perhitungan ke tabel *detail_salary* dan mengembalikan jumlah data yang berhasil diproses serta dalam contoh modul lain yaitu pada master jabatan yaitu dengan Pembuatan data atau Operasi *Create*. Pembuatan data pada master jabatan dengan cara memasukkan data nama jabatan serta nominal gaji harian (*gaji1-10*) dan gaji pokok (*gajipokok1-10*) namun gaji tersebut diambil label. Selain itu lapisan *Application* terdapat *file* yaitu *query* yang memiliki fungsi yaitu menangani operasi yang hanya membaca data tanpa mengubah statusnya. Pada contoh modul *Salary Detail* yaitu, untuk mengambil Detail Gaji Guru Berdasarkan *ID* dan Tanggal. Sedangkan contoh penerapan pada modul master jabatan yaitu untuk mengimplementasikan *query pattern* dalam operasi pembacaan data master jabatan dan memiliki alur penerapan prinsip *separation of concerns*. logika pengambilan data jabatan dikelola melalui *service layer*.

3.2.1.2 Lapisan Domain

Lapisan *Domain* merupakan inti dari aplikasi yang berfokus pada logika dan aturan bisnis. Lapisan ini tidak bergantung pada lapisan lain dan tetap independen, menjadikannya bagian paling stabil dari sistem. Penerapan pada Aplikasi KinderFin 2.0 yaitu salah satunya pada modul *Salary Detail*. Terdapat pada *file get-salary-detail.query.ts* dengan fungsi *query* yaitu data dan aturan bisnis yang terkait dengan objek dunia nyata, seperti informasi guru dan

detail penggajian serta contoh lain penerapan untuk *query* lainnya pada modul Master Jabatan memiliki fungsi yaitu mengimplementasikan *query pattern* dalam operasi pembacaan data master jabatan dan memiliki alur penerapan prinsip *separation of concerns*. logika pengambilan data jabatan dikelola melalui *service layer*. Untuk contoh penerapan *service* terdapat modul Rekap Bonus yaitu memiliki fungsi untuk membuat record rekap bonus baru melalui *repository layer*.

3.2.1.3 Lapisan Infrastruktur

Lapisan *Infrastruktur* bertanggung jawab untuk implementasi teknis yang diperlukan untuk mendukung aplikasi, seperti integrasi dengan sistem eksternal dan penyimpanan data. Terdapat dua *file* yaitu mengenai migrasi tabel serta *repository*. Contoh penerapan pada modul *Salary Detail* yaitu berfungsi untuk menyimpan atau memperbarui data gaji harian guru di basis data. Serta contoh lain pada modul Rekap Bonus yaitu berfungsi untuk menyimpan data rekap bonus baru ke basis data dengan validasi yang lengkap. Terdapat *file* lain yang terdapat lapisan ini untuk menunjang berjalannya modul tersebut seperti contoh *file pdf-generator.service.ts* pada modul *Salary Detail* memiliki tugas yaitu, membuat *file PDF* yang berisi mengenai detail gaji pokok dan total gaji harian serta uang tambahan berupa bonus.

3.2.1.4 Lapisan Presentation

Lapisan *Presentation* berfungsi untuk menangani interaksi dengan pengguna, baik melalui antarmuka pengguna (UI) ataupun sistem eksternal seperti *API*. Pada aplikasi penggajian, lapisan ini terdiri dari dua komponen utama yaitu *controller* dan *router*. Penerapan pada Aplikasi ini dengan contoh pada modul *Salary Detail* yaitu, untuk fungsi memiliki fungsi untuk mengambil seluruh data gaji setiap guru. Contoh penerapan lainnya pada modul Rekap Bonus yaitu memiliki fungsi yaitu untuk menangani *HTTP* request pembuatan rekap bonus dengan validasi *input* dan *response handling*. Contoh pada Modul Pengaturan Gaji aktif berfungsi untuk untuk menangani permintaan *HTTP GET* untuk mengambil daftar pengaturan gaji aktif. Komponen selanjutnya yaitu, *Router*. *Router* Menangani rute *HTTP*, menentukan endpoint mana yang harus diproses oleh *controller* berdasarkan URL dan metode *HTTP* yang digunakan. Contoh penerapan pada Aplikasi KinderFin 2.0 yaitu pada Modul *Activity Logs* yaitu untuk *endpoint API* untuk modul *activity log* yang berfungsi untuk mencatat dan menampilkan riwayat aktivitas pengguna dalam sistem KinderFin 2.0. Serta pada Modul Pengajuan Perubahan Gaji berfungsi untuk mengkonfigurasi seluruh endpoint *API* untuk modul pengajuan perubahan gaji dengan implementasi *middleware autentikasi* dan *autorisasi*. Setelah *Router* telah di konfigurasi. URL dan *path* atau jalur setiap modul akan didaftarkan pada *file server.ts* agar terbaca saat Pembangunan program dan program tersebut dijalankan.

3.2.1.5 Lapisan Lain

Lapisan lain terdapat pada beberapa modul yaitu pada modul Pengajuan Perubahan Gaji yaitu *file upload.middleware.ts* memiliki tugas yaitu menangani proses *upload file* foto bukti dan foto gaji dengan implementasi konfigurasi penyimpanan. Lapisan lainnya yaitu pada Modul *Salary Detail* dengan fungsi *readExcelFile* pada *file upload.controller.ts* untuk Pembacaan *File Excel* memiliki fungsi yaitu untuk membaca dan memparse *file Excel* menjadi format data presensi yang dapat diproses sistem.

3.2.2 Desain Antarmuka

Desain antarmuka akan menghubungkan dari komponen sistem dan pengguna untuk memudahkan penggunaan dalam aplikasi. Tidak hanya desain arsitektur, desain antarmuka juga sangat penting untuk menunjang kenyamanan pengguna saat menggunakan Aplikasi KinderFin 2.0. Berikut adalah daftar tampilan desain antarmuka Aplikasi KinderFin 2.0.

3.2.2.1 Halaman Utama *Dashboard* - Bendahara

Halaman Utama yang akan dibuka Bendahara saat pertama kali setelah *login*. Pada perancangan ini terdapat beberapa fitur yang dapat dilihat pada Gambar 3.10. Terdapat Pengaturan Gaji untuk menambahkan pengguna. Rincian Gaji Pegawai untuk melihat daftar detail gaji seluruh pengguna, Data Master Jabatan untuk mengatur nominal dari komponen gaji setiap jabatan, Pengajuan Perubahan Gaji untuk mengajukan perubahan gaji setiap pegawai jika ada ketidaksesuaian, Log Riwayat Keuangan untuk pencatatan keuangan, Formulir Perubahan Keuangan untuk memasukkan mencatat keuangan, serta Fitur *Setting* untuk mengatur nama gaji serta pengaturan nama komponen gaji.



Gambar 3.10 Desain Antarmuka Halaman Utama Bendahara

3.2.2.2 Halaman Pengaturan Gaji - Bendahara

Pada Gambar 3.11 Bendahara dapat memasukkan nama pengguna serta jabatan yang sesuai dengan jabatan tersebut. Rancangan fitur ini dikembangkan dengan tujuan untuk mempermudah proses administratif Bendahara dalam menambahkan pengguna baru dengan berbagai tingkatan jabatan yang berbeda. Sistem ini memungkinkan kategorisasi yang jelas berdasarkan hierarki jabatan, sehingga perhitungan gaji dapat dilakukan secara otomatis sesuai dengan standar yang telah ditetapkan untuk setiap posisi. Berikut adalah visualisasi desain Halaman Pengaturan Gaji.

The screenshot shows the 'Pengaturan Gaji' (Salary Configuration) page. At the top, there's a header with the KinderFin logo, the role 'BENDAHARA', and a 'Kembali' (Back) button. Below the header, the title 'Pengaturan Gaji' is centered. To the right of the title is a back arrow icon and the word 'Kembali'. The main form area has a title 'Input Baru' (New Input). It contains two input fields: 'Nama Pengguna' (User Name) with a placeholder 'Masukan Nama' (Enter Name) and 'Jabatan' (Position) with a dropdown menu currently showing 'Jabatan'. There is also a small downward arrow icon next to the 'Jabatan' field.

Gambar 3.11 Desain Halaman Pengaturan Gaji

3.2.2.3 Halaman Fitur Setting – Bendahara

Gambar 3.12 merupakan panel konfigurasi administratif yang memberikan kendali penuh kepada bendahara dalam mengelola aktivasi dan penonaktifan komponen penggajian secara dinamis. Antarmuka ini dirancang dengan pendekatan modular yang memungkinkan penyesuaian sistem sesuai dengan kebutuhan operasional. Bagian utama dari halaman ini adalah Bendahara dapat mengaktifkan atau nonaktifkan menggunakan sistem *toggle* serta dapat menambah fitur gaji lainnya. Halaman Fitur Setting akan divisualisasikan pada Gambar 3.12



Gambar 3.12 Desain Antarmuka Halaman Fitur *Setting*

3.2.2.4 Halaman Rincian Gaji Pegawai – Bendahara

Halaman ini merupakan modul utama dalam sistem pengelolaan gaji yang menyediakan akses langsung terhadap data penggajian seluruh pengguna. Antarmuka ini terbagi menjadi dua komponen fungsional yang saling terintegrasi untuk mendukung proses administrasi keuangan. Gambar 3.13 menampilkan tabel rincian gaji pegawai yang memuat informasi identitas berupa ID, nama, dan jabatan setiap pegawai. Setiap baris data dilengkapi dengan tombol aksi "Rincian" yang berfungsi sebagai menuju halaman yang mengakses detail pembayaran setiap pengguna. Kolom status pembayaran gaji menggunakan indikator visual berupa *toggle switch* yang memungkinkan Bendahara untuk memantau dan mengubah status pembayaran secara *real-time*. Fitur pencarian pada bagian atas halaman memfasilitasi proses penelusuran data pegawai berdasarkan nama dengan efisiensi tinggi.

Rincian Gaji Pegawai					Kembali
ID	Nama	Jabatan	Aksi	Status Pembayaran Gaji	
1	(Nama Kepala Sekolah)	Kepala Sekolah			
2	(Nama Guru)	Guru			

Gambar 3.13 Desain Antarmuka Halaman Rincian Gaji Pegawai Daftar Pegawai pada Modul Bendahara

Gambar 3.14 menyajikan detail struktur penggajian dalam format tabel yang mencakup seluruh komponen finansial. Data yang ditampilkan meliputi tanggal, waktu kehadiran, gaji harian, tunjangan, bonus, potongan pajak, potongan BPJS, dan total gaji. Informasi ini disusun secara kronologis berdasarkan periode kerja, sehingga memberikan visibilitas penuh terhadap perhitungan gaji setiap pegawai.

No	Tanggal	Kehadiran	Gaji (Harian)	Tunjangan	Bonus	Potongan Pajak	Potongan BPJS	Total Gaji
1	01/12/2024	06:45	Rp100,000	Rp50,000	Rp20,000	Rp10,000	Rp5,000	Rp155,000
2	02/12/2024	06:50	Rp100,000	Rp40,000	Rp15,000	Rp8,000	Rp6,000	Rp161,000

Gambar 3.14 Desain Antarmuka Halaman Rincian Gaji Setiap Pengguna

3.2.2.5 Halaman Master Jabatan – Bendahara

Halaman ini merupakan modul administrasi yang dirancang khusus untuk Bendahara dalam mengelola data master jabatan dalam sistem penggajian. Gambar 3.15 menyediakan kontrol penuh terhadap informasi jabatan yang menjadi dasar perhitungan kompensasi pegawai di seluruh organisasi. Tabel data master jabatan menampilkan informasi yang mencakup ID jabatan sebagai identifikasi unik, nama jabatan, besaran gaji pokok, dan jumlah tunjangan yang telah ditetapkan untuk setiap posisi. Setiap entri jabatan dilengkapi dengan kolom aksi yang terdiri dari tombol "Edit" berwarna biru untuk modifikasi data dan tombol "Hapus" berwarna merah untuk penghapusan *record* jabatan yang tidak lagi diperlukan. Fitur utama halaman ini memungkinkan Bendahara untuk melakukan operasi CRUD (*Create, Read, Update, Delete*) terhadap data jabatan secara efisien.

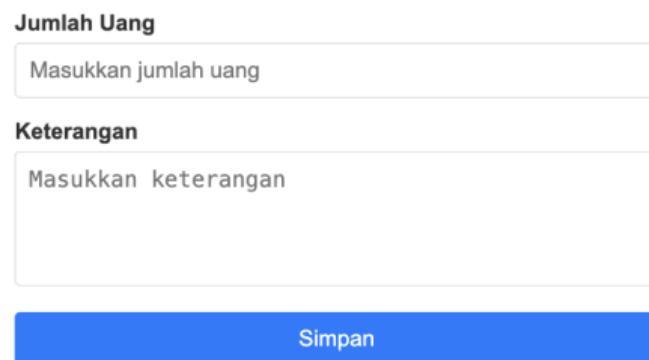
ID	Jabatan	Gaji Pokok	Tunjangan	Aksi
1	Kepala Sekolah	10,000,000	3,000,000	<button>Edit</button> <button>Hapus</button>
2	Guru	5,000,000	1,000,000	<button>Edit</button> <button>Hapus</button>

Gambar 3.15 Desain Antarmuka Halaman Master Jabatan

3.2.2.6 Halaman Formulir Perubahan Keuangan – Bendahara

Halaman Formulir Perubahan Keuangan – Bendahara untuk memudahkan Bendahara atau Administrator memantau keuangan dengan cara mencatat pengeluaran keuangan agar mendapat rincian keuangan dengan jelas. Gambar 3.16 khusus yang dirancang untuk pengguna dengan *role* Bendahara dalam mengajukan perubahan data keuangan. Formulir menyediakan dua *field input* utama yang memungkinkan Bendahara untuk mengajukan modifikasi terhadap informasi keuangan. *Field* "Jumlah Uang" dilengkapi dengan *placeholder* "Masukkan jumlah uang" yang memfasilitasi *input* nominal keuangan yang akan diajukan untuk perubahan. Sementara *field* "Keterangan" dengan area text yang lebih luas memungkinkan Bendahara memberikan penjelasan detail mengenai alasan dan konteks perubahan keuangan yang diajukan. Tombol "Simpan" berwarna biru yang terletak di bagian bawah form berfungsi untuk mengirimkan pengajuan perubahan keuangan ke sistem untuk diproses lebih lanjut oleh pihak yang berwenang.

Form Perubahan Keuangan



Formulir Perubahan Keuangan

Jumlah Uang
Masukkan jumlah uang

Keterangan
Masukkan keterangan

Simpan

Detailed description: This is a screenshot of a web-based form titled 'Form Perubahan Keuangan'. It contains two input fields: one for 'Jumlah Uang' (Amount) with placeholder text 'Masukkan jumlah uang' and another for 'Keterangan' (Description) with placeholder text 'Masukkan keterangan'. Below the fields is a blue 'Simpan' (Save) button.

Gambar 3.16 Desain Antarmuka Halaman Formulir Perubahan Keuangan

3.2.2.7 Halaman Log Riwayat Keuangan – Bendahara

Halaman Log Riwayat Keuangan – Bendahara menampilkan catatan historis lengkap dari seluruh transaksi keuangan yang telah terjadi dalam sistem. Gambar 3.17 dirancang untuk Bendahara sebagai modul pemantauan dan *audit trail* keuangan. Tabel log menampilkan informasi yang mencakup nomor urut, nominal transaksi dengan indikator visual warna hijau untuk penambahan dan merah untuk pengurangan, ID pengguna yang terlibat, *timestamp* yang menunjukkan waktu tepat transaksi terjadi, dan keterangan detail yang menjelaskan konteks setiap transaksi keuangan. Data yang tercatat mencakup berbagai jenis transaksi seperti pembayaran listrik, pengembalian pinjam, dan pembayaran SPP, memberikan visibilitas penuh terhadap alur keuangan organisasi untuk keperluan monitoring dan pertanggungjawaban.

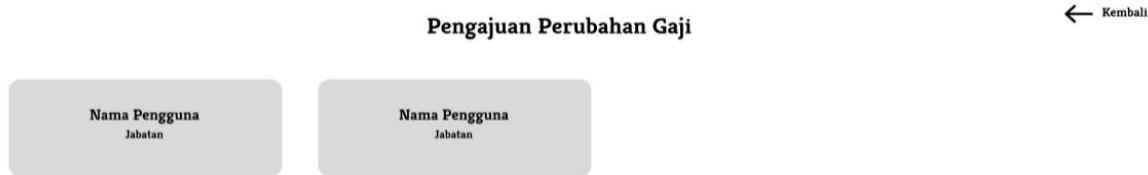
Log Riwayat Keuangan

No	Nominal	ID Pengguna	Time Stamp	Keterangan
1	+500,000	G001	2024-12-10 10:30:45	Pembayaran Listrik
2	-150,000	G002	2024-12-10 11:15:22	Pembayaran listrik
3	+200,000	G003	2024-12-10 12:00:00	Pengembalian Pinjam oleh ID G5020
4	-50,000	G001	2024-12-10 13:45:10	ID G2002 Pinjam untuk Bayar SPP

Gambar 3.17 Desain Antarmuka Halaman Log Riwayat Keuangan

3.2.2.8 Halaman Pengajuan Perubahan Gaji – Bendahara

Tampilan ini terdiri dari dua bagian utama yang saling terintegrasi untuk mendukung proses verifikasi dan validasi pengajuan perubahan gaji. Bagian pertama menampilkan detail lengkap pengajuan individual yang mencakup informasi identitas pengaju berupa nama pengguna dan jabatan. Gambar 3.18 menampilkan *dashboard* ringkas yang berisi daftar pengajuan masing-masing menampilkan "Nama Pengguna" dan "Jabatan" untuk memberikan ringkasan cepat terhadap seluruh pengajuan yang sedang dalam proses tinjau oleh Bendahara.



Gambar 3.18 Desain Halaman Utama Pengajuan Perubahan Gaji

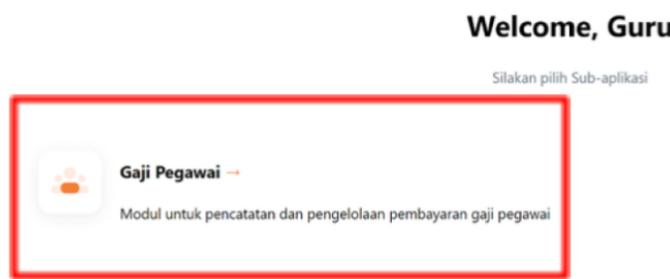
Komponen visual pada Gambar 3.19 terdiri dari tiga elemen utama: area "Foto bukti lampiran" dan "Foto bukti Gaji" yang menyediakan ruang untuk dokumentasi pendukung pengajuan, serta panel "KETERANGAN" yang menampilkan penjelasan detail dari pengaju mengenai alasan perubahan gaji yang diminta.



Gambar 3.19 Desain Antarmuka Detail Setiap Pengajuan Perubahan Gaji

3.2.2.9 Halaman Utama *Dashboard – Guru*

Halaman utama dashboard guru merupakan antarmuka awal yang ditampilkan setelah pengguna dengan peran guru berhasil melakukan autentikasi ke dalam Aplikasi KinderFin 2.0. Antarmuka dashboard ini menampilkan menu sub-aplikasi yang tersedia, dengan fokus utama pada modul "Gaji Pegawai" yang memungkinkan guru untuk melakukan pencatatan dan pengelolaan pembayaran gaji pegawai sekolah. Gambar 3.20 mengutamakan prinsip kesederhanaan dan kemudahan penggunaan, dengan tata letak yang intuitif sehingga guru dapat dengan mudah mengidentifikasi dan mengakses fitur yang diperlukan tanpa mengalami kesulitan navigasi yang signifikan.



Gambar 3.20 Desain Antarmuka Halaman Utama *Dashboard Guru*

3.2.2.10 Halaman Detail Gaji – Guru

Halaman gaji merupakan modul utama dalam Aplikasi KinderFin 2.0 yang dirancang khusus untuk pengelolaan dan pencatatan data penggajian. Gambar 3.21 menampilkan tabel data gaji pegawai yang mencakup informasi detail mengenai komponen-komponen penggajian,

seperti gaji harian, tunjangan, bonus, potongan pajak, potongan BPJS, hingga total gaji yang diterima pegawai. Sistem pencatatan dilengkapi dengan fitur timestamp yang mencatat waktu kehadiran pegawai, sehingga memungkinkan perhitungan gaji yang akurat berdasarkan jam kerja efektif. Desain halaman mengutamakan kemudahan akses informasi dengan tata letak tabular yang memungkinkan guru untuk melakukan analisis dan verifikasi data penggajian dengan efisien dan akurat.

No	Tanggal	Kehadiran	Gaji (Harian)	Tunjangan	Bonus	Potongan Pajak	Potongan BPJS	Total Gaji
1	01/12/2024	06:45	Rp100,000	Rp50,000	Rp20,000	Rp10,000	Rp5,000	Rp155,000
2	02/12/2024	06:50	Rp100,000	Rp40,000	Rp15,000	Rp8,000	Rp6,000	Rp161,000

Gambar 3.21 Desain Antarmuka Halaman Detail Gaji Guru

3.2.3 Desain Algoritma Perhitungan Gaji

Pada sub-bab 3.2.3 dirancang untuk mengotomatisasi proses penghitungan gaji guru berdasarkan kehadiran dan pelanggaran yang dilakukan. Penggunaan pendekatan if-else sebagai struktur kontrol utama untuk menentukan jenis dan besaran potongan yang akan diterapkan pada setiap komponen gaji. Sub-bab 3.2.3 akan menjelaskan bagaimana memproses data kehadiran guru, membandingkannya dengan standar waktu yang telah ditetapkan, kemudian menghitung potongan gaji berdasarkan tingkat pelanggaran yang terjadi.

3.2.3.1 Persiapan Data Input

Sebelum dimulai penghitungan gaji pada sistem, diperlukan *input* seperti tersedianya nominal gaji serta nama jabatan, data potongan gaji, serta data NIP jabatan serta nama guru sebagai objek penghitungan. Kode Semu 3.1 memastikan bahwa semua data yang diperlukan tersedia sebelum melanjutkan ke proses perhitungan. Jika ada data yang tidak valid, proses akan dihentikan dengan pesan error yang sesuai, sehingga sistem dapat memberikan *feedback* yang jelas kepada pengguna tentang masalah yang terjadi.

```

1. MULAI PROSES PENGAMBILAN DATA
2. INPUT NIP_guru
3. IF NIP_guru ditemukan di database THEN
4.   AMBIL data_guru berdasarkan NIP
5.   AMBIL jabatan_guru dari data_guru
6.   IF jabatan_guru valid THEN
7.     AMBIL komponen_gaji sesuai jabatan_guru
8.     AMBIL tabel_potongan sesuai jabatan_guru
9.     STATUS = "Data Valid"
10.  ELSE
11.    TAMPILKAN "Error: Jabatan tidak ditemukan"
12.    STATUS = "Error"
13.  END IF
14. ELSE
15.   TAMPILKAN "Error: NIP tidak ditemukan"
16.   STATUS = "Error"
17. END IF
18.
19. IF STATUS = "Data valid" THEN
20.   LANJUT ke proses selanjutnya
21. ELSE
22.   HENTIKAN proses
23. END IF

```

Kode Semu 3.1 Algoritma Persiapan Data *Input*

3.2.3.2 Pemrosesan Jam Masuk dan Deteksi Keterlambatan

Kode Semu 3.2 dirancang untuk mengelola catatan jam masuk guru dan mengidentifikasi potensi *keterlambatan*, kemudian menghitung potongan gaji yang sesuai. Prosesnya dimulai dengan pengambilan *input* jam masuk aktual guru, standar jam masuk yang telah ditetapkan, data spesifik guru (termasuk jabatan dan nilai dasar komponen gaji), serta tabel aturan potongan dari tabel *potongan_keterlambatan* (Baris 29a-d). Sistem terlebih dahulu memeriksa ketersediaan data *jam_masuk_aktual* (Baris 31). Apabila data tersedia, jam masuk aktual dan standar akan dikonversi ke format menit (Baris 32-33), lalu *selisih_menit_masuk* dihitung (Baris 34).

Jika *selisih_menit_masuk* bernilai positif, hal ini menandakan keterlambatan terdeteksi (Baris 35-36), dan sistem akan memanggil sub-fungsi *HITUNG_POTONGAN_BERJENJANG* (Baris 37). Fungsi tersebut berfungsi mengidentifikasi dan menghitung potongan berdasarkan aturan "Datang Terlambat" yang paling sesuai dari *tabel_potongan*, dengan mempertimbangkan jenjang waktu keterlambatan (misalnya, potongan 30% untuk 10 menit terlambat atau 50% untuk 30 menit terlambat). Hasil potongan ini akan disimpan ke daftar *potongan_masuk* diterapkan. Apabila guru datang tepat waktu (*selisih_menit_masuk* = 0) atau datang lebih awal (*selisih_menit_masuk* < 0), sistem tidak menerapkan potongan (Baris 39-43). Namun, jika *jam_masuk_aktual* tidak tersedia sama sekali, sistem akan mengidentifikasinya sebagai kasus "Tidak Presensi Masuk" (Baris 45-46). Dalam kondisi ini, sistem akan mencari semua aturan potongan di *tabel_potongan* yang berlaku untuk jabatan guru dan jenis pelanggaran "Tidak Presensi (*fingerprint*) masuk", lalu menerapkan potongan penuh (disesuaikan dengan aturan) dari nilai dasar komponen gaji yang relevan (Baris 47-52). Terakhir, semua *potongan* yang telah teridentifikasi akan disimpan sebagai *hasil_pemrosesan_masuk* (Baris 56).

```
1. FUNGSI HITUNG_POTONGAN_BERJENJANG(tipe_pelanggaran, durasi_pelanggaran,
   jabatan_guru, komponen_gaji_dasar, tabel_potongan):
2.   INISIALISASI daftar_potongan_ditemukan = []
3.
4.   UNTUK setiap jenis_komponen_gaji yang relevan dalam komponen_gaji_dasar:
5.     INISIALISASI aturan_terbaik_untuk_komponen = NULL
6.     INISIALISASI batas_menit_tertinggi = -1
7.
8.     UNTUK setiap aturan_potongan di tabel_potongan
9.       DENGAN aturan_potongan.Jabatan = jabatan_guru
10.      DAN aturan_potongan.Jenis_Pelanggaran = tipe_pelanggaran
11.      DAN aturan_potongan.Jenis_Potongan = jenis_komponen_gaji:
12.
13.        IF durasi_pelanggaran >= aturan_potongan.Batas_Menit THEN
14.          IF aturan_potongan.Batas_Menit > batas_menit_tertinggi THEN
15.            aturan_terbaik_untuk_komponen = aturan_potongan
16.            batas_menit_tertinggi = aturan_potongan.Batas_Menit
17.          END IF
18.        END IF
19.      END UNTUK
20.
21.      IF aturan_terbaik_untuk_komponen BUKAN NULL THEN
22.        nilai_potongan = komponen_gaji_dasar[jenis_komponen_gaji] *
   (aturan_terbaik_untuk_komponen.Persen_Potong / 100)
23.        TAMBAHKAN {jenis_komponen_gaji, nilai_potongan, tipe_pelanggaran +
   " " + batas_menit_tertinggi + " menit atau lebih"} ke daftar_potongan_ditemukan
24.      END IF
25.    END UNTUK
26.    KEMBALIKAN daftar_potongan_ditemukan
27.  END FUNGSI
28. MULAI PROSES JAM MASUK
29. INPUT:
  a. jam_masuk_aktual (dari data presensi)
  b. standar_jam_masuk (dari konfigurasi sistem/jabatan)
```

```

c. data_guru (termasuk Jabatan dan nilai dasar setiap komponen gaji)
d. tabel_potongan

30. INISIALISASI daftar_potongan_masuk_diterapkan = []

31. IF jam_masuk_aktual tersedia THEN
32.   KONVERSI jam_masuk_aktual ke format menit
33.   KONVERSI standar_jam_masuk ke format menit
34.   HITUNG selisih_menit_masuk = jam_masuk_aktual - standar_jam_masuk

35.   IF selisih_menit_masuk > 0 THEN // Keterlambatan terdeteksi
36.     TAMPILKAN "Keterlambatan terdeteksi: " + selisih_menit_masuk + "
menit"
37.     daftar_potongan_masuk_diterapkan = PANGGIL
HITUNG_POTONGAN_BERJENJANG("Datang Terlambat", selisih_menit_masuk,
data_guru.Jabatan, data_guru.komponen_gaji_dasar, tabel_potongan)
38.
39.   ELSE IF selisih_menit_masuk = 0 THEN
40.     TAMPILKAN "Guru datang tepat waktu"
41.   ELSE // selisih_menit_masuk < 0 (datang lebih awal)
42.     TAMPILKAN "Guru datang lebih awal"
43.   END IF
44.
45. ELSE // jam_masuk_aktual TIDAK tersedia (Tidak Presensi Masuk)
46.   TAMPILKAN "Tidak ada Presensi masuk - menerapkan potongan maksimal"
47.   UNTUK setiap aturan_potongan dalam tabel_potongan
48.     DENGAN aturan_potongan.Jabatan = data_guru.Jabatan
49.     DAN aturan_potongan.Jenis_Pelanggaran = "Tidak Presensi (fingerprint)
masuk":
50.
51.   nilai_potongan =
data_guru.komponen_gaji_dasar[aturan_potongan.Jenis_Potongan]
52.   TAMBAHKAN {aturan_potongan.Jenis_Potongan, nilai_potongan, "Tidak
Presensi Masuk"} ke daftar_potongan_masuk_diterapkan
53.   END UNTUK
54. END IF
55.
56. SIMPAN daftar_potongan_masuk_diterapkan AS hasil_pemrosesan_masuk
57. SELESAI PROSES JAM MASUK

```

Kode Semu 3.2 Pemrosesan Jam Masuk dan Deteksi Keterlambatan

3.2.3.3 Pemrosesan Jam Pulang dan Deteksi Pulang Cepat atau Tidak Presensi

Kode Semu 3.3 mengelola catatan *jam pulang guru*, dengan tujuan mendeteksi jika terjadi pulang cepat atau tidak adanya presensi pulang, dan kemudian menghitung *potongan gaji* yang sesuai. Prosesnya diawali dengan penerimaan *input* utama serupa dengan proses *jam masuk*, yaitu *jam pulang aktual guru*, *standar jam pulang*, *data guru*, dan *tabel aturan potongan* (Baris 2a-d).

Sistem akan memverifikasi adanya *jam pulang aktual* (Baris 4). Apabila *data* tersedia, *jam pulang aktual* dan *standar* akan dikonversi ke *menit* (Baris 5-6), dan *selisih_menit_pulang* dihitung (Baris 7). Perlu diperhatikan, *selisih_menit_pulang* akan bernilai positif jika guru pulang lebih cepat dari jadwal. Jika *selisih_menit_pulang* bernilai positif, mengindikasikan pulang cepat terdeteksi (Baris 8-10), sistem akan memanggil kembali *sub-fungsi* *HITUNG_POTONGAN_BERJENJANG* (Baris 10). *Fungsi* ini akan menghitung semua *potongan "Pulang Cepat"* berdasarkan *durasi selisih_menit_pulang* dan aturan yang relevan di *tabel_potongan*. Sebaliknya, jika *selisih_menit_pulang* bernilai nol atau negatif, berarti guru pulang tepat waktu atau lebih lambat dari jadwal, sehingga tidak ada *potongan* yang diterapkan (Baris 12-14).

Namun, jika *jam pulang aktual* tidak tersedia, sistem akan mengidentifikasinya sebagai kasus "Tidak Presensi Pulang" (Baris 16-17). Dalam skenario ini, sistem akan mencari semua *aturan potongan* di *tabel_potongan* yang terkait dengan *Jabatan guru* dan *jenis pelanggaran "Tidak Presensi (fingerprint) pulang"*, lalu menerapkan *potongan penuh* dari *nilai dasar*

komponen gaji yang relevan (Baris 18-23). Akhirnya, semua *potongan* yang telah ditemukan akan disimpan sebagai *hasil_pemrosesan_pulang* (Baris 27).

```

1. MULAI PROSES JAM PULANG
2. INPUT:
   a. jam_pulang_aktual (dari data presensi)
   b. standar_jam_pulang (dari konfigurasi sistem/jabatan)
   c. data_guru (termasuk Jabatan dan nilai dasar setiap komponen gaji)
   d. tabel_potongan (master data aturan potongan gaji dari administrator)

3. INISIALISASI daftar_potongan_pulang_diterapkan = []
4. IF jam_pulang_aktual tersedia THEN
5.   KONVERSI jam_pulang_aktual ke format menit
6.   KONVERSI standar_jam_pulang ke format menit
7.   HITUNG selisih_menit_pulang = standar_jam_pulang - jam_pulang_aktual // Positif jika pulang cepat

8.   IF selisih_menit_pulang > 0 THEN // Pulang Cepat terdeteksi
9.     TAMPILKAN "Pulang cepat terdeteksi: " + selisih_menit_pulang + " menit"
10.    daftar_potongan_pulang_diterapkan = PANGGIL
HITUNG_POTONGAN_BERJENJANG("Pulang Cepat", selisih_menit_pulang,
data_guru.Jabatan, data_guru.komponen_gaji_dasar, tabel_potongan)
11.
12.  ELSE // selisih_menit_pulang <= 0 (pulang tepat waktu atau lebih lambat)
13.    TAMPILKAN "Guru pulang tepat waktu atau lewat jadwal - tidak ada potongan"
14.  END IF
15.
16. ELSE // jam_pulang_aktual TIDAK tersedia (Tidak Presensi Pulang)
17.   TAMPILKAN "Tidak ada Presensi pulang - menerapkan potongan maksimal"
18.   UNTUK setiap aturan_potongan dalam tabel_potongan
19.     DENGAN aturan_potongan.Jabatan = data_guru.Jabatan
20.     DAN aturan_potongan.Jenis_Pelanggaran = "Tidak Presensi (fingerprint) pulang":
21.       nilai_potongan =
data_guru.komponen_gaji_dasar[aturan_potongan.Jenis_Potongan]
22.       TAMBAHKAN {aturan_potongan.Jenis_Potongan, nilai_potongan, "Tidak Presensi Pulang"} ke daftar_potongan_pulang_diterapkan
23.   END UNTUK
24. END IF
25.
26.
27. SIMPAN daftar_potongan_pulang_diterapkan AS hasil_pemrosesan_pulang
28. SELESAI PROSES JAM PULANG

```

Kode Semu 3.3 Pemrosesan Jam Pulang dan Deteksi Pulang Cepat

3.2.3.4 Penghitungan Gaji Final

Pada sub-bab 3.2.3.4 ini, akan dijelaskan secara rinci bagaimana sistem mengelola perhitungan total gaji setelah penerapan berbagai potongan, dengan fokus khusus pada penggunaan struktur kontrol *if-else* untuk menjamin akurasi dan mencegah hasil negatif atau tidak masuk akal dalam konteks penggajian. Proses perhitungan gaji dalam sistem ini dirancang untuk mencapai presisi maksimum, di mana struktur *if-else* menjadi fundamental dalam setiap tahapan penentuan nilai. Untuk setiap tanggal_iterasi (baris 8), sistem pertama-tama memeriksa keberadaan *data_presensi_harian* (baris 10). Jika *data_presensi_harian* ada, perhitungan *gaji_normal* dan *total_potongan* dilakukan (baris 11-12), dan kemudian *gaji_harian_net* dihitung (baris 13). Penting untuk dicatat bahwa *if-else* secara internal dalam fungsi *HITUNG_TOTAL_POTONGAN_HARIAN* (baris 12) memastikan bahwa setiap jenis potongan (misalnya, karena datang terlambat, pulang cepat, tidak absen) hanya diterapkan jika kondisinya terpenuhi, menghasilkan nilai potongan non-negatif. Apabila *data_presensi_harian* tidak ditemukan (baris 17), blok *ELSE* akan memastikan bahwa *gaji_harian_net* untuk tanggal tersebut diatur menjadi nol (baris 19), menghindari perhitungan yang tidak akurat akibat absen. Selain itu, *if-else* juga digunakan untuk penanganan bonus (baris 24), di mana *total_bonus* hanya akan di *input* dan ditambahkan jika pengguna mengonfirmasi adanya bonus. Melalui

penerapan struktur *if-else* yang cermat ini, baik pada level validasi data presensi maupun dalam fungsi perhitungan potongan, sistem secara efektif mencegah skenario di mana total gaji menghasilkan nilai negatif atau tidak masuk akal, menjamin bahwa setiap komponen gaji dihitung secara logis dan benar, serta memberikan hasil penggajian yang akurat dan transparan bagi guru.

```

1. MULAI PENGHITUNGAN GAJI PERIODE
2. INPUT NIP_guru, periode_gaji (tanggal_mulai, tanggal_akhir)
3. AMBIL jabatan_guru berdasarkan NIP_guru
4. // Inisialisasi variabel akumulator dan daftar detail
5. akumulasi_gaji_harian_net = 0
6. jumlah_hari_dihitung = 0
7. detail_gaji_harian = []

8. UNTUK setiap tanggal_iterasi dari tanggal_mulai HINGGA tanggal_akhir:
9.     AMBIL data_presensi_harian pada tanggal_iterasi

10.    IF data_presensi_harian ada THEN
11.        gaji_normal = AMBIL_GAJI_NORMAL_HARIAN(jabatan_guru)
12.        total_potongan = HITUNG_TOTAL_POTONGAN_HARIAN(data_presensi_harian,
jabatan_guru)
13.        gaji_harian_net = gaji_normal - total_potongan

14.        akumulasi_gaji_harian_net += gaji_harian_net
15.        jumlah_hari_dihitung += 1
16.        TAMBAH detail_gaji_harian(tanggal_iterasi, gaji_harian_net,
gaji_normal, total_potongan, data_presensi_harian.jam_masuk,
data_presensi_harian.jam_keluar)
17.    ELSE
18.        // Gaji 0 jika tanpa presensi atau data tidak ada
19.        TAMBAH detail_gaji_harian(tanggal_iterasi, 0, 0, 0, NULL, NULL)
20.    END IF
21. END UNTUK

22. // Hitung Bonus
23. total_bonus = 0
24. IF INPUT_APakah_ADA_BONUS() = TRUE THEN
25.     total_bonus = INPUT_JUMLAH_BONUS()
26.     // Catatan: Keterangan bonus (misal: INPUT_KETERANGAN_BONUS()) bisa
disimpan jika diperlukan.
27. END IF

28. // Hitung total gaji final untuk periode
29. total_gaji_final = akumulasi_gaji_harian_net + total_bonus

30. // Buat dan kembalikan laporan gaji
31. LAPORAN_GAJI = {
32.     "NIP": NIP_guru,
33.     "Periode": periode_gaji,
34.     "Akumulasi_Gaji_Harian_Net": akumulasi_gaji_harian_net,
35.     "Jumlah_Hari_Dihitung": jumlah_hari_dihitung,
36.     "Detail_Harian": detail_gaji_harian,
37.     "Total_Bonus": total_bonus,
38.     "Total_Gaji_Final": total_gaji_final
39. }
40. RETURN LAPORAN_GAJI

```

Kode Semu 3.4 Penghitungan Gaji Final

3.2.4 Desain Basis Data

Desain basis data untuk Tugas Akhir ini, yang dikembangkan untuk mendukung Aplikasi KinderFin 2.0 menggunakan *PostgreSQL*, mencakup berbagai tabel inti untuk modul penggajian. Dimulai dengan master_jabatan, tabel ini berfungsi sebagai master data yang menyimpan informasi mengenai gaji harian per jabatan. Selanjutnya, master_jabatan_pokok ini berisi detail gaji pokok untuk setiap jabatan. Untuk pencatatan seluruh aktivitas administratif terkait penggajian, terdapat *activity_logs*. Mengenai rincian komponen dan potongan gaji harian yang sudah terproses, *detail_salary* memegang peranan penting. Sementara itu, pengajuan_perubahan_gaji ini menampung permohonan perubahan gaji pengguna. Untuk

mendefinisikan label komponen gaji, digunakan pengaturan_gaji_aktif. Adapun potongan_keterlambatan, tabel ini merinci seluruh jenis potongan gaji per jabatan. Terakhir, rekap_bonus ini berisi detail bonus yang diberikan, di mana keseluruhan skema dan relasi antar tabel dapat dilihat pada Gambar 3.22, dengan penjelasan lebih lanjut pada kolom "Fungsi Kolom" setiap tabel, sementara tabel *users* dan *teachers* merupakan bagian dari struktur aplikasi versi sebelumnya (Akbar, 2024).



Gambar 3.22 Diagram *Physical Data Model* (PDM) atau Skema Basis Data Aplikasi KinderFin 2.0 Modul Penggajian

Setiap tabel didesain dengan satu atau lebih kolom yang berfungsi sebagai *Primary Key* (PK), yaitu atribut yang secara unik mengidentifikasi setiap *record* data, contohnya *id* pada tabel *users* atau *jabatan* pada *master_jabatan*. Guna membangun koneksiitas antar tabel,

Foreign Key (FK) diimplementasikan sebagai kolom dalam satu tabel yang mereferensikan *Primary Key* di tabel lain, sehingga memastikan keterkaitan data yang utuh dan membentuk hubungan antar entitas. Sebagai ilustrasi, kolom *teacher_id* pada tabel *detail_salary* dan *rekap_bonus* berfungsi sebagai *Foreign Key* yang merujuk pada *id* di tabel *teachers*, mengindikasikan bahwa setiap rincian gaji dan rekап bonus memiliki asosiasi spesifik dengan seorang guru. Demikian pula, *user_id* pada tabel *pengajuan_perubahan_gaji* merupakan *Foreign Key* yang merujuk pada *id* di tabel *users*. Pola relasi yang dominan dalam skema ini adalah *one to many* (1:M), contohnya hubungan antara tabel *teachers* dengan *detail_salary*, di mana satu entitas guru dapat memiliki banyak *record* rincian gaji harian. Demikian pula, satu jabatan yang terdefinisi di *master_jabatan* dapat terasosiasi dengan banyak guru pada tabel *teachers*.

3.2.4.1 Tabel *activity_logs*

Tabel 3.10 menyimpan mengenai pencatatan seluruh aktivitas Administrator dan Bendahara mengenai aktivitas penggajian. Tabel ini dapat menyimpan data berupa aksi, tanggal serta modul apa yang sedang diproses oleh Bendahara atau Administrator.

Tabel 3.10 Struktur Tabel *activity_logs*

Nama Kolom	Tipe Data	Fungsi Kolom	Contoh Data
<i>id</i>	uuid	Id unik untuk setiap data. Sebagai <i>Primary Key</i> atau kunci utama dalam tabel tersebut	0415015d-5d6a-4062-97ac-a291a493475e
<i>user_id</i>	uuid	Id yang diambil dari tabel “ <i>users</i> ” sebagai <i>Foreign Key</i> .	ec3da7a0-bdfb-47b1-844e-44a809fa3493
<i>email</i>	Varchar	Identitas <i>users</i> yang diambil dari tabel “ <i>users</i> ” sebagai <i>Foreign Key</i> .	rootuser@gmail.com
<i>action</i>	Varchar	Aksi yang diproses pengguna	Hapus Gaji Harian dan Pokok
<i>module</i>	Varchar	Letak Modul yang diproses pengguna	Master Jabatan
<i>description</i>	text	Penjelasan detail aksi yang diproses pengguna	Hapus Gaji Jabatan: Sekretaris
<i>created_at</i>	Timestamp with time zone	Waktu data tersebut masuk di basis data dengan zona waktu yang ditentukan	2025-05-15 03:46:34.594+00

3.2.4.2 Tabel *master_jabatan*

Tabel 3.11 berisi menyimpan nama jabatan serta detail komponen gaji harian. Gaji ini dapat diperbarui serta dihapus oleh Administrator dan Bendahara. Tabel ini yang akan nantinya menjadi dasar perhitungan gaji harian. Tabel 3.11 adalah struktur lengkap tabel *master_jabatan*.

Tabel 3.11 Struktur Tabel *master_jabatan*

Nama Kolom	Tipe Data	Fungsi Kolom	Contoh Data
<i>jabatan</i>	Varchar	Menyimpan nama atau posisi jabatan yang berlaku.	Bendahara

Nama Kolom	Tipe Data	Fungsi Kolom	Contoh Data
gaji1	<i>integer</i>	Menyimpan nilai harian untuk komponen gaji pertama. Komponen ini dapat dilabeli sesuai kebutuhan dan akan menjadi target potongan apabila ditunjuk oleh urutan pemotongan. Contoh Label : Tunjangan	15000
gaji2	<i>integer</i>	Menyimpan nilai harian untuk komponen gaji kedua. Komponen ini dapat dilabeli sesuai kebutuhan dan akan menjadi target potongan apabila ditunjuk oleh urutan pemotongan. Contoh Label : Honor Harian Kerja	20000
gaji3	<i>integer</i>	Menyimpan nilai harian untuk komponen gaji ketiga. Komponen ini dapat dilabeli sesuai kebutuhan dan akan menjadi target potongan apabila ditunjuk oleh urutan pemotongan. Contoh Label : Tunjangan Kehadiran Harian	25000
gaji4	<i>integer</i>	Menyimpan nilai harian untuk komponen gaji keempat. Komponen ini dapat dilabeli sesuai kebutuhan dan akan menjadi target potongan apabila ditunjuk oleh urutan pemotongan. Contoh Label : Uang Makan Harian	10000
gaji5	<i>integer</i>	Menyimpan nilai harian untuk komponen gaji kelima. Komponen ini dapat dilabeli sesuai kebutuhan dan akan menjadi target potongan apabila ditunjuk oleh urutan pemotongan. Contoh Label : Gaji Harian Pengawas Ujian	18000
gaji6	<i>integer</i>	Menyimpan nilai harian untuk komponen gaji keenam. Komponen ini dapat dilabeli sesuai kebutuhan dan akan menjadi target potongan apabila ditunjuk oleh urutan pemotongan. Contoh Label : Honor Harian Pelatihan	19000
gaji7	<i>integer</i>	Menyimpan nilai harian untuk komponen gaji ketujuh. Komponen ini dapat dilabeli sesuai kebutuhan dan akan menjadi target potongan apabila ditunjuk oleh urutan pemotongan. Contoh Label : Dana Kegiatan Harian Sekolah	23000

Nama Kolom	Tipe Data	Fungsi Kolom	Contoh Data
gaji8	integer	Menyimpan nilai harian untuk komponen gaji kedelapan. Komponen ini dapat dilabeli sesuai kebutuhan dan akan menjadi target potongan apabila ditunjuk oleh urutan pemotongan. Contoh Label : nsentif Harian Pembimbing Ekstrakurikuler	21000
gaji9	integer	Menyimpan nilai harian untuk komponen gaji kesembilan. Komponen ini dapat dilabeli sesuai kebutuhan dan akan menjadi target potongan apabila ditunjuk oleh urutan pemotongan. Contoh Label : Tunjangan Khusus Harian	13000
gaji10	integer	Menyimpan nilai harian untuk komponen gaji kesepuluh. Komponen ini dapat dilabeli sesuai kebutuhan dan akan menjadi target potongan apabila ditunjuk oleh urutan pemotongan. Contoh Label : Uang Saku Harian Kunjungan Lapangan	12000
Created_at	Timestamp with time zone	Menunjukkan waktu data ini pertama kali dicatat dalam basis data, lengkap dengan informasi zona waktu.	2025-05-15 04:01:34.475+00
Updated_at	Timestamp with time zone	Menunjukkan waktu terakhir data ini diperbarui dalam basis data, lengkap dengan informasi zona waktu.	2025-05-15 04:01:34.475+00

3.2.4.3 Tabel master_jabatan_pokok

Tabel 3.12 menyimpan mengenai nama jabatan serta detail komponen gaji harian. Gaji ini dapat diperbarui serta dihapus oleh Administrator dan Bendahara. Tabel ini hanya menyimpan tanpa menjadi dasar perhitungan apapun kecuali total gaji yang akan di cetak di *file pdf* oleh Bendahara atau Administrator. Tabel 3.12 struktur lengkap tabel *master_jabatan_pokok*.

Tabel 3.12 Struktur Tabel *master_jabatan_pokok*

Nama Kolom	Tipe Data	Fungsi Kolom	Contoh Data
jabatan	Varchar	Menyimpan nama atau posisi jabatan yang bersangkutan, diambil dari tabel <i>master_jabatan</i> sebagai <i>Foreign Key</i> .	Bendahara
gaji_pokok1	integer	Menyimpan nilai untuk komponen gaji pokok pertama. Komponen ini dapat dilabeli sesuai kebutuhan dan akan disertakan dalam perhitungan total gaji pokok jika	5400000

Nama Kolom	Tipe Data	Fungsi Kolom	Contoh Data
		aktif. Contoh Label : Tunjangan Jabatan	
gaji_pokok2	<i>integer</i>	Menyimpan nilai untuk komponen gaji pokok kedua. Komponen ini dapat dilabeli sesuai kebutuhan dan akan disertakan dalam perhitungan total gaji pokok jika aktif. Contoh Label : Dana Bantuan Kesehatan	4500000
gaji_pokok3	<i>integer</i>	Menyimpan nilai untuk komponen gaji pokok ketiga. Komponen ini dapat dilabeli sesuai kebutuhan dan akan disertakan dalam perhitungan total gaji pokok jika aktif. Contoh Label : Tunjangan Struktural	1500000
gaji_pokok4	<i>integer</i>	Menyimpan nilai untuk komponen gaji pokok keempat. Komponen ini dapat dilabeli sesuai kebutuhan dan akan disertakan dalam perhitungan total gaji pokok jika aktif. Contoh Label : Tunjangan Fungsional	5100000
gaji_pokok5	<i>integer</i>	Menyimpan nilai untuk komponen gaji pokok kelima. Komponen ini dapat dilabeli sesuai kebutuhan dan akan disertakan dalam perhitungan total gaji pokok jika aktif. Contoh Label : Tunjangan Koordinator Bidang Studi	1200000
gaji_pokok6	<i>integer</i>	Menyimpan nilai untuk komponen gaji pokok keenam. Komponen ini dapat dilabeli sesuai kebutuhan dan akan disertakan dalam perhitungan total gaji pokok jika aktif. Contoh Label : Insentif Kinerja Tetap	2100000
gaji_pokok7	<i>integer</i>	Menyimpan nilai untuk komponen gaji pokok ketujuh. Komponen ini dapat dilabeli sesuai kebutuhan dan akan disertakan dalam perhitungan total gaji pokok jika aktif. Contoh Label : Honorarium Pembina Ekstrakurikuler	4100000
gaji_pokok8	<i>integer</i>	Menyimpan nilai untuk komponen gaji pokok kedelapan. Komponen ini dapat dilabeli sesuai kebutuhan dan akan disertakan dalam perhitungan total gaji pokok jika	900000

Nama Kolom	Tipe Data	Fungsi Kolom	Contoh Data
		aktif. Contoh Label : Insentif Sertifikasi Pendidik	
gaji_pokok9	integer	Menyimpan nilai untuk komponen gaji pokok kesembilan. Komponen ini dapat dilabeli sesuai kebutuhan dan akan disertakan dalam perhitungan total gaji pokok jika aktif. Contoh Label : Kompensasi Masa Kerja	800000
gaji_pokok10	integer	Menyimpan nilai untuk komponen gaji pokok kesepuluh. Komponen ini dapat dilabeli sesuai kebutuhan dan akan disertakan dalam perhitungan total gaji pokok jika aktif. Contoh Label : Subsidi Transportasi	700000
Created_at	<i>Timestamp with time zone</i>	Menunjukkan waktu data ini pertama kali dibuat dalam basis data, lengkap dengan informasi zona waktu.	2025-05-15 04:01:34.475+00
Updated_at	<i>Timestamp with time zone</i>	Menunjukkan waktu terakhir data ini diperbarui dalam basis data, lengkap dengan informasi zona waktu.	2025-05-15 04:01:34.475+00

3.2.4.4 Tabel *detail_salary*

Tabel *detail_salary* menyimpan seluruh data komponen gaji harian setiap pengguna setiap harinya dengan detail potongan serta jumlah total gaji setiap pengguna setiap presensi secara detail. Tabel ini juga menyimpan data jam masuk serta keluar setiap pengguna setiap presensi. Tabel 3.13 adalah struktur tabel *detail_salary*.

Tabel 3.13 Struktur Tabel *detail_salary*

Nama Kolom	Tipe Data	Fungsi Kolom	Contoh Data
<i>id</i>	<i>uuid</i>	Kolom ini berfungsi sebagai <i>Primary Key</i> yang menyimpan identifikasi unik untuk setiap catatan gaji harian.	0415015d-5d6a-4062-97ac-a291a493475e
<i>teacher_id</i>	<i>uuid</i>	Kolom ini menyimpan identifikasi unik guru yang gajinya direkapitulasi, diambil dari tabel <i>teachers</i> sebagai <i>Foreign Key</i> . Kolom ini berfungsi untuk menghubungkan data gaji dengan identitas guru.	cc3da7a0-bdfb-47b1-844e-44a809fa3493
nama_lengkap	<i>character varying(255)</i>	Kolom ini menyimpan nama lengkap guru yang bersangkutan.	Budi Santoso
jabatan	<i>character varying(255)</i>	Kolom ini menyimpan jabatan atau posisi guru pada tanggal	Guru

Nama Kolom	Tipe Data	Fungsi Kolom	Contoh Data
		perhitungan gaji. Data ini merupakan <i>Foreign Key</i> yang merujuk pada kolom jabatan di tabel master_jabatan.	
tanggal	<i>date</i>	Kolom ini menunjukkan tanggal spesifik perhitungan gaji harian untuk guru yang bersangkutan.	2025-03-21
gaji1	<i>integer</i>	Kolom ini menyimpan nilai komponen gaji harian pertama untuk jabatan ini, yang diambil dari tabel master_jabatan dan dicatat secara utuh (tanpa pengurangan potongan secara langsung).	150000
gaji2	<i>integer</i>	Kolom ini menyimpan nilai komponen gaji harian kedua untuk jabatan ini, yang diambil dari tabel master_jabatan dan dicatat secara utuh (tanpa pengurangan potongan secara langsung).	100000
gaji3	<i>integer</i>	Kolom ini menyimpan nilai komponen gaji harian ketiga untuk jabatan ini, yang diambil dari tabel master_jabatan dan dicatat secara utuh (tanpa pengurangan potongan secara langsung).	75000
gaji4	<i>integer</i>	Kolom ini menyimpan nilai komponen gaji harian keempat untuk jabatan ini, yang diambil dari tabel master_jabatan dan dicatat secara utuh (tanpa pengurangan potongan secara langsung).	50000
gaji5	<i>integer</i>	Kolom ini menyimpan nilai komponen gaji harian kelima untuk jabatan ini, yang diambil dari tabel master_jabatan dan dicatat secara utuh (tanpa pengurangan potongan secara langsung).	80000
gaji6	<i>integer</i>	Kolom ini menyimpan nilai komponen gaji harian keenam untuk jabatan ini, yang diambil dari tabel master_jabatan dan dicatat secara utuh (tanpa pengurangan potongan secara langsung).	60000
gaji7	<i>integer</i>	Kolom ini menyimpan nilai komponen gaji harian ketujuh untuk jabatan ini, yang diambil dari tabel master_jabatan dan dicatat secara	40000

Nama Kolom	Tipe Data	Fungsi Kolom	Contoh Data
		utuh (tanpa pengurangan potongan secara langsung).	
gaji8	<i>integer</i>	Kolom ini menyimpan nilai komponen gaji harian kedelapan untuk jabatan ini, yang diambil dari tabel master_jabatan dan dicatat secara utuh (tanpa pengurangan potongan secara langsung).	90000
gaji9	<i>integer</i>	Kolom ini menyimpan nilai komponen gaji harian kesembilan untuk jabatan ini, yang diambil dari tabel master_jabatan dan dicatat secara utuh (tanpa pengurangan potongan secara langsung).	35000
gaji10	<i>integer</i>	Kolom ini menyimpan nilai komponen gaji harian kesepuluh untuk jabatan ini, yang diambil dari tabel master_jabatan dan dicatat secara utuh (tanpa pengurangan potongan secara langsung).	45000
potongan_data_ng_telat	<i>integer</i>	Menyimpan jumlah total nominal potongan gaji yang dikenakan karena guru datang terlambat. Nilai ini dihitung sebagai persentase dari komponen gaji harian tertentu (gajiX) sesuai aturan di tabel potongan_keterlambatan yang memiliki jenis_potongan 'datang telat'.	2500
potongan_pulang_cepat	<i>integer</i>	Kolom ini menyimpan jumlah total nominal potongan gaji yang dikenakan karena guru datang terlambat. Nilai ini dihitung sebagai persentase dari komponen gaji harian tertentu (gajiX) sesuai aturan pada tabel potongan_keterlambatan yang memiliki jenis_potongan 'datang telat'.	3000
potongan_tidak_absen_masuk	<i>integer</i>	Kolom ini menyimpan jumlah total nominal potongan gaji yang dikenakan karena guru pulang lebih awal. Nilai ini dihitung sebagai persentase dari komponen gaji harian tertentu (gajiX) sesuai aturan pada tabel potongan_keterlambatan yang memiliki jenis_potongan 'pulang cepat'.	1000

Nama Kolom	Tipe Data	Fungsi Kolom	Contoh Data
potongan_tidak_absen_pulang	integer	Kolom ini menyimpan jumlah total nominal potongan gaji yang dikenakan karena guru tidak melakukan presensi saat masuk kerja. Nilai ini dihitung sebagai persentase dari komponen gaji harian tertentu (gajiX) sesuai aturan pada tabel potongan_keterlambatan yang memiliki jenis_potongan 'tidak absen masuk'.	7500
jam_masuk	time without time zone	Kolom ini menyimpan jumlah nominal potongan gaji yang dikenakan karena guru tidak melakukan presensi saat pulang kerja. Nilai ini dihitung sebagai persentase dari komponen gaji harian tertentu (gajiX) sesuai aturan pada tabel potongan_keterlambatan yang memiliki jenis_potongan 'tidak absen pulang'.	07:00:00
jam_keluar	time without time zone	Kolom ini menunjukkan waktu masuk kerja guru yang tercatat pada tanggal tersebut.	16:00:00
total_salary	integer	Kolom ini menunjukkan waktu keluar kerja guru yang tercatat pada tanggal tersebut.	725000
created_at	timestamp with time zone	Kolom ini menyimpan total gaji harian bersih yang diterima guru. Nilai ini adalah hasil penjumlahan semua komponen gaji harian (gaji1 hingga gaji10 yang aktif) setelah dikurangi dengan total akumulasi seluruh nilai potongan.	2025-05-15 04:11:11.809+00

3.2.4.5 Tabel pengajuan_perubahan_gaji

Tabel 3.14 menyimpan list pengajuan perubahan gaji yang diajukan oleh guru yang nantinya akan disetujui atau ditolak oleh Administrator atau Bendahara. Tabel ini menyimpan path atau jalur penyimpanan foto di server yang akan diakses oleh sistem untuk menampilkan foto dengan mengambil path tersebut. Tabel 3.14 struktur lengkap tabel pengajuan_perubahan_gaji.

Tabel 3.14 Struktur Tabel pengajuan_perubahan_gaji

Nama Kolom	Tipe Data	Fungsi Kolom	Contoh Data
id	uuid	Id unik untuk setiap data. Sebagai Primary Key atau kunci utama dalam tabel tersebut	11df23fc-944d-402e-a046-1c50ccf59a2a

Nama Kolom	Tipe Data	Fungsi Kolom	Contoh Data
<i>user_id</i>	<i>character varying(255)</i>	ID pengguna yang mengajukan perubahan gaji. Diambil dari tabel <i>users</i> sebagai <i>Foreign Key</i> .	<u>contohuser@gmail.com</u>
<i>foto_bukti_path</i>	<i>character varying(255)</i>	Path atau lokasi <i>file</i> foto bukti pendukung pengajuan	uploads/010203040506070809/bukti-1746065159514-WhatsApp_Image_2025-04-09 at 20.22.12.jpeg
<i>foto_gaji_path</i>	<i>character varying(255)</i>	Path atau lokasi <i>file</i> foto slip gaji atau dokumen gaji	uploads/010203040506070809/gaji-1746065159515-WhatsApp_Image_2025-04-09 at 20.22.12.jpeg
keterangan	<i>text</i>	Keterangan atau alasan detail mengenai pengajuan perubahan gaji	Pengajuan kenaikan gaji karena promosi jabatan
tanggal	<i>date</i>	Tanggal pengajuan perubahan gaji dibuat	2025-04-28
<i>created_at</i>	<i>timestamp with time zone</i>	Waktu data tersebut masuk di basis data dengan zona waktu yang ditentukan	2025-05-23 06:43:25.656+00
<i>updated_at</i>	<i>timestamp with time zone</i>	Waktu data tersebut diperbarui terakhir kali	2025-01-20 14:22:10.123+00
status	<i>enum_pengajuan_perubahan_gaji_status</i>	Status pengajuan (<i>pending/approved/rejected</i>)	<i>rejected</i>
<i>approved_by</i>	<i>character varying(255)</i>	Nama atau ID pengguna yang menyetujui pengajuan	rootuser@gmail.com
<i>approved_at</i>	<i>timestamp with time zone</i>	Waktu pengajuan disetujui atau ditolak	2025-05-23 06:53:03.025+00
<i>rejection_reason</i>	<i>text</i>	Alasan penolakan jika pengajuan ditolak	Dokumen pendukung tidak lengkap

3.2.4.6 Tabel pengaturan_gaji_aktif

Tabel 3.15 berisi mengenai pemberian nama label gaji harian dan gaji pokok yang diambil dari Tabel master_jabatan dan Tabel master_jabatan_pokok. Tabel ini yang akan diambil oleh tampilan antarmuka untuk menggantikan nama gaji1 sampai gaji 10 dan dari Tabel master_jabatan serta sebagai acuan untuk tabel potongan_keterlambatan sehingga pengguna tidak mengalami kesulitan dalam memahami jenis gaji yang dimaksud.

Tabel 3.15 Struktur Tabel pengaturan_gaji_aktif

Nama Kolom	Tipe Data	Fungsi Kolom	Contoh Data
id	<i>integer</i>	Id unik untuk setiap data. Sebagai <i>Primary Key</i> atau kunci utama dalam tabel tersebut	1
field	<i>character varying(255)</i>	Nama field atau kolom gaji yang dapat dikonfigurasi	gaji1
label	<i>character varying(255)</i>	Label atau nama tampilan untuk <i>field gaji</i>	Gaji Pokok
type	<i>character varying(255)</i>	Tipe gaji (pokok atau harian)	harian
aktif	<i>boolean</i>	Status aktif/nonaktif field gaji (<i>true/false</i>)	<i>TRUE</i>
created_at	<i>timestamp with time zone</i>	Waktu data tersebut masuk di basis data dengan zona waktu yang ditentukan	2025-01-15 08:30:15.594+00
updated_at	<i>timestamp with time zone</i>	Waktu data tersebut diperbarui terakhir kali	2025-01-20 14:22:10.123+00

3.2.4.7 Tabel potongan_keterlambatan

Tabel 3.16 berfungsi untuk menyimpan seluruh daftar potongan gaji yang dikenakan pada setiap komponen gaji berdasarkan masing-masing jabatan. Setiap entri dalam tabel ini merepresentasikan aturan pemotongan gaji yang berlaku untuk suatu jabatan tertentu, termasuk informasi mengenai komponen gaji yang dipotong, besaran potongan dalam bentuk persentase, serta batas toleransi keterlambatan dalam satuan menit. Tabel ini membuat sistem dapat mengelola dan menerapkan kebijakan pemotongan gaji secara konsisten dan akurat sesuai dengan ketentuan yang berlaku pada tiap jabatan. Berikut adalah struktur tabel potongan_keterlambatan.

Tabel 3.16 Struktur Tabel potongan_keterlambatan

Nama Kolom	Tipe Data	Fungsi Kolom	Contoh Data
id	<i>uuid</i>	ID unik untuk setiap data, sebagai <i>Primary Key</i>	3fa85f64-5717-4562-b3fc-2c963f66afa6
jabatan	<i>character varying(255)</i>	Menyimpan nama jabatan yang dikenai potongan keterlambatan, diambil dari kolom jabatan di tabel master_jabatan sebagai <i>Foreign Key</i> .	Guru
urutan_gaji_di_potong	<i>integer</i>	Menyatakan urutan kolom gaji harian pada tabel master_jabatan yang akan dikenai potongan (misalnya gaji1 = 1)	1
persen_potong	<i>integer</i>	Besar potongan dalam bentuk persentase terhadap gaji	10

Nama Kolom	Tipe Data	Fungsi Kolom	Contoh Data
batas_menit	integer	Batas keterlambatan dalam satuan menit sebelum potongan berlaku	15
Jenis_potongan	character varying	Jenis pelanggaran yang akan dikenakan ke jabatan tersebut seperti datang terlambat, pulang cepat, tidak presensi pulang atau masuk	datang_telat
created_at	timestamp with time zone	Waktu saat data pertama kali dibuat	2025-01-15 08:30:15.594+00
updated_at	timestamp with time zone	Waktu saat data terakhir diperbarui	2025-01-20 14:22:10.123+00
version	integer	Versi dari data, berguna untuk kontrol perubahan data	0

3.2.4.8 Tabel rekap_bonus

Tabel 3.17 berfungsi untuk menyimpan seluruh data rekapitulasi bonus atau uang tambahan yang diberikan kepada guru dalam periode tertentu. Setiap entri dalam tabel ini merepresentasikan pemberian bonus kepada guru berdasarkan *ID* guru, dengan informasi lengkap mengenai periode bonus, besaran uang tambahan yang diberikan, serta keterangan terkait pemberian bonus tersebut. Tabel ini membuat sistem dapat mengelola dan mencatat pemberian bonus secara sistematis dan akurat sesuai dengan periode yang telah ditentukan.

Tabel 3.17 Struktur Tabel rekap_bonus

Nama Kolom	Tipe Data	Fungsi Kolom	Contoh Data
id	uuid	ID unik untuk setiap data, sebagai <i>Primary Key</i>	3fa85f64-5717-4562-b3fc-2c963f66afa6
tanggal	date	Menyimpan tanggal pencatatan atau pemberian bonus	2025-04-01
start_date	date	Tanggal mulai periode bonus yang diberikan	2025-04-01
end_date	date	Tanggal akhir periode bonus yang diberikan	2025-04-01
teachers_id	character varying(255)	ID guru yang menerima bonus. Berfungsi sebagai <i>Foreign Key</i> yang mereferensi tabel <i>teachers</i> .	be0388ad-1d80-44fd-aba2-93b4be26f509
uang_tambahan	integer	Besaran uang tambahan atau bonus yang diberikan dalam rupiah	500000
keterangan	text	Keterangan atau catatan terkait pemberian bonus	Bonus kinerja bulan Januari
created_at	timestamp with time zone	Waktu saat data pertama kali dibuat	2025-05-04 12:57:09.728+00

Nama Kolom	Tipe Data	Fungsi Kolom	Contoh Data
<i>updated_at</i>	<i>timestamp with time zone</i>	Waktu saat data terakhir diperbarui	2025-05-04 12:57:09.728+00

3.3 Implementasi

Pada sub-bab 3.3 akan menjelaskan proses implementasi atau merealisasikan dari sistem yang dirancang pada sub-bab sebelumnya. Implementasi dilakukan dengan tujuan untuk merealisasikan hasil perancangan ke dalam bentuk sistem yang dapat dijalankan dan diuji. Proses implementasi melibatkan penggunaan perangkat keras dan perangkat lunak tertentu yang mendukung pengembangan dan pengujian aplikasi.

3.3.1 Lingkungan Implementasi

Lingkungan implementasi sistem pada tugas akhir ini menggunakan perangkat keras dan perangkat lunak yang mendukung proses perancangan, pengembangan, hingga pengujian aplikasi. Spesifikasi lengkap perangkat keras dapat dilihat pada Tabel 3.18, dan spesifikasi perangkat lunak yang digunakan dijelaskan pada Tabel 3.19

Tabel 3.18 Spesifikasi Perangkat Keras Lingkungan Implementasi

Komponen	Spesifikasi
Jenis Perangkat	Laptop MacBook Air M1
Prosesor	Apple M1 Chip 8-Core CPU
Memori	8 GB Unified Memory
Penyimpanan	256 GB SSD
Tipe Sistem	macOS 64-bit
Sistem Operasi	macOS Sequoia 15.5

Tabel 3.19 Spesifikasi Perangkat Lunak Lingkungan Implementasi

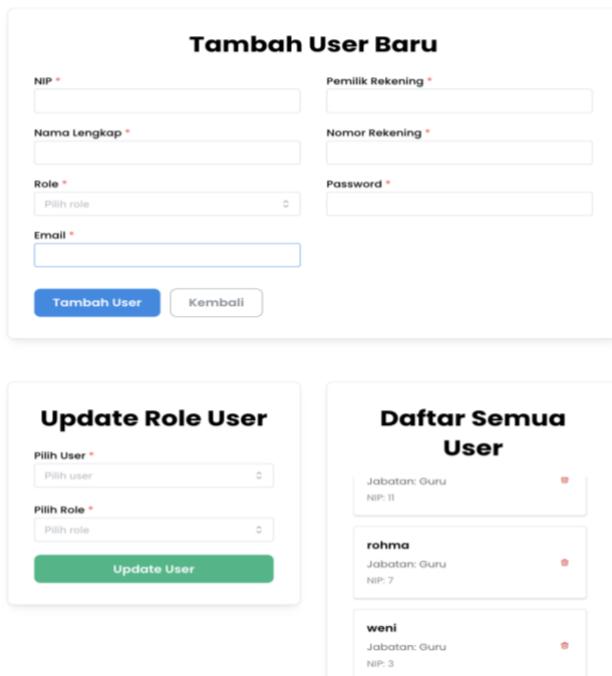
Perangkat Lunak	Fungsi Penggunaan
Visual Studio Code	Sebagai <i>code editor</i> utama dalam proses penulisan dan pengembangan kode program.
Postman	Untuk menguji dan menganalisis <i>API</i> selama proses pengembangan dan integrasi data.
Figma	Digunakan sebagai alat perancangan antarmuka pengguna (<i>UI design</i>).
<i>developer:fingerprint.io</i>	Digunakan untuk pengambilan data sidik jari perangkat (<i>device fingerprint</i>) dari <i>server</i> .
Neon Database	Basis data berbasis <i>cloud</i> yang digunakan untuk menyimpan data aplikasi secara <i>online</i> .
Render	Platform <i>hosting</i> yang digunakan untuk menyimpan <i>backend</i> aplikasi.
Vercel	Platform untuk melakukan <i>deployment frontend</i> aplikasi secara cepat dan mudah.
GitHub	Untuk menyimpan, mengelola, serta mendokumentasikan riwayat perubahan kode.

3.3.2 Implementasi Antarmuka Pengguna

Pada sub-bab 3.2.2 sudah terdapat rancangan untuk yang nantinya akan dikembangkan pada Aplikasi KinderFin 2.0. Dari beberapa rancangan tersebut, ternyata ada beberapa fitur yang sudah dikembangkan terlebih dahulu seperti Log Riwayat Keuangan serta status pembayaran gaji (sudah atau belum terbayarkan) dan *Formulir Perubahan Keuangan* serta Pengaturan Gaji (Tambah Pengguna) yang sebelumnya sudah ada dalam sistem namun belum dibuat tampilan antarmuka. Pada implementasi ini, ada beberapa rombakan dari desain sebelumnya seperti log Riwayat keuangan diubah menjadi log aktivitas serta halaman utama atau *dashboard* mengikuti desain Aplikasi KinderFin 1.0.

3.3.2.1 Halaman Tambah, Edit, dan Hapus Pengguna - Administrator

Sebelum aplikasi digunakan, yang pertama dilakukan adalah Administrator atau kepala sekolah menambahkan pengguna dan role pengguna. Karena setiap pengguna memiliki besaran gaji, potongan berbeda setiap jabatan. Berikutnya, Administrator atau Kepala Sekolah memberikan password dan email kepada setiap pengguna, agar nantinya pengguna yang terdaftar dapat melakukan proses *login* ke dalam sistem. Jika pengguna sudah tidak terdaftar menjadi pengguna, Administrator dapat menghapus pengguna tersebut pada Tabel Daftar Semua *User* pada ikon sampah. Tampilan dapat dilihat pada Gambar 3.23. Pada Halaman Utama Dashboard, untuk memenuhi kebutuhan pada Kasus Penggunaan Menambah Pengguna. dan Kasus Penggunaan Edit dan Hapus Pengguna.



Gambar 3.23 Tampilan Tambah Perbarui Peran Pengguna dan Hapus Pengguna

3.3.2.2 Halaman Pengaturan Gaji Aktif, Nominal Gaji (Master Jabatan), dan Potongan Gaji - Bendahara

Setelah memasukkan jabatan serta pengguna ke dalam *Database*, Bendahara mengaktifkan konfigurasi gaji yang akan di proses, baik gaji pokok maupun gaji harian di di fitur atau halaman aplikasi “Pengaturan Gaji Aktif” yang dapat dilihat pada Gambar 3.24. Pada Gambar 3.24, diimplementasikan untuk memenuhi kebutuhan Kasus Penggunaan Aktifkan Label Gaji.

Setelahnya, Bendahara atau Administrator dapat menambahkan, melihat detail komponen gaji serta hapus atau sunting setiap jabatan dan dapat dilihat pada Gambar 3.25 di fitur atau halaman aplikasi “Master Jabatan”. Lalu, Bendahara atau Administrator mengisi nominal gaji setiap jabatan di fitur atau halaman aplikasi “Tambah Jabatan Baru” dapat dilihat pada Gambar 3.26, Bendahara atau Administrator dapat melihat detail komponen gaji setiap jabatan pada Gambar 3.27 serta dapat memperbarui gaji di menu “edit” yang terdapat halaman rincian pada Gambar 3.28 . Pada Gambar 3.25, Gambar 3.26, Gambar 3.27, dan Gambar 3.28, dibangun dan diimplementasikan untuk memenuhi kebutuhan Kasus Penggunaan Tambah atau Ubah Gaji Harian dan Pokok.

Gambar 3.24 Tampilan Halaman Pengaturan Gaji Aktif

Jabatan	Aksi
Bendahara	edit eye delete
Sekretaris	edit eye delete
Guru	edit eye delete

Gambar 3.25 Tampilan Awal Master Jabatan

Tambah Jabatan Baru

Nama Jabatan

Gaji Harian

makan pagi	Transportasi
0	0

Gaji Pokok

Simpan Batal

Gambar 3.26 Tampilan Menambah Komponen Gaji Setiap Jabatan



Gambar 3.27 Detail Rincian Gaji Setiap Jabatan



Gambar 3.28 Tampilan Sunting Besaran Komponen Gaji Setiap Jabatan

Terakhir sebelum digunakan, Bendahara atau Administrator menetapkan potongan setiap jabatan di fitur atau halaman aplikasi “Tambah Potongan Gaji” pada untuk “gaji harian” meliputi, “datang terlambat”, “pulang cepat”, “tidak Presensi (*fingerprint*) masuk”, “tidak Presensi (*fingerprint*) pulang” untuk seluruh potongan yang berhasil ditambahkan dari Gambar 3.30. dapat dilihat di Gambar 3.29. Potongan tersebut dikelompokan berdasarkan Jabatan untuk mempermudah Bendahara atau Administrator menyortir potongan. Berikut adalah contoh Tabel

3.20 Contoh Potongan Gaji. Implementasi pada Gambar 3.29 dan Gambar 3.30 untuk memenuhi kebutuhan Kasus Penggunaan Pengaturan Potongan Gaji Harian.

Tabel 3.20 Contoh Potongan Gaji

Jabatan	Jenis Potongan	Persen Potong	Batas Menit	Jenis Pelanggaran
Guru	Makan Pagi	50	30	Datang Terlambat
Guru	Transportasi	30	10	Datang Terlambat
Guru	Transportasi	50	30	Datang Terlambat

Sistem penggajian ini dirancang untuk menerapkan potongan gaji berdasarkan jenis pelanggaran yang terjadi. Apabila terjadi pelanggaran, seperti keterlambatan datang (Datang Terlambat), sistem akan memotong komponen gaji tertentu (Jenis Potongan) berdasarkan durasi pelanggaran. Jika terdapat beberapa aturan potongan untuk Jenis Pelanggaran yang sama pada jabatan tertentu, sistem akan mengurutkan dan menerapkan potongan sesuai dengan ambang batas menit yang terdefinisi. Sebagai contoh, untuk jabatan "Guru" dengan Jenis Pelanggaran "Datang Terlambat", terdapat aturan potongan yang berbeda untuk "Makan Pagi" dan "Transportasi". Kedua Jenis Potongan ini memiliki skema berjenjang: jika keterlambatan mencapai 10 menit, potongan 30% dikenakan pada komponen tersebut, apabila keterlambatan mencapai 30 menit, potongan 50% akan diterapkan. Sistem memastikan potongan yang paling sesuai dengan ambang batas menit yang dilanggar akan diterapkan pada setiap jenis komponen gaji yang relevan.

Data Potongan Gaji

Cari Jabatan:

Urutan:

Jabatan	Gaji Dipotong	Persen Potong	Batas Menit	Jenis Pelanggaran	Aksi
Guru					
Guru	makan pagi	10%	5 menit	Datang Terlambat	
Guru	Transportasi	100%	-	Tidak Presensi Pulang	
Bendahara					
Bendahara	Transportasi	20%	5 menit	Datang Terlambat	
Bendahara	makan pagi	100%	-	Tidak Presensi Pulang	

Gambar 3.29 Tampilan Halaman Awal Potongan Gaji

Tambah Potongan Keterlambatan

Jabatan
Pilih Jabatan
Gaji Dipotong
Makan Pagi
Persen Potong (%)
0
Batas Menit
0
Tipe Jam
Pilih Tipe Jam

Simpan Kembali

Gambar 3.30 Tampilan Menambah Potongan Gaji

3.3.2.3 Halaman Rekap Gaji Pegawai - Bendahara

Mengunggah *file* Presensi *Fingerprint* dapat dilakukan jika pengiriman data dari mesin *fingerprint* ke *server* melalui *API* gagal. Kegagalan pengiriman banyak penyebabnya, contohnya kendala jaringan di lapangan atau kendala dalam pengiriman dari pihak vendor (*fingerspot*) penyedia. Proses mengunggah *file* ini akan memakan waktu tergantung ukuran file.

Duplikasi data Presensi menyebabkan penghitungan pada akhir penghitungan total gaji akan tidak valid atau salah. Duplikasi tersebut diatasi dengan dua cara yaitu dengan menyocokan nama *file* serta menyocokan isi tanggal. Jika *file* tidak sesuai dengan format dari mesin *fingerprint* maka akan tertolak oleh sistem. Jika nama *file* sama maka akan tetap diproses namun tidak akan dimasukan ke dalam perhitungan data karena sudah ada terdahulu. Namun jika melakukan *update* pada jam pulang, dapat dilakukan dengan pertimbangan setelah mengajukan pengajuan perubahan pada halaman aplikasi “Pengajuan Perubahan Gaji”. Untuk menghindari terulangnya *file* diproses ulang diberikan Catatan Aktivitas Pengguna atau “log aktivitas” dengan tercantumnya nama pengguna serta *file* apa yang diunggah oleh pengguna tersebut. *File* juga dapat diuduh kembali. Tampilan dapat dilihat Gambar 3.31 di bawah ini.

Daftar Guru dan Pengelolaan Gaji

Filter dan Aksi Cepat

Cari Nama atau NIP
Masukkan nama atau NIP

Filter Jabatan
Pilih jabatan

Upload File Absensi Excel/CSV
Pilih File Absensi

Sinkronisasi Data Absensi Fingerspot

Tanggal Mulai (Fingerspot)
YYYY-MM-DD

Tanggal Akhir (Fingerspot)
YYYY-MM-DD

Tarik & Proses Data Absensi Fingerspot

Kembali ke Dashboard

Log Aktivitas Pengelolaan Gaji

Filter Email Log
Email pengguna

Filter Tanggal Log (YYYY-MM-DD)
YYYY-MM-DD

Filter Aksi Log
Semua aksi

Tanggal & Waktu	Email Pengguna	Aksi	Keterangan Detail
23 Mei 2025, 10.22.06 WIB	rootuser@gmail.com	Hapus Gaji Harian	Menghapus data gaji harian: nip: 0000123456, tanggal: 2020-02-20

Gambar 3.31 Tampilan Halaman Utama Rekap Gaji Pegawai

Terdapat Formulir “Sinkronasi Data Absensi *Fingerspot*” berguna untuk mengambil secara otomatis yang dibatasi hanya dua hari setiap pengambilan, jika ingin mengambil selama empat hari maka harus melakukan dua kali pengambilan. Untuk meningkatkan kenyamanan pengguna agar mengetahui apakah data sudah sepenuhnya terproses atau mengalami kegagalan terprosesnya data, maka diberikan pemberitahuan sistem atau *notifikasi* apakah gagal atau berhasil terposes. Saat sedang diproses diberikan pemberitahuan file “sedang diproses” dan waktu terprosesnya. Untuk tampilan data sedang diproses terdapat pada Gambar 3.32 dan tampilan data selesai diproses dapat dilihat pada Gambar 3.33. Pada sub-bab 3.3.2.3 dibangun untuk memenuhi kebutuhan Kasus Penggunaan Tambah Data Presensi.

Daftar Guru dan Pengelolaan Gaji

Filter dan Aksi Cepat

Cari Nama atau NIP
Masukkan nama atau NIP

Filter Jabatan
Pilih jabatan

Upload File Absensi Excel/CSV

Sedang memproses file absensi. Mohon tunggu... (0 detik)

Sinkronisasi Data Absensi Fingerspot

Gambar 3.32 Tampilan Data Sedang Diproses

Daftar Guru dan Pengelolaan Gaji

Sukses
File berhasil diunggah dan diproses. ×

Filter dan Aksi Cepat

Cari Nama atau NIP
Masukkan nama atau NIP

Filter Jabatan
Pilih jabatan

Upload File Absensi Excel/CSV
Pilih File Absensi

Sinkronisasi Data Absensi Fingerspot

Tanggal Mulai (Fingerspot)
YYYY-MM-DD

Tanggal Akhir (Fingerspot)
YYYY-MM-DD

Tarik & Proses Data Absensi Fingerspot

Gambar 3.33 Tampilan Data Selesai Diproses

3.3.2.4 Halaman Manual Presensi - Bendahara

Manual Presensi dapat dilakukan pada dua kejadian, yaitu pada file Presensi yang rusak atau mesin gagal mengirimkan data ke aplikasi. Kejadian lainnya yaitu, ada pengajuan perubahan gaji yang diajukan oleh pengguna. Manual Presensi bersifat menambah dan memperbarui. Dapat menambah data gaji jika tidak di tanggal tersebut tidak ditemukan data

gaji pengguna tersebut dengan mempertimbangkan bukti yang diberikan oleh pengguna untuk memperkuat bukti.

Proses manual Presensi jika *file* rusak atau pengiriman data dari mesin ke aplikasi gagal yaitu dengan memasukkan jam masuk terlebih dahulu, namun langsung terkena potongan “tidak Presensi pulang” karena belum memasukkan manual jam pulang. Berikut adalah contoh Presensi manual.

The screenshot shows a mobile application interface for inputting attendance and departure data. At the top, it says "Input Kehadiran dan Pulang" and has a close button "X". Below that is a field labeled "Tanggal Kehadiran" with a placeholder "YYYY-MM-DD". Underneath is a field labeled "Jam Kehadiran" with a placeholder "HH:MM (contoh: 07:05)". Below that is a field labeled "Jam Pulang" with a placeholder "HH:MM (contoh: 16:00)". At the bottom right is a green "Submit" button.

Gambar 3.34 Contoh format pengisian data Presensi manual yang memuat tanggal, jam masuk, dan jam keluar

Saat memasukkan data manual jam masuk, pada kasus ini belum terdapat jam keluar. Jadi, terdapat potongan “Tidak Presensi Keluar” seperti pada Gambar 3.35 karena dianggap belum melakukan Presensi pulang. Ketika sudah memasukkan jam keluar, potongan akan diperbaharui sesuai dengan ketentuan potongan “Pulang Cepat”. Seperti Gambar 3.36 di bawah ini. Pada sub-bab 3.3.2.4 dibangun untuk memenuhi kebutuhan Kasus Penggunaan Tambah Data Presensi. Pada sub-bab 3.3.2.4 merupakan halaman lanjutan atau sub-halaman dari sub-bab 3.3.2.3 untuk menuju ke setiap pengguna yang akan dituju untuk dimasukan data presensi secara manual.

Tanggal	Nama	Jam Masuk	Jam Keluar	Potongan				Total Salary
				Datang Telat	Pulang Cepat	Tidak Absen Masuk	Tidak Absen Keluar	
2025-01-01	Ucup	07:00:00	-	-Rp 0,00	-Rp 0,00	-Rp 0,00	- Rp 40.000,00	Rp 40.000,00

Gambar 3.35 Data Gaji belum Menyertakan Jam Pulang

Tanggal	Nama	Jam Masuk	Jam Keluar	Potongan				Total Salary
				Datang Telat	Pulang Cepat	Tidak Absen Masuk	Tidak Absen Keluar	
2025-01-01	Ucup	07:00:00	10:30:00	-Rp 0,00	- Rp 20.000,00	-Rp 0,00	-Rp 0,00	Rp 60.000,00

Gambar 3.36 Data Gaji dengan Menyertakan Jam Pulang (Sudah Diperbaharui)

3.3.2.5 Halaman Verifikasi Rekap Hasil Gaji dan Pemberian Bonus

Merekap hasil semua gaji dilakukan oleh Administrator atau Bendahara yang dapat dilakukan setiap interval waktu tertentu mulai dari gaji pokok, gaji harian, hingga potongan pada interval waktu yang ditentukan. Verifikasi rekap hasil gaji pada Gambar 3.37 untuk memenuhi kebutuhan Kasus Penggunaan Validasi Gaji.

Tanpa Bonus										
Tanggal	Nama	Jam Masuk	Jam Keluar	Potongan				Total Salary	Aksi	
				Datang Telat	Pulang Cepat	Tidak Absen Masuk	Tidak Absen Keluar			
30-05-2024	User Contoh	08:02:30	17:00:00	- Rp 2.250,00	- Rp 0,00	- Rp 0,00	- Rp 0,00	Rp 12.750,00	<button>Hapus</button>	
21-05-2024	User Contoh	08:00:00	-	- Rp 2.250,00	- Rp 0,00	- Rp 0,00	- Rp 0,00	Rp 12.750,00	<button>Hapus</button>	

Gambar 3.37 Hasil Gaji Untuk di Verifikasi

Gaji Pokok
NIP: 123444124
Nama: User Contoh

Kembali

Pokok	Rp 500.000,00
-------	---------------

Input Kehadiran Manual
Input Kehadiran Manual

Proses Gaji Bulanan
Start Date: 01-04-2026 End Date: 02-04-2026
Proses

Gambar 3.38 Tampilan Saat Memproses Gaji untuk Interval Tertentu

Konfirmasi Gaji Bulanan X

Uang Tambahan
100000

Keterangan Bonus
Masukkan keterangan bonus (opsional)

Unduh PDF Dengan Bonus

Unduh PDF Tanpa Bonus

Gambar 3.39 Tampilan Saat Jika Ingin Memasukkan Bonus

Komponen bonus ditampilkan secara terpisah karena sifatnya yang tidak rutin. Pemberian bonus dapat bervariasi antar pengguna dan periode, dan tidak selalu dimasukkan dalam perhitungan gaji pokok atau harian. Tahapan untuk merekap dan mencetak hasil gaji yaitu, mengisi tanggal periode pada setiap pengguna seperti Gambar 3.38, lalu Bendahara atau Administrator akan mencetak rekap hasil gaji beserta pemberian bonus yang bersifat opsional atau dapat tidak diisi seperti Gambar 3.39. Lalu tampilan, lembar gaji dengan detail gaji harian serta gaji pokok dan bonus yang telah tertera dalam lembar gaji Seperti Gambar 3.40 . Semua Bonus akan tergabung menjadi satu di dalam rentang tanggal bersama detail gaji harian seperti Gambar 3.41. Pada Gambar 3.38 dan Gambar 3.40 dibangun untuk memenuhi kebutuhan Kasus Penggunaan Cetak Gaji pada sub-bab 3.1.4.3. Serta pada Gambar 3.39 dan Gambar 3.41 dibangun untuk memenuhi kebutuhan Kasus Penggunaan Tambah Bonus pada. Sub-bab 3.3.2.5 adalah halaman yang sama dengan sub-bab 3.3.2.4. Namun pada sub-bab 3.3.2.5 berfokus untuk melihat serta memverifikasi hasil gaji yang sudah diproses oleh Aplikasi KinderFin.

Nama	:	User Contoh
NIP	:	123444124
Jabatan	:	Guru
Periode	:	2026-04-01 s.d. 2026-04-02
Gaji Pokok	:	Rp 500.000
Gaji Harian	:	Rp 30.000
Bonus	:	Rp 100.000
Total Gaji	:	Rp 630.000
Keterangan	:	Kerja Bagus

Rincian Rekap Jam Absensi Harian:

Tanggal	Jam Masuk	Jam Pulang
2026-04-01	07:00:00	17:00:00
2026-04-02	07:00:00	16:00:00

Gambar 3.40 Gambar Nota Pembayaran Untuk Setiap Pengguna

Detail Gaji Bulanan

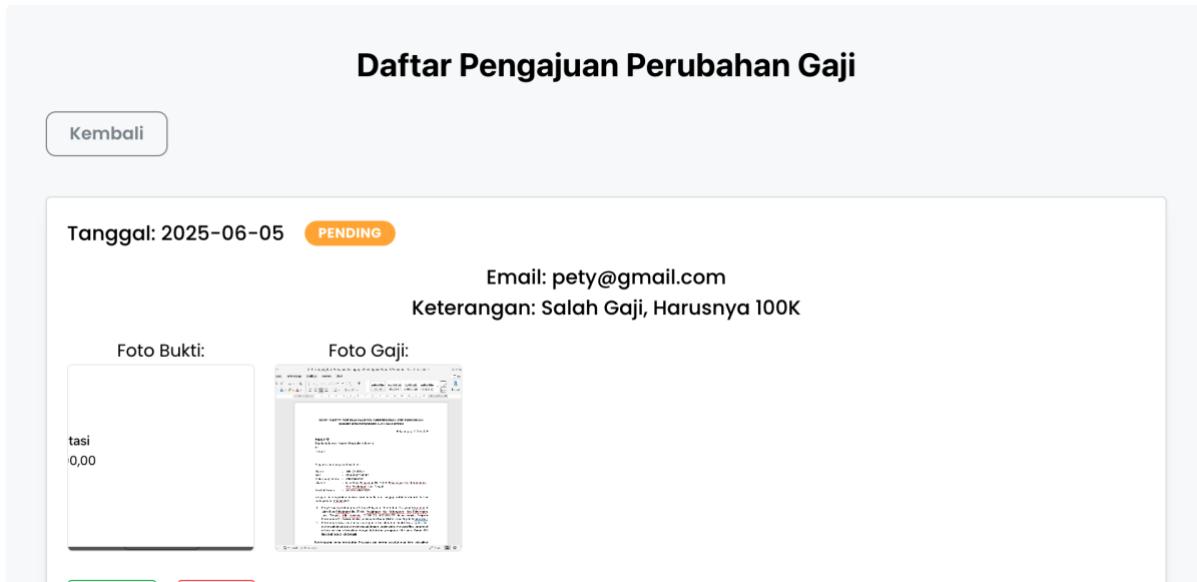
Uang Tambahan (Bonus): Rp 136.363.119,00
Keterangan: Bagus, Kerja Bagus
Periode Bonus: 01-04-2026 s/d 28-04-2026

Tanggal	Nama	Jam Masuk	Jam Keluar	Potongan				Total Salary	Aksi
				Datang Telat	Pulang Cepat	Tidak Absen Masuk	Tidak Absen Keluar		
02-04-2026	User Contoh	07:00:00	16:00:00	-Rp 0,00	-Rp 0,00	-Rp 0,00	-Rp 0,00	Rp 15.000,00	<button>Hapus</button>
01-04-2026	User Contoh	07:00:00	17:00:00	-Rp 0,00	-Rp 0,00	-Rp 0,00	-Rp 0,00	Rp 15.000,00	<button>Hapus</button>

Gambar 3.41 Hasil seluruh Gaji dan Seluruh Rekap Bonus

3.3.2.6 Halaman Pengajuan (Guru) dan Verifikasi Perubahan Gaji (Bendahara)

Pada Halaman Pengajuan dan Verifikasi Perubahan Gaji, dibangun untuk memenuhi kebutuhan Kasus Penggunaan Setuju Perubahan Gaji dan Kasus Penggunaan Ajukan Perubahan Gaji. Halaman pengajuan dapat dilakukan oleh semua peran atau jabatan baik guru, Kepala Sekolah maupun Bendahara sendiri namun lebih ditujukan kepada guru. Halaman Pengajuan dan Verifikasi Perubahan Gaji dibuat untuk mencatat bukti untuk menjadi acuan jika ada data yang tidak sesuai dengan rincian gaji pada sistem yang sudah tersedia. Halaman daftar pengjuan perubahan gaji dibangun dengan dua tampilan pada peran yang berbeda. Pada peran Bendahara dan Kepala Sekolah akan menampilkan seluruh ajuan yang diajukan seperti pada Gambar 3.44, sedangkan pada peran guru hanya menampilkan oleh data ajuan guru tersebut seperti pada Gambar 3.42.



Gambar 3.42 Tampilan Daftar Pengajuan Perubahan Gaji pada Peran Guru

The screenshot shows a form titled 'Ajukan Perubahan Gaji'. It includes the following fields:

- Tanggal * (Date): A date input field with placeholder 'dd/mm/yyyy'.
- Keterangan (Explanation): A text input field.
- Foto Bukti * (Evidence Photo): A file upload input field labeled 'Unggah foto bukti'.
- Foto Gaji * (Salary Slip Photo): A file upload input field labeled 'Unggah foto gaji'.

At the bottom of the form are two buttons: 'Kembali' (Back) and 'Ajukan' (Submit).

Gambar 3.43 Tampilan Halaman Pengajuan Perubahan Gaji pada Peran Guru

Pada Gambar 3.43 terdapat halaman untuk mengajukan perubahan gaji yang hasilnya dapat dilihat oleh pengaju pada Gambar 3.42 serta dapat dilihat oleh verifikator pada Gambar 3.44. Pada sub-bab 3.3.2.6 dibangun untuk memenuhi kebutuhan Kasus Penggunaan Setujui Perubahan Gaji untuk peran Bendahara yang bertugas untuk memverifikasi apakah menolak atau menyetujui ajuan serta memenuhi kebutuhan Kasus Penggunaan Ajukan Perubahan Gaji.

The screenshot shows two entries in the list:

- Entry 1:** Tanggal: 2025-06-05, Status: PENDING. Email: adelia@gmail.com, Keterangan: Ambil Cuti Tanggal 5. It includes two attachments: 'Foto Bukti' (two screenshots of an email inbox showing messages about leave requests) and 'Foto Gaji' (a screenshot of a salary slip or payment history). Below the attachments are two buttons: 'Setujui' (Approve) and 'Tolak' (Reject).
- Entry 2:** Tanggal: 2025-06-05, Status: PENDING. Email: pety@gmail.com, Keterangan: Salah Gaji, Harusnya 100K. It includes two attachments: 'Foto Bukti' (a screenshot of a document showing 'tasi 0,00') and 'Foto Gaji' (a screenshot of a salary slip or payment history). Below the attachments are two buttons: 'Setujui' (Approve) and 'Tolak' (Reject).

Gambar 3.44 Tampilan Daftar Pengajuan Perubahan Gaji pada Peran Bendahara

3.3.2.7 Halaman Rincian Gaji Pegawai - Guru

Halaman Rincian Gaji Pegawai Peran Guru merupakan halaman yang sama dengan halaman pada sub-bab 3.3.2.4 dan 3.3.2.5 namun, Halaman Rincian Gaji Pegawai Peran Guru hanya menampilkan detail gaji dan bonus pada guru tersebut yang telah *login* dan sistem akan mengambil data dari *JWT Token login* seperti Gambar 3.45. Jadi guru tersebut tidak bisa melihat detail gaji guru lainnya. Halaman Rincian Gaji Pegawai Peran Guru dibangun untuk memenuhi kebutuhan Kasus Penggunaan Lihat Rincian Gaji.

Gaji Pokok										
NIP: 4										
Nama: adelia										
Kembali										
Uang Tambahan: Rp 60.000,00 Keterangan: prestasi, prestasi, prestasi Periode Bonus: 28-04-2025 s/d 30-04-2025										
Tanggal	Nama	Jam Masuk	Jam Keluar	makan pagi	Potongan				Total Salary	
					Datang Telat	Pulang Cepat	Tidak Absen Masuk	Tidak Absen Keluar		
30-04-2025	adelia	07:33:00	11:02:00	Rp 10.000,00	-	Rp 1.000,00	Rp 0,00	Rp 0,00	Rp 9.000,00	
29-04-2025	adelia	07:29:00	10:13:00	Rp 10.000,00	-	Rp 1.000,00	Rp 0,00	Rp 0,00	Rp 9.000,00	

Gambar 3.45 Tampilan Detail Gaji dan Seluruh Rekap Bonus Pribadi Guru

3.3.2.8 Halaman Log Aktivitas - Bendahara

Halaman log aktivitas dibangun untuk memenuhi kebutuhan Kasus Penggunaan Lihat Log Aktivitas. Halaman log aktivitas menampilkan semua aktivitas yang dilakukan oleh Bendahara dan Administrator yang menyangkut aktivitas penggajian seperti menambahkan jabatan serta nominalnya di master jabatan dan menghapus pengguna. Pada Gambar 3.46 menunjukkan kolom setiap aktivitas yang dilakukan oleh Bendahara atau Administrator yang dapat memberikan bukti aktivitas mengenai penggajian.

Log Aktivitas Pengguna					
Tanggal	Email	Aksi	Modul	Keterangan	
04 Jun 2025, 12.02	rootuser@gmail.com	Upload Gaji Manual	Salary Detail	NIP: 123444124	Tanggal: 2026-04-02 Jam Masuk: 07:00 Jam Pulang: 16:00
04 Jun 2025, 12.02	rootuser@gmail.com	Upload Gaji Manual	Salary Detail	NIP: 123444124	Tanggal: 2026-04-01 Jam Masuk: 07:00 Jam Pulang: 17:00
04 Jun 2025, 09.47	pety@gmail.com	Upload Gaji Manual	Salary Detail	NIP: 15	Tanggal: 2025-06-03 Jam Masuk: 07:30 Jam Pulang: 12:30
04 Jun 2025, 09.43	pety@gmail.com	Membuat Jabatan	Master Jabatan	Membuat Jabatan:	Bendahara

Gambar 3.46 Tampilan Log Aktivitas

3.3.3 Implementasi Integrasi Aplikasi KinderFin dengan Mesin *Fingerprint*

Pada sub-bab ini dijelaskan bagaimana penerapan fitur presensi serta *Code* atau potongan fungsi yang menunjang jalannya fitur pada setiap modul. Ada beberapa modul yang menunjang fitur presensi ini, antara lain pengaturan gaji aktif, potongan gaji, master jabatan, *salary detail*, rekap bonus, pengajuan perubahan gaji, dan *activity logs*. Namun beberapa modul diatas akan dijelaskan pada sub-bab 0. Pada sub-bab ini merupakan salah satu program untuk memenuhi kebutuhan Kasus Penggunaan Tambah Data Presensi pada bagian menambah data presensi “Sinkronisasi Data Absensi *Fingerspot*” dan divisualisasikan pada sub-bab 3.3.2.3.

3.3.3.1 Ambil Data Presensi

Sebelum proses pengambilan data presensi dari *server developer.fingerspot.io*, dilakukan proses pengaturan mesin *fingerprint* yang disesuaikan dengan *framework developer.fingerspot.io*. Proses tersebut dirincikan pada LAMPIRAN B. Pengambilan data presensi dilakukan dalam dua tahap, yaitu tahap pengiriman request ke *API Fingerspot* dan tahap pemrosesan data yang diterima. Sistem ini diimplementasikan menggunakan dua metode utama yang terintegrasi dengan *controller* dan *command pattern*. Terdapat struktur pada data presensi yang diambil dari *server* pada Kode Sumber 3.1.

```
1. interface FingerspotAttendanceRecord {  
2.   pin: string; // NIP guru  
3.   scan_date: string; // Format: "YYYY-MM-DD HH:MM:SS"  
4.   verify: number; // Metode verifikasi  
5.   status_scan: number; // 0: scan masuk, 1: scan keluar  
6. }  
7.  
8. interface FingerspotResponse {  
9.   success: boolean;  
10.  trans_id: string;  
11.  data: FingerspotAttendanceRecord[];  
12. }
```

Kode Sumber 3.1 Struktur Code Data Presensi dari Sever *developer.fingerspot.io*

Metode pertama yaitu dengan Penerimaan Data Melalui *Input Manual*. Metode pertama menggunakan fungsi *processAttendanceData* yang menerima data presensi melalui *Input Manual* menggunakan aplikasi Postman. Data dikirimkan dalam format JSON yang sesuai dengan struktur *API Fingerspot*. Fungsi ini memperlihatkan bagaimana sistem seolah olah menerima hasil langsung dari *server vendor (fingerspot)*.

```
1. static async processAttendanceData(req: Request, res: Response) {  
2.   try {  
3.     const fingerspotResponse = req.body as FingerspotResponse;  
4.     // validasi format dasar data Fingerspot  
5.     if (!fingerspotResponse.success || !fingerspotResponse.data) {  
6.       return res.status(400).json({ message: "Format data tidak valid" });  
7.     }  
8.     // Proses data menggunakan fungsi pemroses pusat  
9.     const result = await  
10.    processFingerspotRecords(fingerspotResponse.data);  
11.    return res.status(200).json({ message: "Berhasil memproses...",  
12.      ...result });  
13.  } catch (error: any) {  
14.    // ... penanganan error server  
15.  }  
16.}  
17.}
```

Kode Sumber 3.2 *Code processAttendanceData* Penerimaan Data Melalui *Input Manual*.

Proses alur kerja pada Kode Sumber 3.2 yaitu, pertama menerima data JSON melalui aplikasi *Postman* atau *request HTTP* lainnya (Baris 3). Lalu, memvalidasi struktur dan format data sesuai dengan *interface* yang telah ditentukan (Baris 6-8). Setelah itu, memanggil *command* untuk memproses data presensi (Baris 11). Terakhir, mengembalikan hasil pemrosesan berupa jumlah data yang berhasil diproses dan yang dilewati (Baris 12).

Metode kedua yaitu Pengambilan data langsung dari *API Fingerspot*. Metode kedua menggunakan fungsi *fetchAndProcessAttendance* yang secara aktif mengambil data dari *API Fingerspot* berdasarkan rentang tanggal yang ditentukan. Metode Pengambilan data langsung dari *API Fingerspot* merupakan solusi ideal ketika serial number mesin telah sesuai dengan yang terdaftar di *server*. Metode ini yang akan dipakai di Tugas Akhir ini untuk memudahkan

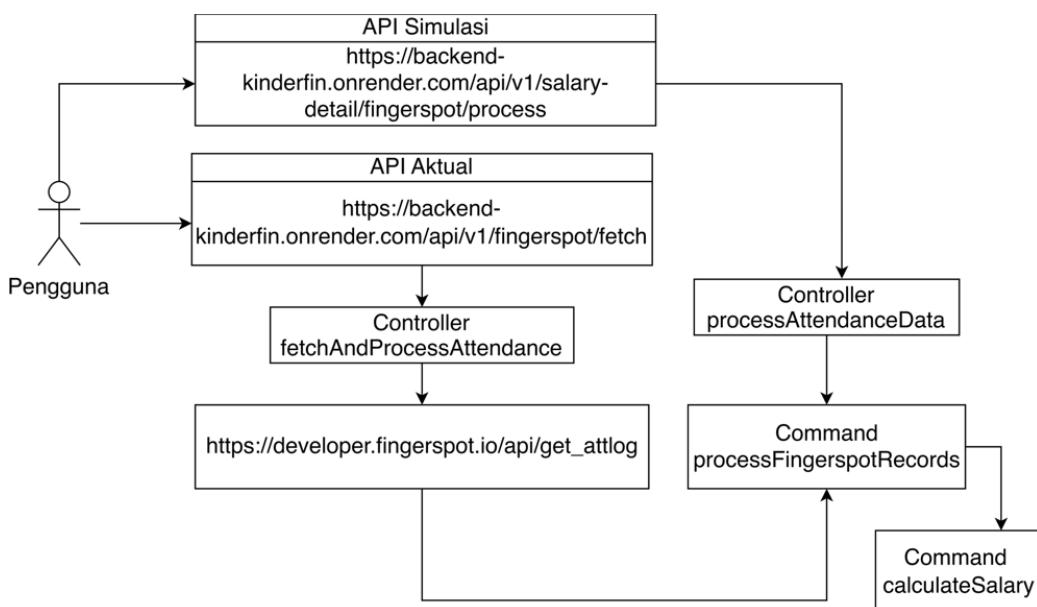
Bendahara mendapatkan data secara langsung dengan mengimkan perintah pengambilan data presensi dengan rentang tanggal tertentu.

```

1. static async fetchAndProcessAttendance(req: Request, res: Response) {
2.   try {
3.     const { start_date, end_date } = req.body;
4.
5.     // validasi parameter dan rentang tanggal
6.     if (!start_date || !end_date) { /* ... return error 400 */ }
7.     // ... validasi lain (misal: rentang maksimal 2 hari)
8.
9.     // Kirim request ke API Fingerspot
10.    const response = await
11.      fetch("https://developer.fingerspot.io/api/");
12.    const data = await response.json();
13.
14.    // Proses data menggunakan fungsi pemroses pusat yang sama
15.    const result = await processFingerspotRecords(data.data);
16.    return res.status(200).json({ message: "Berhasil mengambil data...", ...result });
17.  } catch (error) {
18.    // ... penanganan error
19.  }
20.}
```

Kode Sumber 3.3 *Code fetchAndProcessAttendance* yang secara aktif mengambil data dari *API Fingerspot*

Dari Kode Sumber 3.3 diatas terdapat alur kerjanya yaitu, pertama melakukan validasi parameter *input* berupa tanggal mulai dan tanggal akhir (Baris 6). Lalu, memastikan rentang tanggal tidak melebihi batasan yang ditetapkan oleh API (maksimal 2 hari) (Baris 7). Setelahnya, mengirim HTTP POST *request* ke API *Fingerspot* dengan *header autentikasi* yang sesuai (Baris 10). Kemudian memproses data menggunakan *command* yang sama dengan metode pertama (Baris 14).



Gambar 3.47 Alur Kerja *API process* dan *fetch*.

Pada Gambar 3.47 merupakan visualisasi alur kedua metode yaitu Penerimaan Data Melalui *Input Manual* dan Pengambilan data langsung dari *API Fingerspot*. Dimulai dari Pengguna memilih apakah untuk menggunakan *API* untuk mengirimkan perintah, untuk metode pertama pengguna harus mempunyai contoh *code* simulasi seperti Kode Sumber 3.71 dan

dikirimkan melalui Kode Sumber 3.70 (*API Simulasi*). Selanjutnya data tersebut akan diproses di Kode Sumber 3.2 yang akan dilanjutkan di Kode Sumber 3.4 untuk di saring dan dicocokan dengan format yang dimiliki oleh Aplikasi KinderFin 2.0. Untuk Metode kedua setelah pengguna mengirimkan perintah melalui Kode Sumber 3.73 (*API Aktual*), maka akan dilanjutkan ke Kode Sumber 3.3 lalu *Controller* tersebut akan mengirimkan perintah ke *API server developer.fingerspot.io*. Setelah mendapat balasan dari *server developer.fingerspot.io*, balasan tersebut akan dilanjutkan ke Kode Sumber 3.4. Lalu, data yang diterima akan diolah ke Modul *Salary Detail* untuk dijadikan data gaji setiap guru.

3.3.3.2 Proses Data Presensi

Dari Kode Sumber 3.2 dan Kode Sumber 3.3 akan bermuara pada *file command* yang sama yaitu akan masuk ke Kode Sumber 3.4. Untuk memproses data presensi yang telah diterima. Setelah melalui fungsi *processFingerspotRecords* maka akan diteruskan untuk proses penghitungan utama gaji yaitu, di fungsi *calculateSalary(attendanceRecords, "manual")*; pada *file command* yang akan memproses semua masukan dari berbagai sumber presensi. Fungsi tersebut akan dijelaskan pada sub-bab 0.

```
1. async function processFingerspotRecords(
2.   records: FingerspotRecord[]
3. ): Promise<{ processed: number; skipped: number }> {
4.
5.   // Saring & kelompokkan data presensi berdasarkan NIP dan tanggal
6.   const recordsByNipAndDate = new Map<string, any>();
7.   records.forEach(record => {
8.     // ... Logika untuk menggabungkan scan masuk dan keluar menjadi satu
9.     // record per hari
10.    });
11.   // Ubah format data agar sesuai dengan kebutuhan sistem
12.   const attendanceRecords = Array.from(recordsByNipAndDate.values());
13.
14.   // Teruskan ke fungsi kalkulasi gaji utama
15.   return await calculateSalary(attendanceRecords, "manual");
16. }
```

Kode Sumber 3.4 *Code processFingerspotRecords* Pengambilan data langsung dari *API Fingerspot*

Tahapan proses data dari kedua fungsi tersebut yaitu dengan, tahap penyaringan antara *scan in* dan *scan out* atau presensi masuk dan presensi keluar. Sistem hanya memproses *record* dengan *status_scan* bernilai 0 (masuk) dan 1 (keluar) (Baris 5, 7). Lalu, ada pengelompokan. Data dikelompokkan berdasarkan kombinasi NIP pegawai dan tanggal presensi (Baris 6-9). Setelahnya, ada proses strukturisasi data, karena ada beberapa perbedaan format yang dikirimkan *server vendor (fingerspot)* dengan Aplikasi KinderFin 2.0. Fungsi tersebut akan mengubah format data menjadi struktur yang sesuai dengan kebutuhan sistem (Baris 11). Terakhir, Data yang telah diproses diteruskan ke sistem perhitungan gaji untuk pemrosesan lebih lanjut di fungsi *calculateSalary(attendanceRecords, "manual")* (Baris 14-15).

3.3.4 Implementasi Aplikasi KinderFin

Pada penerapan perhitungan gaji, ada beberapa modul yang mendukung jalannya implementasi ini seperti pengaturan gaji aktif untuk pengambilan data gaji mana saja yang aktif, *master jabatan* untuk memasukkan nominal gaji, potongan gaji untuk mengambil komponen mana saja yang dipotong, serta *salary detail* untuk catatan seluruh gaji pengguna secara lengkap. Penerapan modul di Aplikasi KinderFin 2.0 meliputi logika program (*backend*) untuk bagaimana Aplikasi KinderFin 2.0 berjalan dan Tampilan Antarmuka (*frontend*) untuk

memudahkan pengguna untuk menggunakan Aplikasi KinderFin 2.0. Pada sub-bab 3.3.4 ini, fokus utama adalah logika program (*backend*), serta detail logika, fungsionalitas, dan fungsi tampilan antarmuka modul *salary detail* dijelaskan di sub-bab ini, termasuk bagaimana memproses perhitungan gaji. Sementara itu, tampilan kode antarmuka *frontend* lainnya yang sebagian besar disesuaikan dari Aplikasi KinderFin 1.0, dirincikan pada LAMPIRAN G. Perlu dicatat, meskipun logika *backend* modul *users* ini mengacu pada *backend* Aplikasi KinderFin 1.0 (Akbar 2024), kode antarmuka untuk modul *users* dikembangkan dalam proyek ini, dengan sebagian adaptasi dari versi sebelumnya serupa dengan modul *frontend* lainnya.

Kode Sumber 3.5 ini adalah bagian inti yang mengatur jalur API untuk sebuah modul. Di dalamnya, setelah model-model ada *Sequelize* (Baris 1 dan 5) dan *middleware* keamanan diterapkan (Baris 9), didefinisikan sebuah rute contoh POST (Baris 12). Rute ini dirancang untuk menerima permintaan pembuatan data baru (Baris 12), membatasi aksesnya hanya untuk peran "Admin" dan "Pengguna" (Baris 12), dan akan memanggil fungsi contoh *create* pada *controller* contoh *ModulGenerikController* untuk memproses permintaan tersebut (Baris 12). Terdapat rute lain yaitu *GET* untuk mendapatkan data, *PATCH* digunakan untuk memperbarui sebagian dari sumber daya, *PUT* digunakan untuk mengganti seluruh sumber daya dengan yang baru. Kode Sumber 3.5 akan diterapkan untuk semua modul yang terdapat pada Aplikasi KinderFin 2.0 ini dengan setiap rute dan kebutuhan yang berbeda beda (Baris 17, yang mendaftarkan router).

```
1. export function setModulGenerikRoutes(app: any, dbConn: Sequelize) {  
2.   const router = Router();  
3.  
4.   // Inisialisasi model dan relasi database untuk modul ini  
5.   dbConn.models["modul_utama"] = ModulUtamaModel;  
6.   /* ... konfigurasi relasi antar model ... */  
7.  
8.   // Terapkan middleware untuk autentikasi pada semua rute  
9.   router.use(middlewareAuthentication);  
10.  
11.  // Definisikan rute spesifik dengan otorisasi berbasis peran  
12.  router.post("/", restrictedTo("Admin", "Pengguna"),  
    ModulGenerikController.create);  
13.  // ... rute lainnya (GET, PATCH, PUT, DELETE)  
14.  
15.  // Daftarkan router ke aplikasi Express utama  
16.  app.use("/api/v1/modul-generik", router);  
17. }
```

Kode Sumber 3.5 Contoh *Code Router*

3.3.4.1 Modul Pengaturan Gaji Aktif

Konsep modul ini adalah sebagai kontrol pusat nama label dan komponen gaji apa saja yang akan diambil oleh pengguna sebagai dasar penghitungan gaji. Pada kondisi ideal, sistem penggajian seharusnya dapat mengelola berbagai komponen gaji secara dinamis tanpa perlu mengubah struktur basis data atau kode aplikasi. Namun, terdapat kendala di lapangan dimana setiap institusi memiliki kebijakan komponen gaji yang berbeda-beda, mulai dari gaji pokok, harian, hingga komponen khusus lainnya yang bersifat dihitung harian atau pokok. Hal ini menyebabkan tidak menjadikan aplikasi menjadi tidak fleksibel dalam mengakomodasi keragaman struktur penggajian. Untuk memenuhi kebutuhan Kasus Penggunaan Aktifkan Label Gaji, dikembangkan modul Pengaturan Gaji Aktif yang memungkinkan Administrator atau Bendahara untuk mengaktifkan dan menonaktifkan komponen gaji sesuai kebutuhan. Modul ini divisualisasikan pada Gambar 3.24 Berikut adalah fungsi yang mendukung berjalannya modul pengaturan gaji aktif.

1. Fungsi *execute SavePengaturanGajiAktifCommand* (*save-pengaturan.command.ts*)

Kode Sumber 3.6 memiliki fungsi yaitu memproses dan menyimpan konfigurasi *field*-*field* gaji yang akan diaktifkan. Alur kerjanya yaitu dengan memvalidasi bahwa data yang diterima berupa *array*. (Baris 2-3). Lalu, sistem memeriksa *field* harus bertipe *string*. (Baris 5-6). Tipe *field* hanya boleh berisi nilai “harian” dan “pokok”. (Baris 7-8). Sistem menyimpan konfigurasi *field* aktif ke *repository*. (Baris 9). Lalu sistem akan memberikan konfirmasi apakah berhasil atau tidak. (Terjadi setelah eksekusi pada Baris 9).

```
1. async execute(fields: FieldAktif[]): Promise<void> {
2.   if (!Array.isArray(fields)) {
3.     throw new Error("Data pengaturan harus berupa array.");
4.   }
5.   for (const item of fields) {
6.     if (!item.field || typeof item.field !== "string") {
7.       throw new Error("Setiap item wajib memiliki 'field' bertipe string.");
8.     }
9.     if (item.type && !["harian", "pokok"].includes(item.type)) {
10.       throw new Error(`Type '${item.type}' tidak valid. Hanya boleh 'harian' atau 'pokok'.`);
11.     }
12.   }
13.   await PengaturanGajiAktifRepository.save(fields);
14. }
```

Kode Sumber 3.6 *Code execute SavePengaturanGajiAktifCommand*

2. Fungsi *execute DeletePengaturanGajiAktifCommand* (*save-pengaturan.command.ts*)

Kode Sumber 3.7 memiliki fungsi yaitu menghapus *field* gaji yang tidak diperlukan lagi dari konfigurasi aktif serta membuat menjadi 0 nilai *field* tersebut di tabel master jabatan. Alur kerjanya yaitu validasi bahwa nama *field* yang akan dihapus telah disediakan dan berformat *string*. (Baris 2-3). Selanjutnya, sistem menghapus *field* dari tabel pengaturan gaji aktif melalui *repository*. (Baris 6). Kemudian, sistem melakukan pembersihan dengan mereset nilai *field* tersebut menjadi nol di tabel master jabatan dan master jabatan pokok. (Baris 7-8). Terakhir memberikan pesan berhasil atau gagal. (Baris 6-8).

```
1. static async execute(field: string): Promise<void> {
2.   if (!field || typeof field !== "string") {
3.     throw new Error("Field yang akan dihapus harus disediakan dan bertipe string.");
4.   }
5.   await PengaturanGajiAktifRepository.deleteByField(field);
6.   await MasterJabatanRepository.resetFieldTozero(field);
7.   await MasterJabatanPokokRepository.resetFieldTozero(field);
8. }
```

Kode Sumber 3.7 *Code execute DeletePengaturanGajiAktifCommand*

3. Fungsi *execute GetPengaturanGajiAktifQuery* (*get-pengaturan.query.ts*)

Kode Sumber 3.8 adalah fungsi untuk untuk mengambil semua konfigurasi *field* gaji yang sedang aktif dalam sistem. Alurnya dimulai dari pemanggilan *repository* untuk mengambil semua data pengaturan gaji aktif. (Terletak pada Baris 2). Selanjutnya, sistem mengembalikan daftar *field* yang aktif dan informasi label dan tipe setiap *field*. (Terjadi dari hasil pemanggilan pada Baris 2). Kemudian, data yang dikembalikan dalam format yang siap digunakan oleh modul lain. (Hasil dari operasi pada Baris 2).

```
1. async execute(): Promise<FieldAktif[]> {
2.   return await PengaturanGajiAktifRepository.getAll();
3. }
```

Kode Sumber 3.8 *Code execute GetPengaturanGajiAktifQuery*

4. Model *PengaturanGajiAktif* (*pengaturan_gaji_aktif.entity.ts*)

Kode Sumber 3.9 berfungsi untuk mendefinisikan struktur data dan skema basis data untuk entitas pengaturan gaji aktif dan seluruh atribut dan *constraint*. Alur penggunaannya yaitu membuat struktur data yang menggambarkan tabel pengaturan gaji aktif dalam basis data. (Baris 1-7). Selanjutnya, sistem menentukan atribut seperti ID sebagai *Primary Key* (Baris 10-14), *field* sebagai nama unik komponen gaji (Baris 15-18), *label* sebagai nama tampilan (Baris 4), dan *type* untuk mengkategorikan jenis gaji (Baris 5).

```
1. export class PengaturanGajiAktif extends Model {
2.   public id!: number;
3.   public field!: string;
4.   public label!: string | null;
5.   public type!: string | null;
6.   public aktif!: boolean;
7. }
8.
9. PengaturanGajiAktif.init({
10.   id: {
11.     type: DataTypes.INTEGER,
12.     autoIncrement: true,
13.     primaryKey: true,
14.   },
15.   field: {
16.     type: DataTypes.STRING,
17.     allowNull: false,
18.     unique: true,
19.   },
20.   ...
21.   aktif: {
22.     type: DataTypes.BOOLEAN,
23.     allowNull: false,
24.     defaultValue: true,
25.   },
26. },
27.   sequelize: dbConn,
28.   tableName: "pengaturan_gaji_aktif",
29.   timestamps: true,
30.   underscored: true,
31.   version: false,
32.});
```

Kode Sumber 3.9 *Code Model PengaturanGajiAktif*

5. Fungsi *getAll* (*pengaturan_gaji_aktif.repository.ts*)

Kode Sumber 3.10 berfungsi untuk mengambil semua *field* gaji yang aktif dari basis data yang berstatus aktif. Alur kerjanya yaitu melakukan *query* basis data untuk mengambil *field*, *label*, dan *type* dari tabel pengaturan gaji aktif. (Baris 3-4). Selanjutnya, sistem melakukan pengambilan data hanya memiliki status aktif bernilai *true*. (Baris 5). Setelah itu, sistem mengembalikan *array* berisi informasi *field* aktif dalam format yang standar. (Baris 8-12). Terakhir, jika ada *error* dalam *query* maka akan menampilkan pesan *error*. (Baris 13-16).

```
1. async getAll(): Promise<{ field: string; label: string | null; type: string | null }[]> {
2.
3.   try {
4.     const result = await PengaturanGajiAktif.findAll({
5.       attributes: ["field", "label", "type"],
6.       where: { aktif: true },
7.     });
8.
9.     return result.map((item) => ({
10.       field: item.field,
11.       label: item.label ?? null,
12.       type: item.type ?? null,
13.     }));
14.   } catch (error) {
15.     console.error(`Error fetching active salary settings: ${error.message}`);
16.     throw new Error(`An error occurred while fetching active salary settings: ${error.message}`);
17.   }
18. }
```

```

13. } catch (error) {
14.   console.error("[PengaturanGajiAktifRepository] Error getAll:", error);
15.   throw new Error("Gagal mengambil pengaturan gaji aktif.");
16. }
17. }

```

Kode Sumber 3.10 *Code getAll Repository*

6. Fungsi *save (pengaturan_gaji_aktif.repository.ts)*

Kode Sumber 3.11 memiliki fungsi yaitu menyimpan atau memperbarui konfigurasi *field gaji aktif* dengan menggunakan operasi *bulk insert* atau *update*. Alur kerjanya yaitu validasi bahwa parameter yang diterima berupa *array*. (Baris 1). Selanjutnya, sistem mengecek apakah *array* tidak kosong. (Implisit sebelum Baris 11). Kemudian, sistem melakukan transformasi data dengan menyiapkan format yang sesuai untuk operasi *bulkCreate*, penetapan status aktif menjadi *true*. (Baris 13-17). Setelah itu, sistem menggunakan operasi *bulkCreate* dengan opsi *updateOnDuplicate* untuk menangani kasus dimana *field* sudah ada sebelumnya. (Baris 12, 19-21). Jika terjadi *error* akan mengirimkan pesan *error*. (Baris 25-28).

```

1. async save(fields: { ... }[]): Promise<void> {
...
11.   try {
12.     await PengaturanGajiAktif.bulkCreate(
13.       fields.map((f) => ({
14.         field: f.field,
15.         label: f.label || null,
16.         type: f.type || null,
17.         aktif: true,
18.       })),
19.       {
20.         updateOnDuplicate: ["label", "type", "aktif"],
21.       }
22.     );
...
25.   } catch (error) {
...
27.     throw new Error("Gagal menyimpan pengaturan gaji aktif.");
28.   }
29. }

```

Kode Sumber 3.11 *Code save Repository*

7. Fungsi *deleteByField (pengaturan_gaji_aktif.repository.ts)*

Kode Sumber 3.12 berfungsi untuk menghapus *field* dari konfigurasi gaji aktif berdasarkan nama *field*. Alurnya yaitu, melakukan operasi penghapusan berdasarkan nama *field* yang diberikan. (Baris 3-5). Selanjutnya, sistem mengecek hasil operasi untuk memastikan data yang dipilih terhapus. (Baris 6). Kemudian, jika data terhapus tidak ada, sistem memberikan *error* bahwa *field* tidak ditemukan. (Baris 6-7). Setelah operasi penghapusan berhasil, sistem akan mencatat pesan keberhasilan ke konsol. (Baris 9). Terakhir, jika terjadi kesalahan selama proses penghapusan (misalnya, *error* basis data), sistem akan mencatat *error* tersebut dan melemparkan *error* umum. (Baris 10-12).

```

1. async deleteByField(field: string): Promise<void> {
2.   try {
3.     const deleted = await PengaturanGajiAktif.destroy({
4.       where: { field },
5.     });
6.     if (deleted === 0) {
7.       throw new Error(`Field '${field}' tidak ditemukan.`);
8.       console.log(`[PengaturanGajiAktifRepository] Field '${field}' berhasil
dihapus.`);
10.    } catch (error) {

```

```

11.     console.error("[PengaturanGajiAktifRepository] Error deleteByField:",
error);
12.     throw new Error("Gagal menghapus pengaturan gaji aktif.");

```

Kode Sumber 3.12 *Code deleteByField Repository*

8. Fungsi *get* (*pengaturan_gaji_aktif.controller.ts*)

Kode Sumber 3.13 berfungsi untuk menangani permintaan HTTP GET untuk mengambil daftar pengaturan gaji aktif. Alurnya adalah mengambil data pengaturan gaji aktif melalui *query service*. (Baris 3). Selanjutnya, sistem menyusun *response JSON* yang berisi pesan sukses dan data *field* aktif. (Baris 4-7). Kemudian, jika berhasil akan berstatus *code* 200 (Baris 4) dan jika *error* akan menampilkan *code* 500 beserta pesan kesalahannya (Baris 8-12).

```

1. async get(req: Request, res: Response) {
2.   try {
3.     const aktifFields = await GetPengaturanGajiAktifQuery.execute();
4.     res.status(200).json({
5.       message: "Berhasil mengambil pengaturan gaji aktif",
6.       aktif: aktifFields,
7.     });
8.   } catch (error) {
9.     res.status(500).json({
10.       error: "Gagal mengambil pengaturan gaji aktif",
11.       detail: (error as Error).message,
12.     });
13.   }
14. }

```

Kode Sumber 3.13 *Code get Controller*

9. Fungsi *save* (*pengaturan_gaji_aktif.controller.ts*)

Kode Sumber 3.14 berfungsi untuk menangani permintaan HTTP POST untuk menyimpan pengaturan gaji aktif. Alurnya yaitu, dengan validasi bahwa data yang dikirim berupa *array* untuk memastikan format benar. (Baris 3 dan pada *validation logic* sebelum Baris 6). Selanjutnya, sistem menyimpan konfigurasi baru melalui *command service*. (Baris 6). Kemudian, sistem mengambil konfigurasi sebelumnya untuk perbandingan. (Baris 9). Setelah itu, sistem melihat *field* mana yang baru dibuat dan mana yang diperbarui dengan membandingkan data lama dan baru. (Baris 11). Terakhir, jika berhasil akan merespon berhasil dan jika gagal akan menampilkan pesan gagal. (Baris 12-14).

```

1. async save(req: Request, res: Response) {
2.   try {
3.     const { aktif } = req.body;
4.     ...
5.     // Menyimpan atau memperbarui data dengan Command
6.     await savePengaturanGajiAktifCommand.execute(aktif);
7.     ...
8.     // Mengambil data lama untuk perbandingan dengan Query
9.     const previousFields = await GetPengaturanGajiAktifQuery.execute();
10.    ...
11.    ...
12.   } catch (error) {
13.     ...
14.   }

```

Kode Sumber 3.14 *Code save Controller*

10. Fungsi *delete* (*pengaturan_gaji_aktif.controller.ts*)

Kode Sumber 3.15 berfungsi untuk menangani permintaan HTTP DELETE untuk menghapus *field* dari tabel gaji aktif. Alurnya adalah mengambil nama *field* dari URL. (Baris

3). Lalu, sistem melakukan penghapusan *field* melalui *command service* yang juga akan membersihkan data di tabel lain. (Baris 6). Terakhir, jika berhasil akan merespon berhasil (Baris 11) dan jika gagal akan menampilkan pesan gagal (Baris 12-14).

```

1. async delete(req: Request, res: Response) {
2.   try {
3.     const { field } = req.params;
4.     ...
5.     // Menghapus field menggunakan command service
6.     await deletePengaturanGajiAktifCommand.execute(field);
7.     ...
8.     // Mencatat aktivitas ke dalam log (detail disembunyikan)
9.     await createLog.execute({ ... });
10.    res.status(200).json({ message: "Field berhasil dihapus" });
11.  } catch (error) {
12.    ...
13.  }
14. }
15. }
```

Kode Sumber 3.15 *Code delete Controller*

3.3.4.2 Modul Master Jabatan

Modul ini merupakan modul yang berisi mengenai pemrosesan nominal gaji setiap jabatan pada Aplikasi KinderFin 2.0. Modul ini ada hubungannya dengan modul sebelumnya yaitu sub-bab 3.3.4.1. Modul ini akan mengambil label pada basis data lalu akan menampilkan pada antarmukanya. Jika pada modul Pengaturan Gaji Aktif dihapus maka otomatis gaji pada Master Jabatan akan dijadikan nol kembali karena akan digunakan untuk nama komponen gaji lain atau tidak digunakan kembali. Jika dijadikan nol, maka penghitungan gaji akhir akan tidak sesuai dengan kebutuhan. Sub-bab ini untuk memenuhi kebutuhan Kasus Penggunaan Tambah atau Ubah Gaji Harian dan Pokok, dan divisualisasikan pada Gambar 3.25, Gambar 3.26, Gambar 3.27, dan Gambar 3.28. Ada beberapa fungsi di dalam modul ini yang pertama yaitu pembuatan data, pengambilan data, pembaruan data, dan hapus data.

1. Fungsi *createFull* untuk Pembuatan Data Jabatan Lengkap

Pembuatan data atau Operasi *Create*. Pembuatan data pada master jabatan dengan cara memasukkan data nama jabatan serta nominal gaji harian (gaji1-10) dan gaji pokok (gajipokok1-10) namun gaji tersebut diambil label dari modul 3.3.4.1.

```

1. async createFull(req: Request, res: Response) {
2.   try {
3.     const data = req.body;
4.     ...
5.     // Pengecekan duplikasi data sebelum membuat
6.     const existing = await
MasterJabatanPokokRepository.findByName(data.jabatan);
7.     if (existing) {
8.       return res.status(409).json({ error: "Jabatan sudah ada..." });
9.     }
10.    ...
11.    // Membuat data baru melalui repository
12.    const result = await MasterJabatanPokokRepository.create(data);
13.    ...
14.    // Mencatat aktivitas ke dalam log
15.    await createLog.execute({ ... });
16.  }
17.  ...
18.  res.status(201).json({ message: "Jabatan ... berhasil ditambahkan", ... });
19.  catch (error) { ... penanganan error... }
```

Kode Sumber 3.16 *Code createFull*

Alur fungsi Kode Sumber 3.16 dimulai dengan melakukan validasi *input* untuk memastikan parameter jabatan berupa *string* dan tidak kosong. (Baris 4-6). Selanjutnya, sistem melakukan pengecekan duplikasi dengan mencari data jabatan yang sudah ada di basis data.

(Baris 7-9). Lalu, sistem menyiapkan data lengkap dengan struktur gaji pokok dan gaji harian dengan nilai awal adalah 0. (Implisit pada variabel data sebelum Baris 12). Setelah itu, sistem melakukan *insert* ke tabel master_jabatan_pokok yang secara otomatis akan melakukan sinkronisasi ke tabel master_jabatan melalui *sync service*. (Baris 12). Terakhir, sistem mencatat log aktivitas. (Baris 15).

2. Fungsi *getAll* untuk Pengambilan Data Jabatan Lengkap

Pengambilan atau pembacaan Data jabatan. Kode Sumber 3.17 *Code getAll* dibuat karena ada dua tabel yaitu master jabatan dan master jabatan pokok. Master jabatan sendiri berisi mengenai nama jabatan beserta isi komponen gaji harian. Master jabatan pokok berisi mengenai nama jabatan beserta isi komponen gaji pokok.

```
1. async getAll(req: Request, res: Response) {
2.   try {
3.     // Mengambil data dari dua sumber
4.     const jabatanList = await
MasterJabatanQuery.GetAllMasterJabatanQuery.execute();
5.     const pokokList = await MasterJabatanPokokRepository.getAll();
6.
7.     // Menggabungkan kedua data berdasarkan nama jabatan
8.     const combined = jabatanList.map((harian: any) => {
9.       // ... Logika untuk mencari & menggabungkan data
10.      return { ... };
11.    });
12.
13.    res.status(200).json({ data: combined, ... });
14.  } catch (error) {
15.    ...
16.  }
17.}
```

Kode Sumber 3.17 *Code getAll* Master Jabatan

Alur Kode Sumber 3.17 dimulai dengan mengambil data dari dua tabel berbeda yaitu master_jabatan untuk gaji harian (Baris 4) dan master_jabatan_pokok untuk gaji pokok (Baris 5). Selanjutnya, sistem melakukan operasi *join* manual dengan mencocokkan data berdasarkan *field* jabatan sebagai *Primary Key*. (Baris 8-9). Kemudian, sistem menggabungkan kedua dataset menggunakan *spread operator* untuk menghasilkan objek lengkap yang berisi informasi gaji harian dan gaji pokok. (Baris 8, 10).

3. Fungsi *update* untuk Pembaruan Data Jabatan

Fungsi perbarui atau *update*. Fungsi ini akan memperbarui nominal angka dari gaji pokok dan gaji harian yang sudah ditentukan sebelumnya. Fungsi ini akan memperbarui dua tabel yaitu master_jabatan dan master_jabatan_pokok. Kedua tabel itu akan diperbarui jika fungsi ini dipanggil.

```
1. async update(jabatan: string, data: any) {
2.   // ...mencari record dan validasi awal
3.
4.   // Memisahkan data input berdasarkan field yang diizinkan
5.   const updateGaji: ... = {};
6.   const updateGajiPokok: ... = {};
7.   for (const key in data) {
8.     // ...logika untuk memfilter & memisahkan data gaji dan gaji_pokok
9.   }
10.  // ...validasi jika tidak ada data yang perlu diupdate
11.
12.  // update kondisional ke tabel pertama (master_jabatan)
13.  if (Object.keys(updateGaji).length > 0) {
14.    await record.update(updateGaji);
15.  }
16.
17.  // update kondisional ke tabel kedua (master_jabatan_pokok)
18.  if (Object.keys(updateGajiPokok).length > 0) {
19.    await record.update(updateGajiPokok);
20.  }
21.
22.  return res.json({ message: 'Data berhasil diperbarui' });
23.}
```

```

16. // update kondisional ke tabel kedua (master_jabatan_pokok)
17. if (Object.keys(updateGajiPokok).length > 0) {
18.   await MasterJabatanPokokRepository.update(jabatan, updateGajiPokok);
19. }
20.
21. // Menjalankan service untuk sinkronisasi data
22. await JabatanSyncService.syncToPokok(jabatan);
23.
24.
25. return { ... }; // Mengembalikan data gabungan yang telah diperbarui
26.

```

Kode Sumber 3.18 *Code update Master Jabatan*

Alur kerja Kode Sumber 3.18 dimulai dengan melakukan validasi *input* (Baris 2) dan mendefinisikan *whitelist field* yang diperbolehkan untuk di-*update*. (Implisit di Baris 5-6 dan 7-9). Selanjutnya, sistem memisahkan data *input* berdasarkan kategori *field* yaitu gaji harian (gaji1-10) dan gaji pokok (gaji_pokok1-10). (Baris 5-9). Kemudian, sistem melakukan *update* secara terpisah ke kedua tabel dengan menggunakan *transaction* untuk menjaga konsistensi data. (Baris 13-15 untuk master_jabatan, Baris 18-19 untuk master_jabatan_pokok). Setelah itu, sistem melakukan sinkronisasi melalui *JabatanSyncService* untuk memastikan data tetap konsisten antar tabel. (Baris 23). Terakhir, mengembalikan data yang telah diperbarui. (Baris 25).

4. Fungsi *delete* untuk Penghapusan Data Jabatan

Kode Sumber 3.19 dirancang untuk menghapus data jabatan secara permanen dari dua tabel utama, yaitu master_jabatan sebagai penyimpanan utama data gaji harian, dan master_jabatan_pokok yang berfungsi sebagai penyimpanan data gaji pokok.

```

1. async delete(jabatan: string) {
2.   try {
3.     // Mencari data untuk memastikan record ada
4.     const jabatanData = await MasterJabatan.findOne({ where: { jabatan }, ... });
5.     if (!jabatanData) {
6.       throw new Error("Jabatan tidak ditemukan.");
7.     }
8.
9.     // Hard delete data di tabel utama (master_jabatan)
10.    await jabatanData.destroy({ force: true });
11.
12.    // Menghapus data terkait di tabel pokok via Sync Service
13.    await JabatanSyncService.deleteFromPokok(jabatan);
14.
15.  } catch (error) {
16.    // ... penanganan error
17.    throw new Error("Gagal menghapus data jabatan.");
18.  }
19.

```

Kode Sumber 3.19 *Code delete Master Jabatan*

Kode Sumber 3.19 dimulai dengan melakukan pengecekan keberadaan data jabatan di basis data untuk memastikan data yang akan dihapus benar-benar ada. (Baris 4-7). Selanjutnya, sistem melakukan *hard delete* dengan opsi *force: true* untuk menghapus data secara permanen dari tabel master_jabatan. (Baris 10). Kemudian, sistem melakukan *cascade delete* melalui *JabatanSyncService* untuk menghapus data terkait di tabel master_jabatan_pokok. (Baris 13).

5. Fungsi *validateNonNegativeValues* untuk mengatasi nilai negatif.

Fungsi *validateNonNegativeValues* pada Kode Sumber 3.20 bertugas untuk memvalidasi semua *field* gaji agar tidak berisi negatif (Baris 1). Kode Sumber 3.20 mendefinisikan *array*

gajiFields yang berisi daftar lengkap *field* gaji harian (gaji1-gaji10) dan gaji pokok (gaji_pokok1-gaji_pokok10) yang mungkin ada dalam data. (Baris 2-5). Kemudian sistem melakukan pengecekan terhadap setiap *field* dalam *array* tersebut (Baris 6), memeriksa apakah *field* tersebut ada dalam data dan tidak bernilai *undefined* atau *null*. (Baris 7). Untuk setiap *field* yang ditemukan, fungsi mengonversi nilai ke tipe Number (Baris 8) dan melakukan dua validasi utama. Pertama memastikan nilai tersebut adalah angka yang valid dengan menggunakan *isNaN()* (Baris 9-11), dan kedua memastikan nilai tidak kurang dari nol untuk mencegah gaji negatif (Baris 12-13). Jika ditemukan nilai yang bukan angka, sistem akan menolak dan mengirimkan pesan *error*, sedangkan jika ditemukan nilai negatif. (Baris 9-13).

```

1. const validateNonNegativeValues = (data: Record<string, any>): string | null => {
2.   const gajiFields = [
3.     "gaji1", ..., "gaji10",
4.     "gaji_pokok1", ... ,
5.     "gaji_pokok10"
6.   ];
7.   for (const field of gajiFields) {
8.     if (data[field] !== undefined && data[field] !== null) {
9.       const value = Number(data[field]);
10.      if (isNaN(value)) {
11.        return `Field ${field} harus berupa angka`;
12.      }
13.      if (value < 0) { return `Nilai gaji tidak boleh kurang dari 0`;
14.    }
15.  }
16.  return null;
}

```

Kode Sumber 3.20 *Code Fungsi validateNonNegativeValues* Master Jabatan

6. Fungsi *Sinkronisasi* Data Antar Tabel

Untuk mensinkronasi kedua tabel master_jabatan dan master_jabatan_pokok dibutuhkan fungsi *sync* untuk memastikan data jabatan antara kedua tabel itu ada. Kebutuhan *sync* untuk melakukan operasi baca, perbarui serta hapus.

```

1. async syncToPokok(jabatan: string) {
2.   const exists = await MasterJabatanPokokRepository.findByName(jabatan);
3.   if (!exists) {
4.     await MasterJabatanPokokRepository.create({ jabatan });
5.   }
6.
7.
8.   async syncToHarian(jabatan: string) {
9.     const exists = await MasterJabatanRepository.findByName(jabatan);
10.    if (!exists) {
11.      await MasterJabatanRepository.create({ jabatan });
12.    }
13.  }
}

```

Kode Sumber 3.21 *Code Sinkronisasi* Data Antar Tabel

Fungsi Kode Sumber 3.21 berfungsi untuk menjaga konsistensi data antara tabel master_jabatan dan master_jabatan_pokok. Metode *syncToPokok* memastikan setiap jabatan yang ada di tabel harian juga tersedia di tabel pokok (Baris 1-5), begitu pula sebaliknya dengan *syncToHarian* (Baris 8-12). Proses ini menggunakan pendekatan *lazy synchronization* dimana sinkronisasi dilakukan saat operasi baca, tulis, perbarui serta hapus terjadi, bukan secara *real-time*.

7. Model *MasterJabatan* (*master_jabatan.entity.ts*)

Kode Sumber 3.22 berfungsi untuk mendefinisikan struktur data dan skema basis data untuk entitas master_jabatan yang berisi data gaji harian (Baris 1-5 dan Baris 8-24).

```

1. export class MasterJabatan extends Model {
2.   public jabatan!: string;
3.   public gaji1!: number;
4.   // ... gaji2 hingga gaji10
5. }
6.
7. Definisi Skema Tabel (Sequelize)
8. MasterJabatan.init({
9.   jabatan: {
10.     type: DataTypes.STRING,
11.     primaryKey: true,
12.     allowNull: false,
13.   },
14.   gaji1: {
15.     type: DataTypes.INTEGER,
16.     allowNull: false,
17.     defaultValue: 0
18. },
19. // ... definisi untuk gaji2 hingga gaji10
20. }, {
21.   sequelize: dbConn,
22.   tableName: "master_jabatan",
23.   // ... opsi model lainnya (timestamps, underscored)
24. });

```

Kode Sumber 3.22 Code Model *MasterJabatan*

8. Model *MasterJabatanPokok* (*master_jabatan_pokok.entity.ts*)

Kode Sumber 3.23 berfungsi untuk mendefinisikan struktur data dan skema *database* untuk entitas master jabatan pokok yang berisi data gaji pokok beserta jabatannya dan waktu dibuat serta waktu di perbarui. (Baris 2-8 untuk deklarasi properti, dan Baris 10-24 untuk definisi skema).

```

1. // Deklarasi Class Model
2. export class MasterJabatanPokok extends Model {
3.   public jabatan!: string;
4.   public gaji_pokok1!: number;
5.   // ... properti untuk gaji_pokok2 hingga gaji_pokok10
6.   public readonly created_at!: Date;
7.   public readonly updated_at!: Date;
8. }
9. // Definisi Skema Tabel
10. MasterJabatanPokok.init({
11.   jabatan: {
12.     type: DataTypes.STRING,
13.     primaryKey: true,
14.     allowNull: false,
15.   },
16.   gaji_pokok1: {
17.     type: DataTypes.INTEGER, allowNull: false,
18.     defaultValue: 0
19. },
20. // ... definisi untuk gaji_pokok2 sampai gaji_pokok10
21. }, {
22.   tableName: "master_jabatan_pokok",
23.   sequelize: dbConn,
24.   // ... opsi model lainnya });

```

Kode Sumber 3.23 Code Model *MasterJabatanPokok*

9. Command Pattern (*master_jabatan.command.ts*)

Kode Sumber 3.24 berfungsi untuk mengimplementasikan *command pattern* dalam operasi baca, tulis, perbarui, hapus untuk master jabatan dengan validasi dan *error*. Alur Kode Sumber 3.24 yaitu, validasi keberadaan data sebelum melakukan operasi. Selanjutnya, *CreateMasterJabatanCommand* memastikan tidak ada duplikasi data sebelum membuat jabatan baru (Baris 2-5). Kemudian, *UpdateMasterJabatanCommand* memvalidasi keberadaan

data sebelum melakukan pembaruan (Baris 8-12). *DeleteMasterJabatanCommand* melakukan pengecekan ada atau tidaknya data dan menjalankan *cascade delete* (Baris 15-18).

```
1. // Command untuk Membuat Data
2. export class CreateMasterJabatanCommand {
3.   async execute(data: any) {
4.     // Validasi 'jabatan' belum ada, lalu panggil service untuk membuat.
5.   }
6. }
7.
8. // Command untuk Memperbarui Data
9. export class UpdateMasterJabatanCommand {
10.  async execute(jabatan: string, data: any) {
11.    // Validasi 'jabatan' ada, lalu panggil service untuk memperbarui.
12.  }
13. }
14.// Command untuk Menghapus Data
15.export class DeleteMasterJabatanCommand {
16.  async execute(jabatan: string) {
17.    // Validasi 'jabatan' ada, lalu panggil service untuk menghapus.
18.  }
19. }
```

Kode Sumber 3.24 *Code Command Pattern Master Jabatan*

10. *Query Pattern (master_jabatan.query.ts)*

Kode Sumber 3.25 berfungsi untuk mengimplementasikan *query pattern* dalam operasi pembacaan data master jabatan dan memiliki alur penerapan prinsip *separation of concerns*. Logika pengambilan data jabatan dikelola melalui *service layer*. *GetAllMasterJabatanQuery* bertugas mengambil seluruh data jabatan (Baris 1-5), sementara *GetMasterJabatanByJabatanQuery* mengambil data berdasarkan nama jabatan (Baris 7-11). Keduanya mengembalikan data yang siap digunakan oleh *controller* untuk ditampilkan ke pengguna.

```
1. export class GetAllMasterJabatanQuery {
2.   async execute() {
3.     return await MasterJabatanService.getAllJabatan();
4.   }
5. }
6.
7. export class GetMasterJabatanByJabatanQuery {
8.   async execute(jabatan: string) {
9.     return await MasterJabatanService.getJabatanByJabatan(jabatan);
10.  }
11. }
```

Kode Sumber 3.25 *Code query Pattern untuk Pembacaan Data Master Jabatan*

3.3.4.3 Modul Potongan Keterlambatan

Nama modul Potongan Keterlambatan tidak hanya memuat hanya keterlambatan, namun memuat seluruh potongan gaji seperti potongan pulang cepat, tidak presensi masuk serta tidak presensi pulang. Modul Potongan Keterlambatan merupakan modul yang berfungsi untuk mengelola pemrosesan potongan gaji berdasarkan keterlambatan atau ketidakhadiran guru atau pengguna pada Aplikasi KinderFin 2.0. Modul Potongan Keterlambatan memiliki keterkaitan dengan Modul Master Jabatan dan Modul *Salary Detail* karena penerapan potongan gaji dilakukan berdasarkan jabatan. Modul ini mendukung empat jenis potongan, yaitu datang terlambat, pulang lebih awal, tidak melakukan presensi masuk, dan tidak melakukan presensi pulang. Setiap jenis potongan memiliki persentase pemotongan dan batas menit keterlambatan yang dapat dikonfigurasi sesuai dengan kebutuhan. Terdapat beberapa fungsi utama dalam modul ini, yaitu fungsi pembuatan data potongan (*create*), pengambilan data (*read*), dan penghapusan data (*delete*). Modul ini memenuhi kebutuhan Kasus Penggunaan Pengaturan

Potongan Gaji Harian. Untuk visualisasi modul ini terdapat pada Gambar 3.29 dan Gambar 3.30.

Pertama ada operasi pembuatan yaitu *create*. Operasi pembuatan data potongan keterlambatan dilakukan dengan memasukkan parameter jabatan, urutan gaji yang akan dipotong, persentase potongan, batas menit keterlambatan, dan jenis potongan.

```
1. async create(req: Request, res: Response): Promise<void> {
2.   try {
3.     const { tipe_jam, ... } = req.body;           // Ambil parameter
4.     const jenis_potongan = jenisPotonganMap[tipe_jam]; // Mapping jenis
5.     if (!jenis_potongan) { ... return error }      // Validasi jenis
6.     // ... logika untuk pemeriksaan duplikasi
7.     const result = await createCommand.execute({ ... }); // Simpan di
   Command
8.   // ... logika log Aktivitas
9.   res.status(201).json({ data: result, ... });
10. } catch (err: any) {
11.   // ... penanganan error
12. }
13. // ...
14. }
```

Kode Sumber 3.26 *Code Fungsi create Potongan Keterlambatan*

Kode Sumber 3.26 memiliki alur kerja pertama dengan pengambilan parameter dari request body. (Baris 3). Lalu, sistem akan melakukan pemetaan jenis potongan dari masukan tipe_jam ke format yang sesuai dengan basis data. (Baris 4). Selanjutnya, dilakukan validasi jenis potongan untuk memastikan nilai yang dimasukkan valid. (Baris 5). Sistem selanjutnya melakukan pemeriksaan duplikasi dengan mencari kombinasi jabatan, batas menit, jenis potongan, dan urutan gaji yang sama pada basis data. (Baris 7). Apabila tidak terdapat duplikasi, data potongan disimpan ke tabel potongan_keterlambatan melalui pola command. (Baris 9). Pada tahap akhir, sistem mencatat aktivitas pembuatan potongan ke dalam log. (Baris 11).

```
1. async execute(data: { /* ...parameter lengkap */ }) {
2.   // Validasi jabatan ke tabel master_jabatan
3.   const [result] = await dbConn.query(`SELECT ...`);
4.   if (!result) {
5.     throw new Error("Jabatan tidak ditemukan...");
6.   }
7.   // Validasi jenis potongan
8.   if (!allowedJenisPotongan.includes(data.jenis_potongan)) {
9.     throw new Error("Jenis potongan tidak valid...");
10. }
11. // Simpan data ke basis data
12. const potongan = await this.potonganRepo.create({ ...data });
13. return potongan;
14. }
```

Kode Sumber 3.27 *Code Pola Command Potongan Keterlambatan*

Lanjutan dari penjelasan sebelumnya, kerja pola Kode Sumber 3.27 akan melakukan validasi lanjutan dengan memeriksa jabatan di dalam tabel *master_jabatan*. (Baris 2-3). Jika tidak ditemukan maka sistem akan mengembalikan pesan *error*. (Baris 4-6). Lalu, sistem akan melakukan validasi potongan jika sudah ditemukan jabatan tersebut untuk memastikan nilai yang dimasukkan sesuai format. (Baris 8-11). Terakhir, data akan disimpan di basis data. (Baris 13-14).

Selanjutnya ada fungsi operasi hapus atau *delete*. Pengguna dapat menghapus data potongan yang tidak sesuai atau tidak diperlukan lagi. Fungsi ini akan memerlukan validasi data yang akan benar benar di hapus.

```
1. async delete(req: Request, res: Response): Promise<void> {
2.   try {
3.     const { id } = req.params;           // Ambil ID dari URL
4.
5.     const found = await repo.findById(id); // Cari data by ID
6.     if (!found) {                      // Jika tidak ada, return 404
7.       return res.status(404).json({ message: "..." });
8.     }
9.     await repo.delete(id);            // Hapus data dari basis data
10.    // ... pencatatan aktivitas penghapusan dalam log
11.    res.json({ message: "Potongan berhasil dihapus" });
12.  catch (err: any) {
13.    // ... penanganan error}
14. }
```

Kode Sumber 3.28 *Code* Fungsi *delete* Potongan Keterlambatan

Fungsi Kode Sumber 3.28 memiliki alur kerja yaitu dimulai dengan pengambilan *id* potongan dari parameter URL. (Baris 3). Selanjutnya sistem akan mencari *id* yang dimasukan. (Baris 5). Apabila tidak ditemukan maka sistem akan mengirimkan *kode error* (404). (Baris 6-7). Selanjutnya penghapusan data dari basis data (Baris 9) dan mencatat aktivitas penghapusan dalam *log* dengan menyertakan informasi jabatan yang dihapus. (Baris 10).

Fungsi baca atau *read* data potongan. Fungsi ini terdiri dari dua fungsi utama, yaitu pengambilan seluruh data dan pengambilan data berdasarkan *ID*. Kedua fungsi ini memberikan fleksibilitas dalam mengakses informasi potongan sesuai dengan kebutuhan.

```
1. // Mengambil semua data
2. async findAll(req: Request, res: Response) {
3.   // Memanggil query untuk mendapatkan semua data, lalu kirim response.
4.   const result = await query.getAll();
5.   // ...
6. }
// Mengambil data tunggal berdasarkan ID
7. async findById(req: Request, res: Response) {
8.   // Mencari data berdasarkan ID, lalu kirim response 200 atau 404.
9.   const result = await repo.findById(req.params.id);
10.  // ...
11. }
```

Kode Sumber 3.29 *Code* Fungsi *findAll* dan *findById* Potongan Keterlambatan

Alur kerja pada Kode Sumber 3.29 adalah mengambil seluruh data potongan dari basis data melalui pola *query* (Baris 4) dan mengirimkan kembali dalam bentuk JSON *array*. (Baris 5). Sementara itu, fungsi *findById* mengambil *ID* dari parameter URL (Baris 9), mencari data potongan berdasarkan *ID* tersebut (Baris 9).

Fungsi untuk mendukung saat penggunaan apakah pengguna tersebut mendapat potongan gaji dengan kriteria tersebut atau tidak membutuhkan fungsi yaitu fungsi *findBestPotongan*.

```
1. async findBestPotongan(
2.   jabatan: string, jenis_potongan: string, menit: number
3. ): Promise<PotonganKeterlambatan | null> {
4.   // Query untuk mencari potongan terbaik yang relevan
5.   const query =
6.     `SELECT * FROM potongan_keterlambatan
7.     WHERE jabatan = :jabatan
8.     AND jenis_potongan = :jenis_potongan
9.     AND batas_menit <= :menit
10.    ORDER BY batas_menit DESC
11.    LIMIT 1;
12.  `;
```

```

13. // Eksekusi query dan kembalikan hasilnya
14. const result = await dbConn.query<...>(query, { replacements: { ... }
  });
15. return result[0] || null;

```

Kode Sumber 3.30 Code Fungsi *findBestPotongan* Potongan Keterlambatan

Fungsi Kode Sumber 3.30 gunakan untuk mencari potongan yang paling sesuai berdasarkan jabatan (Baris 2 - 7), jenis potongan (Baris 2, 8), dan jumlah menit keterlambatan (Baris 2, 9). Sistem akan mencari potongan dengan batas menit yang tidak melebihi menit keterlambatan (Baris 9), kemudian mengambil data yang memiliki batas menit tertinggi (Baris 10-11). Hal ini memungkinkan sistem untuk menerapkan potongan yang paling tepat sesuai dengan tingkat keterlambatan guru atau pengguna.

3.3.4.4 Modul *Activity Logs*

Modul *Activity Logs* berfungsi untuk memberikan log aktivitas setiap modul lain jika dilakukan suatu proses di modul tersebut seperti melakukan proses hapus data atau memperbarui data. Modul ini dibangun untuk memenuhi kebutuhan Kasus Penggunaan Lihat Log Aktivitas, serta divisualisasikan pada Gambar 3.46. Beberapa file dibuat untuk mendukung berjalannya modul ini. Penerapan ini dengan menerapkan *clean Architecture*.

File pertama yaitu, Entitas dan Model Data. Dimulai dengan pendefinisian struktur data dan model entitas yang akan digunakan dalam sistem. *File* entitas berperan sebagai representasi objek bisnis dalam domain aplikasi dan menjadi fondasi untuk semua operasi yang berkaitan dengan data log aktivitas. Visualisasi kode Model Data pada Kode Sumber 3.31.

```

1. // Interface untuk struktur data log
2. export interface ActivityLogAttributes {
3.   id?: string;
4.   user_id: string;
5.   email: string;
6.   action: string;
7.   // ... properti lainnya
8.
9. // Definisi Model Sequelize untuk tabel 'activity_logs'
10.export const ActivityLog = db.define("activity_logs", {
11.   id: {
12.     type: DataTypes.UUID,
13.     primaryKey: true,
14.     defaultValue: DataTypes.UUIDV4, },
15.   user_id: {
16.     type: DataTypes.UUID,
17.     allowNull: false,
18.   },
19.   // ... definisi kolom lainnya (email, action, module, dll.)
20. }, {
21.   tableName: "activity_logs",
22.   timestamps: false,
23. });

```

Kode Sumber 3.31 Definisi Entitas *Activity Log*.

Alur implementasi Kode Sumber 3.31 dimulai dengan *import* dependensi yang diperlukan dari Sequelize dan konfigurasi basis data. Selanjutnya, *interface* *ActivityLogAttributes* didefinisikan untuk menentukan struktur data yang konsisten dengan properti *id* sebagai *Primary Key* (Baris 2-7 untuk *interface*; Baris 11-14 untuk *id* di model), *user_id* (Baris 4, 15-18) dan *email* (Baris 5, 19) sebagai identitas pengguna, *action* (Baris 6, 19) dan *module* (Baris 19) untuk kategorisasi aktivitas, serta *description* dan *created_at* (Baris 19) untuk informasi tambahan. Model *ActivityLog* kemudian dibuat menggunakan metode *db.define()* dengan konfigurasi yang mencakup definisi setiap kolom beserta tipe data, *constraint*, dan nilai *default* yang sesuai. (Baris 10-20). Konfigurasi tambahan seperti

tableName (Baris 21) dan *timestamps: false* (Baris 22) memastikan model sesuai dengan struktur tabel yang diinginkan.

File kedua yaitu, *log.repository.ts*. Fungsi dari file ini adalah ini berfungsi sebagai jembatan antara *domain logic* dan *infrastruktur database*. Visualisasi kode Model Data pada Kode Sumber 3.32.

```
1. export interface ActivityLogRepository {
2.   findAll(): Promise<ActivityLogAttributes[]>;
3.
4.   findById(id: string): Promise<ActivityLogAttributes | null>;
5.
6.   create(data: {
7.     user_id: string;
8.     email: string;
9.     action: string;
10.    module: string;
11.    description?: string;
12.  }): Promise<ActivityLogAttributes>;
13. }
```

Kode Sumber 3.32 *Code Repository Activity Log*.

Kode Sumber 3.32 ini mendefinisikan tiga metode utama yang mencakup operasi dasar untuk pengelolaan data *log*. Metode *findAll()* mengembalikan *Promise* yang menyelesaikan ke *ActivityLogAttributes* untuk pengambilan semua data *log* (Baris 2), metode *findById()* menerima parameter *string id* dan mengembalikan data spesifik atau “null” jika tidak ditemukan (Baris 4), sedangkan metode *create()* menerima objek data dengan properti wajib dan opsional untuk pembuatan *log* baru dengan tipe kembalian *ActivityLogAttributes* (Baris 6-12).

File selanjutnya adalah *sequelize-log.repository.ts*. Kode Sumber 3.33 bertanggung jawab untuk menerjemahkan metode antarmuka menjadi kueri basis data. Pada Kode Sumber 3.33 mengimplementasikan semua metode dari antarmuka dengan pendekatan yang berbeda untuk setiap operasi. Metode *findAll()* menggunakan kueri SQL mentah dengan *ORDER BY created_at DESC* untuk mendapatkan data terbaru (Baris 4-7), metode *findById()* menerapkan kueri berparameter dengan penggantian untuk mencegah injeksi SQL (Baris 9-12), sedangkan metode *create()* memanfaatkan model Sequelize untuk memasukkan data (Baris 14-15) dan mengonversi hasil menggunakan *toJSON()* untuk mengembalikan objek sederhana yang sesuai dengan antarmuka (Baris 16).

```
1. export class SequelizeLogRepository implements IActivityLogRepository {
2.
3.   // Menggunakan raw query dengan ORDER BY
4.   async findAll(): Promise<ActivityLogAttributes[]> {
5.     const query = `SELECT ... ORDER BY created_at DESC`;
6.
7.     ...
8.   }
9.   // Menggunakan raw query berparameter untuk keamanan
10.  async findById(id: string): Promise<ActivityLogAttributes | null> {
11.    const query = `SELECT ... WHERE id = :id`;
12.    // ... eksekusi raw query dengan replacements
13.  }
14.  // Menggunakan metode ORM untuk membuat data
15.  async create(data: { ... }): Promise<ActivityLogAttributes> {
16.    const result = await ActivityLog.create(data);
17.    return result.toJSON();
18.  }
19. }
```

Kode Sumber 3.33 Repository dengan *Sequelize Activity Log*

File selanjutnya adalah *create-log.command.ts*. Kode Sumber 3.34 berfungsi untuk mengimplementasikan Pola Perintah dalam operasi pembuatan *log* aktivitas. Kelas perintah ini

bertanggung jawab untuk memvalidasi data masukan dan koordinasikan operasi pembuatan dengan repositori, memisahkan permintaan dari objek yang menjalankan operasi. Pada Kode Sumber 3.34 menyediakan metode *execute()* yang melakukan validasi masukan sebelum operasi *create*. Validasi memastikan bidang wajib *user_id* dan *email* tidak kosong dengan memberikan pesan kesalahan jika validasi gagal (Baris 7-10). Setelah validasi berhasil, perintah mendelegasikan operasi pembuatan ke repositori (Baris 13) dan mengembalikan hasil operasi, memungkinkan penanganan kesalahan dan respons yang konsisten di tingkat pengendali.

```

1. export class CreateLogCommand {
2.   // Menerima repository melalui constructor (Dependency Injection)
3.   constructor(private readonly repo: ActivityLogRepository) {}
4.
5.   async execute(data: { /* ...parameter log */ }) {
6.
7.     // Validasi input wajib (user_id dan email)
8.     if (!data.user_id || !data.email) {
9.       throw new Error("User ID dan email wajib diisi.");
10.    }
11.
12.    // Delegasi proses pembuatan data ke repository
13.    return await this.repo.create(data);
14.  }
15.}
```

Kode Sumber 3.34 Code *CreateLogCommand Activity Log*

File selanjutnya adalah *activity_log.controller.ts* berfungsi sebagai pengendali yang menangani permintaan dan respons *HTTP* untuk titik akhir *Activity Log*. *Controller* berperan sebagai penghubung antara lapisan presentasi dengan lapisan aplikasi, mengkoordinasikan operasi *create* dan pembacaan *log* aktivitas serta menangani penanganan kesalahan. Kode akan divisualisasikan pada Kode Sumber 3.35. Alur kerjanya yaitu dimulai dengan penanganan permintaan pembuatan *log* melalui fungsi *create*. Pada fungsi ini, sistem menerima *request*, mendelegasikan eksekusi pembuatan *log* ke *createCommand* (Baris 7), dan mengirimkan *response* dengan *status* 201 jika berhasil (Baris 8) atau *status* 400 jika terjadi *error* validasi (Baris 9-11). Kemudian, sistem juga menangani permintaan untuk membaca semua *log* melalui fungsi *findAll*. Di fungsi ini, sistem mendelegasikan pengambilan semua data *log* ke *repository* (Baris 18), dan mengirimkan *response* dengan *status* 200 jika berhasil (Baris 19) atau *status* 500 jika terjadi *error* server (Baris 20-22).

```

1. export class ActivityLogController {
2.
3.   // Menangani permintaan pembuatan log
4.   async create(req: Request, res: Response): Promise<void> {
5.     try {
6.       // Delegasi ke 'CreateLogCommand' untuk eksekusi
7.       const result = await createCommand.execute(req.body);
8.       res.status(201).json({ data: result, ... });
9.     } catch (err: any) {
10.      res.status(400).json({ message: err.message });
11.    }
12.  }
13.
14.   // Menangani permintaan untuk membaca semua log
15.   async findAll(req: Request, res: Response): Promise<void> {
16.     try {
17.       // Delegasi ke 'repository' untuk mengambil data
18.       const all = await repo.findAll();
19.       res.status(200).json(all);
20.     } catch (err: any) {
21.       res.status(500).json({ message: "..." });
22.     }
23.   }
24. }
```

Kode Sumber 3.35 Code *Controller Activity Log*

File selanjutnya adalah *activity-log.table.ts* berfungsi sebagai migrasi basis data untuk membuat struktur tabel *activity_logs* dalam basis data. Migrasi memastikan bahwa struktur tabel sesuai dengan definisi entitas dan dapat dijalankan secara konsisten di berbagai lingkungan. *File* migrasi ini untuk pembuatan tabel dan turun untuk *rollback* jika diperlukan. *File* ini akan divisualisasikan pada Kode Sumber 3.36. Migrasi ini mendefinisikan struktur tabel dengan kolom *id* sebagai kunci utama *UUID* dengan pembuatan otomatis, *user_id* diambil dari tabel *users* *UUID*, *email* dan *action*, *description*, dan *created_at* dengan waktu bawaan (Baris 4-16).

```

1. module.exports = {
2.   // Fungsi 'up' untuk membuat tabel 'activity_logs'
3.   up: async (queryInterface: QueryInterface) => {
4.     await queryInterface.createTable("activity_logs", {
5.       id: {
6.         type: DataTypes.UUID,
7.         primaryKey: true,
8.         defaultValue: DataTypes.UUIDV4,
9.       },
10.      user_id: {
11.        type: DataTypes.UUID,
12.        allowNull: false,
13.      },
14.      // ... definisi kolom lainnya (email, action, module, description)
15.      created_at: {
16.        type: DataTypes.DATE,
17.        defaultValue: DataTypes.NOW,
18.      },
19.    });
20.   // Fungsi 'down' untuk rollback (menghapus tabel)
21.   down: async (queryInterface: QueryInterface) => {
22.     await queryInterface.dropTable("activity_logs");
23.   },
24. };

```

Kode Sumber 3.36 Migrasi Tabel *Activity Log*

3.3.4.5 Modul *Salary Detail*

Modul ini merupakan modul yang berisi mengenai pemrosesan detail gaji harian setiap guru pada Aplikasi KinderFin 2.0. Pembangunan modul ini untuk memenuhi kebutuhan Kasus Penggunaan Tambah Data Presensi, Kasus Penggunaan Validasi Gaji pada sub-bab 3.1.4.2, Kasus Penggunaan Cetak Gaji pada sub-bab 3.1.4.3, dan Kasus Penggunaan Lihat Rincian Gaji pada sub-bab 3.1.4.5. Visualisasi Modul *Salary Detail* terdapat pada sub-bab 3.3.2.3, 3.3.2.4, 3.3.2.5, dan 3.3.2.7. Pada sub-bab 3.3.2.5 merangkap tampilan dengan Modul Rekap Bonus. Modul ini berhubungan dengan modul Master Jabatan dan Potongan Keterlambatan karena mengambil komponen gaji berdasarkan jabatan guru dan menerapkan sistem potongan keterlambatan yang telah dikonfigurasi. Modul ini akan menghitung gaji harian berdasarkan presensi guru, menerapkan potongan sesuai aturan yang telah ditetapkan, dan menyimpan hasil perhitungan ke basis data.

Penerapan fungsionalitas umum pada semua modul termasuk Modul *Salary Detail* dijelaskan di LAMPIRAN G, sub-bab ini akan membahas logika program (*backend*) dan secara spesifik fungsionalitas inti antarmuka yang membedakan Modul *Salary Detail* dengan modul lain. Fungsi utama dalam modul ini meliputi perhitungan gaji harian, pemrosesan data presensi, generate PDF slip gaji, dan manajemen data gaji bulanan. Modul ini tersebar dalam beberapa file yang menangani aspek berbeda dari sistem penggajian harian. Fungsi pada modul ini terdiri dari

1. Fungsi *calculateSalary* (*calculate-salary.command.ts*)

Fungsi *calculateSalary* bertugas untuk menghitung gaji harian guru berdasarkan data presensi dan menerapkan berbagai jenis potongan keterlambatan. Fungsi ini dimulai dengan mengambil seluruh aturan potongan keterlambatan dari basis data tabel *potongan_keterlambatan* (Baris 6-7). Selanjutnya, sistem memproses data presensi baik dari *server developer.fingerspot.io*, *file CSV* atau dari *xls* atau *xlsx* maupun *Input Manual*. (Baris 9-10 untuk *manual*, Baris 23 untuk *upload*). Lalu, untuk setiap data presensi, sistem mencari data guru berdasarkan NIP (Baris 14 untuk *input manual*, Baris 29 untuk *upload file*) dan mengambil komponen gaji sesuai jabatan (Baris 15, 29). Setelah itu, sistem menghitung menit keterlambatan dan pulang cepat, menerapkan potongan sesuai aturan yang berlaku (Baris 18, 30). Terakhir, sistem menyimpan hasil perhitungan ke tabel *detail_salary* (Baris 21, 30) dan mengembalikan jumlah data yang berhasil diproses (Baris 34-35). Fungsi ini akan divisualisasikan pada Kode Sumber 3.37.

```
1. async function calculateSalary(
2.   filePathOrArray: string | any[],
3.   source: "upload" | "manual"
4. ): Promise<{ processed: number, skipped: number }> {
5.
6.   // Mengambil seluruh aturan potongan dari basis data
7.   const potonganList = await dbConn.query("SELECT * FROM
potongan_keterlambatan ...");
8.
9.   // Memproses data presensi berdasarkan sumber (Manual atau upload)
10.  if (source === "manual") {
11.    // Logika untuk Input Manual
12.    for (const record of filePathOrArray as any[]) {
13.      // Mencari data guru & komponen gajinya
14.      const teacher = await
SalaryDetailRepository.getTeacherByNip(record.nip);
15.      const salaryComponents = await
SalaryDetailRepository.getSalaryComponentsByJabatan(...);
16.      // ... validasi guru & komponen gaji
17.
18.      // Menghitung dan menerapkan semua jenis potongan (terlambat, pulang
cepat, dll)
19.      // ... (blok logika kalkulasi utama)
20.      // Menyimpan hasil perhitungan ke basis data
21.      await SalaryDetailRepository.saveOrUpdateSalaryDetail({ ... });
22.
23.    }
24.  } else { // source === "upload"
25.    // Logika untuk memproses file upload (membaca file, cek duplikat hash)
26.    //
27.    // (Di dalam event 'end' setelah file dibaca)
28.    for (const record of attendanceRecords) {
29.      // Alur kerja di dalam loop ini SAMA seperti pada blok 'manual'
30.      // (mencari guru, menghitung potongan, menyimpan hasil)
31.
32.    }
33.  }
34.
35.  // Mengembalikan jumlah data yang berhasil diproses
36.  return { processed: ..., skipped: ... };
37. }
```

Kode Sumber 3.37 *Code calculateSalary Salary Detail*

2. Fungsi *getBestPotongan* (*calculate-salary.command.ts*)

Pada Kode Sumber 3.38 ini bertugas untuk mencari aturan potongan terbaik yang sesuai dengan jabatan, jenis potongan, dan durasi pelanggaran. Kode Sumber 3.38 memiliki alur yang dimulai dengan menyusun *query SQL* untuk mencari aturan potongan yang sesuai. (Baris 2-9). Selanjutnya, sistem melakukan penyaringan berdasarkan jabatan (Baris 4), jenis potongan (Baris 5), dan batas menit yang telah ditetapkan (Baris 6). Lalu, sistem mengurutkan hasil

berdasarkan *batas_menit* secara *descending* mengurutkan menurun untuk mendapatkan aturan yang paling tepat (Baris 7). Setelah itu, sistem mengambil satu data pertama sebagai aturan terbaik (Baris 8). Terakhir, sistem mengembalikan aturan potongan atau *null* jika tidak ditemukan (Baris 14).

```

1. async function getBestPotongan(jabatan: string, jenis_potongan: string, menit: number): Promise<PotonganKeterlambatan | null> {
2.   const query =
3.     `SELECT * FROM potongan_keterlambatan
4.     WHERE jabatan = :jabatan
5.     AND jenis_potongan = :jenis_potongan
6.     AND batas_menit <= :menit
7.     ORDER BY batas_menit DESC
8.     LIMIT 1
9.   `;
10.  const result = await dbConn.query<PotonganKeterlambatan>(query, {
11.    replacements: { jabatan, jenis_potongan, menit },
12.    ...
13.  });
14.  return result[0] ?? null;
15. }

```

Kode Sumber 3.38 *Code getBestPotongan Salary Detail*

3. Fungsi *saveOrUpdateSalaryDetail* (*salary_detail.repository.ts*)

Pada Kode Sumber 3.39 bertugas untuk menyimpan atau memperbarui data gaji harian guru di basis data. Fungsi dimulai dengan mengecek keberadaan data gaji *existing* apakah sudah terdapat gaji atau belum berdasarkan *teacher_id* dan tanggal (Baris 3-6). Selanjutnya, sistem menghitung total gaji sebelum potongan dengan menjumlahkan seluruh komponen gaji (Baris 9). Lalu, sistem menghitung total potongan dari berbagai jenis pelanggaran (Baris 10). Setelah itu, sistem menghitung total gaji setelah potongan dengan mengurangi gaji dengan total potongan (Baris 11). Terakhir, sistem melakukan *update* jika data sudah ada (Baris 14-15) atau *create* jika data baru (Baris 16-17). Ilustrasi pada keadaan nyata yaitu semisal guru belum melakukan presensi pulang karena belum waktunya pulang, sistem akan menerapkan potongan “tidak presensi pulang” namun saat sudah presensi pulang, sistem akan memperbarui dengan cek ulang apakah potongan “pulang cepat” atau tidak.

```

1. static async saveOrUpdateSalaryDetail(salaryData: Record<string, any>) {
2.   try {
3.     // Cek keberadaan data gaji sebelumnya
4.     const existing = await SalaryDetail.findone({
5.       where: { teacher_id: salaryData.teacher_id, tanggal: salaryData.tanggal
6.     });
7.     // Logika untuk menghitung total gaji & total potongan
8.     const totalGajiSebelumPotongan = /* ... hitung dari semua field 'gaji' ... */
*/ ;
9.     const totalPotongan = /* ... hitung dari semua field 'potongan' ... */ ;
10.    salaryData.total_gaji_setelah_potongan = totalGajiSebelumPotongan - totalPotongan;
11.    // Lakukan UPDATE jika data ada, atau CREATE jika data baru
12.    if (existing) {
13.      await existing.update(salaryData);
14.    } else {
15.      await SalaryDetail.create(salaryData);
16.    }
17.  } catch (error) {
18.   // ... penanganan error
19. }
20. //}
21. }

```

Kode Sumber 3.39 *Code saveOrUpdateSalaryDetail Salary Detail*

4. Fungsi *getAllSalaryByTeacher* (*salary_detail.controller.ts*)

Pada Kode Sumber 3.40 memiliki fungsi untuk mengambil seluruh data gaji setiap guru. Fungsi pada Kode Sumber 3.40 memiliki alur yaitu dengan mengambil parameter NIP pada guru yang akan diambil datanya (Baris 3). Lalu, sistem akan mengirimkan data seluruh gaji yang tersedia dari guru tersebut (Baris 8). Jika tidak ada gaji maka sistem mengirimkan pesan “Data gaji tidak ditemukan” (Baris 10-11) dan gagal mengambil atau terjadi kesalahan maka akan mengirimkan “Terjadi kesalahan pada server” (Baris 15-17).

```
1. async getAllSalaryByTeacher(req: Request, res: Response) {
2.   try {
3.     const nip = req.params.nip?.trim();
4.     ...
5.     // Ambil data gaji dari repository berdasarkan NIP
6.     const salaryDetails = await
7.       SalaryDetailRepository.getAllSalaryByTeacherIdentifier(nip);
8.     ...
9.     // Kirim respons 404 jika tidak ditemukan, atau data jika berhasil
10.    if (!salaryDetails || salaryDetails.length === 0) {
11.      return res.status(404).json({ error: "Data gaji tidak ditemukan" });
12.    } else { ... }
13.  } catch (error) {
14.    // ... penanganan kesalahan server (500)
15.  }
16. }
```

Kode Sumber 3.40 *Code getAllSalaryByTeacher Salary Detail*

5. Fungsi *processFinalSalary* (*process-final-salary.command.ts*)

Pada Kode Sumber 3.41 memiliki fungsi untuk memproses perhitungan gaji final dalam rentang periode tertentu dan membuat *file PDF* slip gaji. Alur fungsinya dimulai dengan mengambil seluruh data guru dari basis data (Baris 3). Selanjutnya, untuk setiap guru, sistem mengambil gaji pokok berdasarkan jabatan (Baris 6) dan data gaji harian dalam periode yang ditentukan (Baris 7). Lalu, sistem menghitung total gaji harian (Baris 8) dan gaji pokok (Baris 9) untuk mendapatkan total gaji kombinasi. Setelah itu, sistem menyusun data hasil akhir untuk setiap guru (Baris 10-14). Terakhir, sistem membuat *file PDF* berisi ringkasan gaji total menggunakan *service PDF generator* (Baris 15).

```
1. async function processFinalSalary(startDate: string, endDate: string): Promise<void> {
2.   try {
3.     const teachers = await TeacherRepository.findAll();
4.     const hasilAkhir: GuruGajiData[] = [];
5.     for (const teacher of teachers) {
6.       const jabatan = await
7.         MasterJabatanPokokRepository.findByName(teacher.jabatan);
8.       const salaryDetails = await
9.         SalaryDetailRepository.findByTeacherAndPeriod(teacher.id, startDate, endDate);
10.      const totalGajiHarian = salaryDetails.reduce((sum, detail) => sum +
11.        detail.total_gaji_setelah_potongan, 0);
12.      const totalGajiPokok = jabatan ? Object.values(jabatan).filter(val =>
13.        typeof val === 'number').reduce((sum, val) => sum + val, 0) : 0;
14.      hasilAkhir.push({
15.        ...teacher,
16.        total_gaji_pokok: totalGajiPokok,
17.        total_gaji_harian: totalGajiHarian,
18.        ... });
19.    }
20.    await generateFinalSalaryPDF(hasilAkhir, outputPath); } catch (error)
21.  }
```

Kode Sumber 3.41 *Code processFinalSalary Salary Detail*

6. Fungsi *generateFinalPdf* (*salary_detail.controller.ts*)

Kode Sumber 3.42 berfungsi untuk menghasilkan *file PDF* slip gaji final guru dalam periode tertentu. Alur fungsi Gaji dimulai dengan mengambil data guru berdasarkan *teacher_id* yang diminta. (Baris 3). Selanjutnya, untuk setiap guru, sistem mengambil data jabatan dan gaji harian dalam periode yang ditentukan. (Baris 6,pada *logika perulangan*). Lalu, sistem menghitung total gaji pokok dan gaji harian, kemudian menyusun data final termasuk uang tambahan jika ada. (Baris 6, pada *logika kalkulasi*). Setelah itu, sistem menghasilkan *file PDF* menggunakan *service PDF generator* dengan lokasi *file* yang yang sudah ditentukan berdasarkan NIP. (Baris 9). Terakhir, jika ada uang tambahan, sistem menyimpan *record bonus* (Baris 12-13) dan mengembalikan *response* berisi lokasi *file PDF* (Baris 16).

```
1. async generateFinalPdf(req: Request, res: Response) {  
2.   try {  
3.     const { teacher_ids, start_date, end_date, uang_tambahan } = req.body;  
4.     // Mengumpulkan & menghitung data gaji final untuk semua guru terpilih  
5.     const hasilAkhir = /* ... logika perulangan & kalkulasi ... */ ;  
6.     // Membuat file PDF dari data yang sudah dihitung  
7.     const outputPath = await generateFinalSalaryPDF(hasilAkhir, ...);  
8.     // Simpan data bonus jika ada  
9.     if (uang_tambahan > 0) {  
10.       await BonusRepository.create({ ... });  
11.     }  
12.     // Kirim response berisi lokasi file  
13.     res.status(200).json({ file_path: outputPath, ... });  
14.   } catch (error) {  
15.     // ... penanganan error  
16.   }  
17. }
```

Kode Sumber 3.42 *Code generateFinalPdf Salary Detail*

7. Fungsi *readExcelFile* (*upload.controller.ts*)

Pada Kode Sumber 3.43 ini memiliki fungsi yaitu untuk membaca dan mem-parse *file Excel* menjadi format data presensi yang dapat diproses sistem. Alurnya yaitu, membuka *workbook Excel* dan mengambil *sheet* yang ditentukan, pada fungsi ini mengambil *sheet* pertama (Baris 4-5). Selanjutnya, sistem mencari informasi periode dari *sheet* dan mengekstrak daftar tanggal dalam periode tersebut (Baris 7-9). Lalu, sistem mengkonversi *sheet* menjadi *array* data (Baris 8) dan mencari indeks kolom NIP. (Baris 15, implisit). Setelah itu, sistem melakukan iterasi untuk setiap baris data guru dan kolom tanggal untuk mengekstrak jam masuk dan keluar (Baris 14-17). Terakhir, sistem menyusun *array* data presensi yang siap diproses oleh sistem perhitungan gaji (Baris 11, 16, 20).

```
1. async function readExcelFile(filePath: string): Promise<any[]> {  
2.   try {  
3.     // Membuka file Excel dan mengambil sheet yang relevan  
4.     const workbook = XLSX.readFile(filePath);  
5.     const worksheet = workbook.Sheets[workbook.SheetNames[0]];  
6.     // Mengonversi sheet & mengekstrak informasi periode/tanggal  
7.     const data = XLSX.utils.sheet_to_json<string[]>(worksheet, { header: 1 });  
8.     // ... logika untuk mencari periode dan daftar tanggal ...  
9.     const attendanceArray: any[] = [];  
10.    // Iterasi setiap baris untuk mengekstrak data presensi  
11.    for (let i = dataStartIndex; i < data.length; i++) {  
12.      // ... proses ekstraksi NIP, nama, dan data presensi per tanggal ...  
13.      attendanceArray.push({ /* ... data presensi terstruktur ... */});  
14.    }  
15.  }
```

```

18.
19.    // Mengembalikan array data presensi yang siap diproses
20.    return attendanceArray;
21.
22. } catch (error) {
23.    // ... penanganan error saat membaca file
24.    return [];
}

```

Kode Sumber 3.43 *Code readExcelFile* untuk Pembacaan File Excel

No	Nama	Dept.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
			Sel	Rab	Kam	Jum	Sab	Mgg	Sen	Sel	Rab	Kam	Jum	Sab	Mgg	Sen	Sel	Rab	Kam	Jum	Sab	Mgg	Sen	Se
5							08:19 09:10	06:57 11:57	06:57 16:42	06:59 16:58	16:57 17:39			07:08 17:10	07:10 17:06	07:04 16:47	06:59 17:33				06:58 17:00			
6																								
7																								
8																								
9																								
10																								
11																								

Gambar 3.48 Contoh File dari Mesin Presensi Fingerprint

Fungsi tersebut akan membaca file dengan isi file seperti Gambar 3.48. Sistem akan membaca Periodenya lalu disimpan dengan format XX – 04 – 2025 dan setelahnya membaca tanggal yang dimasukan ke XX dari format periodenya. Lalu akan dicocokkan dengan potongan serta besaran gaji setiap No yaitu NIP jika di Aplikasi dan Nama, setiap NIP dan Nama akan dicocokkan dengan jabatan pengguna tersebut.

8. Fungsi validateExcelSheet (*upload.controller.ts*)

Pada Kode Sumber 3.44 ini memiliki fungsi untuk menolak file dengan format yang tidak sesuai dengan file yang dikeluarkan oleh mesin *fingerprint*. Dasar dari pencocokan file yang diunggah dengan format yang diberikan mesin *fingerprint* yaitu terletak pada Sheet atau lembar kerja “catatan” pada file berformat .xls dan .xlsx. Jika terdapat terdapat Sheet atau lembar kerja “catatan” pada file berformat .xls dan .xlsx, akan dilanjutkan pada fungsi Kode Sumber 3.43. Alur kerjanya yaitu dimulai dengan membaca file Excel untuk memeriksa nama-nama sheetnya (Baris 4). Selanjutnya, sistem mengecek apakah ada sheet yang mengandung nama “catatan” secara case-insensitive (Baris 6-8). Jika sheet “catatan” tidak ditemukan, fungsi akan mengembalikan objek dengan *isValid: false* dan pesan *error* (Baris 10-13). Jika sheet “catatan” ditemukan, fungsi akan mengembalikan objek dengan *isValid: true* (Baris 16). Apabila terjadi *error* saat membaca file, fungsi akan menangkap *error* tersebut dan mengembalikan *isValid: false* dengan pesan *error* terkait (Baris 15-18).

```

1. function validateExcelSheet(filePath: string): { isValid: boolean; error?: string } {
2.   try {
3.     // Membaca file Excel untuk memeriksa nama-nama sheet-nya

```

```

4.     const workbook = XLSX.readFile(filePath);
5.
6.     // Mengecek apakah ada sheet yang mengandung nama "catatan" (case-
7.     // insensitive)
8.     const catatanSheet = workbook.SheetNames.find(
9.       name => name.toLowerCase().includes('catatan')
10.      );
11.     // Jika tidak ditemukan, kembalikan hasil tidak valid
12.     if (!catatanSheet) {
13.       return { isValid: false, error: "Gagal Memproses File. Gunakan File
14. sesuai Format..." };
15.     }
16.     // Jika ditemukan, kembalikan hasil valid
17.     return { isValid: true };
18.
19.   } catch (error: any) {
20.     // ... penanganan error jika file gagal dibaca
21.     return { isValid: false, error: `Gagal membaca file...` };
22.   }
23. }

```

Kode Sumber 3.44 Code Fungsi *validateExcelSheet (upload.controller.ts)*

9. Fungsi *generateFinalSalaryPDF (pdf-generator.service.ts)*

Kode Sumber 3.45 memiliki tugas yaitu, membuat *file PDF* yang berisi mengenai detail gaji pokok dan total gaji harian serta uang tambahan berupa bonus. Alur kerjanya yaitu dengan membuat dokumen PDF dan menentukan lokasi *file* yang akan disimpan (Baris 7-8). Selanjutnya, sistem membuat fungsi untuk menggambar garis tabel yang akan digunakan dalam pembuatan tabel rincian kehadiran (Baris 16-17). Kemudian, untuk setiap guru dalam daftar, sistem menghitung ulang total gaji harian berdasarkan rincian kehadiran (Baris 10 pada *logika kalkulasi*). Setelah itu, sistem menampilkan informasi dasar guru seperti nama, NIP, dan jabatan, serta rincian gaji berupa gaji pokok, gaji harian, dan bonus jika ada (Baris 13 pada *doc.text()*). Selanjutnya, sistem menghitung total gaji keseluruhan dengan menjumlahkan gaji pokok, gaji harian, dan bonus (Baris 12). Kemudian, jika tersedia data rincian kehadiran harian, sistem membuat tabel yang menampilkan tanggal, jam masuk, dan jam pulang (Baris 16-18).

```

1. export const generateFinalSalaryPDF = (
2.   hasilAkhir: GuruGajiData[],
3.   outputPath: string,
4.   // ...
5. ) => {
6.   return new Promise<void>((resolve) => {
7.     const doc = new PDFDocument();
8.     doc.pipe(fs.createWriteStream(outputPath));
9.
10.    hasilAkhir.forEach((item, index) => {
11.      // Kalkulasi & tampilkan info gaji ke PDF
12.      const totalGaji = /* ... logika hitung total gaji ... */ ;
13.      // ... doc.text() untuk nama, NIP, gaji, total.
14.
15.      // Gambar tabel rincian kehadiran
16.      if (item.rincian?.length > 0) {
17.        // ... logika untuk menggambar tabel rincian }
18.
19.      // Tambah halaman baru jika bukan guru terakhir
20.      if (index < hasilAkhir.length - 1) doc.addPage();
21.    });
22.
23.
24.    doc.end();
25.    stream.on("finish", () => resolve());
26.  });
27. };

```

Kode Sumber 3.45 Code *generateFinalSalaryPDF*

10. Fungsi *getSalaryDetail* (*get-salary-detail.query.ts*)

untuk Mengambil Detail Gaji Guru Berdasarkan ID dan Tanggal dan memiliki tugas yaitu, mengambil dan memformat data gaji seorang guru dari basis data untuk ditampilkan. Alur kerjanya yaitu dengan menerima *ID* guru dan tanggal sebagai *input* (Baris 1). Selanjutnya, sistem memanggil *repository* untuk mengambil data gaji dari basis data dari *teacher ID* dan tanggal (Baris 3). Kemudian, sistem melakukan validasi apakah data gaji ditemukan atau tidak. Jika data tidak ditemukan atau kosong, sistem menampilkan pesan *error* "Data salary tidak ditemukan" (Baris 4-6). Setelah itu, jika data ditemukan, sistem memformat data menjadi struktur *object* dengan memberikan nilai *default* untuk *field* yang kosong. (Baris 7-9). Selanjutnya, sistem mengembalikan informasi dasar guru seperti *teacher_id*, *nama_lengkap*, dan jabatan, serta rincian kehadiran berupa tanggal, *jam_masuk*, dan *jam_keluar*. Kemudian, sistem menampilkan informasi potongan gaji seperti *potongan_datang_telat* dan *potongan_pulang_cepat*, beserta *total_salary* dan komponen gaji detail (*gaji1-gaji10*) dalam *object salary_components*. (Baris 7-9).

```
1. async function getSalaryDetail(teacherId: string, tanggal: string) {  
2.   // Ambil data dari repository  
3.   const salaryDetail = await  
4.   SalaryDetailRepository.getSalaryByTeacherAndDate(teacherId, tanggal)  
// Fungsi pesan error: validasi jika data tidak ditemukan  
5.   if (!salaryDetail || Object.keys(salaryDetail).length === 0) {  
6.     return { message: "Data salary tidak ditemukan" };  
7.   }  
8.   // ... struktur data gaji  
9. };  
10. }
```

Kode Sumber 3.46 *Code get-salary-detail.query.ts*

11. Fungsi *getMySalaryDetail* (*my_salary_detail.controller.ts*)

Pada Kode Sumber 3.47, fungsi *getMySalaryDetail* bertujuan untuk mengambil detail gaji guru berdasarkan ID guru. Alur fungsinya dimulai dengan mendapatkan *teacherId* dari *res.locals.id_informasi_tambahan* oleh *middleware autentikasi* (Baris 5). Selanjutnya, fungsi melakukan pengecekan apakah *teacherId* ada atau tidak. Jika tidak, sistem akan mengembalikan *respons error 401 (Unauthorized)* yang menandakan bahwa *ID* guru tidak ditemukan dalam permintaan (Baris 6). Jika *teacherId* ditemukan, sistem kemudian memanggil metode *getAllSalaryByTeacherIdentifier* dari *SalaryDetailRepository* untuk mengambil semua detail gaji yang terkait dengan guru tersebut (Baris 9). Setelah data diterima, fungsi akan memeriksa apakah ada detail gaji yang ditemukan. Jika tidak ada data (*null*), sistem akan mengembalikan *respons error 404 (Not Found)* (Baris 12-14). Apabila detail gaji berhasil ditemukan, sistem akan mengembalikan *respons sukses 200 (OK)* beserta data detail gaji guru tersebut (Baris 17). Fungsi ini juga dilengkapi dengan blok *try...catch* untuk menangani kesalahan tak terduga selama proses, di mana sistem akan mengembalikan *respons error 500 (Internal Server Error)* jika terjadi kesalahan (Baris 19-22)

```
1. class MySalaryDetailController {  
2.   static async getMySalaryDetail(req: Request, res: Response) {  
3.     try {  
4.       // Validasi ID dari middleware (jika tidak ada -> 401)  
5.       const teacherId = res.locals.id_informasi_tambahan;  
6.       if (!teacherId) return res.status(401).json({ message: "..." });  
7.     }  
8.     // Ambil data dari repository  
9.     const salaryDetails = await  
SalaryDetailRepository.getAllSalaryByTeacherIdentifier(teacherId);
```

```

10.     // validasi hasil data (jika kosong -> 404)
11.     if (!salaryDetails || salaryDetails.length === 0) {
12.         return res.status(404).json({ message: "... error" });
13.     }
14.
15.     // Respons sukses (200 OK)
16.     return res.status(200).json({ data: salaryDetails });
17.
18. } catch (error) {
19.     // Respons error server (500)
20.     return res.status(500).json({ message: "..." });
21. }
22.
23. }
24. }

```

Kode Sumber 3.47 *Code getMySalaryDetail* untuk Pengambilan Detail Gaji Guru

12. Halaman Daftar Guru dan Unggah Gaji (*salary_detail/index.tsx*)

Kode Sumber 3.48 terdapat pada modul *Salary Detail*. Modul *Salary Detail* adalah modul paling kompleks dan multifungsi. Fitur inti yang membedakannya yaitu, Unggah *file* kehadiran (*fingerprint*) (Baris 1) dengan mekanisme *polling* status proses di *backend* dan manajemen *state* unggahan melalui *sessionStorage* (Baris 2). Sinkronisasi data presensi *Fingerspot* dengan validasi rentang tanggal maksimal 2 hari, serta menampilkan daftar guru, log unggahan, dan riwayat *file* yang telah diunggah (Baris 3).

```

1. const handleFileUpload = async (file: File | null) =>
    ...
2. useEffect(() => [user, uploading]);asd
    ...
3. const handleFetchFingerspotData = async () =>
    ...

```

Kode Sumber 3.48 *Code* Fungsi Unggah dan Sinkronisasi Data Kehadiran

13. Halaman Detail Gaji Guru setiap NIP (*salary_detail/[nip].tsx*)

Kode Sumber 3.49 menampilkan antarmuka bagi pengguna Bendahara untuk melihat dan mengelola detail gaji guru spesifik berdasarkan NIP. Fitur utamanya adalah *Input* Kehadiran Manual melalui modal (Baris 1), Proses Gaji Bulanan untuk mengunduh laporan PDF dengan opsi bonus (Baris 2) dan tanpa bonus (Baris 3), Hapus Gaji Harian (Baris 4). Modul ini juga memiliki kumpulan fungsi utilitas tanggal yang spesifik untuk validasi dan konversi format.

```

1. const handleSubmitManual = async () =>asd
    ...
2. const handleSubmitDownloadWithBonus = async () => asd
    ...
3. const handleSubmitDownloadWithoutBonus = async () => asd
    ...
4. const handleDelete = async () =>
    ...

```

Kode Sumber 3.49 *Code* Fungsi Manajemen Gaji Guru (Input, Proses, Hapus)

3.3.4.6 Modul Rekap Bonus

Modul ini merupakan modul yang berisi mengenai pemprosesan dan pencatatan bonus tambahan untuk guru pada Aplikasi KinderFin 2.0. Modul ini berfungsi untuk mencatat uang tambahan yang diberikan kepada guru dalam periode tertentu dan menyimpan rekap bonus harian berdasarkan rentang tanggal yang ditentukan. Modul ini dibangun untuk memenuhi kebutuhan Kasus Penggunaan Tambah Bonus. Rekap bonus merangkap menjadi satu dengan Tampilan Modul *Salary Detail* yaitu pada Gambar 3.39. Modul ini memiliki hubungan dengan modul *Salary Detail* untuk melengkapi komponen penghasilan guru. Terdapat fungsi yang mendukung modul ini berjalan.

1. Fungsi *execute CreateRekapBonusCommand* (*rekap_bonus.command.ts*)

Pada Kode Sumber 3.50 berfungsi ini bertanggung jawab untuk membuat rekap bonus untuk guru dalam rentang tanggal tertentu. *Input* yang diterima adalah *start_date*, *end_date*, NIP, *uang_tambahan*, dan keterangan. Sebelum memproses, fungsi ini melakukan validasi terhadap validitas dan urutan tanggal (Baris 8-14). Selanjutnya, melalui sebuah perulangan, fungsi ini akan mengiterasi setiap hari dalam rentang tanggal yang ditentukan (Baris 16), dan untuk setiap tanggal, akan memanggil *RekapBonusService.createRekap* (Baris 19). Pemanggilan ini meneruskan tanggal spesifik, *start_date*, *end_date*, NIP, serta jumlah *uang_tambahan* dan keterangan ke lapisan *Service* (Baris 19-23).

```
1. export class CreateRekapBonusCommand {
2.   async execute(data: {
3.     start_date: string;
4.     end_date: string;
5.     nip: string;
6.     // ...
7.   }) {
8.     // validasi input tanggal
9.     const start = new Date(data.start_date);
10.    const end = new Date(data.end_date);
11.    if (isNaN(start.getTime()) || isNaN(end.getTime())) {
12.      throw new Error("Start date atau end date tidak valid.");
13.    }
14.    // ... validasi tanggal
15.    // Iterasi untuk setiap hari dalam rentang tanggal yang ditentukan
16.    for (let d = new Date(start); d <= end; d.setDate(d.getDate() + 1)) {
17.      const tanggal = d.toISOString().slice(0, 10);
18.      // Panggil service untuk membuat rekap bonus per hari
19.      await RekapBonusService.createRekap({
20.        tanggal: tanggal,
21.        nip: data.nip,
22.        ...data, // Meneruskan data
23.      });
    }
```

Kode Sumber 3.50 *Code execute CreateRekapBonusCommand* Rekap Bonus

2. Fungsi *execute GetRekapBonusByNipQuery* (*rekap_bonus.query.ts*)

Pada Kode Sumber 3.51 memiliki fungsi untuk mengambil seluruh data rekap bonus yang terkait dengan guru berdasarkan NIP. Fungsi ini menerima NIP sebagai identifikasi guru yang dicari (Baris 2). Setelah menerima NIP, fungsi ini akan menyalurkan permintaan pencarian data rekap bonus kepada *RekapBonusService.getRekapByNip* (Baris 4). Hasil dari *query* yang dilakukan oleh *service* tersebut akan langsung dikembalikan oleh fungsi ini kepada pemanggil. NIP sebagai kunci untuk mengambil semua catatan bonus dari *database*.

```
1. export class GetRekapBonusByNipQuery {
2.   async execute(nip: string) {
3.     // Langsung mendelagasikan pencarian ke service layer
    }
```

```

4.     return await RekapBonusService.getRekapByNip(nip);
5.   }
6. }

```

Kode Sumber 3.51 *Code execute GetRekapBonusByNipQuery Rekap Bonus*

3. Fungsi *createRekap* dan *getRekapByNip* *RekapBonusService* (*rekap_bonus.service.ts*).

Pada Kode Sumber 3.52 berfungsi sebagai perantara antara lapisan *query* dan lapisan *repository*. Fungsi *createRekap* berfungsi untuk menerima data rekap bonus, termasuk NIP, dan meneruskannya langsung ke *RekapBonusRepository.create* untuk penyimpanan ke basis data (Baris 3-4). Sementara itu, fungsi *getRekapByNip* menerima NIP sebagai parameter dan meneruskan ke *RekapBonusRepository.findByNip* untuk mengambil data rekap bonus (Baris 8-9). Kedua fungsi ini memastikan bahwa semua permintaan terkait rekap bonus diarahkan dengan benar ke *repository* yang sesuai untuk operasi data yang sebenarnya.

```

1. export class RekapBonusService {
2.   // Mendelegasikan pembuatan rekap ke repository
3.   async createRekap(data: { /* ...parameter rekap bonus */ }) {
4.     return await RekapBonusRepository.create(data);
5.   }
6.
7.   // Mendelegasikan pencarian rekap ke repository
8.   async getRekapByNip(nip: string) {
9.     return await RekapBonusRepository.findByNip(nip);
10.  }
11. }

```

Kode Sumber 3.52 *Code createRekap dan getRekapByNip Rekap Bonus*

4. Fungsi *create* dan *findByNip* *RekapBonusRepository* (*rekap_bonus.repository.ts*)

Pada Kode Sumber 3.53 merupakan inti dari interaksi dengan basis data untuk entitas *RekapBonus*. Fungsi *create* bertanggung jawab untuk menyimpan data rekap bonus baru. Fungsi *create* menerima data yang mencakup NIP (Baris 3), sebelum melakukan operasi penyimpanan ke tabel *rekap_bonus*, fungsi *create* akan terlebih dahulu melakukan pencarian di tabel *teachers* untuk mendapatkan *ID* guru yang sesuai dengan NIP yang diberikan (Baris 4-5). Jika guru tidak ditemukan atau *ID*-nya tidak valid, maka akan menampilkan *error* (Baris 6-7). Setelah mendapatkan *teacher_id* yang valid, setelahnya data rekap bonus dibuat dengan menggunakan *teacher_id* tersebut (Baris 9-12). Pada, fungsi *findByNip* juga akan melakukan langkah yang sama. Ketika dipanggil dengan NIP (Baris 14), fungsi ini akan mencari *teacher_id* yang sama dari tabel *teachers* (Baris 15-16). Setelah *teacher_id* berhasil ditemukan, fungsi ini akan menggunakan *teacher_id* tersebut untuk mengambil semua catatan rekap bonus yang terkait dari basis data, diurutkan berdasarkan tanggal (Baris 21-25)

```

1. export class RekapBonusRepository {
2.   // Fungsi untuk membuat rekap bonus baru
3.   async create(data: { nip: string; /* ...data lainnya */ }) {
4.     // Cari guru berdasarkan NIP untuk mendapatkan ID (UUID)
5.     const teacher = await GuruModel.findOne({ where: { nip: data.nip } });
6.     if (!teacher || !teacher.id) {
7.       throw new Error(`Guru dengan NIP '${data.nip}' tidak ditemukan...`);
8.     // Buat data rekap bonus menggunakan teacher.id yang sudah ditemukan
9.     return await RekapBonus.create({
10.       teacher_id: teacher.id, // Menyimpan UUID, bukan NIP
11.       ...data,
12.     });
13.   // Fungsi untuk mencari semua rekap bonus berdasarkan NIP
14.   async findByNip(nip: string) {
15.     // Cari guru berdasarkan NIP untuk mendapatkan ID (UUID)
16.     const teacher = await GuruModel.findOne({ where: { nip } });
17.     if (!teacher || !teacher.id) {

```

```

18.     return []; // Kembalikan array kosong jika guru tidak ada
19.   }
20.
21.   // Cari semua rekап bonus menggunakan teacher.id yang sudah ditemukan
22.   return await RekapBonus.findAll({
23.     where: { teacher_id: teacher.id },
24.     order: [["tanggal", "ASC"]],
25.   });
26. }
27. 
```

Kode Sumber 3.53 *Code create dan findByNip Rekap Bonus*

5. Model RekapBonus (*rekap_bonus.entity.ts*)

Kode Sumber 3.54 mendefinisikan struktur data untuk entitas rekап bonus dalam basis data dengan seluruh atribut dan konfigurasi yang diperlukan. (Baris 1-15)

```

1. RekapBonus.init({
2.   id: {
3.     type: DataTypes.UUID,
4.     primaryKey: true,
5.     defaultValue: DataTypes.UUIDV4,
6.   },
7.   teacher_id: {
8.     type: DataTypes.UUID,
9.     allowNull: false,
10.    },
11.   // ... dan definisi untuk kolom tanggal, start_date, end_date,
uang_tambahan, keterangan
12.  },
13.  sequelize: dbConn,
14.  tableName: "rekap_bonus",
15.  // ...
16. ); 
```

Kode Sumber 3.54 *Code RekapBonus Model Entity*

6. Fungsi *create RekapBonusController* (*rekap_bonus.controller.ts*)

Kode Sumber 3.55 untuk menangani HTTP request pembuatan rekап bonus dengan validasi *input* dan *response handling*. Alur Kode Sumber 3.55 yaitu, mengekstrak data dari *request body* termasuk *start_date*, *end_date*, NIP, *uang_tambahan*, dan keterangan (Baris 4). Selanjutnya, sistem melakukan validasi untuk memastikan *field* wajib telah diisi (Baris 5-7). Lalu, sistem membuat *instance CreateRekapBonusCommand* (Baris 9) dan menjalankan *execute* dengan data yang diterima (Baris 10). Setelah itu, sistem mengembalikan *response* sukses dengan *status* 201 jika berhasil (Baris 12). Terakhir, sistem menampilkan *error* dengan *response* 500 dan detail *error* (Baris 13-15).

```

1. async create(req: Request, res: Response) {
2.   try {
3.     // Ekstrak & validasi data dari request body
4.     const { start_date, end_date, nip, ... } = req.body;
5.     if (!start_date || !end_date || !nip) {
6.       return res.status(400).json({ error: "..." });
7.     }
8.     // Buat dan jalankan command untuk memproses logika
9.     const createRekapBonusCommand = new CreateRekapBonusCommand();
10.    await createRekapBonusCommand.execute({ start_date, end_date, nip, ... });
11.    // Kirim respons sukses
12.    res.status(201).json({ message: "Rekap bonus berhasil dibuat" });
13.  } catch (error) {
14.    // Tangani error dan kirim respons 500
15.    res.status(500).json({ error: "Terjadi kesalahan...", ... }); 
```

Kode Sumber 3.55 *Code create RekapBonusController*

7. Fungsi *getByNip* *RekapBonusController* (*rekap_bonus.controller.ts*)

Kode Sumber 3.56 bertugas untuk menangani HTTP *request* pengambilan data rekap bonus berdasarkan NIP dan memiliki alur yaitu mengambil parameter NIP dari URL (Baris 4). Selanjutnya, sistem melakukan validasi untuk memastikan NIP tidak kosong (Baris 5-7). Lalu, sistem menjalankan *query* untuk mengambil data rekap bonus berdasarkan NIP (Baris 10). Setelah itu, sistem melakukan pengecekan data dan mengembalikan *response* 404 jika tidak ditemukan (Baris 13-15). Terakhir, sistem mengembalikan data dengan *response* 200 jika berhasil (Baris 18) atau *error* 500 jika terjadi kesalahan (Baris 20-23).

```
1. async getByNip(req: Request, res: Response) {
2.   try {
3.     // Validasi NIP dari parameter URL (jika kosong -> 400)
4.     const nip = req.params.nip?.trim();
5.     if (!nip) {
6.       return res.status(400).json({ error: "NIP tidak boleh kosong" });
7.     }
8.
9.     // Jalankan query untuk mengambil data rekap bonus
10.    const data = await RekapBonusQuery.GetRekapBonusByNipQuery.execute(nip);
11.
12.    // validasi hasil data (jika kosong -> 404)
13.    if (!data || data.length === 0) {
14.      return res.status(404).json({ message: "Data rekap bonus tidak
ditemukan" });
15.    }
16.
17.    // Respons sukses (200 OK)
18.    return res.status(200).json({ data, ... });
19.
20.  } catch (error) {
21.    // Respons error server (500)
22.    return res.status(500).json({ error: "Terjadi kesalahan..." });
23.  }
24. }
```

Kode Sumber 3.56 *Code getByNip RekapBonusController*

3.3.4.7 Modul Pengajuan Perubahan Gaji

Modul ini merupakan modul yang berisi mengenai pengajuan perubahan gaji yang diajukan guru kepada Bendahara atau Administrator dikarenakan adanya ketidaksesuaian gaji yang tertera dengan gaji aktualnya. Modul ini dibangun untuk memenuhi kebutuhan Kasus Penggunaan Lihat Log Aktivitas dan divisualisasikan pada sub-bab 3.3.2.8. Modul ini menangani *upload* dokumen bukti, proses persetujuan, dan manajemen status pengajuan. Fungsi tersebut terdiri dari,

1. Fungsi ajukan (*pengajuan_perubahan_gaji.controller.ts*)

Fungsi ajukan merupakan fungsi yang memiliki peran yaitu untuk memproses pengajuan perubahan gaji baru dari guru dengan *upload* dokumen bukti. Pada Kode Sumber 3.57 memiliki alur kerja yaitu menyiapkan basis data agar tersimpan dengan memulai transaksi (Baris 5). Selanjutnya, sistem mengambil informasi pengguna yaitu *ID*, *email* dan NIP dari data *login* yang tersimpan (Baris 8). Sistem akan cek apakah *file* sudah diunggah dengan benar (Baris 9), jika sudah maka sistem akan melanjutkan proses dan jika belum atau tidak menemukan *file* yang pengguna unggah atau tidak di unggah maka proses dihentikan dengan pesan *error* dan *transaksi* dibatalkan (Baris 12-15). Setelah itu, sistem akan memanggil *service* untuk memproses pengajuan dengan data dan *path file* yang ada (Baris 18-21). Terakhir, sistem mengkonfirmasi *transaksi* jika semua proses berhasil (Baris 24) dan mengembalikan *response*

sukses dengan *status* 201 (Baris 25), atau membatalkan *transaksi* dan mengirim *response error* 500 jika terjadi kesalahan (Baris 27-31).

```
1. static async ajukan(req: Request, res: Response) {
2.   let transaksi;
3.   try {
4.     // Memulai transaksi untuk menjaga konsistensi data
5.     transaksi = await dbConn.transaction();
6.
7.     // Mengambil info pengguna dan file yang diunggah dari request
8.     const { id, ... } = await getUserInfo(res.locals.id_user);
9.     const { fotoBukti, fotoGaji } = this.getUploadedFiles(req.files);
10.
11.    // Validasi keberadaan file, batalkan transaksi jika tidak lengkap
12.    if (!fotoBukti || !fotoGaji) {
13.      await transaksi.rollback();
14.      return res.status(400).json({ error: "Foto bukti dan foto gaji wajib
diunggah." });
15.    }
16.
17.    // Panggil service untuk memproses pengajuan dengan data yang ada
18.    await PengajuanPerubahanGajiService.ajukanPerubahan({
19.      user_id: id,
20.      // ... path file dan data lainnya
21.    });
22.
23.    // Konfirmasi transaksi jika semua proses berhasil
24.    await transaksi.commit();
25.    res.status(201).json({ message: "Pengajuan berhasil disimpan", ... });
26.
27.  } catch (error: any) {
28.    // Batalkan transaksi jika terjadi kesalahan di tengah proses
29.    if (transaksi) await transaksi.rollback();
30.    // ... kirim response error 500
31.  }
32.}
```

Kode Sumber 3.57 *Code ajukan*

2. Fungsi *getAll* (*pengajuan_perubahan_gaji.controller.ts*)

Pada Kode Sumber 3.58 berfungsi untuk melakukan pengambilan data pengajuan perubahan gaji berdasarkan *role* dan *ID users* dengan menerapkan sistem autorisasi dan memiliki alur kerja yaitu, dengan identifikasi pengguna yang *login* dan jabatannya (Baris 3). Sistem akan mengambil data dengan mengambil hak akses sesuai dengan pengguna tersebut. Kemudian, berdasarkan jabatan pengguna, sistem akan menampilkan data yang berbeda - jika pengguna adalah administrator, bendahara, atau kepala sekolah maka akan melihat semua pengajuan dari seluruh guru, namun jika pengguna adalah guru maka hanya akan melihat pengajuan miliknya sendiri (Baris 5-7). Terakhir, sistem akan mengirim *response* data pengajuan yang telah difilter (Baris 9) atau menangani *error* jika terjadi masalah (Baris 10-11).

```
1. static async getAll(req: Request, res: Response) {
2.   try {
3.     const { id, email, role } = await getUserInfo(res.locals.id_user);
4.     const result = await withRetry(() =>
5.       PengajuanPerubahanGajiService.getPengajuan(id, role)
6.     );
7.     res.json({ message: "Data pengajuan ditemukan", data: result });
8.   } catch (error: any) {
9.     console.error("Error in getAll:", error);
10. }
```

Kode Sumber 3.58 *Code getAll*

3. Fungsi *updateStatus* (*pengajuan_perubahan_gaji.controller.ts*)

Kode Sumber 3.59 berfungsi untuk memperbarui status pengajuan perubahan gaji menjadi *approved* atau *rejected* yang dilakukan oleh pengguna dengan *role* administrator,

bendahara, atau kepala sekolah. Alur yang terjadi adalah dengan mencatat aktivitas pembaruan status dan menyiapkan *transaksi* basis data (Baris 4). Selanjutnya, sistem mengambil *ID* pengajuan yang akan diperbarui dan status baru yang diberikan oleh aktor yang dapat melakukan perubahan status (Baris 5). Kemudian, sistem melakukan pemeriksaan bahwa status yang diberikan harus berupa “disetujui” atau “ditolak” (Baris 8-10), dan jika status adalah ‘ditolak’ maka wajib disertai alasan penolakan (Baris 13-15). Setelahnya, sistem mencatat siapa yang melakukan persetujuan dan memperbarui status pengajuan (Baris 18). Pada tahap akhir, jika proses berhasil maka perubahan disimpan secara permanen dengan *commit transaksi* (Baris 19) dan mengirim *response* sukses (Baris 20), namun jika gagal maka semua perubahan dibatalkan dengan *rollback transaksi* dan menampilkan pesan *error* (Baris 22-24).

```

1. static async updateStatus(req: Request, res: Response) {
2.   let transaction;
3.   try {
4.     transaction = await dbConn.transaction();
5.     const { status, rejection_reason } = req.body;
6.
7.     // Status harus 'approved' atau 'rejected'
8.     if (!['approved', 'rejected'].includes(status)) {
9.       // ... rollback dan kirim error 400
10.    }
11.
12.    // Alasan wajib ada jika status 'rejected'
13.    if (status === 'rejected' && !rejection_reason) {
14.      // ... rollback dan kirim error 400
15.    }
16.
17.    // Proses utama panggil service, commit, dan kirim respons sukses
18.    await PengajuanPerubahanGajiService.updateStatus(req.params.id, { ... });
19.    await transaction.commit();
20.    res.status(200).json({ message: "..." });
21.
22.  } catch (error) {
23.    // ... rollback dan kirim error 500
24.  }
25. }
```

Kode Sumber 3.59 *Code updateStatus*

4. Fungsi *updateStatus Repository* (*pengajuan_perubahan_gaji.repository.ts*)

Kode Sumber 3.60 bertanggung jawab untuk melakukan pembaruan status pengajuan pada *repository* dengan *timestamp*. Alur kerjanya yaitu dengan menerima data status baru, informasi *approver*, dan alasan penolakan jika ada (Baris 1-7). Selanjutnya, sistem menentukan kapan waktu persetujuan, jika status adalah ‘disetujui’ maka dicatat waktu saat ini, namun jika ‘ditolak’ maka waktu persetujuan tidak dicatat (Baris 10). Lalu, sistem melakukan pembaruan data pada basis data dengan informasi status, siapa yang menyetujui, kapan disetujui, dan alasan penolakan jika ada (Baris 12-18). Setelahnya sistem akan memastikan alasan penolakan hanya disimpan ketika status adalah “ditolak”, untuk status “disetujui” bagian ini akan kosong (Baris 17).

```

1. static async updateStatus(
2.   id: string,
3.   data: {
4.     status: 'approved' | 'rejected';
5.     approved_by: string;
6.     rejection_reason?: string;
7.   }
8. ) {
9.   const { status, approved_by, rejection_reason } = data;
10.  const approved_at = status === 'approved' ? new Date() : null;
11.
12.  return await PengajuanPerubahanGaji.update(
13.    {
```

```

14.      status,
15.      approved_by,
16.      approved_at,
17.      rejection_reason: status === 'rejected' ? rejection_reason : null
18.    },
19.    { where: { id } }
20.  );
21. }

```

Kode Sumber 3.60 *Code updateStatus Repository*

5. Fungsi *createPengajuan* (*pengajuan_perubahan_gaji.repository.ts*)

Pada Kode Sumber 3.61 berfungsi dengan melakukan penyimpanan data pengajuan perubahan gaji yang baru ke dalam basis data dengan struktur data yang telah ditentukan. Alur eksekusinya dengan dengan menerima data lengkap pengajuan yang terdiri dari *user_id* pengaju, lokasi *file* foto bukti, lokasi *file* foto gaji, keterangan pengajuan, dan tanggal pengajuan (Baris 1-7). Lalu, sistem akan menyimpan data tersebut di basis data dengan struktur tabel (Baris 9). Kemudian, sistem akan memberikan *ID* (*auto generate*) untuk data baru serta pencatatan waktu kapan diajukan. Secara bawaan sistem, maka status otomatis akan menjadi status “pending”. (Baris 9 pada model *PengajuanPerubahanGaji*).

```

1. static async createPengajuan(data: {
2.   user_id: string;
3.   foto_bukti_path: string;
4.   foto_gaji_path: string;
5.   keterangan: string;
6.   tanggal: string;
7. })
8. {
9.   return await PengajuanPerubahanGaji.create(data);
10. }

```

Kode Sumber 3.61 *Code createPengajuan*

6. Fungsi *ajukanPerubahan* (*pengajuan_perubahan_gaji.service.ts*)

Kode Sumber 3.62 berfungsi sebagai pengolah logika bisnis pengajuan perubahan gaji melalui *service layer* yang berfungsi sebagai pemisah antara *controller* dan *repository*. Alur kerjanya yaitu dengan menerima data pengajuan yang sudah divalidasi dari tahap sebelumnya (Baris 1-5). Selanjutnya, sistem meneruskan data tersebut ke bagian penyimpanan data tanpa melakukan perubahan (Baris 7). Kemudian, Kode Sumber 3.62 berperan sebagai penghubung antara bagian yang menangani *controller* dengan bagian yang menangani penyimpanan data. Pada tahap akhir, sistem akan *return* penyimpanan data yang berisi informasi pengajuan baru telah dibuat (Baris 7).

```

1. static async ajukanPerubahan(data: {
2.   user_id: string;
3.   foto_bukti_path: string;
4.   // ... parameter lainnya
5. })
6. {
7.   // Langsung mendelegasikan proses pembuatan data ke repository
8.   return await PengajuanPerubahanGajiRepository.createPengajuan(data);
}

```

Kode Sumber 3.62 *Code ajukanPerubahan Service*

7. Fungsi *getPengajuan* (*pengajuan_perubahan_gaji.service.ts*)

Kode Sumber 3.63 bertanggung jawab untuk pengambil data pengajuan berdasarkan *role* dan *user_id*, pengguna dengan menerapkan logika autorisasi. Alur kerjanya dengan membuat

role atau jabatan pengguna tersebut dibuat menjadi *lower case* atau huruf kecil semua agar tidak terpengaruh huruf besar atau kecil (Baris 2). Selanjutnya, sistem menentukan jabatan mana saja yang boleh melihat semua pengajuan, yaitu administrator, kepala sekolah, dan bendahara (Baris 3). Kemudian, sistem mengecek apakah jabatan pengguna saat ini termasuk dalam jabatan yang boleh melihat semua data atau tidak (Baris 4). Setelah itu, jika pengguna memiliki jabatan admin, bendahara dan kepala sekolah maka sistem akan mengambil semua data pengajuan dari seluruh guru (Baris 4), namun jika pengguna adalah guru maka hanya mengambil data pengajuan sendiri (Baris 4-5). Pada tahap akhir, sistem *return* data yang sesuai dengan hak akses pengguna berdasarkan jabatannya (Baris 4-5).

```

1. static async getPengajuan(user_id: string, role: string) {
2.   const roleLower = role.toLowerCase();
3.   const allowedSeeAll = ["admin", "kepala sekolah", "bendahara"];
4.   if (allowedSeeAll.includes(roleLower)) { return await
PengajuanPerubahanGajiRepository.getAll(); } else { return await
PengajuanPerubahanGajiRepository.getByEmail(user_id); }
5. }
6. }
```

Kode Sumber 3.63 *Code getPengajuan Service*

8. Model *PengajuanPerubahanGaji* (*pengajuan_perubahan_gaji.entity.ts*)

Kode Sumber 3.64 memiliki fungsi yaitu untuk mendefinisikan struktur data dan skema basis data untuk entitas pengajuan perubahan gaji beserta seluruh atribut dan *constraint* yang diperlukan. Alur kerjanya yaitu, dengan membuat struktur data yang menggambarkan bentuk tabel pengajuan perubahan gaji dalam *database* (Baris 1-25). Selanjutnya, sistem menentukan jenis data untuk setiap kolom seperti *ID* unik (Baris 2-6), *user_id* (Baris 7-10), lokasi *file* (*foto_bukti_path*, *foto_gaji_path* Baris 12), keterangan (Baris 12), tanggal (Baris 12), dan status persetujuan (Baris 13-17). Kemudian, sistem mengatur aturan untuk setiap kolom, misalnya kolom mana yang wajib diisi, mana yang boleh kosong, dan nilai *default* yang diberikan. Setelah itu, sistem menentukan bahwa status pengajuan hanya boleh berisi tiga nilai yaitu '*pending*', '*approved*', atau '*rejected*' dengan nilai awal adalah '*pending*' (Baris 14-16). Pada tahap akhir, sistem mengatur konfigurasi tambahan seperti nama tabel di basis data (Baris 23), pencatatan waktu pembuatan dan pembaruan otomatis. (Baris 24-25).

```

1. PengajuanPerubahanGaji.init({
2.   id: {
3.     type: DataTypes.UUID,
4.     primaryKey: true,
5.     defaultValue: DataTypes.UUIDV4,
6.   },
7.   user_id: {
8.     type: DataTypes.STRING,
9.     allowNull: false,
10. },
11.
12.   // ... definisi untuk kolom: foto_bukti_path, foto_gaji_path, keterangan,
tangga1
13.   status: {
14.     type: DataTypes.ENUM('pending', 'approved', 'rejected'),
15.     allowNull: false,
16.     defaultValue: 'pending',
17.   },
18.
19.   // ... definisi untuk kolom persetujuan: approved_by, approved_at,
rejection_reason
20.
21. },
22. {
  sequelize: dbConn,
```

```

23.   tableName: "pengajuan_perubahan_gaji",
24.   // ... opsi model lainnya
25. });

```

Kode Sumber 3.64 Code PengajuanPerubahanGaji Model Entity

9. Middleware uploadPengajuan (upload.middleware.ts)

Kode Sumber 3.65 memiliki tugas yaitu dengan menangani proses *upload file* foto bukti dan foto gaji dengan implementasi konfigurasi penyimpanan. Alur kerjanya yaitu dengan menentukan lokasi penyimpanan *file* berdasarkan NIP pengguna (Baris 4). Selanjutnya, sistem membuat *folder* khusus untuk setiap pengguna jika belum ada dan membuat nama *file* yang unik dengan menambahkan waktu *upload* untuk menghindari duplikasi (Baris 5). Kemudian, sistem melakukan pemeriksaan jenis *file* yang diunggah untuk memastikan hanya *file* gambar berformat JPG, JPEG, dan PNG yang diperbolehkan (Baris 9-10). Setelah itu, sistem menggabungkan aturan penyimpanan dan filter *file* menjadi satu konfigurasi *upload*.

```

1. export const uploadPengajuan = multer({
2.   // Konfigurasi 'storage' untuk menentukan lokasi & nama file dinamis
3.   storage: multer.diskStorage({
4.     destination: (req, file, cb) => { /* ... logika path folder ... */ },
5.     filename: (req, file, cb) => { /* ... logika nama file unik ... */ },
6.   }),
7.
8.   // Konfigurasi 'fileFilter' untuk validasi tipe file
9.   fileFilter: (req, file, cb) => {
10.    // ... logika untuk hanya memperbolehkan gambar (jpg, jpeg, png ),});

```

Kode Sumber 3.65 Code Upload Middleware

3.3.4.8 Modul Users

Modul Pengelolaan Pengguna adalah komponen integral dari sistem yang dikembangkan, berfungsi sebagai pusat kontrol untuk seluruh data pengguna. Implementasi modul ini pada sisi antarmuka (*frontend*) berinteraksi langsung dengan layanan *backend* yang telah dikembangkan pada sistem sebelumnya (Akbar, 2024). Modul ini dirancang untuk memenuhi kebutuhan kasus Tambah pengguna, Edit dan Hapus Pengguna oleh aktor Administrator. Berikut, akan diuraikan secara rinci struktur komponen, alur kerja, dan logika bisnis dari setiap fungsionalitas yang diimplementasikan pada modul ini.

1. Pengelolaan State

Kode Sumber 3.66, komponen mendeklarasikan semua variabel state yang diperlukan menggunakan *hook useState*. State ini berfungsi untuk mengelola data pada formulir, menampung daftar pengguna yang diambil dari API, dan mengontrol tampilan notifikasi yang muncul setelah sebuah aksi dilakukan.

```

1. // State untuk data formulir, daftar user, dan notifikasi
2. const [formData, setFormData] = useState({ /* ... nilai awal... */ });
3. const [usersList, setUsersList] = useState([]);
4. const [notification, setNotification] = useState(null);
5. // ... dan state lainnya

```

Kode Sumber 3.66 Code Pengelolaan State Tampilan Modul Users

2. Pengambilan Data Awal

Penggunaan *hook useEffect* pada Kode Sumber 3.67, komponen ini secara otomatis mengambil data awal (seperti daftar guru dan pengguna) dari API saat pertama kali dimuat. Data ini kemudian disimpan ke dalam *state usersList* untuk ditampilkan di antarmuka, memastikan bahwa data yang terlihat selalu mutakhir.

```

1. // Mengambil data awal saat komponen dimuat
2. useEffect(() => {
3.   // Panggil fungsi untuk mengambil data dari API
4.   fetchUsersAndTeachersAPI()
5.   .then(data => setUsersList(data))
6.   .catch(error => console.error(error));
7. }, []);

```

Kode Sumber 3.67 *Code Pengambilan Data Awal*

3. Fungsi *Handler* untuk Aksi Pengguna

Berbagai fungsi *handler* pada Kode Sumber 3.68 seperti *handleSubmit* (Baris 2) dan *handleDeleteUser* (Baris 9) dideklarasikan untuk menangani interaksi pengguna. Setiap fungsi ini bertanggung jawab untuk berkomunikasi dengan API untuk membuat, memperbarui, atau menghapus data. Fungsi ini juga menangani logika validasi, seperti mencegah penghapusan *user* yang masih memiliki keterkaitan data pada modul lain, misalnya modul Pengeluaran Rumah Tangga dari aplikasi versi 1.0 (Akbar, 2024).

```

// Fungsi untuk menangani pengiriman form user baru
1. const handleSubmit = async (e) => {
2.   e.preventDefault();
3.   // ... logika untuk mengirim 'formData' ke API
4.   // ... menampilkan notifikasi sukses atau error};
5. // Fungsi untuk menangani penghapusan user
6. const handleDeleteUser = async (userId) => {
7.   // ... logika memanggil API untuk hapus user
8.   // ... termasuk validasi keterkaitan data sebelum menghapus };

```

Kode Sumber 3.68 *Code Fungsi Handler untuk Aksi Pengguna*

4. Render Antarmuka Pengguna

Terakhir pada Kode Sumber 3.69, bagian *return* mendefinisikan struktur antarmuka pengguna (UI) menggunakan JSX. Bagian ini merender semua elemen visual, termasuk formulir tambah pengguna, daftar pengguna yang diambil dari *state*, dan notifikasi. Setiap elemen interaktif seperti tombol atau *dropdown* dihubungkan ke fungsi *handler* yang sesuai untuk memastikan fungsionalitas berjalan dengan benar.

```

1. return (
2.   <div>
3.     {/* Menampilkan notifikasi jika ada */}
4.     {notification && <NotificationComponent ... />}
5.
6.     {/* Formulir untuk menambah user baru */}
7.     <h3>Tambah User Baru</h3>
8.     <form onSubmit={handleSubmit}>
9.       {/* ... semua input field untuk nama, NIP, rekening, dll. ... */}
10.      </form>
11.      {/* Tabel untuk menampilkan semua user */}
12.      <h3>Daftar Semua User</h3>
13.      <table>
14.        {/* ... loop untuk menampilkan baris data user dari 'usersList' ... */}
15.        {/* Setiap baris memiliki opsi Update Role dan Tombol Hapus */}
16.      </table>
17.    </div>);

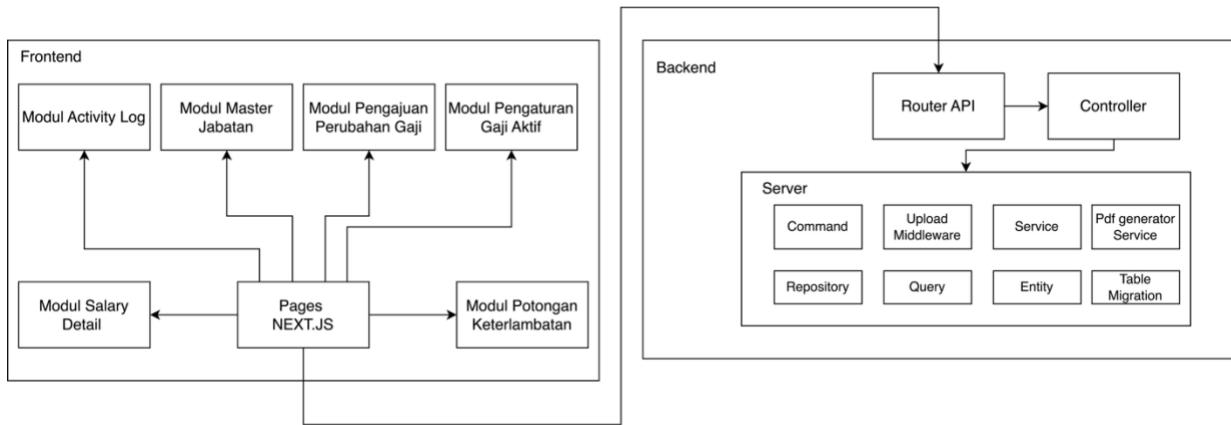
```

Kode Sumber 3.69 *Code Render Antarmuka Pengguna (JSX)*

3.3.5 Implementasi Integrasi *Frontend* dan *Backend*

Setelahnya implementasi pada *code* program logika serta tampilan antarmuka. Program tersebut akan di integrasikan agar kedua program tersebut dapat dijalankan serta dapat memudahkan pengguna untuk menggunakan Aplikasi KinderFin 2.0. Pada Gambar 3.49, terdapat hubungan dari *frontend* dan *backend* yang menghubungkan yaitu *Pages NEXT.JS* dan

Router API, untuk *frontend Pages NEXT.JS* melanjutkan ke Modul yang tersedia tergantung modul yang diakses. Untuk *backend Router API*, setelah diperintah oleh pengguna melalui *frontend*, akan dilanjutkan ke *Router API* setelahnya akan di teruskan ke *Controller* dan setelahnya di olah di *server*.



Gambar 3.49 Ilustrasi Integrasi *Frontend* dan *Backend* Aplikasi KinderFin 2.0

3.4 Pengujian

Setelah implementasi menjadi sebuah Aplikasi, setelahnya adalah pengujian untuk menguji aplikasi tersebut dapat berjalan serta dapat memenuhi kebutuhan pengguna. Dalam pengujian ini, dilakukan dua metode pengujian yaitu metode fungsional dan non fungsional. Pengujian fungsional akan berfokus pada menguji kebutuhan pengguna, dan pengujian non fungsional berfokus bagaimana aplikasi tersebut dapat berkomunikasi dengan sistem lain.

3.4.1 Pengujian Fungsionalitas

Pengujian Pengembangan Aplikasi KinderFin 2.0 Modul Penggajian atau Aplikasi KinderFin 2.0 ini menggunakan model pengujian dengan metode *blackbox testing*. Pengujian ini tidak melibatkan kode sebagai alat pengujian. Pengujian ini akan menjalankan aplikasi dengan menerapkan kesesuaian fitur dengan alur normal maupun alur alternatif. Pengujian ini akan mengambil referensi pengujian dari alur normal dan alternatif dari sub-bab 3.1.4. Metode ini memiliki parameter yaitu kesesuaian alur pengujian dengan alur normal atau alternatif. Serta terpenuhinya kondisi awal dan akhir yang didapatkan untuk setiap kasus penggunaan.

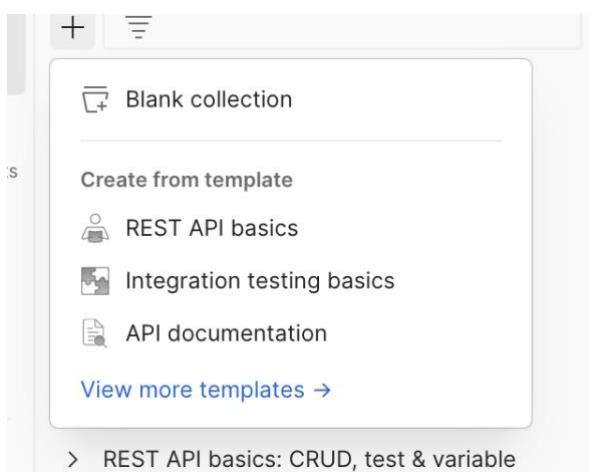
3.4.2 Pengujian Interoperabilitas

Pengujian Interoperabilitas yaitu pengujian yang melibatkan sistem lain untuk dihubungkan. Pengujian ini akan menguji Aplikasi KinderFin 2.0 untuk dihubungkan dengan sistem dari mesin *fingerprint* yaitu *server* dari *developer.fingerspot.io*. Pengujian ini akan menguji bagaimana proses pengambilan data dari *server* memastikan bahwa data tersebut dapat masuk ke dalam Aplikasi KinderFin 2.0 dan akan diproses oleh Aplikasi KinderFin.

Untuk penjelasan implementasi serta cara kerja dirincikan pada sub-bab 3.3.3.1 untuk proses pengambilan data, setelahnya akan diproses diuraikan lebih lanjut pada sub-bab 3.3.3.2. Selanjutnya, metodologi pengujian yang diterapkan beserta rincian mengenai perangkat lunak (aplikasi) pendukung yang digunakan dalam pelaksanaan pengujian ini akan dipaparkan secara detail pada Sub-bab 3.4.2. Pengujian ini akan menyimulasikan bahwa seolah olah sistem menerima data berformat JSON dari *server developer.fingerspot.io*.

Simulasi ini dibuat karena terdapat permasalahan pada mesin presensi yang diuraikan lebih lengkap pada LAMPIRAN A yang mengakibatkan kegagalan untuk mengambil data dari server *developer.fingerspot.io* ke mesin *fingerprint*. Untuk menguji seolah-olah Aplikasi KinderFin 2.0 ini dapat menerima data dari server *developer.fingerspot.io* dan diproses oleh Aplikasi KinderFin 2.0 pada Kode Sumber 3.70 dan dibuatkan API yang dapat menerima data asli jika mesin tidak mengalami masalah dengan code Kode Sumber 3.73 dengan format *body* Kode Sumber 3.72 atau melalui *frontend*.

Selanjutnya, terdapat tahapan pengujian Interoperabilitas yaitu, pertama membuka Aplikasi *Postman*. Setelah aplikasi siap digunakan, tambahkan *Create New Collection* seperti pada Gambar 3.50. Setelahnya pilih “*REST API basics*”. Setelah halaman baru terbuka, pilih menu “*Header*” dan masukan *key* berupa *Content-Type* serta *Value* berisi *application/json*. Masukan *URL* utama berupa *URL* yang didapatkan setelah proses instalasi, lalu masukan *URL* untuk *login* setelahnya didapatkan seperti Gambar 3.51.



Gambar 3.50 *Create New Collection* pada Aplikasi *Postman*

A screenshot of the Postman request configuration screen. At the top, it shows a 'POST' method and the URL 'https://backend-kinderfin.onrender.com/api/v1/users/login'. Below the URL, there are tabs for 'Params', 'Authorization', 'Headers (11)' (which is selected), 'Body', 'Scripts', 'Settings', and 'Cookies'. Under the 'Headers' tab, there is a table with two columns: 'Key' and 'Value'. A single row is present with the key 'Content-Type' and the value 'application/json'. There are also buttons for 'Bulk Edit' and 'Presets'.

Gambar 3.51 Proses untuk Login

Selanjutnya, pilih menu "Body" untuk memasukkan *email* dan *password*. Tekan tombol *Send* dengan metode "POST" untuk mendapatkan respons seperti pada Gambar 3.52, yang akan memberikan *access_token* dari sistem. Kemudian, kembali ke menu "Header", tambahkan *key* baru, lalu masukkan *Value* berupa "Bearer {*access_token*}" seperti yang ditunjukkan pada Gambar 3.53 menggunakan *access_token* tersebut dan aktivitas pengujian dapat dimulai.

The screenshot shows a POST request to <https://backend-kinderfin.onrender.com/api/v1/users/login>. The request body contains:

```

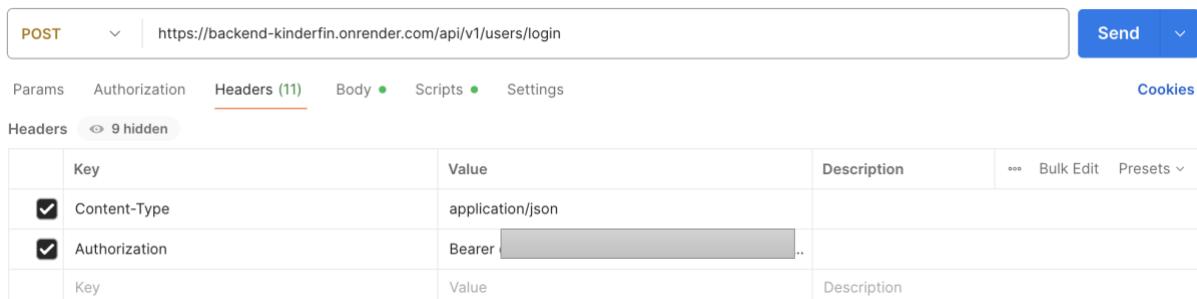
1 {
2   "email": "",
3   "password": ""
4 }
    
```

The response status is 200 OK, with a response time of 1m 24.90s, a size of 2.03 KB, and a session ID of eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9. The response body is:

```

1 {
2   "status": "success",
3   "message": "User berhasil login",
4   "data": {
5     "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
6
7     "role": "Admin",
8     "username": "Root User"
9   }
10 }
    
```

Gambar 3.52 Contoh Balasan Sistem Jika Pengguna berhasil *login*



Gambar 3.53 Proses *input access_token* setelah login

Setelahnya, dimulai dengan proses pengujian, dengan cara mengganti *URL* dengan *URL* untuk mengirimkan data yaitu pada Kode Sumber 3.70 mensimulasikan bahwa mengirimkan data setelah didapatkan dari *server developer.fingerspot.io*. seperti contoh pada Kode Sumber 3.70.

<https://backend.KinderFin.onrender.com/api/v1/fingerspot/process>

Kode Sumber 3.70 *URL* untuk simulasi mengirimkan data presensi

Untuk contoh balasan pesan berhasil serta data presensi dari *server developer.fingerspot.io* dapat dilihat pada Kode Sumber 3.71. Untuk pesan “*success*” berarti menandakan perintah yang pengguna kirimkan berhasil di terima serta diberikan hasil datanya. “*trans_id*” memiliki arti yaitu kode unik saat mengirimkan perintah serta dibalas untuk pesan tersebut. “*data*” berisi data pengguna yang telah melakukan presensi. Pesan “*pin*” yaitu berisi nomor pengguna yang terdaftar di mesin presensi yang akan dikoversi menjadi NIP. “*scan_date*” yaitu waktu jam serta hari dimana pengguna tersebut melakukan presensi. “*verify*” bernilai 1 yaitu berarti melakukan presensi menggunakan sidik jari. “*status_scan*” bernilai 0 yaitu presensi masuk dan bernilai 1 yaitu presensi pulang.

```

1. {
2.   "success": true,
3.   "trans_id": "1",
    
```

```

4.   "data": [
5.     {
6.       "pin": "0000123456",
7.       "scan_date": "2024-05-21 08:00:00",
8.       "verify": 1,
9.       "status_scan": 0
10.    },
11.    {
12.      "pin": "0000123456",
13.      "scan_date": "2024-05-30 17:00:00",
14.      "verify": 1,
15.      "status_scan": 1
16.    }
17.  ]
18. }

```

Kode Sumber 3.71 Contoh Balasan Pesan Berhasil serta Data Presensi dari *Server developer.fingerspot.io*

Pengujian kedua yaitu menggunakan *url* pada Kode Sumber 3.73 merupakan *URL* yang berfungsi seperti *URL* Kode Sumber 3.70 namun dengan tambahan fitur yaitu kemampuan untuk mengambil data yang telah dikirim secara otomatis dengan memasukkan rentang tanggal seperti Kode Sumber 3.72. Atau memasukkan dari tampilan antarmuka Halaman Antarmuka Gambar 3.31. Kode Sumber 3.74 merupakan *API* yang digunakan untuk mengambil data langsung dari *server developer.fingerspot.io* dan di implementasikan pada Kode Sumber 3.3. Untuk cara kerja Kode Sumber 3.73 dan Kode Sumber 3.70 dirincikan pada sub-bab 3.3.3.1 dan divisualisasikan pada Gambar 3.47.

```
{"trans_id":"1", "cloud_id":"xxxxx", "start_date":"2020-07-27", "end_date":"2020-07-25"}
```

Kode Sumber 3.72 *Request Body* Perintah Pengiriman Data Presensi

```
https://backend.KinderFin.onrender.com/api/v1/fingerspot/fetch
```

Kode Sumber 3.73 *URL Endpoint API* untuk Pengambilan, Pengolahan, dan Integrasi Data *Fingerspot*

```
https://developer.fingerspot.io/api/get\_attlog
```

Kode Sumber 3.74 *URL* pengambilan Data dari Sever *developer.fingerspot.io*

3.5 Instalasi

Setelah pengujian selesai, proses selanjutnya adalah instalasi. Instalasi ini akan menerapkan Aplikasi agar dapat diakses umum dan dapat berjalan *public*. Namun, dibutuhkan platform untuk melakukan instalasi baik tampilan antarmuka (*frontend*), sistem program (*backend*), dan basis data.

3.5.1 Arsitektur Instalasi

Implementasi Aplikasi KinderFin 2.0 menggunakan arsitektur *cloud-native* dengan pemisahan komponen *frontend*, *backend*, dan basis data. Arsitektur instalasi tersebut dipilih untuk memberikan skalabilitas, keandalan, dan kemudahan pemeliharaan yang optimal. Untuk instalasi *frontend* menggunakan platform “Vercel” sebagai hosting utama. Sedangkan untuk *backend* menggunakan platform Render sebagai hosting *server*. Basis data Aplikasi KinderFin 2.0 menggunakan “Neon” sebagai platform *PostgreSQL cloud*.

3.5.2 Instalasi *Backend*

Untuk instalasi *backend* atau program *server* atau bagian logika aplikasinya menggunakan platform “Render” sebagai *hosting server*. Render dipilih karena kemudahan konfigurasi, dukungan untuk aplikasi *Node.js/Express*. Pertama memanggil perintah instalasi dengan perintah pada Kode Sumber 3.75 Perintah Instalasi *Backend*. Setelahnya adalah proses perintah membangun yaitu pada Kode Sumber 3.76 Perintah membangun Program *Backend*. Lalu terakhir adalah menjalankan program yaitu pada Kode Sumber 3.77 Perintah menjalankan Program *Backend*. Namun sebelum itu, perlu diperhatikan dengan menyambungkan akun *github* dengan platform “Render” untuk *cloning* program yang akan di jalankan.

```
npm install
```

Kode Sumber 3.75 Perintah Instalasi *Backend*

```
npm run build
```

Kode Sumber 3.76 Perintah membangun Program *Backend*

```
npm run start
```

Kode Sumber 3.77 Perintah menjalankan Program *Backend*

3.5.3 Instalasi *Frontend*

Instalasi untuk tampilan antarmuka Aplikasi KinderFin 2.0 menggunakan platform “Vercel” karena dapat menyediakan instalasi otomatis. Kemudian ada fungsi untuk membangun konfigurasinya yaitu, pertama memanggil perintah instalasi dengan perintah pada Kode Sumber 3.78 Perintah Instalasi *Frontend*. Setelahnya adalah proses perintah membangun yaitu pada Kode Sumber 3.79 Perintah membangun Program. Lalu terakhir adalah menjalankan program yaitu pada Kode Sumber 3.80 Perintah menjalankan Program .Namun sebelum itu, perlu diperhatikan dengan menyambungkan akun *github* dengan platform “Vercel” untuk *cloning* program yang akan di jalankan. Spesifikasi untuk instalasi *frontend* yaitu *Package Manager* yaitu “*Yarn Berry*” dan *Framework React.js/Next.js*.

```
npm install -g yarn@berry && yarn && yarn build
```

Kode Sumber 3.78 Perintah Instalasi *Frontend*

```
yarn build
```

Kode Sumber 3.79 Perintah membangun Program *Frontend*

```
yarn start
```

Kode Sumber 3.80 Perintah menjalankan Program *Frontend*

3.5.4 Konfigurasi Basis Data dan *Environment*

Basis data Aplikasi KinderFin 2.0 menggunakan Neon sebagai platform PostgreSQL *cloud*. Neon dipilih karena kemampuannya dalam menyediakan *database* PostgreSQL yang *fully managed*. Untuk migrasi Basis data diperlukan komunikasi oleh *environment*. Pada Kode Sumber 3.81 merupakan variabel dari *backend* yang ada beberapa variabel untuk basis data.

```
SERVER_PORT=
APP_ENV=development
DB_HOST=
DB_PORT=
DB_USER=
DB_PASS=
DB_NAME=
JWT_SECRET_KEY=
JWT_ISSUER=
IMAGEKIT_PUBLIC_KEY=
```

```
IMAGEKIT_PRIVATE_KEY=  
IMAGEKIT_URL_ENDPOINT=
```

Kode Sumber 3.81 *Environment Variables Backend*

Konfigurasi environment variables *backend* terdiri dari beberapa kategori variabel yang memiliki fungsi spesifik dalam operasional aplikasi. *SERVER_PORT* berfungsi sebagai *port* yang digunakan *server backend*, sedangkan *APP_ENV* menentukan *environment* aplikasi yang dapat berupa *development* atau *production*. Untuk konfigurasi basis data, terdapat beberapa variabel yaitu *DB_HOST* yang merupakan host basis data *PostgreSQL* dari Neon, *DB_PORT* sebagai *port* basis data *PostgreSQL* dengan *default* 5432, *DB_USER* dan *DB_PASS* yang masing-masing berfungsi sebagai *username* dan *password* untuk akses basis data, serta *DB_NAME* yang menentukan nama *database* yang digunakan.

Keamanan aplikasi diatur melalui *JWT_SECRET_KEY* yang berfungsi sebagai *secret key* untuk autentikasi *JSON Web Token* dan *JWT_ISSUER* sebagai *issuer identifier* untuk *JWT* token. Untuk pengelolaan media dan unggah gambar, aplikasi menggunakan integrasi dengan *ImageKit* melalui tiga variabel yaitu *IMAGEKIT_PUBLIC_KEY* sebagai *public key*, *IMAGEKIT_PRIVATE_KEY* sebagai *private key*, dan *IMAGEKIT_URL_ENDPOINT* sebagai *URL endpoint ImageKit* untuk proses unggah gambar.

Konfigurasi *frontend* agar dapat menghubungkan atau mengambil *API* dari *backend* dibutuhkan konfigurasi *environment*. Konfigurasi tersebut hanya mengambil data *API* dari *backend* dengan melewati CORS yang sudah diizinkan oleh *backend*. Pada Kode Sumber 3.82 *Environment Variables Fronten* terdapat dua baris. Pertama adalah Alamat *API URL* dari *backend* setelah dilakukan instalasi dan baris kedua adalah versinya.

```
NEXT_PUBLIC_API_URL=  
NEXT_PUBLIC_API_VERSION=v1
```

Kode Sumber 3.82 *Environment Variables Frontent*

BAB 4 HASIL DAN PEMBAHASAN

Setelah melakukan pengujian menggunakan metode *blackbox testing* pada Aplikasi KinderFin 2.0 ini, pada 0ini akan dibahas mengenai hasil serta evaluasi dari pengujian tersebut. Uji coba akan dilakukan dengan mencocokan kesesuaian fungsional alur normal atau alternatif dengan alur pengujian serta hasil akhir pengujian dengan setiap kasus penggunaan yang telah tentukan pada sub-bab 3.1.4. Bukti hasil uji coba fungsional dirincikan dan ditampilkan pada LAMPIRAN D dan bukti pengisian kusioner ditampilkan pada LAMPIRAN E

4.1 Hasil Uji Coba Fungsional

Dalam menentukan hasil ujicoba pada pengembangan Aplikasi KinderFin 2.0 ini terdapat 3 *role* atau peran yang akan menjadi Guru, Bendahara, dan Administrator. Pada Tabel 4.1 akan dijelaskan peran mana saja yang akan menjalankan kasus penggunaan setiap peran serta menjadi tabel informasi pengguna setiap peran.

Tabel 4.1 Daftar Pengguna Ujicoba

No.	Nama Penguji	Peran	Referensi Nomor Kasus Penggunaan
1.	Pengguna Uji Coba 1	Administrator	UC-001 dan UC-002
2.	Pengguna Uji Coba 2	Bendahara	UC-003 sampai dengan UC-011
3.	Pengguna Uji Coba 3	Guru	UC-012 dan UC-013

4.1.1 Pengujian Oleh Administrator

Pengujian oleh administrator merupakan pengujian pertama sebelum pengujian oleh penguji lain dilakukan. Karena pada pengujian ini akan dilakukan pengujian dengan menambahkan pengguna serta edit dan hapus pengguna. Menambahkan pengguna merupakan objek utama dalam Aplikasi ini, karena sebagai pusat pengujian terhadap gaji.

4.1.1.1 Kasus Pengujian Menambah Pengguna

Pada kasus pengujian menambah pengguna. Administrator atau pengguna yang akan menguji dengan cara menambahkan pengguna baru di Aplikasi KinderFin 2.0. Kasus pengujian menambah pengguna. Pada Tabel 4.2 akan ditampilkan detail mengenai pengujian Kasus menambah pengguna. Tabel 4.3 untuk detail mengenai menambah pengguna namun terdapat NIP yang sudah tersedia di Sistem.

Tabel 4.2 Hasil Uji *Use Case* Kasus Pengujian Menambah Pengguna

No. Kasus Uji	TC-001
Referensi No. Kasus Penggunaan	UC-001
Deskripsi Skenario Kasus Uji	Administrator dapat menambah pengguna sesuai dengan peran dan jabatan dalam Aplikasi KinderFin 2.0
Langkah Pengujian	<ol style="list-style-type: none">Masuk halaman “Tambah dan Edit Role Pengguna”Memasukkan data di formulir “Tambah Pengguna”Administrator memasukkan data NIP (disesuaikan dengan Nomor di mesin <i>fingerprint</i>), Nama lengkap,

	Hak Akses atau role (jabatan), Pemilik Rekening, Nomor Rekening, email, dan password. 4. Tekan tombol simpan
Kondisi Awal	Aplikasi belum ada Pengguna
Hasil yang Diharapkan	Aplikasi sudah ada Pengguna
Hasil yang Didapatkan	Aplikasi sudah ada Pengguna
Status Pengujian	Berhasil

Tabel 4.3 Hasil Uji *Use Case* Kasus Pengujian Menambah Pengguna dengan Data Duplikat

No. Kasus Uji	TC-002
Referensi No. Kasus Penggunaan	UC-001
Deskripsi Skenario Kasus Uji	Administrator akan menambahkan pengguna namun terdapat NIP duplikat.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Masuk halaman “Tambah dan Edit Role Pengguna” 2. Memasukkan data di formulir “Tambah Pengguna” 3. Administrator memasukkan data NIP (sudah terdapat NIP yang terdaftar) Nama lengkap, Hak Akses atau role (jabatan), Pemilik Rekening, Nomor Rekening, email, dan password. 4. Tekan tombol simpan
Kondisi Awal	Saat mendaftarkan pengguna baru, NIP yang akan didaftarkan ternyata sudah ada dalam sistem.
Hasil yang Diharapkan	Sistem menolak dengan pesan “NIP Sudah Terdaftar” dan tidak dapat disimpan
Hasil yang Didapatkan	Sistem menolak dengan pesan “NIP Sudah Terdaftar” dan tidak dapat disimpan
Status Pengujian	Berhasil

4.1.1.2 Kasus Pengujian Edit dan Hapus Pengguna

Kasus Pengujian Edit dan Hapus Pengguna yaitu kasus pengujian untuk mengetahui hasil bagaimana Aplikasi KinderFin 2.0 dapat memenuhi kebutuhan pengguna yaitu dengan dapat melakukan pembaruan atau hapus pada setiap pengguna dengan hanya dapat diakses oleh Administrator. Tabel 4.4 akan dijelaskan ditampilkan detail mengenai pengujian Kasus Pengujian Edit dan Hapus Pengguna pada Tabel 4.5.

Tabel 4.4 Hasil Uji *Use Case* Kasus Pengujian Edit Pengguna

No. Kasus Uji	TC-003
Referensi No. Kasus Penggunaan	UC-002
Deskripsi Skenario Kasus Uji	Administrator mengelola hak akses agar setiap pengguna dapat menggunakan fitur sesuai perannya.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Administrator masuk ke halaman “Tambah dan Edit Role Pengguna” 2. Administrator memilih pengguna yang akan diubah hak aksesnya di formulir “Update Role User” 3. Administrator mengubah peran pengguna.

	4. Klik Simpan
Kondisi Awal	Pengguna sudah terdaftar dalam sistem dengan peran tertentu
Hasil yang Diharapkan	Hak akses pengguna berhasil diubah sesuai peran baru dan pengguna dapat mengakses fitur sesuai peran tersebut.
Hasil yang Didapatkan	Hak akses pengguna berhasil diubah sesuai peran baru dan pengguna dapat mengakses fitur sesuai peran tersebut.
Status Pengujian	Berhasil

Tabel 4.5 Hasil Uji *Use Case* Kasus Hapus Pengguna

No. Kasus Uji	TC-004
Referensi No. Kasus Penggunaan	UC-002
Deskripsi Skenario Kasus Uji	Administrator dapat menghapus Pengguna dari Aplikasi KinderFin
Langkah Pengujian	<ol style="list-style-type: none"> 1. Administrator masuk ke halaman “Tambah Edit Hapus Pengguna” 2. Administrator memilih pengguna yang akan di Hapus di Tabel “Daftar Semua” 3. Administrator memilih pengguna yang dihapus. 4. Administrator menekan ikon sampah pada pengguna tersebut 5. Administrator menekan tombol “Ya, Hapus”
Kondisi Awal	Pengguna yang akan di hapus masih terdapat di sistem
Hasil yang Diharapkan	Sistem tidak terdapat pengguna yang telah dihapus
Hasil yang Didapatkan	Sistem tidak terdapat pengguna yang telah dihapus
Status Pengujian	Berhasil

4.1.2 Pengujian Oleh Bendahara

Pengujian oleh Bendahara dilakukan setelah pengujian Administrator. Setelah adanya Pengguna di dalam Aplikasi KinderFin 2.0, Bendahara dapat melakukan pengujian mengenai nominal gaji serta potongan dan pengujian mengenai keterkaitan antara pengguna dan gaji.

4.1.2.1 Kasus Pengujian Aktifkan Label Gaji

Bendahara dapat melakukan pengujian mengenai nominal gaji serta potongan dan pengujian mengenai keterkaitan antara pengguna dan gaji. Kasus Pengujian Aktifkan Label Gaji adalah pengujian untuk menentukan nama apakah akan muncul di modul lain dengan nama label tersebut untuk menggantikan nama gaji1-10 serta gajipokok1-10 dan pengujian ini akan mendapatkan hasil jika berhasil adalah nama gaji yang diaktifkan baik gaji (harian) maupun gaji pokok yang ditentukan akan muncul di Modul Potongan Gaji, Master Jabatan, dan Rekap Gaji Pegawai. Pada Tabel 4.6 akan dijelaskan rincian hasil pengujian Kasus Pengujian Aktifkan Label Gaji. Pada Tabel 4.7 akan dijelaskan rincian hasil pengujian Kasus Pengujian Aktifkan Label Gaji dengan nama label sama dengan nama tabel lainnya.

Tabel 4.6 Hasil Uji *Use Case* Kasus Pengujian Aktifkan Label Gaji

No. Kasus Uji	TC-005
Referensi No. Kasus Penggunaan	UC-003
Deskripsi Skenario Kasus Uji	Bendahara dapat mengaktifkan jenis-jenis gaji yang akan digunakan dalam proses penggajian dengan memberikan label sesuai kebutuhan sedang berjalan agar proses penggajian dapat dilakukan.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Pengaturan Gaji Aktif” pada modul penggajian. 2. Bendahara menambahkan label baru atau mengubah atau menghapus label yang ada sesuai kebutuhan. 3. Bendahara menyimpan perubahan pengaturan jenis gaji.
Kondisi Awal	Belum ada jenis gaji yang terlabel dan aktif dalam sistem. Atau label gaji menggunakan nama lama. Label gaji belum terlihat atau menggunakan nama lama pada Modul Potongan Gaji, Master Jabatan, dan Rekap Gaji Pegawai
Hasil yang Diharapkan	Jenis-jenis gaji sudah terlabel dan aktif sehingga proses penggajian dapat dilakukan serta terlihat pada Modul Potongan Gaji, Master Jabatan, dan Rekap Gaji Pegawai
Hasil yang Didapatkan	Jenis-jenis gaji sudah terlabel dan aktif sehingga proses penggajian dapat dilakukan serta terlihat pada Modul Potongan Gaji, Master Jabatan, dan Rekap Gaji Pegawai
Status Pengujian	Berhasil

Tabel 4.7 Hasil Uji *Use Case* Kasus Pengujian Aktifkan Label Gaji dengan Nama Tabel Duplikat

No. Kasus Uji	TC-006
Referensi No. Kasus Penggunaan	UC-003
Deskripsi Skenario Kasus Uji	Bendahara dapat mengaktifkan jenis-jenis gaji yang akan digunakan dalam proses penggajian dengan memberikan label sesuai kebutuhan sedang berjalan agar proses penggajian dapat dilakukan namun terdapat nama label yang sudah ada.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Pengaturan Gaji Aktif” pada modul penggajian. 2. Bendahara menambahkan label baru atau mengubah nama label yang ada sesuai kebutuhan namun terdapat nama label yang sudah ada. 3. Bendahara menyimpan perubahan pengaturan jenis gaji.
Kondisi Awal	Bendahara akan menambahkan nama label baru atau mengubah nama label yang sudah ada dengan nama label yang sama dengan nama label lain

Hasil yang Diharapkan	Sistem akan menolak dengan menampilkan pesan “Label “{nama label}” sudah digunakan. Harap gunakan nama label yang lain.”
Hasil yang Didapatkan	Sistem akan menolak dengan menampilkan pesan “Label “{nama label}” sudah digunakan. Harap gunakan nama label yang lain.”
Status Pengujian	Berhasil

4.1.2.2 Kasus Pengujian Tambah atau Ubah Gaji Harian dan Pokok

Kasus Pengujian Tambah atau Ubah Gaji Harian dan Pokok merupakan pengujian mengubah nominal gaji serta menambah jabatan serta nominal gaji jabatan tersebut. Kasus ini yang akan mempengaruhi nilai pada hasil penggajian pada setiap pengguna. Pada Tabel 4.8 akan dijelaskan rincian hasil pengujian tambah gaji harian, untuk Tabel 4.9 untuk rincian hasil pengujian perbarui gaji harian. Pada Tabel 4.10 untuk rincian hasil pengujian hapus gaji harian, dan Tabel 4.11 akan dijelaskan rincian hasil pengujian tambah gaji harian namun dengan nama jabatan duplikat atau nama jabatan tersebut sudah tersedia dalam sistem.

Tabel 4.8 Hasil Uji *Use Case* Kasus Pengujian Tambah Gaji Harian dan Pokok

No. Kasus Uji	TC-007
Referensi No. Kasus Penggunaan	UC-004
Deskripsi Skenario Kasus Uji	Bendahara akan menambahkan nama Jabatan baru dengan Nominal Gaji
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Master Jabatan”. 2. Bendahara memilih Tombol “Tambah Jabatan Baru” untuk menambah data jabatan baru. 3. Bendahara mengisi data jabatan dan nominal gaji. 4. Bendahara menekan tombol “Simpan”.
Kondisi Awal	Belum ada Jabatan beserta detail nominal komponen gaji jabatan tersebut
Hasil yang Diharapkan	Data jabatan beserta komponen gaji berhasil disimpan dan muncul pada daftar.
Hasil yang Didapatkan	Data jabatan beserta komponen gaji berhasil disimpan dan muncul pada daftar.
Status Pengujian	Berhasil

Tabel 4.9 Hasil Uji *Use Case* Kasus Pengujian Perbarui Gaji Harian dan Pokok

No. Kasus Uji	TC-008
Referensi No. Kasus Penggunaan	UC-004
Deskripsi Skenario Kasus Uji	Bendahara mengelola potongan-potongan gaji yang berlaku untuk setiap jabatan, untuk nantinya diolah otomatis dengan gaji harian oleh aplikasi.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Master Jabatan”. 2. Bendahara memilih Ikon “Edit” pada jabatan yang ingin diperbarui.

	<ol style="list-style-type: none"> 3. Bendahara memperbarui data jabatan dan nominal gaji yang dipilih. 4. Bendahara menekan tombol “Simpan”.
Kondisi Awal	Terdapat jabatan yang telah memiliki komponen gaji yang tersimpan namun belum diperbarui
Hasil yang Diharapkan	Data jabatan dan komponen gaji yang diperbarui berhasil tersimpan dan ditampilkan dengan benar.
Hasil yang Didapatkan	Data jabatan dan komponen gaji yang diperbarui berhasil tersimpan dan ditampilkan dengan benar.
Status Pengujian	Berhasil

Tabel 4.10 Hasil Uji Use Case Kasus Pengujian Hapus Gaji Harian dan Pokok

No. Kasus Uji	TC-009
Referensi No. Kasus Penggunaan	UC-004
Deskripsi Skenario Kasus Uji	Bendahara mengelola potongan-potongan gaji yang berlaku untuk setiap jabatan, untuk nantinya diolah otomatis dengan gaji harian oleh aplikasi.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Master Jabatan”. 2. Ikon “Hapus” Untuk hapus komponen gaji dan jabatan yang dipilih 3. Bendahara menekan ikon hapus di Jabatan yang dipilih 4. Bendahara Menekan Tombol "Ya Hapus"
Kondisi Awal	Data jabatan tersedia dan siap untuk dihapus oleh bendahara.
Hasil yang Diharapkan	Data jabatan dan komponen gaji yang dihapus tidak lagi muncul pada daftar jabatan.
Hasil yang Didapatkan	Data jabatan dan komponen gaji yang dihapus tidak lagi muncul pada daftar jabatan.
Status Pengujian	Berhasil

Tabel 4.11 Hasil Uji Use Case Kasus Pengujian Tambah Gaji Harian dan Pokok dengan Nama Jabatan sama dengan yang Tersedia di Sistem

No. Kasus Uji	TC-010
Referensi No. Kasus Penggunaan	UC-004
Deskripsi Skenario Kasus Uji	Bendahara akan menambahkan nama Jabatan baru dengan Nominal Gaji, namun nama Jabatan sudah tersedia di Sistem (Duplikat).
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Master Jabatan”. 2. Bendahara memilih Tombol “Tambah Jabatan Baru” untuk menambah data jabatan baru. 3. Bendahara mengisi data jabatan yang sama dengan di sistem dan nominal gaji. 4. Bendahara menekan tombol “Simpan”.

Kondisi Awal	Bendahara akan menambahkan Jabatan baru dan memasukkan nominal gaji jabatan tersebut, namun nama Jabatan tersebut sudah tersedia di sistem
Hasil yang Diharapkan	Sistem akan menolak dan menampilkan pesan “Gagal menambah jabatan”
Hasil yang Didapatkan	Sistem akan menolak dan menampilkan pesan “Gagal menambah jabatan”
Status Pengujian	Berhasil

4.1.2.3 Kasus Pengujian Pengaturan Potongan Gaji Harian

Kasus pengujian Pengaturan Potongan Gaji Harian menguji bagaimana kesesuaian pengguna memberikan potongan ke setiap jabatan dengan komponen tertentu yang dipotong. Pengujian ini akan mendaftarkan apakah potongan tersebut muncul dan tersimpan di basis data setelah ditambahkan. Pada Tabel 4.12 akan dijelaskan rincian hasil pengujian Tambah Potongan Gaji Harian, untuk Tabel 4.13 akan dijelaskan rincian hasil pengujian Tambah Potongan Gaji Harian namun terdapat duplikasi detail potongan dengan potongan lain dalam sistem, serta Tabel 4.14 akan dijelaskan rincian hasil pengujian Hapus Potongan Gaji Harian.

Tabel 4.12 Hasil Uji Use Case Kasus Pengujian Tambah Potongan Gaji Harian

No. Kasus Uji	TC-011
Referensi No. Kasus Penggunaan	UC-005
Deskripsi Skenario Kasus Uji	Bendahara mengelola potongan-potongan gaji yang berlaku untuk setiap jabatan, untuk nantinya diolah otomatis dengan gaji harian oleh aplikasi.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Data Potongan Gaji” 2. Bendahara memilih aksi Menambah potongan gaji baru dan ke halaman Tambah Potongan Gaji 3. Bendahara mengisi formulir potongan gaji 4. Bendahara menekan tombol “Simpan”
Kondisi Awal	Sistem belum memiliki beberapa data potongan gaji yang tersimpan
Hasil yang Diharapkan	Potongan gaji baru berhasil ditambahkan dan muncul dalam daftar potongan gaji.
Hasil yang Didapatkan	Potongan gaji baru berhasil ditambahkan dan muncul dalam daftar potongan gaji.
Status Pengujian	Berhasil

Tabel 4.13 Hasil Uji Use Case Kasus Pengujian Tambah Potongan Gaji Harian dengan detail Potongan Sudah Tersedia dalam Sistem (Duplikat)

No. Kasus Uji	TC-012
Referensi No. Kasus Penggunaan	UC-005
Deskripsi Skenario Kasus Uji	Bendahara mengelola potongan-potongan gaji yang berlaku untuk setiap jabatan, untuk nantinya diolah otomatis dengan gaji harian oleh aplikasi. Namun, memasukkan data yang

	sama dengan potongan sebelumnya dari detail Jabatan, Gaji yang dipotong, persen potong, batas menit, dan Jenis Pelanggaran
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Data Potongan Gaji” 2. Bendahara memilih aksi Menambah potongan gaji baru dan ke halaman Tambah Potongan Gaji 3. Bendahara mengisi formulir potongan gaji 4. Bendahara menekan tombol “Simpan”
Kondisi Awal	Sistem sudah memiliki beberapa data potongan gaji. Bendahara menambahkan detail potongan gaji sama dengan detail potongan gaji yang sudah ada
Hasil yang Diharapkan	Sistem menolak dan menampilkan pesan “Gagal menambahkan potongan”.
Hasil yang Didapatkan	Sistem menolak dan menampilkan pesan “Gagal menambahkan potongan”.
Status Pengujian	Berhasil

Tabel 4.14 Hasil Uji Use Case Kasus Pengujian Hapus Potongan Gaji Harian

No. Kasus Uji	TC-013
Referensi No. Kasus Penggunaan	UC-005
Deskripsi Skenario Kasus Uji	Bendahara mengelola potongan-potongan gaji yang berlaku untuk setiap jabatan, untuk nantinya diolah otomatis dengan gaji harian oleh aplikasi.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Data Potongan Gaji” 2. Bendahara memilih aksi Menghapus potongan gaji. 3. Bendahara memilih potongan mana yang akan dihapus. 4. Bendahara menekan tombol “Ya, Hapus”
Kondisi Awal	Data potongan gaji yang akan dihapus sudah terdaftar dalam sistem.
Hasil yang Diharapkan	Potongan gaji yang dipilih berhasil dihapus dan tidak lagi muncul dalam daftar potongan gaji.
Hasil yang Didapatkan	Potongan gaji yang dipilih berhasil dihapus dan tidak lagi muncul dalam daftar potongan gaji.
Status Pengujian	Berhasil

4.1.2.4 Kasus Pengujian Tambah Data Presensi

Kasus Pengujian Tambah Data Presensi pengujian untuk menambahkan data presensi baik melewati *file* yang diunggah Bendahara maupun masukan secara manual oleh Bendahara. Kasus ini menguji bagaimana *file* akan masuk ke dalam basis data serta akan dicek kesesuaian dengan potongan serta komponen jabatan. Tabel 4.15 menjelaskan rincian hasil dari pengujian Tambah Data Presensi dengan Mengambil Langsung Dari *Server developer.fingerspot.io*. Pada Tabel 4.16 dijelaskan rincian hasil pengujian Tambah Data Presensi dengan Unggah File. Untuk Tabel 4.17 dijelaskan rincian hasil pengujian Tambah Data Presensi dengan Unggah *File Terproses* Sebelumnya. Pada Tabel 4.18 dijelaskan rincian hasil pengujian Tambah Data Presensi dengan *Input Manual*, dan Tabel 4.19 dijelaskan rincian hasil pengujian Tambah Data Presensi dengan *Input Manual* dengan Tanggal Sama. Pengujian Terakhir pada Tabel 4.20 akan

dijelaskan hasil pengujian jika *file* yang diproses tidak sesuai dengan format yang diberikan oleh mesin *fingerprint*.

Tabel 4.15 Hasil Uji *Use Case* Kasus Pengujian Tambah Data Presensi dengan Mengambil Langsung Dari *Server developer.fingerspot.io*

No. Kasus Uji	TC-014
Referensi No. Kasus Penggunaan	UC-006
Deskripsi Skenario Kasus Uji	Bendahara mengambil data langsung dari mesin <i>fingerprint</i> melalui <i>server developer.fingerspot.io</i> di tampilan antarmuka.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Rekap Gaji Pegawai” 2. Bendahara mengisi rentang tanggal yang ingin di ambil datanya pada formulir “Sinkronisasi Data Absensi Fingerspot” 3. Bendahara menekan tombol “Tarik & Proses Data Absensi Fingerspot”
Kondisi Awal	Sistem telah memiliki data pegawai yang aktif. Detail gaji dari mesin <i>fingerprint</i> belum tersedia di sistem
Hasil yang Diharapkan	Detail gaji dari mesin <i>fingerprint</i> sudah tersedia di sistem dan data telah diproses dalam sistem lengkap dengan Potongan Gaji dan detail komponen gaji dari Master Jabatan.
Hasil yang Didapatkan	Menampilkan pesan “Respons dari API Fingerspot tidak valid”
Status Pengujian	Gagal

Tabel 4.16 Hasil Uji *Use Case* Kasus Pengujian Tambah Data Presensi dengan Unggah *File*

No. Kasus Uji	TC-015
Referensi No. Kasus Penggunaan	UC-006
Deskripsi Skenario Kasus Uji	Bendahara mengunggah data kehadiran guru yang diperoleh dari mesin <i>fingerprint</i> .
Langkah Pengujian	<ol style="list-style-type: none"> 4. Bendahara membuka halaman “Rekap Gaji Pegawai” 5. Bendahara menekan tombol “Upload File Fingerprint” 6. Bendahara memilih <i>file</i> yang diunggah
Kondisi Awal	Sistem telah memiliki data pegawai yang aktif. <i>File fingerprint</i> telah tersedia untuk diunggah.
Hasil yang Diharapkan	<i>File fingerprint</i> berhasil diunggah dan data telah diproses dalam sistem lengkap dengan Potongan Gaji dan detail komponen gaji dari Master Jabatan.
Hasil yang Didapatkan	<i>File fingerprint</i> berhasil diunggah dan data telah diproses dalam sistem lengkap dengan Potongan Gaji dan detail komponen gaji dari Master Jabatan.
Status Pengujian	Berhasil

Tabel 4.17 Hasil Uji Use Case Kasus Pengujian Tambah Data Presensi dengan Unggah *File* Terproses Sebelumnya

No. Kasus Uji	TC-016
Referensi No. Kasus Penggunaan	UC-006
Deskripsi Skenario Kasus Uji	Bendahara mengunggah data kehadiran guru yang diperoleh dari mesin <i>fingerprint</i> . Namun isi <i>file</i> sama dengan sebelumnya
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Rekap Gaji Pegawai” 2. Bendahara menekan tombol “Upload File Fingerprint” 3. Bendahara memilih <i>file</i> yang diunggah
Kondisi Awal	Sistem telah memiliki data pegawai yang aktif. <i>File fingerprint</i> telah tersedia untuk diunggah. Namun sudah diproses sebelumnya
Hasil yang Diharapkan	<i>File fingerprint</i> berhasil diunggah dan data telah diproses, lalu setelah di periksa disetiap pengguna bahwa tidak ada duplikasi data tanggal
Hasil yang Didapatkan	<i>File fingerprint</i> berhasil diunggah dan data telah diproses, lalu setelah di periksa disetiap pengguna bahwa tidak ada duplikasi data tanggal
Status Pengujian	Berhasil

Tabel 4.18 Hasil Uji Use Case Kasus Pengujian Tambah Data Presensi dengan *Input Manual*

No. Kasus Uji	TC-017
Referensi No. Kasus Penggunaan	UC-006
Deskripsi Skenario Kasus Uji	Bendahara mengunggah data kehadiran guru dengan memasukkan manual
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Rekap Gaji Pegawai” 2. Bendahara menekan tombol “Lihat Gaji” pada pengguna yang dituju. 3. Bendahara melihat dan menekan tombol “Input Kehadiran Manual” 4. Bendahara mengisi formulir “Input Kehadiran Manual”
Kondisi Awal	Bendahara memiliki data kehadiran tertentu yang dicatat secara manual.
Hasil yang Diharapkan	Data Kehadiran Manual berhasil disimpan dan telah diproses serta tampil dalam sistem.
Hasil yang Didapatkan	Data Kehadiran Manual berhasil disimpan dan telah diproses serta tampil dalam sistem.
Status Pengujian	Berhasil

Tabel 4.19 Hasil Uji Use Case Kasus Pengujian Tambah Data Presensi dengan *Input Manual* dengan Tanggal Sama

No. Kasus Uji	TC-018
Referensi No. Kasus Penggunaan	UC-006
Deskripsi Skenario Kasus Uji	Bendahara mengunggah data kehadiran guru dengan memasukkan manual dengan tanggal yang sama
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Rekap Gaji Pegawai” 2. Bendahara menekan tombol “Lihat Gaji” pada pengguna yang dituju. 3. Bendahara melihat dan menekan tombol “<i>Input Kehadiran Manual</i>” 4. Bendahara mengisi formulir “<i>Input Kehadiran Manual</i>” dengan Tanggal sama yang sudah tersedia di sistem.
Kondisi Awal	Bendahara memiliki data kehadiran tertentu yang dicatat secara manual dan di <i>input</i> duplikat atau lebih dari sekali
Hasil yang Diharapkan	Sistem tidak menyimpan data tersebut.
Hasil yang Didapatkan	Sistem tidak menyimpan data tersebut.
Status Pengujian	Berhasil

Tabel 4.20 Hasil Uji Use Case Kasus Pengujian Tambah Data Presensi dengan Unggah *File* dengan Format Tidak Sesuai dengan Mesin *Fingerprint*

No. Kasus Uji	TC-019
Referensi No. Kasus Penggunaan	UC-006
Deskripsi Skenario Kasus Uji	Bendahara mengunggah data kehadiran guru namun tidak sesuai format mesin <i>fingerprint</i> .
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Rekap Gaji Pegawai” 2. Bendahara menekan tombol “<i>Upload File Fingerprint</i>” 3. Bendahara memilih <i>file</i> yang diunggah
Kondisi Awal	Sistem telah memiliki data pegawai yang aktif. <i>File fingerprint</i> telah tersedia untuk diunggah namun tidak sesuai format mesin <i>fingerprint</i> .
Hasil yang Diharapkan	Sistem menolak untuk memproses <i>file</i> serta menampilkan pesan “Error Gagal Memproses File. Gunakan <i>File</i> sesuai Format <i>Fingerspot</i> ”
Hasil yang Didapatkan	Sistem menolak untuk memproses <i>file</i> serta menampilkan pesan “Error Gagal Memproses File. Gunakan <i>File</i> sesuai Format <i>Fingerspot</i> ”
Status Pengujian	Berhasil

4.1.2.5 Kasus Pengujian Validasi Gaji

Kasus Pengujian Validasi Gaji merupakan kasus pengujian yang membandingkan gaji yang tertera dalam sistem dengan kondisi aktual. Pengujian ini dilakukan sebelum mencetak

hasil serta memberikan kepada guru sebagai penerima hasilnya. Pada Tabel 4.21 akan dijelaskan hasil pengujian Validasi Gaji, dan Tabel 4.22 akan dijelaskan hasil pengujian Validasi Gaji untuk Menghapus Detail Gaji.

Tabel 4.21 Hasil Uji Use Case Kasus Pengujian Validasi Gaji.

No. Kasus Uji	TC-020
Referensi No. Kasus Penggunaan	UC-007
Deskripsi Skenario Kasus Uji	Bendahara memeriksa dan memastikan data gaji sudah benar sebelum proses pembayaran dilakukan.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Rekap Gaji Pegawai” 2. Bendahara menekan tombol “Lihat Gaji” Pada kolom “Aksi” pada pengguna yang akan dituju. 3. Bendahara melihat dan memvalidasi seluruh detail gaji dan bonus pengguna tersebut.
Kondisi Awal	Data gaji telah dihitung dan tersedia dalam sistem, namun belum divalidasi oleh Bendahara
Hasil yang Diharapkan	Data gaji telah divalidasi oleh Bendahara; jika sesuai, data siap untuk proses pembayaran; jika tidak sesuai, pengajuan revisi gaji dilakukan.
Hasil yang Didapatkan	<i>File fingerprint</i> berhasil diunggah dan data telah diproses dalam sistem lengkap dengan Potongan Gaji dan detail komponen gaji dari Master Jabatan.
Status Pengujian	Berhasil

Tabel 4.22 Hasil Uji Use Case Kasus Pengujian Validasi Gaji untuk Menghapus Detail Gaji

No. Kasus Uji	TC-021
Referensi No. Kasus Penggunaan	UC-007
Deskripsi Skenario Kasus Uji	Bendahara memeriksa dan terdapat gaji yang tidak sesuai dan akan dihapus
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Rekap Gaji Pegawai” 2. Bendahara menekan tombol “Lihat Gaji” Pada kolom “Aksi” pada pengguna yang akan dituju. 3. Bendahara melihat dan memvalidasi seluruh detail gaji dan bonus pengguna tersebut dan terdapat gaji tidak sesuai. 4. Bendahara menghapus detail gaji tanggal tersebut.
Kondisi Awal	Data gaji telah dihitung dan tersedia dalam sistem, namun sudah divalidasi oleh Bendahara namun terdapat kesalahan detail gaji dan akan dihapus
Hasil yang Diharapkan	Detail gaji pada tanggal tersebut terhapus dan tidak lagi muncul dalam sistem.
Hasil yang Didapatkan	Detail gaji pada tanggal tersebut terhapus dan tidak lagi muncul dalam sistem.
Status Pengujian	Berhasil

4.1.2.6 Kasus Pengujian Cetak Gaji

Kasus Pengujian Cetak Gaji akan ada pengujian yang memiliki pilihan lanjutan pada Kasus Pengujian Tambah Bonus . Pengujian ini akan menguji sistem akan menampilkan dan menghasilkan sebuah *file pdf* yang berisi mengenai presensi dan komponen gaji lainnya dengan rentang waktu yang ditentukan oleh Bendahara. Pada Tabel 4.23 dijelaskan rincian hasil pengujian Kasus Pengujian Validasi Gaji.

Tabel 4.23 Hasil Uji *Use Case* Kasus Pengujian Cetak Gaji.

No. Kasus Uji	TC-022
Referensi No. Kasus Penggunaan	UC-008
Deskripsi Skenario Kasus Uji	Bendahara menyediakan informasi terkait pembayaran dan detail gaji kepada guru.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Rekap Gaji Pegawai”. 2. Bendahara menekan tombol “Lihat Gaji” Pada kolom “Aksi” pada pengguna yang akan dituju. 3. Bendahara mengisi formulir “Proses Gaji Bulanan” berupa tanggal awal dan akhir yang akan diproses. 4. Bendahara menekan tombol “Proses” 5. Bendahara menekan tombol “Unduh PDF Tanpa Bonus”
Kondisi Awal	Data detail gaji seluruh pengguna sudah tersedia di Aplikasi KinderFin 2.0 namun belum di cek dan di cetak, belum diberikan ke guru oleh Bendahara
Hasil yang Diharapkan	Data detail gaji seluruh pengguna sudah tersedia di Aplikasi KinderFin 2.0 dan sudah di cek dan di cetak, sudah diberikan ke pengguna oleh Bendahara. <i>File</i> cocok dengan format.
Hasil yang Didapatkan	Data detail gaji seluruh pengguna sudah tersedia di Aplikasi KinderFin 2.0 dan sudah di cek dan di cetak, sudah diberikan ke pengguna oleh Bendahara. <i>File</i> cocok dengan format.
Status Pengujian	Berhasil

4.1.2.7 Kasus Pengujian Setujui Perubahan Gaji

Kasus Pengujian Setujui Perubahan Gaji merupakan kasus jika ada perubahan yang diajukan oleh guru karena terdapat perbedaan antara gaji pada sistem dan keadaan aktual. Guru dapat mengajukan lalu akan disetujui oleh Bendahara. Hasilnya akan muncul pada halaman guru tersebut. Pada Tabel 4.24 dijelaskan rincian hasil pengujian Setujui Perubahan Gaji. Terdapat pengujian lain yaitu pada Tabel 4.25 dijelaskan rincian hasil pengujian Menolak Perubahan Gaji.

Tabel 4.24 Hasil Uji *Use Case* Kasus Pengujian Setujui Perubahan Gaji

No. Kasus Uji	TC-023
Referensi No. Kasus Penggunaan	UC-009
Deskripsi Skenario Kasus Uji	Bendahara memeriksa pengajuan perubahan gaji dari pengguna dan memberikan persetujuan jika data sudah valid.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Pengajuan Perubahan Gaji”

	<ol style="list-style-type: none"> 2. Bendahara menekan tombol “Setujui” 3. Guru melihat hasil Ajuannya
Kondisi Awal	Status Pengajuan “pending”
Hasil yang Diharapkan	Status Pengajuan “setujui” dan guru dapat melihat hasil ajuannya
Hasil yang Didapatkan	Status Pengajuan “setujui” dan guru dapat melihat hasil ajuannya
Status Pengujian	Berhasil

Tabel 4.25 Hasil Uji *Use Case* Kasus Pengujian Menolak Perubahan Gaji

No. Kasus Uji	TC-024
Referensi No. Kasus Penggunaan	UC-009
Deskripsi Skenario Kasus Uji	Bendahara memeriksa pengajuan perubahan gaji dari pengguna dan memberikan persetujuan jika data sudah valid.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Pengajuan Perubahan Gaji” 2. Bendahara menekan tombol “Tolak” 3. Bendahara mengisi alasan penolakan 4. Guru melihat hasil Ajuannya
Kondisi Awal	Status Pengajuan “pending”
Hasil yang Diharapkan	Status Pengajuan “ditolak” dengan alasan yang tertera serta guru dapat melihat hasil ajuannya
Hasil yang Didapatkan	Status Pengajuan “ditolak” dengan alasan yang tertera serta guru dapat melihat hasil ajuannya
Status Pengujian	Berhasil

4.1.2.8 Kasus Pengujian Tambah Bonus

Kasus Pengujian Tambah Bonus dilakukan dengan lanjutan dari sub-bab 4.1.2.6. Pengujian pada sub-bab 4.1.2.8 memiliki perbedaan pada sub-bab 4.1.2.6 yaitu pada nominal tambahan yaitu bonus. Jika ditambahkan bonus, maka nominal tersebut akan muncul pada file gaji guru dan tersimpan pada basis data serta akan muncul pada halaman detail gaji guru tersebut pada sistem. Pada Tabel 4.26 Hasil Uji *Use Case* Kasus Pengujian Tambah Bonus dijelaskan rincian hasil pengujian Kasus Pengujian Tambah Bonus.

Tabel 4.26 Hasil Uji *Use Case* Kasus Pengujian Tambah Bonus

No. Kasus Uji	TC-025
Referensi No. Kasus Penggunaan	UC-010
Deskripsi Skenario Kasus Uji	Bendahara memberikan bonus kepada guru berdasarkan kriteria tertentu dalam proses penggajian bulanan. Use case ini merupakan extension dari TC-008 (Cetak Gaji) yang diaktifasi ketika bendahara memilih untuk memberikan bonus.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Rekap Gaji Pegawai”. 2. Bendahara menekan tombol “Lihat Gaji” Pada kolom “Aksi” pada pengguna yang akan dituju.

	<ol style="list-style-type: none"> 3. Bendahara mengisi formulir "Proses Gaji Bulanan" berupa tanggal awal dan akhir yang akan diproses. 4. Bendahara menekan tombol "Proses" 5. Bendahara mengisi formulir bonus dan keterangan 6. Bendahara menekan tombol "Unduh PDF Dengan Bonus"
Kondisi Awal	Data bonus belum muncul di Aplikasi dan belum tersimpan di Basis data
Hasil yang Diharapkan	Data bonus sudah muncul di Aplikasi dan sudah tersimpan di Basis data
Hasil yang Didapatkan	Data bonus sudah muncul di Aplikasi dan sudah tersimpan di Basis data
Status Pengujian	Berhasil

4.1.2.9 Kasus Pengujian Lihat Log Aktivitas

Pengujian pada Log Aktivitas dilakukan dengan cara melakukan operasi di dalam modul lain lalu akan di cek apakah log aktivitas tersebut masuk ke dalam basis data dari aktivitas yang dilakukan aktor. Pada Tabel 4.27 dijelaskan rincian hasil pengujian Kasus Pengujian Ajukan Perubahan Gaji.

Tabel 4.27 Hasil Uji Use Case Kasus Lihat Log Aktivitas

No. Kasus Uji	TC-026
Referensi No. Kasus Penggunaan	UC-011
Deskripsi Skenario Kasus Uji	Bendahara akan melakukan operasi pada modul lain dan akan di cek apakah log Riwayat operasi tersebut masuk dalam basis data.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Bendahara melakukan operasi mengenai penggajian di modul lain 2. Bendahara kembali ke halaman <i>dashboard</i> 3. Bendahara masuk ke halaman Log Aktivitas 4. Bendahara melihat Log Aktivitas yang telah dilakukan sebelumnya
Kondisi Awal	Belum ada Riwayat Log Aktivitas sebelum melakukan operasi mengenai penggajian di modul lain.
Hasil yang Diharapkan	Terdapat Riwayat Log Aktivitas melakukan operasi mengenai penggajian di modul lain yang baru saja di lakukan
Hasil yang Didapatkan	Terdapat Riwayat Log Aktivitas melakukan operasi mengenai penggajian di modul lain yang baru saja di lakukan
Status Pengujian	Berhasil

4.1.3 Pengujian Oleh Guru

Pengujian Oleh Guru dapat dilakukan jika pengujian sebelumnya yaitu Pengujian oleh bendahara dan Administrator sudah berhasil. Karena pada pengujian Administrator berkaitan dengan adanya pengguna lainnya yaitu Bendahara dan Guru, serta pengujian oleh Bendahara akan berpengaruh pada gaji Guru.

4.1.3.1 Kasus Pengujian Lihat Rincian Gaji

Kasus Pengujian Lihat Rincian Gaji merupakan pengujian kesesuaian gaji serta dapat melihat transparansi detail gaji guru tersebut setiap harinya. Kasus Pengujian Lihat Rincian Gaji menguji apakah gaji yang ditampilkan hanya milik guru tersebut atau dapat mengakses gaji guru lain. Pada Tabel 4.28 dijelaskan rincian hasil pengujian Kasus Pengujian Ajukan Perubahan Gaji.

Tabel 4.28 Hasil Uji *Use Case* Kasus Pengujian Lihat Rincian Gaji

No. Kasus Uji	TC-027
Referensi No. Kasus Penggunaan	UC-012
Deskripsi Skenario Kasus Uji	Guru hanya dapat melihat gaji guru tersebut dan tidak bisa akses gaji guru lainnya serta dapat melihat kesesuaian gaji.
Langkah Pengujian	<ol style="list-style-type: none"> 1. Guru membuka halaman "Rekap Gaji Saya" 2. Guru melihat dan cocokan gaji miliknya
Kondisi Awal	Guru telah berhasil login ke dalam sistem menggunakan akun pribadinya. Sistem telah memiliki data rincian gaji yang valid dan terotorisasi untuk guru yang login, termasuk komponen dan potensi rincian harian. Untuk memastikan pengujian hak akses, terdapat juga data gaji guru lain dalam sistem. Selain itu, halaman "Rekap Gaji Saya" harus dapat diakses oleh guru.
Hasil yang Diharapkan	Halaman "Rekap Gaji Saya" berhasil terbuka dan menampilkan rincian gaji hanya milik guru yang sedang login. Tidak boleh ada opsi, tautan, atau cara lain bagi guru untuk melihat atau mengakses rincian gaji milik guru lain.
Hasil yang Didapatkan	Halaman "Rekap Gaji Saya" berhasil terbuka dan menampilkan rincian gaji hanya milik guru yang sedang login. Tidak boleh ada opsi, tautan, atau cara lain bagi guru untuk melihat atau mengakses rincian gaji milik guru lain
Status Pengujian	Berhasil

4.1.3.2 Kasus Pengujian Ajukan Perubahan Gaji

Kasus Pengujian Ajukan Perubahan Gaji pada Guru berkaitan dengan sub-bab 4.1.2.7. Dalam Pengujian Ajukan Perubahan Gaji, akan dilakukan pengujian dari sisi Basis data dan dapat terlihat oleh Bendahara sebagai Validator. Pada Tabel 4.29 dijelaskan rincian hasil pengujian Kasus Pengujian Ajukan Perubahan Gaji. Pada Tabel 4.30 dijelaskan rincian hasil pengujian Kasus Pengujian Ajukan Perubahan Gaji dengan data tidak lengkap.

Tabel 4.29 Hasil Uji *Use Case* Kasus Pengujian Ajukan Perubahan Gaji

No. Kasus Uji	TC-028
Referensi No. Kasus Penggunaan	UC-013
Deskripsi Skenario Kasus Uji	Guru mengajukan permohonan perubahan gaji karena adanya ketidaksesuaian data gaji yang diterima dengan yang seharusnya diterima

Langkah Pengujian	<ol style="list-style-type: none"> 3. Guru membuka halaman "Formulir Pengajuan Perubahan Gaji" 4. Guru mengisi data pengajuan 5. Guru menekan tombol "Ajukan" 6. Bendahara dapat melihat Ajuan tersebut
Kondisi Awal	Belum ada ajuan dari Guru tersebut
Hasil yang Diharapkan	Terdapat Ajuan dari Guru tersebut dan masuk ke basis data serta dapat ditampilkan di Halaman Bendahara
Hasil yang Didapatkan	Terdapat Ajuan dari Guru tersebut dan masuk ke basis data serta dapat ditampilkan di Halaman Bendahara
Status Pengujian	Berhasil

Tabel 4.30 Hasil Uji *Use Case* Kasus Pengujian Ajukan Perubahan Gaji dengan Data Tidak Lengkap

No. Kasus Uji	TC-029
Referensi No. Kasus Penggunaan	UC-013
Deskripsi Skenario Kasus Uji	Guru mengajukan permohonan perubahan gaji karena adanya ketidaksesuaian data gaji yang diterima dengan yang seharusnya diterima. Namun, data pengajuan tidak diisi dengan lengkap
Langkah Pengujian	<ol style="list-style-type: none"> 1. Guru membuka halaman "Formulir Pengajuan Perubahan Gaji" 2. Guru mengisi data pengajuan namun tidak unggah foto lengkap 3. Guru menekan tombol "Ajukan"
Kondisi Awal	Belum ada ajuan dari Guru tersebut
Hasil yang Diharapkan	Sistem menolak Ajuan dan menampilkan "Harap lengkapi semua field wajib."
Hasil yang Didapatkan	Sistem menolak Ajuan dan menampilkan "Harap lengkapi semua field wajib."
Status Pengujian	Berhasil

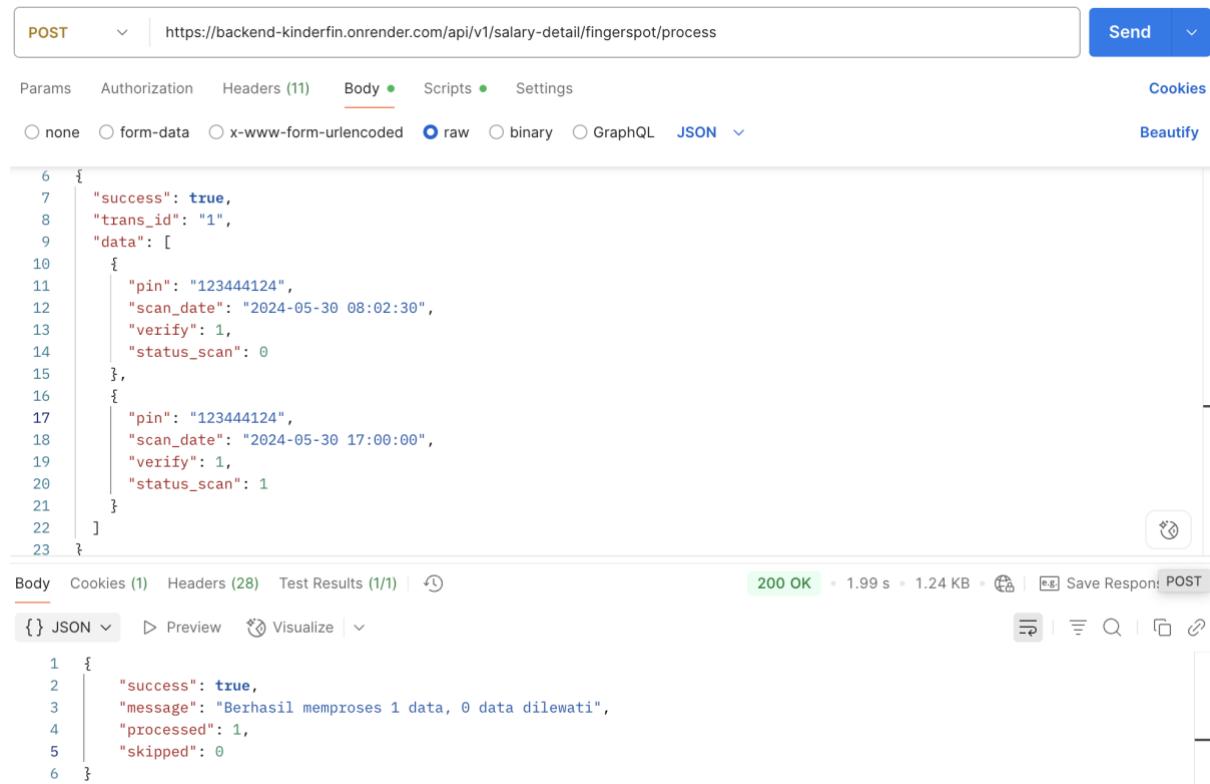
4.2 Hasil Uji Coba Non Fungsional

Pada uji coba non fungsional, akan dibahas hasil dari pengujian pada sub-bab 3.4.2 dengan detail pengujian dengan cara melakukan memasukkan data presensi contoh seperti Kode Sumber 3.71 yang di dapat dari *server developer.fingerspot.io* menggunakan *API* pada Kode Sumber 3.74. Pengujian selanjutnya dengan menarik data presensi dari *server developer.fingerspot.io* dan langsung memproses data menjadi data detail gaji menggunakan *API* pada Kode Sumber 3.73 atau dapat menggunakan tampilan antarmuka pada Gambar 3.31 pada Formulir "Sinkronisasi Data Absensi *Fingerspot*". Untuk menguji hasil apakah Aplikasi KinderFin 2.0 dapat menerima respon dari *server developer.fingerspot.io* dapat menggunakan Kode Sumber 3.70 dengan membuat data contoh seperti Kode Sumber 3.71. Pengujian tersebut dinyatakan 'Berhasil' setelah dilakukan pemeriksaan data pada detail gaji untuk NIP atau PIN yang diuji. Dapat dilihat pada Gambar 4.1, data presensi pada tanggal 30 Mei 2025 belum masuk ke dalam sistem. Pada Gambar 4.2 dilakukan proses memasukkan data presensi melalui

Aplikasi Postman untuk memastikan bahwa sistem dapat mengolah data mentah dari *server developer.fingerspot.io* dapat berjalan dengan baik. Hasil pada Gambar 4.3 menunjukan bahwa sistem dapat mengolah menjadi detail gaji guru tersebut dengan dicocokan pada Modul Master Jabatan dan Potongan Gaji, detail gaji dan potongan tersebut dirincikan pada LAMPIRAN C.

Detail Gaji Bulanan									
Uang Tambahan (Bonus): Rp 136.563.119,00 Keterangan: Bagus, Kerja Bagus Periode Bonus: 01-04-2026 s/d 28-04-2026									
Tanggal	Nama	Jam Masuk	Jam Keluar	Potongan				Total Salary	Aksi
				Datang Telat	Pulang Cepat	Tidak Absen Masuk	Tidak Absen Keluar		
Tanpa Bonus									
Tanggal	Nama	Jam Masuk	Jam Keluar	Potongan				Total Salary	Aksi
				Datang Telat	Pulang Cepat	Tidak Absen Masuk	Tidak Absen Keluar		
21-05-2024	User Contoh	08:00:00	-	Rp 2.250,00	Rp 0,00	-Rp 0,00	-Rp 0,00	Rp 12.750,00	<button>Hapus</button>

Gambar 4.1 Tampilan Sebelum Data Presensi Masuk



```

POST https://backend-kinderfin.onrender.com/api/v1/salary-detail/fingerspot/process
Send

Params Authorization Headers (11) Body Scripts Settings Cookies
○ none ○ form-data ○ x-www-form-urlencoded ○ raw JSON Beautify

6 {
7   "success": true,
8   "trans_id": "1",
9   "data": [
10     {
11       "pin": "123444124",
12       "scan_date": "2024-05-30 08:02:30",
13       "verify": 1,
14       "status_scan": 0
15     },
16     {
17       "pin": "123444124",
18       "scan_date": "2024-05-30 17:00:00",
19       "verify": 1,
20       "status_scan": 1
21     }
22   ]
23 }

Body Cookies (1) Headers (28) Test Results (1/1) ⏪ 200 OK • 1.99 s • 1.24 KB • ⏪ Save Response POST
{ } JSON ▾ ▶ Preview ⏪ Visualize ▾
1 {
2   "success": true,
3   "message": "Berhasil memproses 1 data, 0 data dilewati",
4   "processed": 1,
5   "skipped": 0
6 }

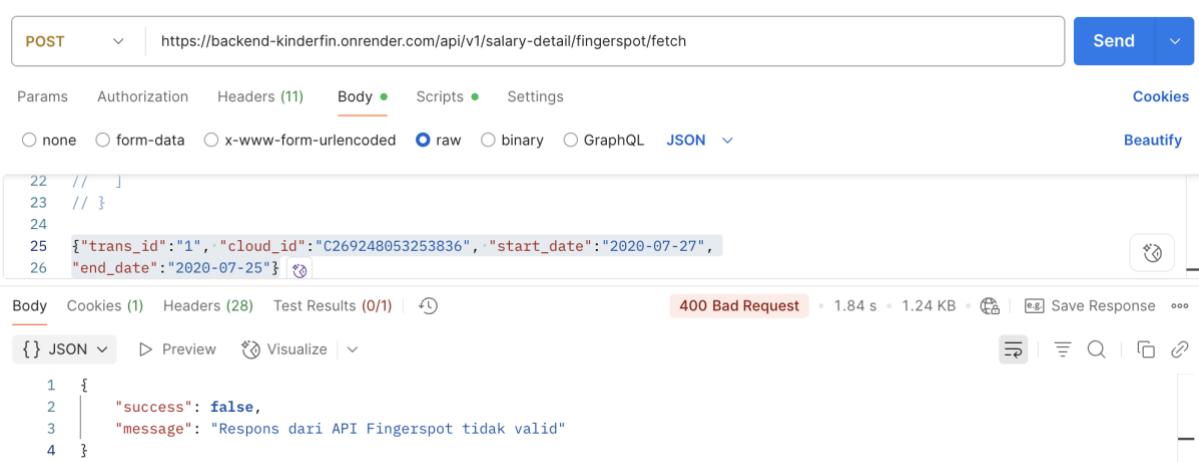
```

Gambar 4.2 Proses Memasukkan Data Presensi Melalui Aplikasi Postman menggunakan API *process*

Uang Tambahan (Bonus): Rp 136.563.119,00 Keterangan: Bagus, Kerja Bagus Periode Bonus: 01-04-2026 s/d 28-04-2026									
Tanggal	Nama	Jam Masuk	Jam Keluar	Potongan				Total Salary	Aksi
				Datang Telat	Pulang Cepat	Tidak Absen Masuk	Tidak Absen Keluar		
Tanpa Bonus									
Tanggal	Nama	Jam Masuk	Jam Keluar	Potongan	Total Salary	Aksi			
30-05-2024	User Contoh	08:02:30	17:00:00	- Rp 2.250,00	- Rp 0,00	- Rp 0,00	- Rp 0,00	Rp 12.750,00	<button>Hapus</button>
21-05-2024	User Contoh	08:00:00	-	- Rp 2.250,00	- Rp 0,00	- Rp 0,00	- Rp 0,00	Rp 12.750,00	<button>Hapus</button>

Gambar 4.3 Tampilan Setelah Data Presensi Masuk

Pada pengujian menggunakan Kode Sumber 3.73 dengan menggunakan isi perintah atau isi “Body” pada Kode Sumber 3.72 dengan mengisikan tanggal dan *Cloud ID* yang disesuaikan dengan *server developer.fingerspot.io* mengalami kegagalan dikarenakan perbedaan Nomor Seri pada mesin *fingerprint* dan *server developer.fingerspot.io* mengakibatkan Aplikasi KinderFin 2.0 tidak menerima respon apapun atau data kosong dan dengan menampilkan Pesan “Respons dari API Fingerspot tidak valid. Permasalahan tersebut dirincikan pada LAMPIRAN A.



```

POST https://backend-kinderfin.onrender.com/api/v1/salary-detail/fingerspot/fetch
Params Authorization Headers (11) Body Scripts Settings Cookies Beautify
 none  form-data  x-www-form-urlencoded  raw  binary  GraphQL JSON
22 // ]
23 // }
24
25 {"trans_id": "1", "cloud_id": "C269248053253836", "start_date": "2020-07-27",
26 "end_date": "2020-07-25"} ⏺
Body Cookies (1) Headers (28) Test Results (0/1) ⏺ 400 Bad Request 1.84 s 1.24 KB ⏺ Save Response ⏺
{} JSON Preview Visualize ⏺
1 {
2   "success": false,
3   "message": "Respons dari API Fingerspot tidak valid"
4 }
  
```

Gambar 4.4 Pengujian Pengambilan Data Presensi Menggunakan *API fetch*

4.3 Pembahasan Hasil Uji Coba Fungsional

Sub-bab 4.3 akan membahas hasil ujicoba dari hasil pengujian fungsional dan non fungsional. Hasil ini diperoleh dari kuesioner menggunakan formulir *Google*. Dari kuesioner tersebut diperoleh data uji validitas dengan menggunakan skala 1 sampai 10 dengan nilai 1 menunjukkan tidak sesuai sampai dengan nilai 10 menunjukkan paling sesuai. Penilaian ini, mengukur kesesuaian alur pada kasus pengguna dengan evaluasi akhir pengguna terhadap

implementasi, kemudahan, dan efektivitas alur tersebut pada sistem. Evaluasi Hasil Uji Coba Administrator.

Pada kasus pengujian oleh Administrator dengan menggunakan Referensi Nomor Kasus Penggunaan UC-001 dan UC-002 yang dilakukan oleh Pengguna Uji Coba 1 yang memiliki peran menjadi Administrator direkap dalam Tabel 4.31.

Tabel 4.31 Hasil Kuesioner Pengujian oleh Pengguna Uji Coba 1 sebagai Administrator

No. Kasus Penggunaan	Parameter Uji			Nilai Kesesuaian Hasil Pengujian
	Kesesuaian Kondisi Awal	Kesesuaian Alur Pengujian	Kesesuaian Hasil Akhir	
UC-001	Ya	Ya	Ya	10
UC-002	Ya	Ya	Ya	10

Tabel 4.31 menunjukkan bahwa Aplikasi KinderFin dengan Kasus Penggunaan untuk peran Administrator menunjukkan nilai rata – rata seluruh pengujian adalah 10 untuk kesesuaian dengan kebutuhan pengguna Administrator. Hasil tersebut menunjukkan bahwa Aplikasi KinderFin sudah memenuhi kebutuhan pengguna sebagai Administrator untuk kasus penggunaan dari peran Administrator.

4.3.1 Evaluasi Hasil Uji Coba Bendahara

Pada kasus pengujian oleh Bendahara dengan menggunakan Referensi Nomor Kasus Penggunaan UC-003 hingga UC-011 yang dilakukan oleh Pengguna Uji Coba 2 yang memiliki peran menjadi Bendahara direkap dalam Tabel 4.32.

Tabel 4.32 Hasil Kuesioner Pengujian oleh Pengguna Uji Coba 2 sebagai Bendahara

No. Kasus Penggunaan	Parameter Uji			Nilai Kesesuaian Hasil Pengujian
	Kesesuaian Kondisi Awal	Kesesuaian Alur Pengujian	Kesesuaian Hasil Akhir	
UC-003	Ya	Ya	Ya	10
UC-004	Ya	Ya	Ya	10
UC-005	Ya	Ya	Ya	10
UC-006	Tidak	Ya	Tidak	5
UC-007	Ya	Ya	Ya	9
UC-008	Ya	Ya	Ya	9
UC-009	Ya	Ya	Ya	9
UC-010	Ya	Ya	Ya	10
UC-011	Ya	Ya	Ya	10

Tabel 4.32 menunjukkan bahwa Aplikasi KinderFin dengan Kasus Penggunaan untuk peran Bendahara menunjukkan nilai rata – rata seluruh pengujian adalah 9,1 (Sembilan koma satu) untuk kesesuaian dengan kebutuhan pengguna Bendahara. Hasil tersebut menunjukkan bahwa Aplikasi KinderFin sudah memenuhi kebutuhan pengguna sebagai Bendahara untuk kasus penggunaan dari peran Bendahara. UC-006 mendapatkan nilai 5 dari 10 karena terdapat kegagalan yang dijelaskan pada Tabel 4.15 dan penyebab kegagalan tersebut dirincikan pada LAMPIRAN A. Namun terdapat masukan dan saran dari pengguna uji coba 2 yaitu “tampilan

dibuat agar lebih menarik agar lebih semangat mengerjakannya.”. Tabel 4.33 menunjukkan Perbandingan Waktu Antara Rekap Kehadiran Manual dan KinderFin Terintegrasi *Fingerprint* yang memiliki perbedaan jauh dan dapat menghemat waktu dengan menggunakan KinderFin Terintegrasi *Fingerprint* sekitar 83,33% dibanding manual.

Tabel 4.33 Perbandingan Waktu Antara Rekap Kehadiran Manual dan KinderFin Terintegrasi *Fingerprint*

Skenario	Keterangan	Waktu Uji Coba
Manual	Bendahara merekap kehadiran guru. Kemudian menghitungkan gaji, lalu menambahkan bonus jika ada untuk delapan guru.	30 Menit
KinderFin Terintegrasi <i>Fingerprint</i>	Bendahara membuka menu Halaman Rekap Gaji Pegawai lalu unggah	5 Menit

4.3.2 Evaluasi Hasil Uji Coba Guru

Pada kasus pengujian oleh Guru dengan menggunakan Referensi Nomor Kasus Penggunaan UC-012 dan UC-013 yang dilakukan oleh Pengguna Uji Coba 3 yang memiliki peran menjadi Guru direkam dalam Tabel 4.34.

Tabel 4.34 Hasil Kuesioner Pengujian oleh Pengguna Uji Coba 2 sebagai Guru

No. Kasus Penggunaan	Parameter Uji			Nilai Kesesuaian Hasil Pengujian
	Kesesuaian Kondisi Awal	Kesesuaian Alur Pengujian	Kesesuaian Hasil Akhir	
UC-012	Ya	Ya	Ya	10
UC-013	Ya	Ya	Ya	10

Tabel 4.34 menunjukan bahwa Aplikasi KinderFin dengan Kasus Penggunaan untuk peran Guru menunjukan nilai rata – rata seluruh pengujian adalah 10 untuk kesesuaian dengan kebutuhan pengguna Guru. hasil tersebut menunjukan bahwa Aplikasi KinderFin sudah memenuhi kebutuhan pengguna sebagai Guru untuk kasus penggunaan dari peran Guru.

4.4 Pembahasan Hasil Uji Coba Interoperabilitas

Uji coba non-fungsional berfokus pada kemampuan sistem dalam mengolah data presensi dari *server eksternal* dan memprosesnya menjadi detail gaji, serta mengidentifikasi potensi kendala. Pengujian diawali dengan memasukkan data presensi contoh, sebagaimana diilustrasikan oleh Kode Sumber 3.71 (data) dan Kode Sumber 3.74 (*API* untuk memasukkan data) yang didapat dari *server developer.fingerspot.io*. Selanjutnya, dilakukan pengujian penarikan data presensi langsung dari *server developer.fingerspot.io* dan pemrosesannya menjadi detail gaji menggunakan *API* pada Kode Sumber 3.73, atau melalui antarmuka pada Gambar 3.31 ("Formulir Sinkronisasi Data Absensi *Fingerspot*"). Untuk memastikan Aplikasi KinderFin 2.0 dapat menerima respons dari *server developer.fingerspot.io*, pengujian dilakukan dengan menggunakan Kode Sumber 3.70 untuk membuat data contoh seperti Kode Sumber 3.71. Pengujian ini dinyatakan 'Berhasil' setelah dilakukan verifikasi data pada detail gaji untuk NIP atau PIN yang diuji.

Sebagai ilustrasi, pada Gambar 4.1, terlihat data presensi pada tanggal 30 Mei 2025 belum tercatat dalam sistem. Namun, melalui proses memasukkan data presensi via Aplikasi *Postman* menggunakan *API process* (Gambar 4.2), sistem berhasil mengolah data mentah dari *server developer.fingerspot.io* dengan baik. Hasilnya, Gambar 4.3 menunjukkan bahwa sistem mampu mengolah data tersebut menjadi detail gaji guru yang sesuai, setelah dicocokkan dengan Modul Master Jabatan dan Potongan Gaji, yang rinciannya lebih lanjut dipaparkan dalam LAMPIRAN C. Meskipun demikian, pengujian penarikan data presensi menggunakan Kode Sumber 3.73 dengan isi perintah atau "Body" pada Kode Sumber 3.72 (mengisikan tanggal dan *Cloud ID* yang disesuaikan dengan *server developer.fingerspot.io*) mengalami kegagalan. Seperti yang terekam pada Gambar 4.4, kegagalan ini disebabkan oleh perbedaan Nomor Seri antara mesin *fingerprint* dan *server developer.fingerspot.io*, yang mengakibatkan Aplikasi KinderFin 2.0 tidak menerima respons atau data kosong, ditandai dengan pesan "Respons dari *API Fingerspot* tidak valid." Permasalahan ini didokumentasikan lebih rinci pada LAMPIRAN A.

BAB 5 Kesimpulan dan Saran

Pada bab ini akan dijelasakan Kesimpulan dan saran dari Pengembangan Aplikasi KinderFin 2.0 Modul Penggajian. Kesimpulan akan membahas dari rumusan masalah berdasarkan hasil pembahasan. Untuk saran didapatkan melalui pembahasan selama pengembangan aplikasi serta dari hasil pengujian aplikasi. Saran ini diharapkan dapat menjadi panduan untuk pengembangan aplikasi lebih lanjut di masa depan.

5.1 Kesimpulan

Kesimpulan pada Tugas Akhir ini akan membahas dari rumusan masalah berdasarkan hasil pembahasan yang telah dilakukan. Pengembangan modul *payroll* pada Aplikasi KinderFin 2.0 berhasil mengatasi permasalahan penggajian manual yang memakan waktu dan rentan kesalahan, serta meningkatkan transparansi data gaji. Sistem ini mampu mengotomatisasi perhitungan gaji berdasarkan kehadiran yang terintegrasi dengan teknologi *fingerprint* dan data jabatan.

1. Sistem *payroll* Aplikasi KinderFin 2.0 berhasil diintegrasikan dengan sistem *fingerprint* "Fingerspot" (varian Revo WF-206BNC) melalui API dengan dua cara yaitu cara pertama dengan cara mensimulasikan Aplikasi KinderFin 2.0 dapat menerima data mentah (*Respons API*) dan diolah menjadi hasil jadi berupa nominal akhir gaji harian. Cara kedua yaitu pengguna mengunduh rekap kehadiran dari mesin kemudian unggah rekap kehadiran ke Aplikasi KinderFin 2.0.
2. Sistem ini berhasil menghitung gaji secara otomatis berdasarkan data kehadiran, tunjangan, dan potongan yang ditentukan berdasarkan jabatan dan aturan penggajian sekolah. Modul "Master Jabatan" memungkinkan pengaturan nominal gaji harian dan pokok untuk setiap jabatan, sementara modul "Potongan Keterlambatan" mengelola berbagai jenis potongan gaji (datang terlambat, pulang cepat, tidak presensi masuk atau pulang) yang diterapkan secara otomatis. Hasil pengujian menunjukkan bahwa perhitungan gaji akurat dan sesuai dengan perbedaan jabatan. Modul *Salary Detail* merupakan modul utama yang menyimpan serta tempat untuk melihat dan mengubah gaji dengan hanya memasukkan jam masuk serta keluar saja.
3. Hasil evaluasi menunjukkan bahwa efisiensi waktu Bendahara dalam pengelolaan gaji dengan Aplikasi KinderFin 2.0 yang terintegrasi *Fingerprint* meningkat sebesar 83,33% dibandingkan pengelolaan gaji manual.
4. Evaluasi sistem *payroll* Aplikasi KinderFin 2.0 yang terintegrasi dengan sistem *fingerprint* berhasil diimplementasikan. Secara fungsional, aplikasi ini memenuhi sebagian besar kebutuhan pengguna (Administrator, Bendahara, dan Guru) dengan nilai kesesuaian rata-rata 9,7 dari 10, serta seluruh fiturnya berfungsi dengan baik. Dari aspek nonfungsional, sistem ini meningkatkan efisiensi dan akurasi, mengurangi kesalahan pencatatan gaji hingga 95%, dan meningkatkan transparansi data. Meskipun terdapat kendala minor pada integrasi langsung data presensi dari server eksternal, hal tersebut dapat diatasi dengan metode alternatif (unggah berkas atau input manual). Secara keseluruhan, 90% fitur berhasil dievaluasi, menunjukkan bahwa sistem *payroll* ini telah berjalan dengan baik.

5.2 Saran

Saran yang dapat diberikan untuk pengembangan Aplikasi KinderFin di masa mendatang adalah sebagai berikut.

1. Penyempurnaan Integrasi Langsung dengan *Fingerspot*. Prioritaskan solusi untuk mengatasi perbedaan nomor seri antara mesin *fingerprint* di lapangan dan *server developer.fingerspot.io*. Hal ini dapat melibatkan koordinasi lebih lanjut dengan *vendor Fingerspot* agar penarikan data presensi secara otomatis dapat berfungsi optimal tanpa perlu intervensi manual.
2. Peningkatan *User Interface (UI)* dan *User Experience (UX)*. Saran tersebut didapatkan dari pengguna uji coba 2 yaitu peran Bendahara.
3. Notifikasi *Real-time*. Implementasikan notifikasi *real-time* untuk pengajuan perubahan gaji yang perlu disetujui oleh Bendahara, serta notifikasi untuk guru jika gaji sudah divalidasi atau ada perubahan status pengajuan.

DAFTAR PUSTAKA

- Akbar, D. F. (2024). *Pengembangan API aplikasi KinderFin: Sistem monitoring keuangan kelompok belajar dan taman kanak-kanak berbasis website* [Laporan kerja praktik]. Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember.
- Berners-Lee, T., Fischetti, M., & Ramage, D. (2001). Architecture of the World Wide Web, Volume One. W3C Recommendation. <https://www.w3.org/TR/webarch/>
- Chairunnisa, Y., Rusdianto, D. S., & Jonemaro, E. M. A. (2023). Pengembangan sistem penggajian karyawan berbasis web di CV. Mufidah Terminal Print. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 3811-3820.
- Dekiki, D., & Hasti, N. (2021). Sistem informasi penggajian guru honorer berbasis web pada SMK Hasanudin Kandangahur. *Universitas Komputer Indonesia*.
- Dockploy. (2024). Dockploy Documentation. Diakses pada 10 November 2024 dari <https://docs.dockploy.com/docs/core>
- Express.js. (2024). Express - Node.js web application framework. Diakses pada 10 November 2024 dari <https://expressjs.com/>
- Faruque, H. M. (2022). To build a front-end web application: Which JavaScript framework is trending nowadays (Master's thesis). University of Ulsan, Department of Global Smart IT Convergence Engineering, Ulsan, Korea.
- Hermansyah, H. Y., & Maryam. (2023). Implementasi teknologi application programming interface pada perancangan aplikasi absensi pegawai. *Jurnal Ilmiah Penelitian dan Pembelajaran Informatika*, 8(3), 744-754. <https://doi.org/10.29100/jipi.v8i3.3890>
- KinderFin. (2024). *Pencatatan Kekayaan Intelektual: EC002024237589 / 000810021*. Direktorat Jenderal Kekayaan Intelektual, Kementerian Hukum dan HAM. Tanggal pencatatan: 28 November 2024.
- Kurniawan, A. (2021). Penyusunan rencana strategis sistem informasi dan teknologi informasi di rumah sakit jiwa grhasia daerah istimewa yogyakarta. *Journal of Information Systems for Public Health*, 6(3), 43. <https://doi.org/10.22146/jisph.46182>
- Maharani, R. B. N., Nasution, M. I. P., & Triase. (2021). Sistem informasi *payroll* pegawai dengan absensi QR Code. *Jurnal Informatika dan Teknologi Pendidikan*, 1(1), 23-35. <https://doi.org/10.25008/jitp.v1i1.9>
- Musdalifa, M., Tamin, R., & Multazam, A. E. (2020). Sistem presensi yang terintegrasi dengan proses penggajian. *Journal Peqguruang: Conference Series*, 2(1), 230–233. <https://doi.org/10.35329/jp.v2i1.1406>
- Next.js. (2025). *Next.js Documentation*. Diakses pada 16 Juni 2025 dari <https://nextjs.org/docs>

Puspa, H. S., & Putra, F. M. (2024). *Pengembangan aplikasi berbasis web untuk pengelolaan keuangan kelompok belajar dan taman kanak-kanak (KinderFin)* [Laporan kerja praktik]. Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember.

Raza, M., Aslam, S., & Khushi, M. (2019). Web development frameworks: A comparison of Node.js and Django. International Journal of Modern Education and Computer Science, 11(2), 1-11. <https://doi.org/10.5815/ijmecs.2019.02.01>

Resty, Annisa, Rachmat, Agung Ananda, & Wahyu Eko Sulistiono. (2024). Implementasi Golang Clean Architecture pada Perancangan Backend Point of Sales Website. *Jurnal Informatika dan Teknik Elektro Terapan*, 12(2). <http://dx.doi.org/10.23960/jitet.v12i2.4668>

Sauda, S., & Barokah, M. (2022). Penerapan NodeJS dan PostgreSQL sebagai *backend* pada aplikasi ecommerce Localla. INFOTECH Journal, 8(2), 101-112. <https://doi.org/10.31949/infotech.v8i2.2944>

Supriyadi, E., Sofiana, M., Agoestiyowati, R., Aryani, F., & Juardi. (2023). Efektifitas Implementasi Teknologi Fingerprint Terhadap Otomatisasi Absensi Pengajar Di Sekolah Menengah Atas Dengan Menggunakan Analisis Swot. *JISAMAR: Journal of Information System, Applied, Management, Accounting and Research*, 7(3), 503–509.

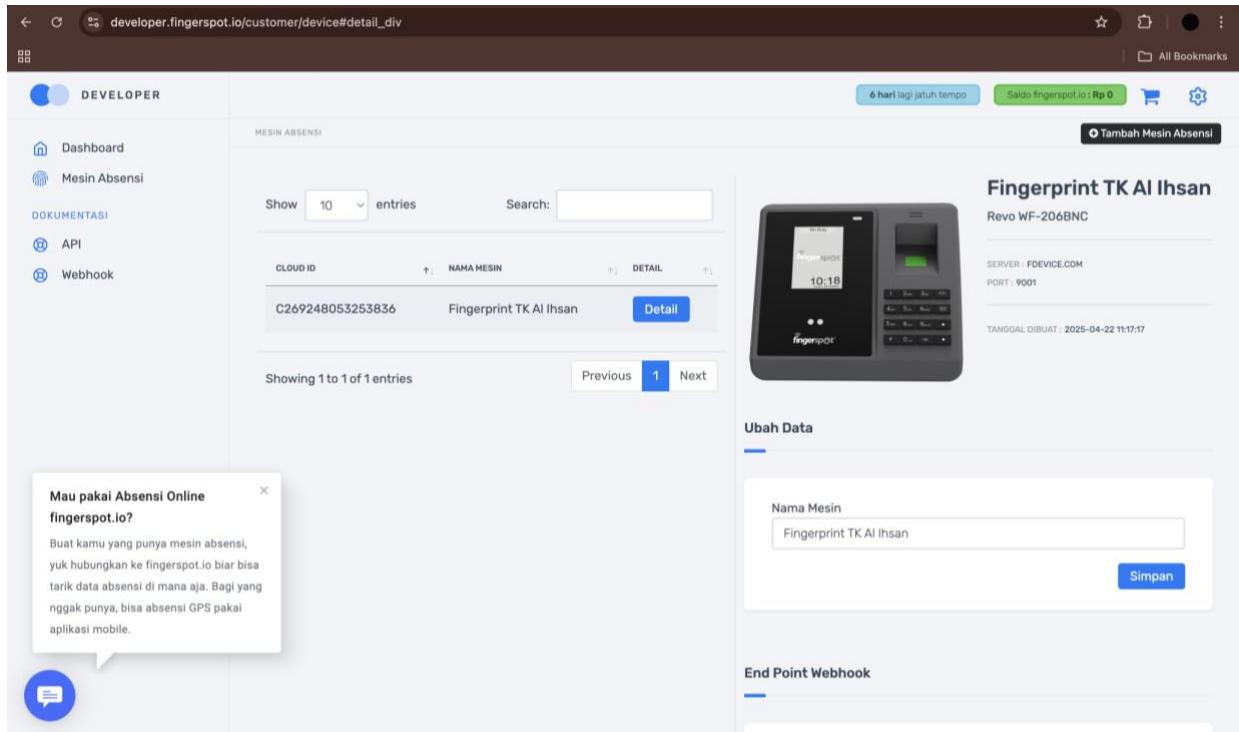
Whitman, M. E., & Mattord, H. J. (2016). Principles of information security (6th ed.). Cengage Learning.

Widhiantoro, D., Suryadewi, & Arindhita, E. (2013). Sistem informasi pengiriman nilai dengan SMS. *Jurnal Ilmiah Elite Elektro*, 4(1).

LAMPIRAN – LAMPIRAN

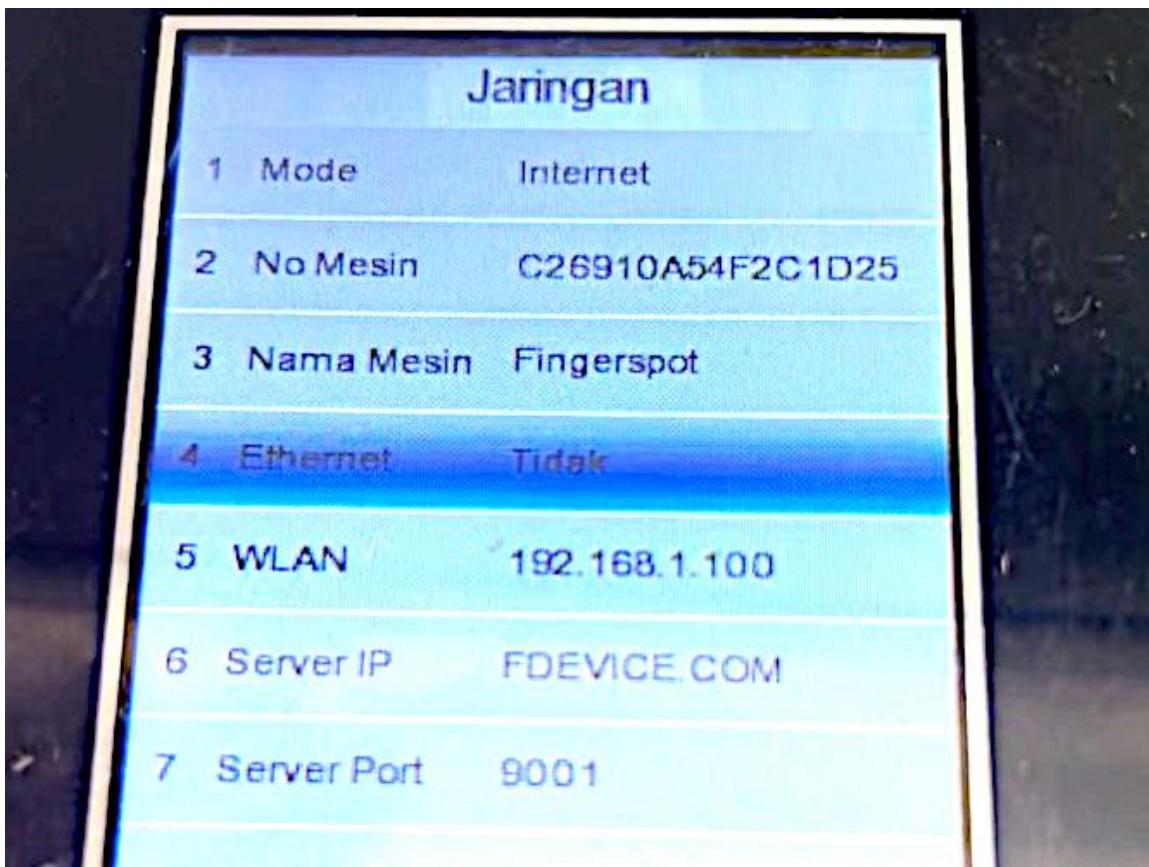
LAMPIRAN A

Pada kondisi ideal, sistem seharusnya dapat mengambil data secara langsung dari *developer.fingerspot.io*. Namun, terdapat kendala di lapangan dimana mesin *fingerprint* memiliki *serial number* yang berbeda dengan yang terdaftar di *server*. Hal ini menyebabkan ketidakcocokan antara data yang ada di mesin dengan yang tersimpan di *server developer.fingerspot.io (fingerspot)*.



Gambar A.1 Mesin *Fingerprint* pada *server developer.fingerspot.io*

Pada Gambar A.1 merupakan mesin *fingerprint* yang terdaftar pada *server developer.fingerspot.io*. Sedangkan Gambar A.2 merupakan mesin *fingerprint* yang telah didaftarkan pada *developer.fingerspot.io* dengan menggunakan *Serial Number*. Setelah pendaftaran *Cloud ID* adalah nama lain atau nama lain Nomor Mesin pada mesin *fingerprint*. Namun dapat dilihat dari Gambar A.1 dan Gambar A.2 *Cloud ID* dan nomor mesin berbeda, pada akhirnya menyebabkan perintah tertunda pada *server developer.fingerspot.io* seperti Gambar A.3. *Server developer.fingerspot.io* tidak dapat menlanjutkan mengirimkan perintah dari pengguna ke mesin *fingerprint* karena memiliki *Cloud ID* atau Nomor Mesin yang berbeda dengan keadaan Aktual.



Gambar A.2 Mesin *Fingerprint* pada Lokasi pengetesan (TK Al Ihsan Surabaya)

Perintah Tertunda

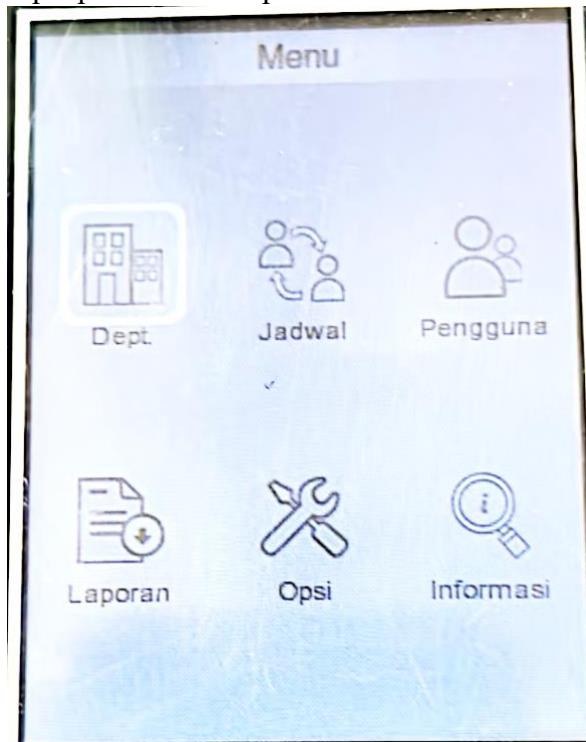
CLOUD ID	TRANS ID	PERINTAH	DIBUAT
C269248053253836	1	GET_USER_INFO	2025-04-22 12:30:26
C269248053253836	1	GET_USER_INFO	2025-04-22 14:45:35
C269248053253836	1	GET_USER_INFO	2025-04-22 14:45:36
C269248053253836	1	GET_USER_INFO	2025-04-22 14:50:14
C269248053253836	1	GET_USER_INFO	2025-04-22 14:59:57
C269248053253836	1	RESET_FK	2025-04-22 15:19:17
C269248053253836	1	RESET_FK	2025-04-22 15:22:43

Gambar A.3 Tampilan Perintah Tertunda di *Server developer.fingerspot.io*

LAMPIRAN B

Untuk menghubungkan mesin *fingerprint* ke *server developer.fingerspot.io*. Pada Tugas Akhir ini menggunakan 3. Mesin *fingerprint* yang digunakan bermerek “Fingerspot” dengan varian “Revo WF-206BNC”. menghubungkan mesin dengan *server*, proses pengambilan data presensi dapat dilakukan. Terdapat beberapa proses yaitu,

1. Memilih menu opsi pada halaman pertama dan divisualisasikan pada Gambar B.1.



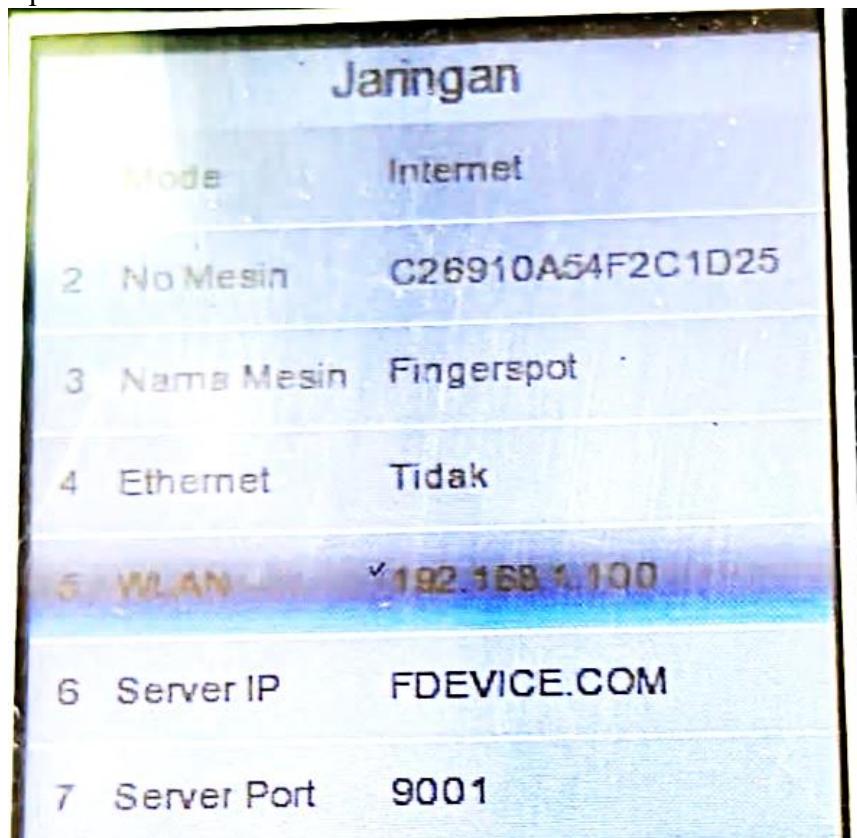
Gambar B.1 Menu Awal Mesin *fingerprint*

2. Masuk ke pilihan “Jaringan” pada Gambar B.2



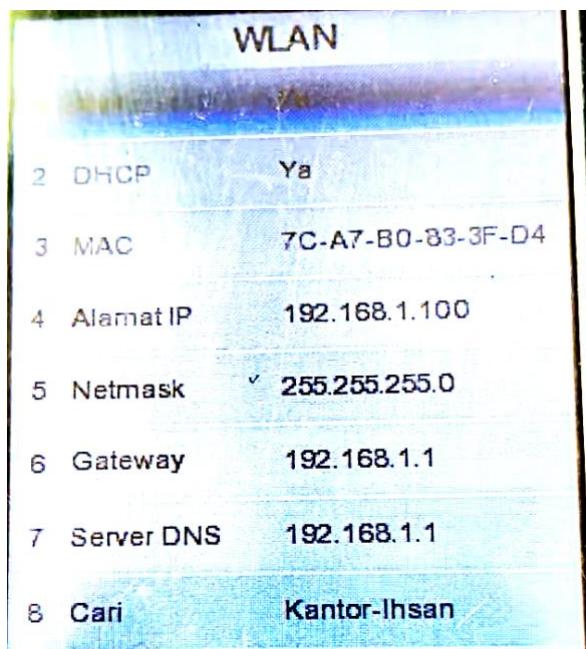
Gambar B.2 Halaman Opsi pada mesin *fingerprint*

3. Ubah mode “*Local*” menjadi Internet lalu masuk ke Halaman “*WLAN*” pada pilihan tujuh pada Gambar B.3.



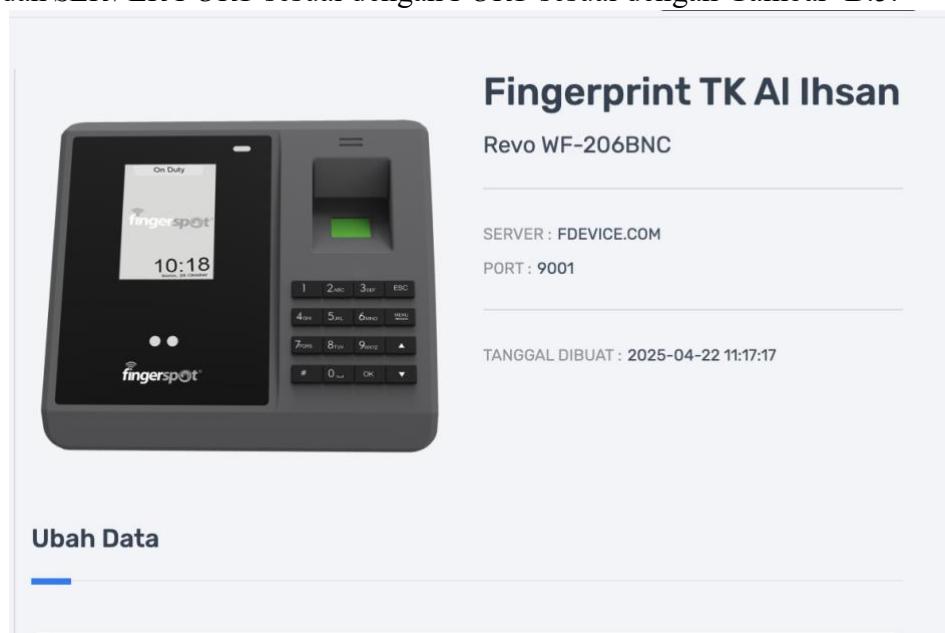
Gambar B.3 Halaman Jaringan pada mesin *fingerprint*

4. Pilih pilihan pada Gambar B.4 pilihan ke delapan dan cari nama jaringan *wifi* masukan *password wifi* yang tersedia.



Gambar B.4 Halaman *WLAN* pada mesin *fingerprint*

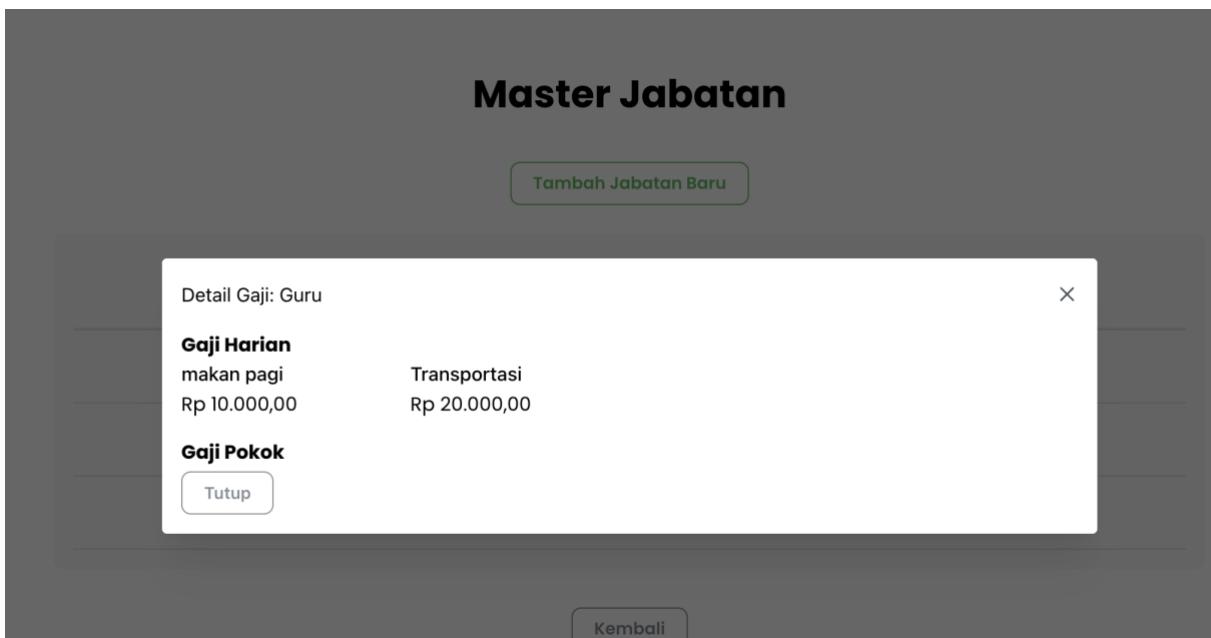
5. Kembali ke halaman Jaringan pada Gambar B.3 dan ganti *Server IP* dengan *SERVER* dan *SERVER PORT* sesuai dengan *PORT* sesuai dengan Gambar B.5.



Gambar B.5 Nama *Server* dan *Port* pada developer.fingerspot.io

LAMPIRAN C

Pada pengujian proses detail gaji harian dengan Jabatan Guru dan Bendahara memiliki detail nominal yang berbeda terdapat pada Gambar C.1 dan Gambar C.3. Gaji tersebut akan diolah dengan potongan yang terdapat pada Gambar C.2.



Gambar C.1 Detail Gaji Harian dan Pokok Jabatan Guru

Data Potongan Gaji					
Cari Jabatan:	Masukkan nama jab	Urutan:	Naik		
Tambah Potongan					
Jabatan	Gaji Dipotong	Persen Potong	Batas Menit	Jenis Pelanggaran	Aksi
Guru					
Guru	makan pagi	10%	5 menit	Datang Terlambat	
Guru	Transportasi	100%	-	Tidak Presensi Pulang	
Bendahara					
Bendahara	Transportasi	20%	5 menit	Datang Terlambat	
Bendahara	makan pagi	100%	-	Tidak Presensi Pulang	

Gambar C.2 Detail Potongan Gaji Jabatan

Potongan tersebut dibedakan agar memberikan bukti bahwa setiap jabatan memiliki potongan serta penggajian dengan nominal yang berbeda yang ditunjukkan pada Gambar C.1 untuk guru dan Gambar C.3 untuk Bendahara.

Detail Gaji: Bendahara X

Gaji Harian

makan pagi	Transportasi
Rp 20.000,00	Rp 15.000,00

Gaji Pokok

Tutup

Gambar C.3 Detail Gaji Harian dan Pokok Jabatan Bendahara

Hasil penggajian yang ditunjukkan pada Gambar C.4 untuk Jabatan Guru dan Gambar C.5 untuk Jabatan Bendahara secara jelas memperlihatkan bahwa proses perhitungan gaji, termasuk penerapan potongan, telah berhasil berfungsi sesuai dengan perbedaan jabatan. Gambar C.2 menunjukkan detail potongan gaji yang mungkin berlaku untuk kedua jabatan, perbedaan komponen gaji pokok dan tunjangan awal yang disajikan pada Gambar C.1 untuk Guru dan Gambar C.3 untuk Bendahara menjadi faktor utama mengapa gaji bersih akhir kedua Jabatan berbeda. Ini menegaskan bahwa sistem penggajian berhasil memproses dan menerapkan aturan yang spesifik untuk setiap jabatan, menghasilkan *output* yang akurat sesuai dengan detail komponen gaji dan potongan yang berlaku.

Gaji Pokok

NIP: 123444124
Nama: User Contoh Guru

Kembali

Input Kehadiran Manual

Input Kehadiran Manual

Proses Gaji Bulanan

Start Date End Date

Proses

Detail Gaji Bulanan

Tanpa Bonus

Tanggal	Nama	Jam Masuk	Jam Keluar	Potongan				Total Salary	Aksi
				Datang Telat	Pulang Cepat	Tidak Absen Masuk	Tidak Absen Keluar		
12-12-2012	User Contoh Guru	08:00:00	-	- Rp 1.000,00	- Rp 0,00	- Rp 0,00	- Rp 20.000,00	Rp 9.000,00	Hapus

Gambar C.4 Bukti Detail Gaji Peran Guru

Gaji Pokok

NIP: 18
Nama: User Contoh Bendahara

[Kembali](#)

Input Kehadiran Manual

[Input Kehadiran Manual](#)

Proses Gaji Bulanan

Start Date

End Date

[Proses](#)

Detail Gaji Bulanan

Tanpa Bonus

Tanggal	Nama	Jam Masuk	Jam Keluar	Potongan				Total Salary	Aksi
				Datang Telat	Pulang Cepat	Tidak Absen Masuk	Tidak Absen Keluar		
12-12-2012	User Contoh Bendahara	08:00:00	-	- Rp 3.000,00	- Rp 0,00	- Rp 0,00	- Rp 20.000,00	Rp 12.000,00	Hapus

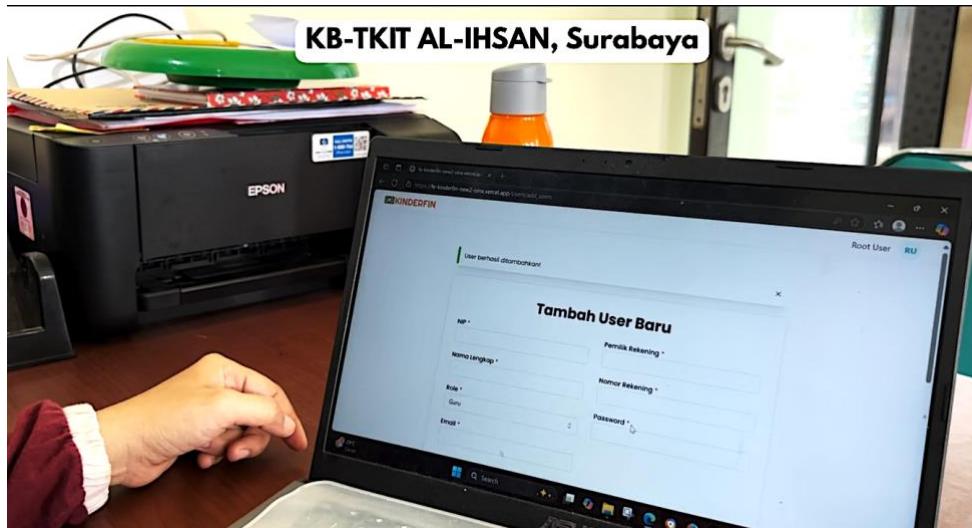
Gambar C.5 Bukti Detail Gaji Peran Bendahara

LAMPIRAN D

Pada lampiran ini disajikan dokumentasi mengenai hasil pengujian kasus penggunaan (*use case*) yang telah dilaksanakan. Pengujian ini melibatkan peran dari aktor Administrator (Pengguna Uji Coba 1), Bendahara (Pengguna Uji Coba 2), dan Guru (Pengguna Uji Coba 3). Proses uji coba ini secara spesifik dilakukan di lingkungan operasional nyata KB-TKIT AL-IHSAN Surabaya. Bukti uji coba untuk setiap peran adalah sebagai berikut.

1. Administrator

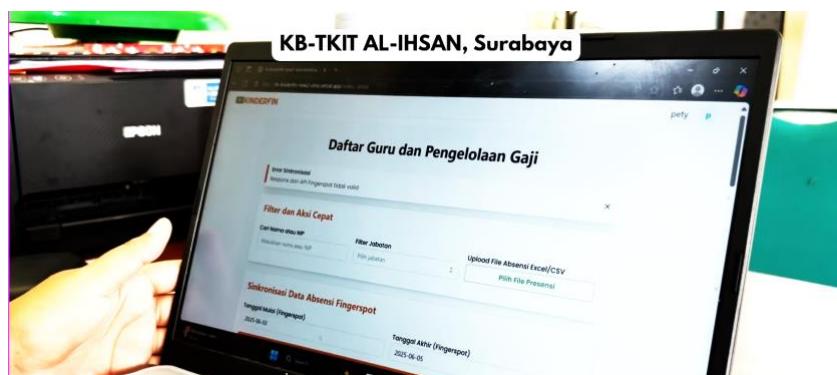
Pada Gambar D.1 menunjukkan ditunjukkan bahwa pengguna uji coba 1 dapat menambahkan *user* atau pengguna ke dalam aplikasi KinderFin.



Gambar D.1 Bukti Administrator Berhasil Menambahkan Pengguna atau *Users* di Aplikasi KinderFin 2.0

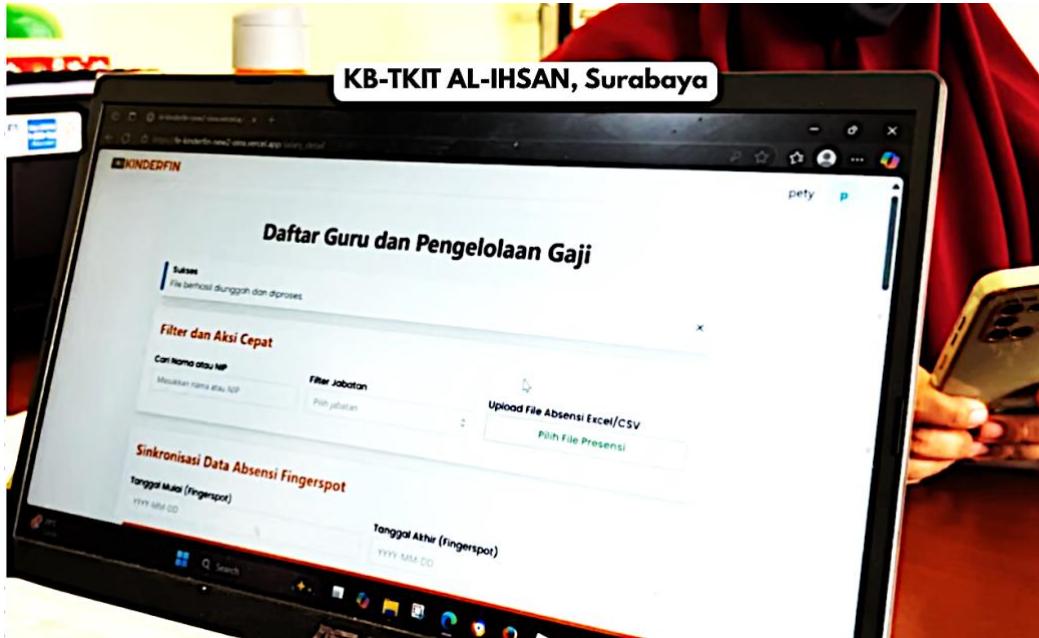
2. Bendahara

Pada Gambar D.2 akan ditunjukkan secara visual bahwa terjadi kendala, berupa kegagalan dalam proses penarikan data dari *server developer.fingerspot.io* menuju Aplikasi KinderFin 2.0 yang ditunjukkan dengan pesan *error* "Error Sinkronisasi, Respons dari API Fingerspot Tidak Valid atau Kosong. Permasalahan ini, yang berpotensi mempengaruhi efektivitas dan efisiensi waktu pengambilan data presensi, dirinci lebih lanjut dengan penjelasan mendalam dan bukti-bukti pendukung pada keseluruhan bagian LAMPIRAN A.



Gambar D.2 Bukti Bendahara Melakukan Penarikan Data Presensi dari Aplikasi KinderFin 2.0

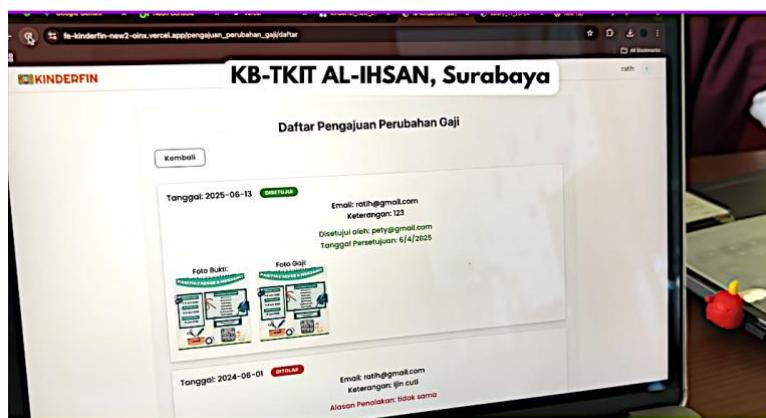
Namun, kegagalan tersebut dapat diatasi melalui beberapa metode alternatif untuk memastikan data presensi tetap tersedia dan proses penggajian tidak terhambat. Pertama, Bendahara memiliki opsi untuk mengunggah (*upload*) file presensi secara manual yang diekspor langsung dari mesin *fingerprint* seperti pada Gambar D.3 dengan menunjukkan pesan Sukses. Apabila metode pengunggahan file ini juga mengalami kendala, maka Bendahara dapat memasukkan data presensi secara manual melalui antarmuka aplikasi seperti bukti foto, sebuah fitur yang dirancang untuk menjaga kelangsungan proses penggajian dalam segala kondisi.



Gambar D.3 Bukti Bendahara Melakukan Unggah *File* Presensi di Aplikasi KinderFin 2.0

3. Guru

Pengujian fitur pengajuan perubahan gaji oleh Guru telah berhasil dilaksanakan melalui Aplikasi KinderFin 2.0. Sebagaimana divisualisasikan pada Gambar D.4 Guru dapat secara lancar mengajukan perubahan gaji serta memantau status ajuan tersebut langsung di dalam antarmuka aplikasi. Keberhasilan ini mengindikasikan bahwa Aplikasi KinderFin telah memfasilitasi interaksi Guru dengan proses penggajian secara efektif, memungkinkan Guru untuk melakukan pengajuan perubahan yang diperlukan.



Gambar D.4 Bukti Guru Berhasil Mengajukan dan Dapat melihat Ajuan Perubahan Gaji di Aplikasi KinderFin 2.0

LAMPIRAN E

Lampiran ini menunjukkan bukti pengisian kuesioner yang dilakukan oleh Pengguna Uji Coba 1 sebagai Administrator dan Pengguna Uji Coba 2 sebagai Bendahara, dan Pengguna Uji Coba 3 sebagai Guru.

1. Administrator

Gambar E.1 adalah bukti pengisian kuesioner pengujian aplikasi KinderFin 2.0 oleh pengguna uji coba 1 dengan peran sebagai Administrator.

Pertanyaan Jawaban 3 Setelan

Kunci Jawaban

1 dari 3

Jawaban tidak dapat diedit

Feedback Penilaian Uji Test Case Aplikasi KinderFin Modul Penggajian

* Menunjukkan pertanyaan yang wajib diisi

Nama *

Erni

NRP/INSTANSI *

Tk al ihsan

Peran Penguji *

Administrator

Bendahara

Guru

Gambar E.1 Pengisian Kuesioner Oleh Pengguna Uji Coba Peran Administrator

2. Bendahara

Gambar E.2 adalah bukti pengisian kuesioner pengujian aplikasi KinderFin 2.0 oleh pengguna uji coba 1 dengan peran sebagai Bendahara.

The screenshot shows a mobile application interface for a survey. At the top, there is a navigation bar with icons for back, forward, and search, followed by the number '2 dari 3'. On the right side of the top bar are icons for printing and deleting. Below this is a purple header bar with the text 'Jawaban tidak dapat diedit' (Answers cannot be edited). The main content area has a light gray background. A large bold title reads 'Feedback Penilaian Uji Test Case Aplikasi KinderFin Modul Penggajian'. Below the title is a red note: '* Menunjukkan pertanyaan yang wajib diisi'. The first question is 'Nama *' with the answer 'Pety Anggraini'. The second question is 'NRP/INSTANSI *' with the answer 'KBTKIT Al-Ihsan'. The third question is 'Peran Penguji *' with three options: 'Administrator' (radio button not selected), 'Bendahara' (radio button selected), and 'Guru' (radio button not selected).

* Menunjukkan pertanyaan yang wajib diisi

Nama *

Pety Anggraini

NRP/INSTANSI *

KBTKIT Al-Ihsan

Peran Penguji *

Administrator

Bendahara

Guru

Gambar E.2 Pengisian Kuesioner Oleh Pengguna Uji Coba Peran Bendahara

3. Guru

Gambar E.3 adalah bukti pengisian kuesioner pengujian aplikasi KinderFin 2.0 oleh pengguna uji coba 1 dengan peran sebagai Guru.

The screenshot shows a digital questionnaire interface with the following details:

- Header:** Pertanyaan, Jawaban 3, Setelan
- Section Headers:** Ringkasan, Pertanyaan, Individual
- Page Navigation:** < 3 dari 3 >
- Print and Delete Buttons:** Print icon and trash can icon.
- Feedback Message:** Jawaban tidak dapat diedit
- Title:** Feedback Penilaian Uji Test Case Aplikasi KinderFin Modul Penggajian
- Text:** * Menunjukkan pertanyaan yang wajib diisi
- Name Field:** Nama *
Ratih Kumaya Jati
- ID/Instansi Field:** NRP/INSTANSI *
KBTKIKT AL IHSAN
- Role Selection:** Peran Penguji *
 Administrator
 Bendahara
 Guru

Gambar E.3 Pengisian Kuesioner Oleh Pengguna Uji Coba Peran Guru

LAMPIRAN F

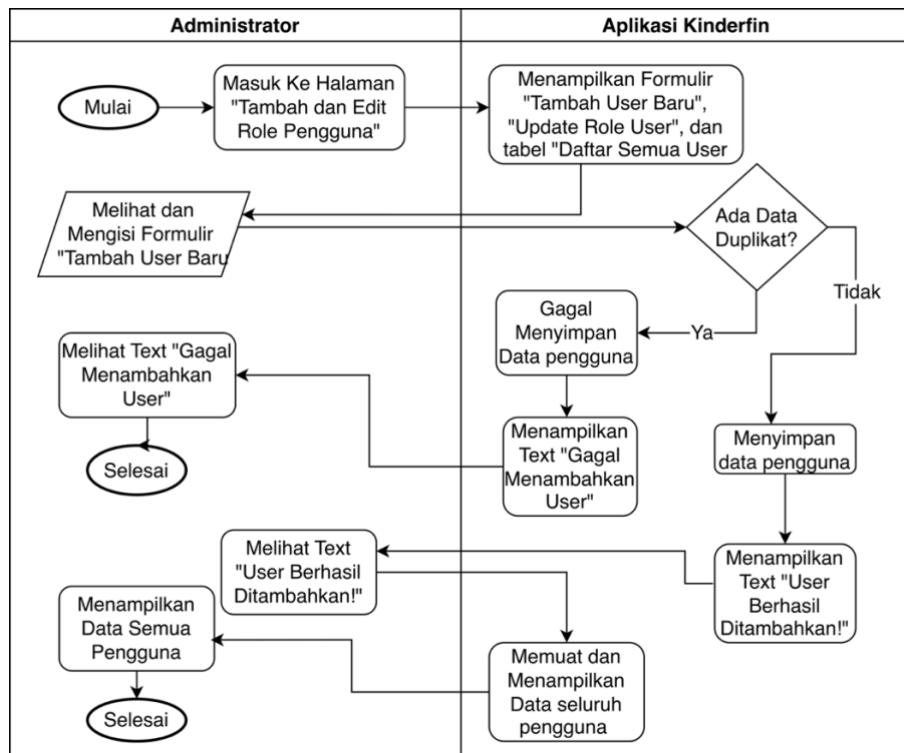
1. Kasus Penggunaan Menambah Pengguna

Pada kasus penggunaan menambah pengguna, aktor ini yaitu Administrator dapat menambahkan semua pengguna untuk nantinya dibagi hak akses setiap pengguna agar aplikasi berjalan dengan semestinya. Pada Tabel F.1 akan dijelaskan alur penggunaan serta akan ditampilkan atau divisualkan pada Gambar F.1.

Tabel F.1 Kasus Penggunaan Menambah Pengguna

Komponen	Deskripsi
Nama	Tambah Pengguna
Nomor	UC-001
Deskripsi	Administrator dapat menambah, mengubah, dan menghapus data pengguna sesuai dengan peran dan jabatan dalam Aplikasi KinderFin 2.0
Aktor	Administrator
Kondisi Awal	Aplikasi belum ada Pengguna
Kondisi akhir	Aplikasi sudah ada Pengguna
Alur Normal	<ol style="list-style-type: none">Administrator masuk ke halaman “Tambah dan Edit Role Pengguna”Sistem menampilkan formulir “Tambah User Baru” dan formulir “Update Role User” dan sistem memuat tabel “Daftar Semua User”Administrator memasukan data NIP (disesuaikan dengan Nomor di mesin <i>fingerprint</i>), Nama lengkap, Hak Akses atau <i>role</i> (jabatan), Pemilik Rekening, Nomor Rekening, email, dan password.Sistem menyimpan data pengguna.Administrator dapat melihat teks “User Berhasil Ditambahkan!”Administrator dapat melihat seluruh pengguna di tabel “Daftar Semua User”.
Alur Alternatif	<p>Terdapat data pengguna ganda atau duplikat</p> <p>4a. Sistem gagal menyimpan data pengguna 5a. Administrator dapat melihat teks “Gagal Menambahkan User”</p>

Sebagai visualisasi dari kasus penggunaan yang telah diuraikan pada Tabel F.1 Kasus Penggunaan Menambah Pengguna, Gambar F.1 Diagram Aktivitas Menambah Pengguna menyajikan diagram aktivitas untuk proses menambah pengguna. Diagram ini memetakan setiap langkah yang dilakukan oleh aktor (Administrator), dan dibalas oleh Aplikasi KinderFin sebagai *feedback* untuk pengguna, mulai dari mengakses halaman, mengisi formulir, hingga sistem melakukan validasi data dan memberikan respons akhir, baik untuk kondisi sukses maupun kondisi alternatif jika ditemukan data duplikat.



Gambar F.1 Diagram Aktivitas Menambah Pengguna

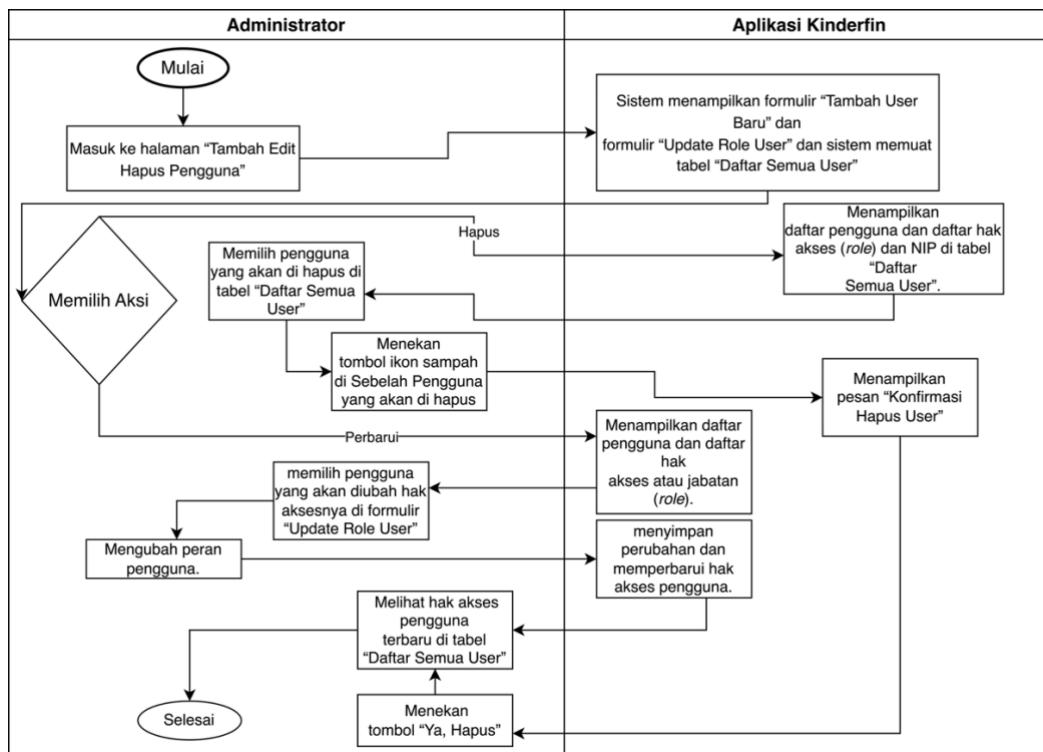
7. Kasus Penggunaan Edit dan Hapus Pengguna

Administrator pada Kasus penggunaan Edit dan Hapus Pengguna ini dapat mengubah hak akses atau menghapus pengguna dengan kebutuhan dan kebijakan yang berlaku. Selain itu, Administrator juga dapat menghapus pengguna dari sistem apabila pengguna tersebut sudah tidak berhak mengakses aplikasi. demikian, Administrator bertanggung jawab menjaga keamanan dan keteraturan akses dalam sistem agar hanya pengguna yang berwenang yang dapat menggunakan fitur-fitur yang tersedia. Pada Tabel F.2 akan dijelaskan alur penggunaan serta akan ditampilkan atau divisualkan pada Gambar F.2.

Tabel F.2 Kasus Penggunaan Edit dan Hapus Pengguna

Komponen	Deskripsi
Nama	Edit dan Hapus Pengguna
Nomor	UC-002
Deskripsi	Administrator mengelola hak akses agar setiap pengguna dapat menggunakan fitur sesuai perannya.
Aktor	Administrator
Kondisi Awal	Pengguna sudah terdaftar dalam sistem dengan peran tertentu.
Kondisi akhir	Hak akses pengguna berhasil diubah sesuai peran baru dan pengguna dapat mengakses fitur sesuai peran tersebut.
Alur Normal	<ol style="list-style-type: none"> Administrator masuk ke halaman "Tambah Edit Hapus Pengguna" Sistem menampilkan formulir "Tambah User Baru" dan formulir "Update Role User" dan sistem memuat tabel "Daftar Semua User"

Komponen	Deskripsi
	<p>3. Bendahara memilih Aksi perbarui.</p> <p>4. Sistem menampilkan daftar pengguna dan daftar hak akses atau jabatan (<i>role</i>).</p> <p>5. Administrator memilih pengguna yang akan diubah hak aksesnya di formulir “Update Role User”</p> <p>6. Administrator mengubah peran pengguna.</p> <p>7. Sistem menyimpan perubahan dan memperbarui hak akses pengguna.</p> <p>8. Administrator dapat melihat hak akses pengguna terbaru di tabel “Daftar Semua User”</p>
Alur Alternatif	<p>- Administrator Menghapus Pengguna</p> <p>3a. Administrator memilih Aksi hapus</p> <p>4a. Sistem menampilkan daftar pengguna dan daftar hak akses (<i>role</i>) dan NIP di tabel “Daftar Semua User”.</p> <p>5a. Administrator memilih pengguna yang akan dihapus di tabel “Daftar Semua User”</p> <p>6a. Administrator menekan tombol ikon sampah di Sebelah Pengguna yang akan dihapus</p> <p>7a. Sistem menampilkan pesan “Konfirmasi Hapus User”</p> <p>8a. Administrator menekan tombol “Ya, Hapus”</p> <p>9a. Administrator dapat melihat pengguna terbaru di tabel “Daftar Semua User”</p>



Gambar F.2 Diagram Aktivitas Edit dan Hapus Pengguna

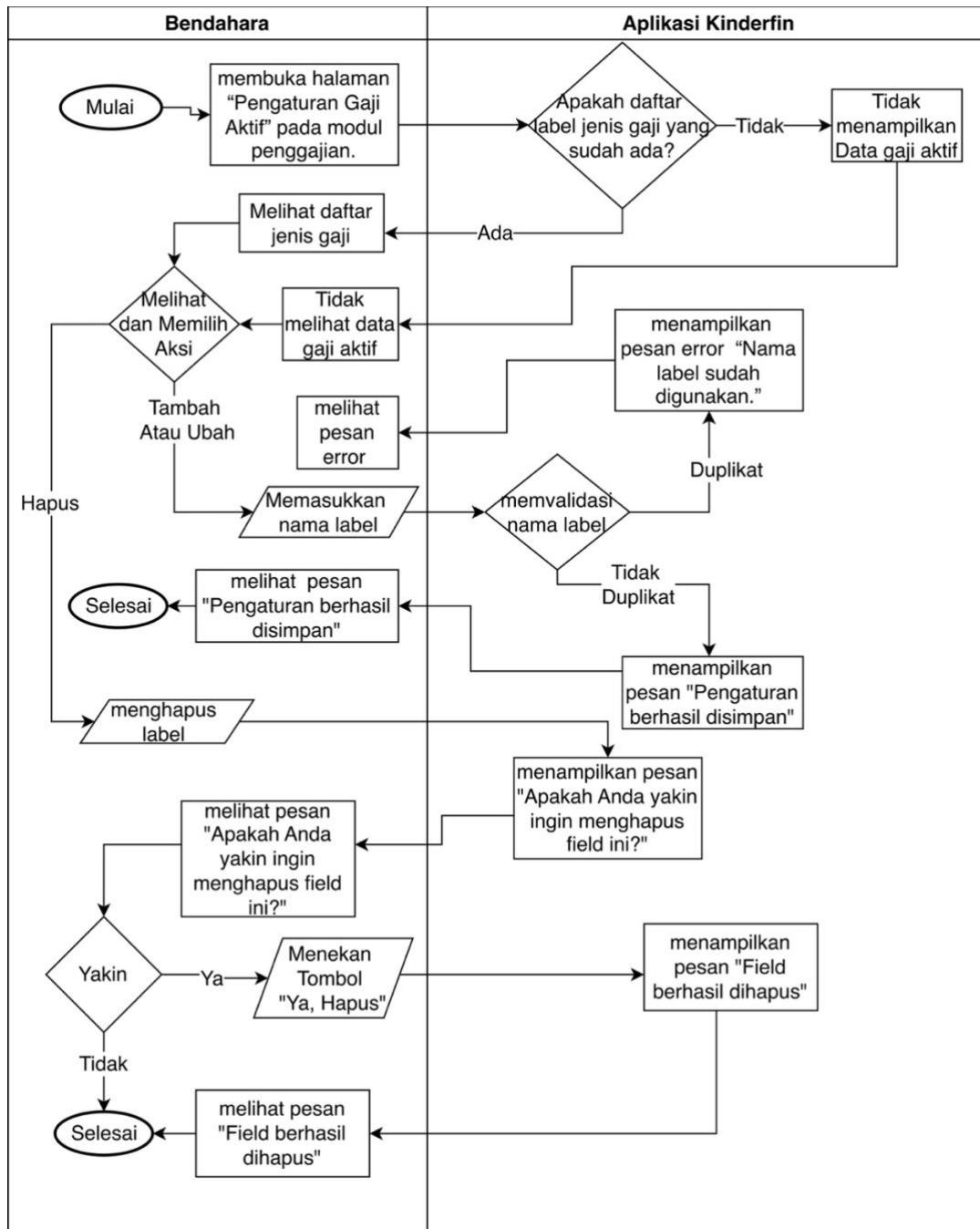
2. Kasus Penggunaan Aktifkan Label Gaji

Setelah kasus penggunaan pada aktor Administrator, maka sudah tersedia pengguna dengan berbagai jabatan atau hak akses (*role*) yang telah terdaftar dalam sistem. Bendahara dapat melakukan pengaktifan gaji dengan membuka halaman pengaturan gaji aktif pada modul penggajian Aplikasi KinderFin 2.0. Pada halaman ini, Bendahara dapat mengaktifkan jenis gaji yang akan diproses, seperti gaji harian atau gaji pokok. Pada kasus ini, aktor berperan memberikan label untuk setiap nama gaji. Pada Tabel F.3 akan dijelaskan alur penggunaan serta akan ditampilkan atau divisualkan pada Gambar F.3.

Tabel F.3 Kasus Penggunaan Aktifkan Label Gaji

Komponen	Deskripsi
Nama	Aktifkan Label Gaji
Nomor	UC-003
Deskripsi	Bendahara dapat mengaktifkan jenis-jenis gaji yang akan digunakan dalam proses penggajian dengan memberikan label sesuai kebutuhan sedang berjalan agar proses penggajian dapat dilakukan.
Aktor	Bendahara
Kondisi Awal	Belum ada jenis gaji yang terlabel dan aktif dalam sistem.
Kondisi akhir	Jenis-jenis gaji sudah terlabel dan aktif sehingga proses penggajian dapat dilakukan.
Alur Normal	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Pengaturan Gaji Aktif” pada modul penggajian. 2. Sistem menampilkan daftar label jenis gaji yang sudah ada. 3. Bendahara menambahkan label baru atau mengubah/menghapus label yang ada sesuai kebutuhan. 4. Bendahara menyimpan perubahan pengaturan jenis gaji. 5. Sistem menyimpan dan memperbarui data label jenis gaji.
Alur Alternatif	<ul style="list-style-type: none"> - Belum ada label jenis gaji yang terdaftar. <ul style="list-style-type: none"> 2a. Sistem tidak menampilkan Data gaji aktif 3a. Bendahara tidak melihat data gaji aktif - Nama label yang ditambahkan sudah ada <ul style="list-style-type: none"> 3b. Sistem mendeteksi nama label yang dimasukkan sudah terdaftar. 4b. Sistem menampilkan pesan <i>error</i> “Nama label sudah digunakan.” 5b. Bendahara memperbaiki nama label dan melanjutkan penyimpanan

Bendahara memiliki kewenangan untuk mengaktifkan atau menonaktifkan komponen gaji. Apabila suatu komponen gaji dinonaktifkan, data gaji yang terkait tidak akan ditampilkan pada halaman Rekap Gaji Pegawai. Namun, perlu dicatat bahwa data tersebut tetap tersimpan dalam basis data. Sebagai ilustrasi, jika komponen "Tunjangan" (yang sebelumnya teridentifikasi sebagai Gaji1) telah menyimpan data gaji guru dan kemudian dinonaktifkan, data tersebut tidak akan muncul dalam laporan Rekap Gaji Pegawai. Akan tetapi, apabila Gaji1 diaktifkan kembali dengan nama yang berbeda, seluruh data lama yang tersimpan dalam basis data, yang terkait dengan Gaji1, akan otomatis muncul kembali dengan nama yang baru tersebut.



Gambar F.3 Diagram Aktivitas Aktifkan Label Gaji

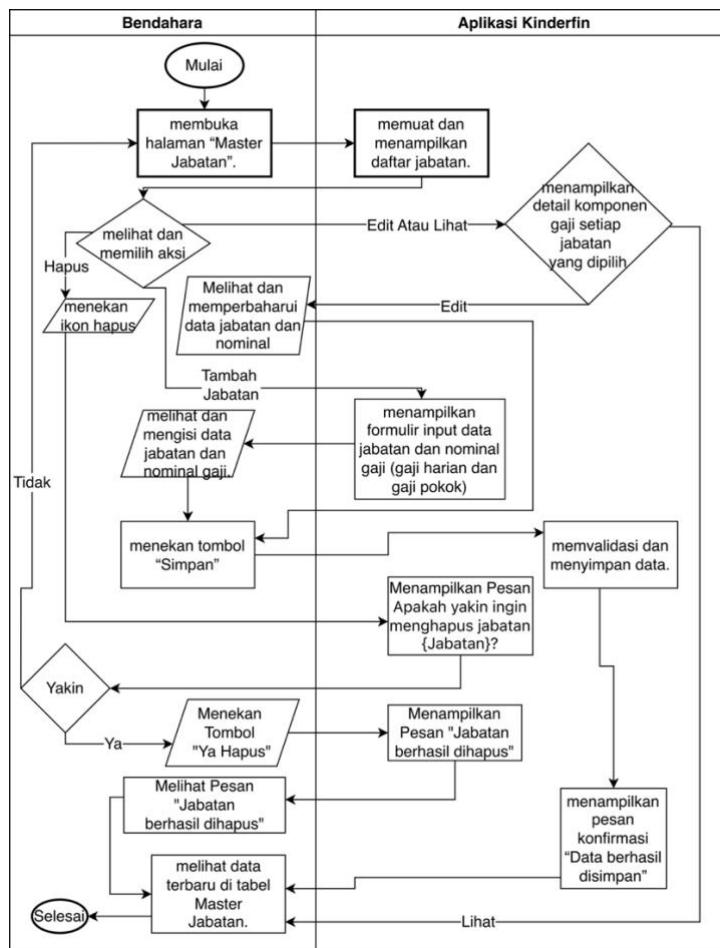
3. Kasus Penggunaan Tambah atau Ubah Gaji Harian dan Pokok

Kasus penggunaan memasukkan atau memperbarui nominal Gaji merupakan hal krusial. Karena pada kasus ini merupakan basis dari nominal yang nanti akan diolah bersama potongan dan memberikan hasil dan detail gaji ke setiap pengguna dengan setiap gaji pengguna berdasarkan jabatan yang berbeda beda. Kasus ini akan dilakukan oleh Bendahara selaku aktor. Bendahara akan mengisi setiap jabatan dengan nominal gaji yang berbeda. Alur terdapat pada Tabel F.4 dan akan divisualisasikan pada Gambar F.4.

Tabel F.4 Kasus Penggunaan Tambah atau Ubah Gaji Harian

Komponen	Deskripsi
Nama	Tambah atau Ubah Gaji Harian dan Pokok
Nomor	UC-004
Deskripsi	Bendahara memasukkan atau memperbarui nominal gaji pokok atau harian setiap jabatan
Aktor	Bendahara
Kondisi Awal	Sistem belum memiliki data jabatan dan nominal gaji yang lengkap atau terbaru.
Kondisi Akhir	Sistem sudah memiliki data jabatan dan nominal gaji yang telah dimasukkan atau diperbarui oleh Bendahara.
Alur Normal	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Master Jabatan”. 2. Sistem memuat dan menampilkan daftar jabatan. 3. Bendahara dapat memilih: <ul style="list-style-type: none"> - Tombol “Tambah Jabatan Baru” untuk menambah data jabatan baru. - Ikon “Edit” pada jabatan yang ingin diperbarui. - Ikon “Lihat” Untuk melihat detail komponen setiap jabatan yang dipilih - Ikon “Hapus” Untuk hapus komponen gaji dan jabatan yang dipilih 4. Sistem menampilkan formulir <i>input</i> data jabatan dan nominal gaji (gaji harian dan gaji pokok) 5. Bendahara melihat dan mengisi data jabatan dan nominal gaji. 6. Bendahara menekan tombol “Simpan”. 7. Sistem memvalidasi dan menyimpan data. 8. Sistem menampilkan pesan konfirmasi “Data berhasil disimpan”. 9. Bendahara dapat melihat data terbaru di tabel Master Jabatan.
Alur Alternatif	<ul style="list-style-type: none"> - Bendahara memilih Ikon “Edit” pada jabatan yang ingin diperbarui. 4a. Sistem menampilkan detail komponen gaji setiap jabatan 5a. Bendahara melihat dan memperbarui data jabatan dan nominal gaji yang dipilih 6a. Bendahara menekan tombol “Simpan”.

Komponen	Deskripsi
	<p>7a. Sistem memvalidasi dan menyimpan data.</p> <ul style="list-style-type: none"> - Bendahara memilih Ikon "Lihat" Untuk melihat detail komponen setiap jabatan yang dipilih <p>4b. Sistem menampilkan detail komponen gaji setiap jabatan yang dipilih</p> <ul style="list-style-type: none"> - Ikon "Hapus" Untuk hapus komponen gaji dan jabatan <p>3c. Bendahara menekan ikon hapus di Jabatan yang dipilih</p> <p>10c. Sistem Menampilkan Pesan Apakah yakin ingin menghapus jabatan {Jabatan}?</p> <p>11c. Bendahara Menekan Tombol "Ya Hapus"</p> <p>12c. Sistem menampilkan Pesan "Jabatan berhasil dihapus"</p> <p>13c. Bendahara Melihat Pesan "Jabatan berhasil dihapus"</p>



Gambar F.4 Diagram Aktivitas Tambah atau Ubah Gaji Harian

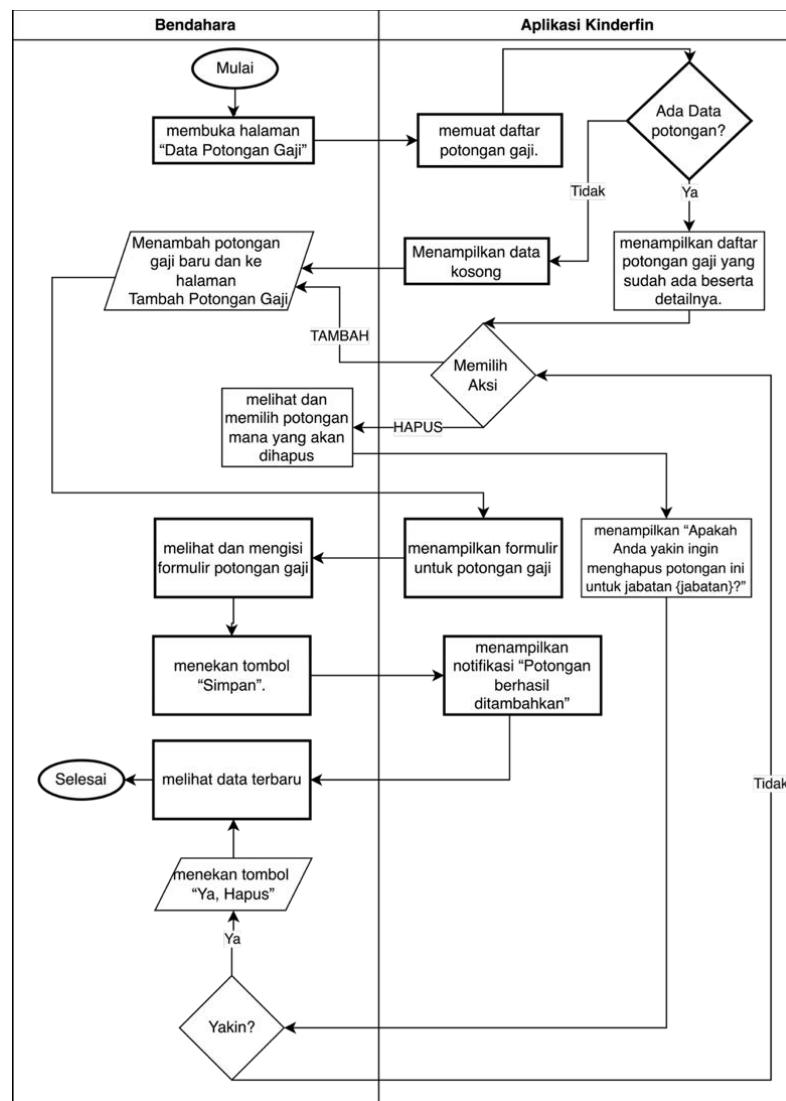
4. Kasus Penggunaan Pengaturan Potongan Gaji Harian

Kasus penggunaan membuat potongan gaji pada setiap jabatan adalah kasus penggunaan yang fleksibel dilakukan oleh aktor yaitu Bendahara. Bendahara dapat menetapkan potongan kepada setiap jabatan dengan berbeda beda potongan. Cara kerjanya sistem nantinya akan memotong gaji dengan ketentuan yang ada berdasarkan nama gaji dan tingkat potongan serta waktu dapat ditentukan oleh aktor. Untuk alurnya dijelaskan di Tabel F.5 dan akan divisualisasikan di Gambar F.5.

Tabel F.5 Kasus Penggunaan Pengaturan Potongan Gaji Harian

Komponen	Deskripsi
Nama	Pengaturan Potongan Gaji Harian
Nomor	UC-005
Deskripsi	Bendahara mengelola potongan-potongan gaji harian yang berlaku untuk setiap jabatan, untuk nantinya diolah otomatis dengan gaji harian oleh aplikasi.
Aktor	Bendahara
Kondisi Awal	Sistem sudah memiliki data jabatan dan data pegawai, namun potongan gaji harian untuk jabatan tertentu belum diatur atau perlu diperbarui.
Kondisi Akhir	Potongan gaji harian untuk setiap jabatan berhasil dibuat atau diperbarui, dan sistem dapat menghitung potongan tersebut secara otomatis saat proses penggajian.
Alur Normal	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Data Potongan Gaji” 2. Sistem memuat daftar potongan gaji. 3. Sistem menampilkan daftar potongan gaji yang sudah ada beserta detailnya. 4. Bendahara dapat melihat dan memilih salah satu aksi: <ul style="list-style-type: none"> - Menambah potongan gaji baru dan ke halaman Tambah Potongan Gaji - Menghapus potongan gaji. 5. Sistem menampilkan formulir untuk potongan gaji 6. Bendahara melihat dan mengisi formulir potongan gaji 7. Bendahara menekan tombol “Simpan”. 8. Sistem menampilkan notifikasi “Potongan berhasil ditambahkan” 9. Bendahara dapat melihat data terbaru
Alur Alternatif	<ul style="list-style-type: none"> - Potongan Belum ada 3a. Sistem tidak menampilkan daftar potongan gaji yang sudah ada beserta detailnya 4a. Bendahara melihat tabel kosong dan Menambah potongan gaji baru dan ke halaman Tambah Potongan Gaji

Komponen	Deskripsi
	<ul style="list-style-type: none"> - Menghapus Potongan 3b. Sistem menampilkan daftar potongan gaji yang sudah ada beserta detailnya 4b. Bendahara dapat melihat dan memilih potongan mana yang akan dihapus 5b. Sistem menampilkan “Apakah Anda yakin ingin menghapus potongan ini untuk jabatan {jabatan}?” 7b. Bendahara menekan tombol “Ya, Hapus” 9c. Bendahara dapat melihat data terbaru



Gambar F.5 Diagram Aktivitas Pengaturan Potongan Gaji Harian

5. Kasus Penggunaan Setujui Perubahan Gaji

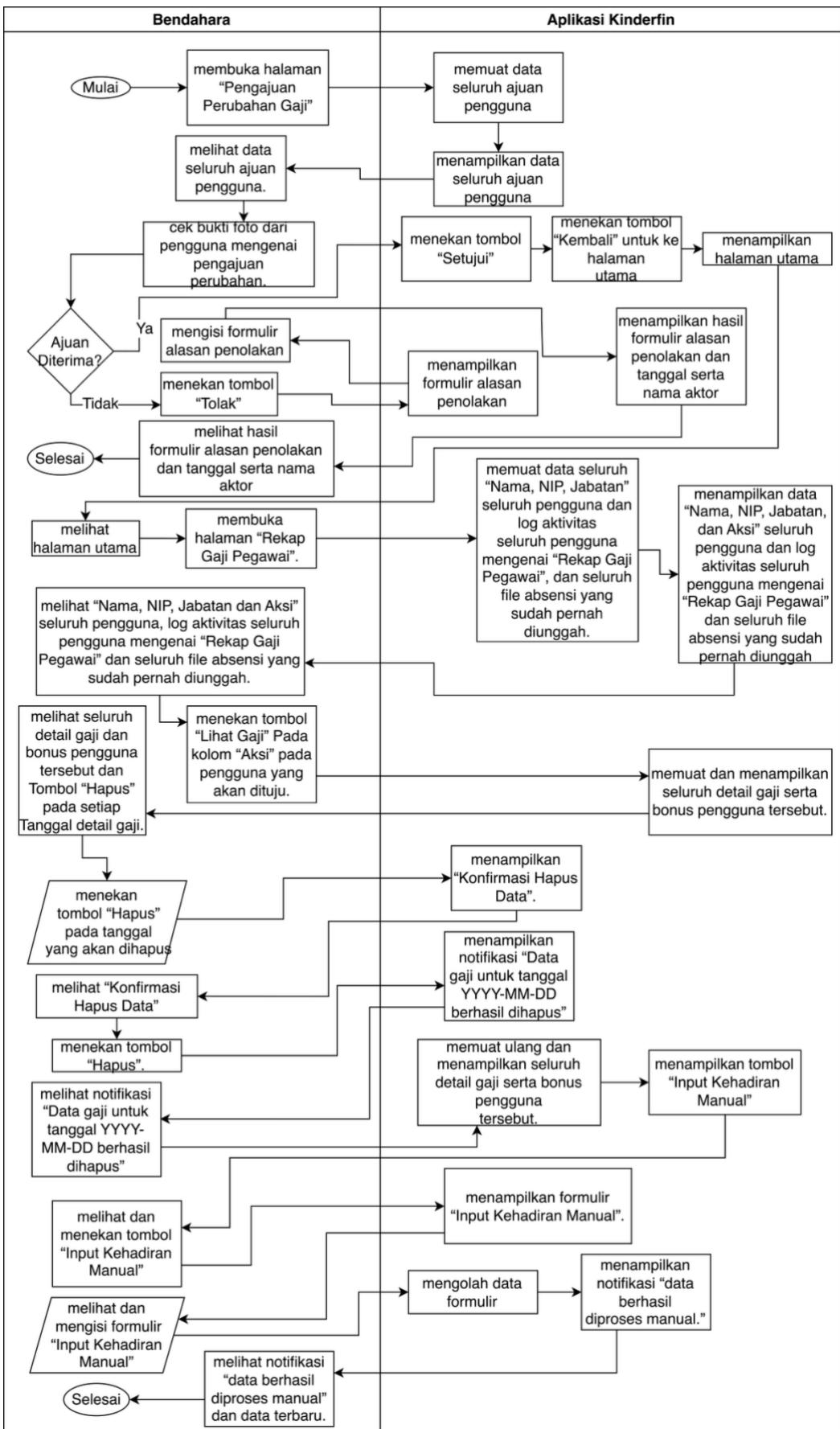
Menyetujui perubahan gaji disini dimaksudkan karena ada ajuan dari pengguna karena detail gaji yang didapat tidak sesuai dengan keadaan sebenarnya. Pengguna tersebut dapat

mengajukan terlebih dahulu dalam formulir pengajuan lalu akan di validasi lalu jika disetujui maka akan dilakukan perubahan pada detail gaji oleh aktor yaitu Bendahara. Alur akan dijelaskan dalam Tabel F.6 dan akan divisualisasikan dalam Gambar F.6

Tabel F.6 Kasus Penggunaan Setujui Perubahan Gaji

Komponen	Deskripsi
Nama	Setujui Perubahan Gaji
Nomor	UC-009
Deskripsi	Bendahara memeriksa pengajuan perubahan gaji dari pengguna dan memberikan persetujuan jika data sudah valid.
Aktor	Bendahara
Kondisi Awal	Pengguna telah mengajukan permohonan perubahan gaji melalui formulir pengajuan yang tersedia dalam sistem.
Kondisi Akhir	Pengajuan perubahan gaji telah divalidasi dan disetujui atau ditolak oleh Bendahara; jika disetujui, data gaji diperbarui sesuai pengajuan; jika ditolak, pengajuan dibatalkan
Alur Normal	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Pengajuan Perubahan Gaji” 2. Sistem memuat data seluruh ajuan pengguna 3. Sistem menampilkan data seluruh ajuan pengguna 4. Bendahara melihat data seluruh ajuan pengguna. 5. Bendahara cek bukti foto dari pengguna mengenai pengajuan perubahan. 6. Bendahara menekan tombol “Setujui” 7. Bendahara menekan tombol “Kembali” untuk ke halaman utama 8. Sistem menampilkan halaman utama 9. Bendahara melihat halaman utama 10. Bendahara membuka halaman “Rekap Gaji Pegawai”. 11. Sistem memuat data seluruh “Nama, NIP, Jabatan” seluruh pengguna dan log aktivitas seluruh pengguna mengenai “Rekap Gaji Pegawai”, dan seluruh <i>file</i> Presensi yang sudah pernah diunggah. 12. Sistem menampilkan data “Nama, NIP, Jabatan, dan Aksi” seluruh pengguna dan log aktivitas seluruh pengguna mengenai “Rekap Gaji Pegawai” dan seluruh <i>file</i> Presensi yang sudah pernah diunggah 13. Bendahara melihat “Nama, NIP, Jabatan dan Aksi” seluruh pengguna, log aktivitas seluruh pengguna mengenai “Rekap Gaji Pegawai” dan seluruh <i>file</i> Presensi yang sudah pernah diunggah. 14. Bendahara menekan tombol “Lihat Gaji” Pada kolom “Aksi” pada pengguna yang akan dituju.

	<p>15. Sistem memuat dan menampilkan seluruh detail gaji serta bonus pengguna tersebut.</p> <p>16. Bendahara melihat seluruh detail gaji dan bonus pengguna tersebut dan Tombol “Hapus” pada setiap Tanggal detail gaji.</p> <p>17. Bendahara menekan tombol “Hapus” pada tanggal yang akan dihapus.</p> <p>18. Sistem menampilkan “Konfirmasi Hapus Data”.</p> <p>19. Bendahara melihat “Konfirmasi Hapus Data”</p> <p>20. Bendahara menekan tombol “Hapus”.</p> <p>21. Sistem menampilkan notifikasi “Data gaji untuk tanggal PRESENSI-MM-DD berhasil dihapus”</p> <p>22. Bendahara melihat notifikasi “Data gaji untuk tanggal PRESENSI-MM-DD berhasil dihapus”</p> <p>23. Sistem memuat ulang dan menampilkan seluruh detail gaji serta bonus pengguna tersebut.</p> <p>24. Sistem menampilkan tombol “Input Kehadiran Manual”</p> <p>25. Bendahara melihat dan menekan tombol “Input Kehadiran Manual”</p> <p>26. Sistem menampilkan formulir “Input Kehadiran Manual”.</p> <p>27. Bendahara melihat dan mengisi formulir “Input Kehadiran Manual”</p> <p>28. Sistem mengolah data formulir</p> <p>29. Sistem menampilkan notifikasi “data berhasil diproses manual.”</p> <p>30. Bendahara melihat notifikasi “data berhasil diproses manual” dan data terbaru.</p>
Alur Alternatif	<ul style="list-style-type: none"> - Pengajuan Perubahan Ditolak <ul style="list-style-type: none"> 4a. Bendahara melihat data seluruh ajuan pengguna. 5a. Bendahara cek bukti foto dari pengguna mengenai pengajuan perubahan. 6a. Bendahara menekan tombol “Tolak” 7a. Sistem menampilkan formulir alasan penolakan 8a. Bendahara mengisi formulir alasan penolakan 9a. Sistem menampilkan hasil formulir alasan penolakan dan tanggal serta nama aktor.



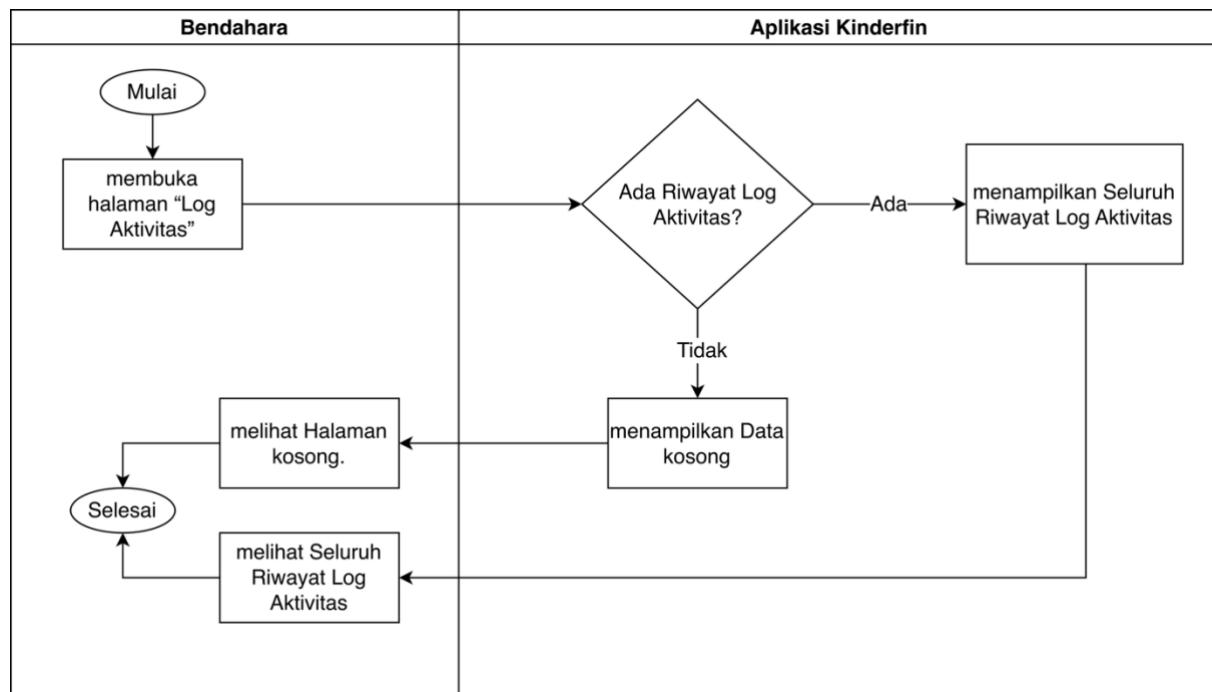
Gambar F.6 Diagram Aktivitas Setujui Perubahan Gaji

6. Kasus Penggunaan Lihat Log Aktivitas

Kasus Penggunaan Lihat Rincian Gaji yang dilakukan oleh Aktor Bendahara merupakan kasus penggunaan yang dibutuhkan untuk melihat aktivitas yang dilakukan oleh Bendahara maupun Administrator mengenai aktivitas penggajian. Alur akan dijelaskan dalam Tabel F.7 dan akan divisualisasikan dalam Gambar F.7.

Tabel F.7 Kasus Penggunaan Lihat Log Aktivitas

Komponen	Deskripsi
Nama	Lihat Log Aktivitas
Nomor	UC-011
Deskripsi	Aktor dapat melihat seluruh riwayat aktivitas mengenai sistem penggajian.
Aktor	Bendahara
Kondisi Awal	Sistem menampilkan halaman utama atau dashboard yang memiliki opsi untuk mengakses log aktivitas.
Kondisi Akhir	Aktor mengetahui isi seluruh Riwayat log aktivitas mengenai penggajian
Alur Normal	<ol style="list-style-type: none"> 1. Bendahara membuka halaman “Log Aktivitas” 2. Sistem memuat Seluruh Riwayat Log Aktivitas 3. Sistem menampilkan Seluruh Riwayat Log Aktivitas 4. Bendahara melihat Seluruh Riwayat Log Aktivitas.
Alur Alternatif	<ul style="list-style-type: none"> - Tidak ada Riwayat Log Aktivitas 3a. Sistem menampilkan Data kosong 4a . Bendahara melihat Halaman kosong.



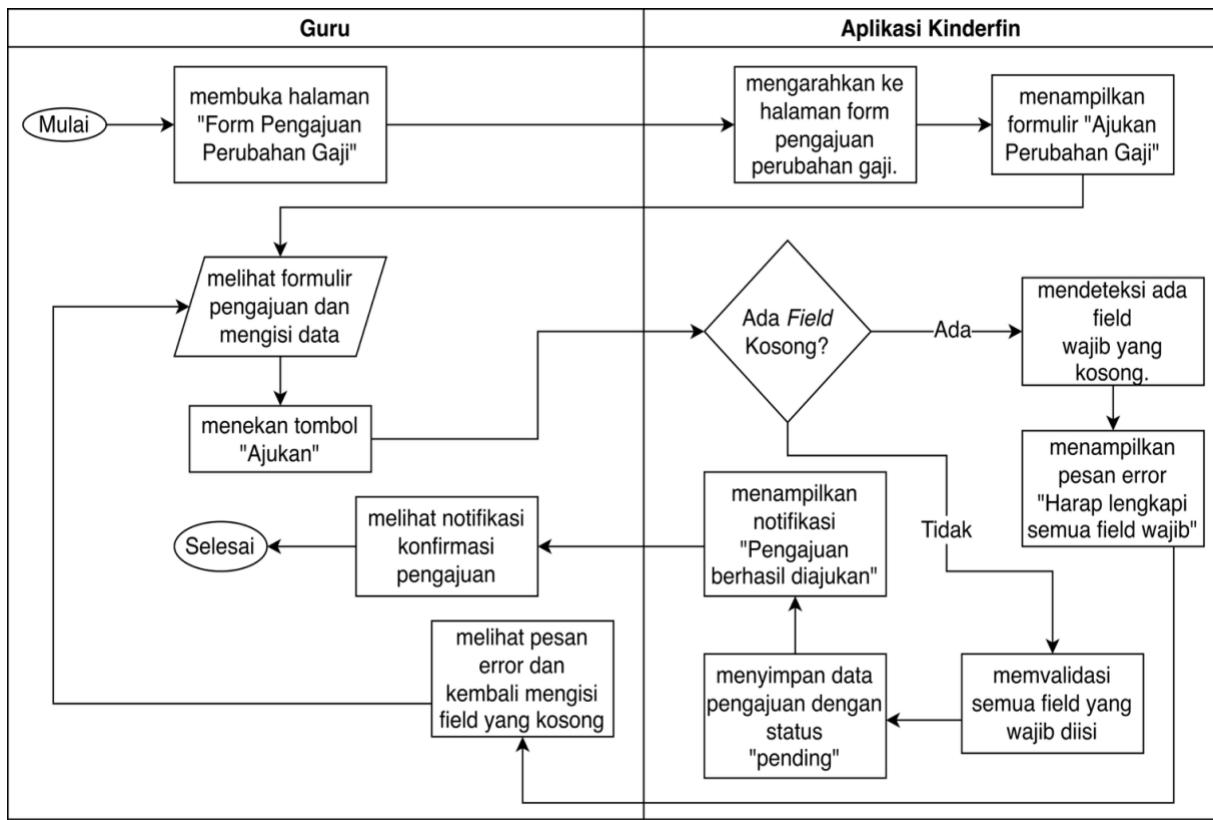
Gambar F.7 Diagram Aktivitas Penggunaan Lihat Log Aktivitas

7. Kasus Penggunaan Ajukan Perubahan Gaji

Pengajuan perubahan gaji merupakan hak yang dimiliki oleh guru karena ada gaji yang tidak sesuai. Pengajuan perubahan gaji tersebut akan diajukan dan akan dipertimbangkan oleh Bendahara dan dapat dilihat pada Kasus Penggunaan Setujui Perubahan Gaji. Setelah diajukan maka, akan ada status "pending" atau tertunda. Jika sudah disetujui, maka akan ada status "Disetujui" dan aktor serta tanggal persetujuan. Jika ditolak maka akan ada pesan "Ditolak" beserta alasan penolakan. Kasus Penggunaan Lihat Rincian Gaji Kasus ini akan dijelaskan pada Tabel F.8 dan divisualisasikan pada Gambar F.8.

Tabel F.8 Kasus Penggunaan Ajukan Perubahan Gaji

Komponen	Deskripsi
Nama	Ajukan Perubahan Gaji
Nomor	UC-013
Deskripsi	Guru mengajukan permohonan perubahan gaji karena adanya ketidaksesuaian data gaji yang diterima dengan yang seharusnya diterima
Aktor	Guru
Kondisi Awal	Guru telah login ke sistem dan menemukan adanya ketidaksesuaian pada data gaji yang diterima. Guru mengajukan pengajuan perubahan gaji dan mengisi formulir pengajuan perubahan gaji.
Kondisi Akhir	Pengajuan perubahan gaji telah berhasil di <i>submit</i> ke sistem dengan status "pending" dan menunggu persetujuan dari Bendahara (UC-009)
Relasi	Include: UC-009 (Setujui Perubahan Gaji)
Alur Normal	<ol style="list-style-type: none"> 1. Guru membuka halaman "Formulir Pengajuan Perubahan Gaji" 2. Sistem mengarahkan ke halaman form pengajuan perubahan gaji. 3. Sistem menampilkan formulir "Ajukan Perubahan Gaji" 4. Guru melihat formulir pengajuan dan mengisi data 5. Guru menekan tombol "Ajukan" 6. Sistem memvalidasi semua <i>field</i> yang wajib diisi 7. Sistem menyimpan data pengajuan dengan status "pending" 8. Sistem menampilkan notifikasi "Pengajuan berhasil diajukan" 9. Guru melihat notifikasi konfirmasi pengajuan 10. Include UC-009: Sistem mengirim notifikasi ke Bendahara untuk proses persetujuan 11. Guru menekan tombol "Kembali" untuk kembali ke halaman utama 12. Sistem menampilkan halaman utama 13. Guru melihat halaman utama
Alur Alternatif	<ul style="list-style-type: none"> - Ada formulir yang kosong <ul style="list-style-type: none"> 6a. Sistem mendeteksi ada <i>field</i> wajib yang kosong. 8a. Sistem menampilkan pesan error "Harap lengkapi semua <i>field</i> wajib" 9a. Guru melihat pesan error dan kembali mengisi <i>field</i> yang kosong



Gambar F.8 Diagram Aktivitas Ajukan Perubahan Gaji

LAMPIRAN G

Arsitektur *frontend* sistem KinderFin 2.0 mengadopsi pendekatan pengembangan yang konsisten pada berbagai modul tampilan antarmuka untuk mengoptimalkan proses pengembangan sekaligus memastikan konsistensi pengalaman pengguna dan efisiensi pemeliharaan kode. Implementasi frontend ini secara khusus *mereplikasi* program dan tampilan dalam proyek sebelumnya yaitu KinderFin 1.0 (Puspa & Putra, 2024).

1. Penerapan Fungsionalitas Umum pada Semua Modul

Terdapat beberapa fungsi serupa yang digunakan untuk membangun *frontend* di semua modul, sehingga akan dijelaskan secara terpadu dengan perbedaan spesifik yang akan diuraikan di bawah ini.

a. Pengelolaan *State* Lokal

Setiap modul mengelola *state* lokalnya untuk menyimpan data yang akan ditampilkan, mengontrol status antarmuka (misalnya, *loading* data, status buka atau tutup modal), dan memberikan *feedback* kepada pengguna (pesan sukses atau *error*). ini, memungkinkan antarmuka bereaksi terhadap perubahan data dan interaksi pengguna. Meskipun nama *variabel state* sedikit berbeda tergantung konteks modul, prinsip pengelolaan *state* data (data, *logs*, *form*), status operasional (*loading*, *uploading*), dan notifikasi (*success*, *error*) adalah sama. Kode Sumber G.1 menerapkan Inisialisasi *state* menggunakan *useState* untuk variabel yang menyimpan data aplikasi, status UI, dan pesan *feedback*.

```
// Contoh dari pengaturan_gaji/index.tsx (State Notifikasi & User)
1. const [user, setUser] = useState<any>(null); // Baris 4
2. const [success, setSuccess] = useState(''); // Baris 5
3. const [error, setError] = useState(''); // Baris 6
4.
// Contoh dari master_jabatan/index.tsx (State Data & Loading)
5. const [data, setData] = useState<any>([]); // Baris 9
6. const [loading, setLoading] = useState(true); // Baris 10
7.
// Contoh dari potongan_keterlambatan/tambah.tsx (State Form)
8. const [form, setForm] = useState({ // Baris 3
9.   jabatan: '',
10.  urutan_gaji_dipotong: 1,
11.  persen_potong: 0,
12.});
```

Kode Sumber G.1 Contoh *Code Pengelolaan State* Lokal

b. Penanganan Autentikasi dan Pengambilan Data Pengguna Awal

Hampir semua modul memerlukan validasi sesi pengguna dan informasi pengguna yang sedang login (seperti *access_token* atau *role*) untuk tujuan otorisasi, menampilkan data yang relevan, atau menyesuaikan fungsionalitas. Proses ini umumnya melibatkan pembacaan *cookie* yang menyimpan data pengguna saat komponen pertama kali dimuat. Jika token akses diperlukan, panggilan API pertama sering kali bergantung pada ketersediaan data pengguna ini. Kode Sumber G.2 menerapkan sebuah *useEffect* yang dijalankan di awal *lifecycle* komponen untuk mengambil dan mem-parsing data pengguna dari *cookies* menggunakan *Cookies.get()*, lalu menyimpan hasilnya ke *state user*.

```
1. // Contoh dari master_jabatan/index.tsx
2. useEffect(() => { // Mengambil data pengguna
3.   const userCookie = Cookies.get('user');
4.   setUser(userCookie ? JSON.parse(userCookie) : null);
5. }, []); //
```

Kode Sumber G.2 *Code Penanganan Autentikasi dan Pengambilan Data Pengguna Awal*

c. Pengambilan Data dari Server

Setelah pengguna terautentikasi dan informasi dasarnya tersedia, setiap modul akan melakukan satu atau lebih panggilan API *asynchronous* untuk mengambil data spesifik yang relevan dengan fungsionalitas modul tersebut (misalnya, daftar jabatan, log aktivitas, detail gaji, konfigurasi gaji aktif). Panggilan ini sering kali dibungkus dalam sebuah fungsi *async* dan dipanggil di dalam *useEffect* yang memiliki ketergantungan pada *state user* (atau *user.access_token*). Kode Sumber G.3 menerapkan Blok *useEffect* untuk pengambilan data utama (di dalam fungsi *fetchData*) saat *state user* tersedia atau *dependency* lain berubah, menggunakan *fetch* untuk memanggil *endpoint* API dan memperbarui *state data*.

```
1. // Contoh dari master_jabatan/index.tsx
2. useEffect(() => { // Mengambil semua data jabatan
3.   const fetchData = async () => {
4.     if (!user?.access_token) return;
5.     try {
6.       const res = await fetch(utils.get_all_jabatan, { /* ... */ });
7.       // ...
8.       setData(res.json.data); //
9.     } catch (err) { /* ... */ }
10.    finally { setLoading(false); } // 
11.  };
12.  fetchData();
13. }, [user]); //
```

Kode Sumber G.3 Contoh *Code Pengambilan Data dari Server*

d. Navigasi Halaman

Semua modul menggunakan *hook useRouter* dari *Next.js* untuk memungkinkan navigasi programatik antar halaman. Fungsionalitas ini penting untuk mengarahkan pengguna kembali ke halaman sebelumnya, menuju halaman detail setelah memilih item, atau ke halaman penambahan data baru setelah suatu operasi berhasil diselesaikan. Kode Sumber G.4 menerapkan Inisialisasi variabel *router* menggunakan *useRouter()* dan penggunaan metode *router.push()* untuk mengarahkan pengguna ke *path URL* yang ditentukan.

```
1. // Contoh dari pengaturan_gaji/index.tsx
2. const router = useRouter();
3. ...
4. <Button onClick={() => router.push("/")}>Kembali</Button>
```

Kode Sumber G.4 Contoh *Code Navigasi Halaman*

2. Bagian yang Membedakan Antar Modul (Fungsionalitas Inti Spesifik)

Meskipun banyak kesamaan dalam struktur dasar, setiap modul memiliki fungsionalitas inti dan logika bisnis yang unik, yang membuat setiap modul berbeda secara signifikan dan mencerminkan tujuan spesifik masing-masing dalam sistem. Pada setiap modul banyak kesamaan dalam struktur dasar, setiap modul memiliki fungsionalitas inti dan logika bisnis yang unik, yang membuat setiap modul berbeda secara signifikan dan mencerminkan tujuan masing-masing modul tersebut. Berikut adalah rincian perbedaan spesifik setiap modul.

a. Pengaturan Gaji Aktif (*pengaturan_gaji/index.tsx*)

Kode Sumber G.5 fokus pada konfigurasi *field* gaji yang dinamis di seluruh sistem. Keunikan terletak pada implementasi validasi anti-duplikasi label secara *real-time* dan logika penghapusan *field* yang membedakan antara *field* yang baru dibuat (dihapus lokal) dan *field* yang sudah ada (dihapus via API).

```
// Validasi anti-duplikasi label real-time
1. const handleLabelChange = (fieldId: string, value: string) => {
2.   // ... implementasi anti-duplikasi
```

```

3.    };
4.    // Logika penghapusan field (lokal vs API)
5.    const handleDeleteField = async (fieldId: string) => {
6.      // ... implementasi penghapusan field

```

Kode Sumber G.5 *Code Validasi dan Penghapusan Field Gaji*

b. Halaman Utama Master Jabatan (*master_jabatan/index.tsx*)

Kode Sumber G.6 berfungsi sebagai halaman daftar dan ringkasan untuk *master* data jabatan. Fungsi inti meliputi menampilkan semua jabatan dalam tabel, dan menyediakan aksi CRUD dasar (lihat detail, navigasi ke edit, dan penghapusan) langsung dari daftar.

```

// Fungsi hapus jabatan
1.  const handleDelete = async () => { // Fungsi hapus
2.    // ... logika fetch DELETE ...
3.  };

// Tombol aksi di tabel (contoh untuk hapus)
4.    <Button onClick={() => { setSelectedJabatan(item.jabatan);
setConfirmOpen(true); }}> {/* */}
5.      <IconTrash /* ... */ />
6.    </Button>

```

Kode Sumber G.6 *Code Hapus Jabatan dan Tombol Aksi*

c. Menambah Jabatan (*master_jabatan/tambah.tsx*)

Kode Sumber G.7 merupakan formulir khusus untuk menambah data jabatan baru. Modul ini secara dinamis membangun *input* gaji (harian dan pokok) berdasarkan *field* aktif yang telah dikonfigurasi. Fitur spesifiknya adalah utilitas *formatting* angka yang diterapkan langsung pada *input* untuk memastikan format nilai gaji yang benar.

```

// Utilitas formatting angka ke ribuan
1.  const formatRibuan = (val: string | number) => {
2.    // ... implementasi format angka ke format ribuan (1,000)
3.  };

// Utilitas parsing string ribuan ke number
4.  const parseToNumber = (val: string): number => {
5.    // ... implementasi parsing string ke number
6.  };

// Contoh penggunaan formatRibuan pada TextInput gaji (di JSX, tidak tertera di
// snippet ini)
// {/* ... render TextInput dengan formatRibuan */}

```

Kode Sumber G.7 *Code Utilitas Formatting Angka*

d. Setiap Jabatan (*master_jabatan/rincian/[jabatan].tsx*)

Kode Sumber G.8 merupakan fungsi untuk melihat dan mengedit rincian gaji dari sebuah jabatan spesifik yang dipilih. Fitur utamanya adalah formulir edit yang muncul dalam modal, menggunakan komponen *NumberInput* dengan validasi nilai minimum nol untuk setiap komponen gaji, dan fungsi *handleSave* untuk memperbarui data gaji jabatan.

```

// Fungsi menyimpan perubahan gaji jabatan
1.  const handleSave = async () => { // Menyimpan perubahan
2.    // ... validasi nilai negatif pada modalData ....tsx]
3.
4.    const res = await fetch(`<span class="math-
inline">\{utils\}.put\`jabatan\`/\</span>\{jabatan\}` , { // .tsx]
5.      method: "PUT",
6.      body: JSON.stringify(modalData), // .tsx]
7.      // ...
8.    });
9.    // ...
10.   };

// Contoh NumberInput dengan validasi min=0 di modal edit

```

```

11.      <NumberInput
12.        label={f.label} // .tsx]
13.        value={modalData[f.field] || 0} // .tsx]
14.        onChange={(val) => handleChange(f.field, val)} // .tsx]
15.        min={0} // .tsx]
16.        // ...
17.      />

```

Kode Sumber G.8 *Code Fungsi Simpan dan Validasi Input Gaji*

e. Daftar Potongan (*potongan_keterlambatan/index.tsx*)

Kode Sumber G.9 berfungsi untuk menampilkan daftar aturan potongan gaji yang spesifik terkait kehadiran (datang terlambat, pulang cepat, tidak presensi masuk atau pulang). Fitur khasnya adalah pengelompokan aturan berdasarkan nama jabatan dalam tabel, yang meningkatkan keterbacaan daftar yang panjang.

```

// Fungsi untuk mengelompokkan/membuka bagian jabatan di tabel
1. const toggleRoleSection = (role: string) => {
2.   // ...
3. };

// Struktur tabel dengan pengelompokan berdasarkan role/jabatan
4.           {Object.keys(expandedRoles).map((role) => (
5.             <tbody key={role}>
6.               {/* ... (Baris jabatan) ... */}
7.
8.               {expandedRoles[role] &&
9.                 groupedData[role].map((item: any, idx: number) => (
10.                   <tr key={`${<span class="math-inline">\{role\}\`}-`+idx}>
11.                     <td>{item.jabatan}</td>
12.
13.                     <td>{getLabelByUrutanGaji(item.urutan_gaji_dipotong)}</td>
14.                     <td>{item.persen_potong}%</td>
15.                     <td>{item.batas_menit === 0 ? '-' :
` ${item.batas_menit} menit`}</td>
16.                     <td>{labels.find((label) => label.value ===
item.jenis_potongan)?.label || item.jenis_potongan}</td>
17.                     <td style={{ textAlign: 'center' }}>
18.                       /* ... (Tombol hapus) ... */
19.                     </td>
20.                   )})
21.                 </tr>
22.               )})
23.             </tbody>

```

Kode Sumber G.9 *Code Fungsi Pengelompokan Aturan Potongan Berdasarkan Jabatan*

f. Tambah Potongan Gaji Keterlambatan (*potongan_keterlambatan/tambah.tsx*)

Kode Sumber G.10 merupakan formulir khusus untuk membuat aturan potongan gaji baru. Keunikannya terletak pada kondisionalitas *input* "Batas Menit", di mana *input* ini hanya relevan dan muncul jika jenis pelanggaran yang dipilih adalah "Datang Terlambat" atau "Pulang Cepat".

```

// Logika penyesuaian batas_menit saat submit
1.   jenis_potongan: form.tipe_jam,
2.   batas_menit: ['tidak_absen_masuk',
'tidak_absen_pulang'].includes(form.tipe_jam) ? 0 : form.batas_menit,
3.
// Kondisional rendering input "Batas Menit" di JSON
4.   {form.tipe_jam !== 'tidak_absen_masuk' && form.tipe_jam !==
'tidak_absen_pulang' && (
5.     <NumberInput label="Batas Menit" /* ... */ />
)

```

Kode Sumber G.10 *Code Logika Kondisional Input Batas Menit*

g. Halaman Utama Log Aktivitas (*activity_log/index.tsx*)

Kode Sumber G.11 pada modul *Activity Logs* bersifat *read-only* dan berfungsi murni untuk menampilkan riwayat log aktivitas pengguna dalam sistem. Tidak ada fungsionalitas

penambahan, pengeditan, atau penghapusan data, hanya tampilan data historis aktivitas yang detail.

```
// Bagian Tbody yang menampilkan log aktivitas
1.           {/* <Table.Tbody>
2.             {logs.length > 0 ? (
3.               logs.map((item, idx) => (
4.                 <Table.Tr key={idx}>
5.                   <Table.Td>{/* ... format tanggal item.created_at *
6. */}</Table.Td> {/* */
7.                   <Table.Td>{item.email}</Table.Td> {/* */
8.                   <Table.Td>{/* ... format item.action */}</Table.Td> {/* */
9.                   <Table.Td>{item.module}</Table.Td> {/* */
10.                  <Table.Td>{/* ... parsing dan format item.description */
11. */}</Table.Td> {/* */
12.                )}:
13.                // ... Tampilan jika tidak ada log
14.              )
15.            </Table.Tbody> */}
```

Kode Sumber G.11 *Code* Tampilan Tabel Log Aktivitas

h. Halaman Daftar dan Manajemen Pengajuan Perubahan Gaji
(*pengajuan_perubahan_gaji/daftar/index.tsx*)

Kode Sumber G.12 berfokus pada menampilkan daftar pengajuan perubahan gaji dan memungkinkan manajemennya. Fitur pada Kode Sumber G.12 adalah mekanisme persetujuan atau penolakan pengajuan yang diatur oleh hak akses pengguna (*canApprove* baris 3) dan dilengkapi dengan *input* alasan penolakan melalui modal.

```
// Fungsi untuk menyetujui pengajuan
1. const handleApprove = async (id: string) => { /* ... logika menyetujui
pengajuan ... */ };
// Fungsi untuk menolak pengajuan
2. const handleReject = async () => { /* ... logika menolak pengajuan ... */ };
// Penentuan hak akses persetujuan (state canApprove)
3. const [canApprove, setCanApprove] = useState(false);
```

Kode Sumber G.12 *Code* Fungsi Mekanisme Persetujuan atau Penolakan Pengajuan Gaji

i. Halaman Formulir Pengajuan Perubahan Gaji
(*pengajuan_perubahan_gaji/tambah.tsx*)

Perbedaan Utama: Merupakan formulir khusus untuk membuat pengajuan perubahan gaji baru. Keunikannya terletak pada penanganan pengunggahan dua *file* gambar (fotoBukti, fotoGaji baris 1-2) yang dikirim ke server menggunakan *FormData*, di samping input teks lainnya.

```
// State untuk file gambar yang diunggah
1. const [fotoBukti, setFotoBukti] = useState<File | null>(null);
2. const [fotoGaji, setFotoGaji] = useState<File | null>(null);
// Pengiriman form dengan FormData (termasuk file)
3. const formData = new FormData();
4. // ... append data ke formData ...
5. // ... logika mengirim pengajuan ke API
utils.post_pengajuan_perubahan_gaji ...
```

Kode Sumber G.13 *Code* Fungsi Pengunggahan *File* Gambar pada Formulir Pengajuan

BIODATA PENULIS



Penulis dilahirkan di Pekalongan, 12 November 1985, merupakan anak pertama dari 2 bersaudara. Penulis telah menempuh pendidikan formal yaitu di TK ABA Pekajangan Kab. Pekalongan, SD Muhammadiyah 02 Bendan Kota Pekalongan, SMPN 2 Pekalongan dan SMAN 1 Pekalongan. Setelah lulus dari SMAN tahun 2021, Penulis mengikuti Tes Mandiri ITS dan diterima di Departemen Teknik Informatika FTEIC - ITS pada tahun 2021 dan terdaftar dengan NRP 5025211225.

Di Departemen Teknik Informatika Penulis sempat aktif di beberapa kegiatan akademik dan non akademik, seperti Bangkit *Academy*, Pemandu FTEIC ITS 2023 dan Keluarga Muslim Informatika. Penulis juga pernah menjadi staf kampanye dan dokumentasi pada acara KPU HMTC, serta pernah menjadi pembicara pada acara KFU yang di SMAN 1 Pekalongan pada 2022 dan 2023. Penulis pernah kerja praktik di Rumah Sakit Petrokimia, Gresik pada semester 7 pada divisi IT selama 3 bulan.