BAB III

PELAKSANAAN KERJA MAGANG

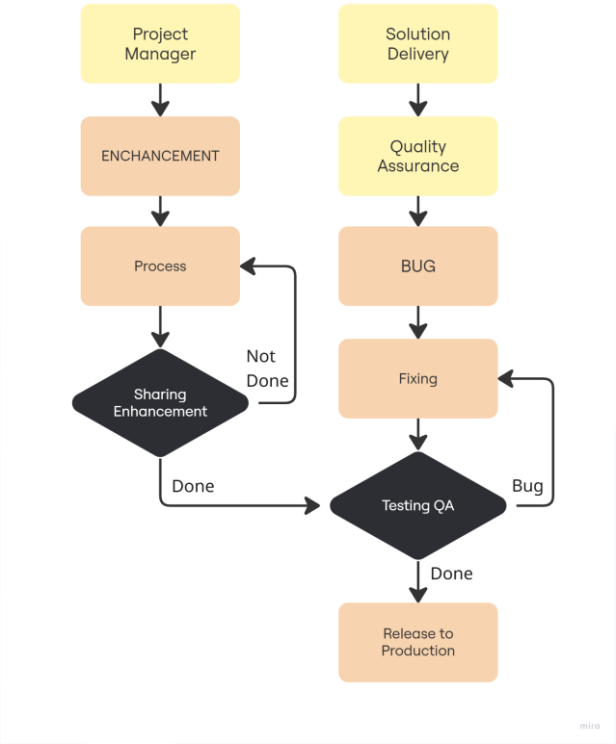
3.1 Kedudukan dan Koordinasi

Posisi penulis di PT Indodev Niaga Internet adalah sebagai *Software Developer Intern*, yang berada didepartemen *HR Development*, di tim *Payroll*, dan bertanggung jawab untuk *maintains Payroll module*, seperti *bug fixing* dan *enchancement*. Bapak Andry selaku *Project Manager* juga sebagai *supervisor* dan *mentor* dari *intern*. Kegiatan magang ini berlangsung dari 17 Maret sampai 16 September 2025 dengan waktu kerja 8 jam sehari selama 5 hari per satu minggu.

Pekerjaan meliputi *maintain* dua versi aplikasi *SunFish* yaitu, *SunFish 6* dan *SunFish 7*, yang keduanya berbasis *web* pada sisi *FrontEnd* dan *BackEnd*.

3.1.1 Alur Kerja Magang

Tugas atau pekerjaan yang akan diberikan dapat dilihat pada aplikasi internal perusahaan *SFsupport* ataupun melalui *SunFish 7* yang berisikan *list ticket* atau tugas. Utamanya ada dua jenis tugas yang dikerjakan yaitu, BUG (*Bug Fixing*) dan ENC (*Enhancement*). Untuk komunikasi antar tim dapat dilakukan via *Telegram* atau *GreatDay HR*, *platform virtual meeting* internal perusahaan, biasanya digunakan untuk *sharing enhancement*. Diagram alur magang dapat dilihat pada gambar 3.1.



Gambar 3.1 Alur Pengerjaan Tiket

Ticket yang dikategorikan sebagai *Bug Fixing* (BUG) dibuat oleh *Quality Assurance* atau *Solution Delivery*. Jika terdapat laporan *bug* dari *client* maka *Solution Delivery* yang akan membuat *ticket* atau tugas tersebut. Sama halnya jika terdapat *bug* dari hasil uji pada dilakukan *Quality Assurance* maka *ticket* juga akan dibuat dan diberikan kepada *Assignee*.

Ticket yang dikategorikan sebagai *Enhancement* (ENC), dibuat oleh *Project Manager*, diberikan ketika terdapat fitur atau perubahan fitur yang disampaikan oleh *client*. Sebelum *feature* ini akan diterapkan di *production*, akan diadakan sesi *virtual meeting* untuk *Sharing Enhancement* yang dikerjakan.

Setiap *ticket* memiliki kode statusnya, ketika *ticket* pertama kali dibuat maka *initial* statusnya adalah *open*. Lalu penerima *ticket* akan *set* status ke *Responded* dan mengerjakan pekerjaan. Jika sudah selesai maka status *ticket* adalah *Fixed* dan akan diverifikasi oleh *Quality Assurance*, jika masih terdapat *bug* atau

kesalahan fungsionalitas maka status menjadi *Reopen*. Selesai tahap verifikasi *status* tiket berubah menjadi *Closed*.

Untuk SF6 digunakan aplikasi *SunFish Cola* untuk melakukan *read/write* ke *module* aplikasi SF6 yang berbasis *ticket* untuk mendapat akses *read/write*. Setelah mendapatkan akses *file* atau *module* tersebut dikunci agar hanya ada satu pengguna pada satu waktu yang melakukan *write operation*. Status tiket *Closed* sebelum di *release* dilakukan *patch* untuk tahap pengujian lebih lanjut, memastikan stabilitas dan mitigasi kerentanan.

SF7 menggunakan *platform GitLab* untuk mengatur pengembangan *FrontEnd* dan *Gitea* untuk bagian *BackEnd*. Dua *platforms* ini digunakan untuk sinkronisasi *modules*, *remote repository* dan kolaborasi dengan anggota yang lain. Status tiket *Closed*, setelah dilakukan *merge* ke *testing branch* dan dikonfirmasi oleh QA, maka perubahan dapat diterapkan (*merge*) untuk *development branch* dan *bug-fixes* atau *enc branch*.

3.2 Tugas dan Uraian Kerja Magang

Selama menjadi *Software Developer Intern*, penulis bertugas untuk *maintaining* dan *enhancing code base SF6* dan *SF7*, menguji hasil komponen, dan analisis *error* atau *bug* yang mungkin ada. Detail *timeline* magang dapat dilihat pada tabel 3.1

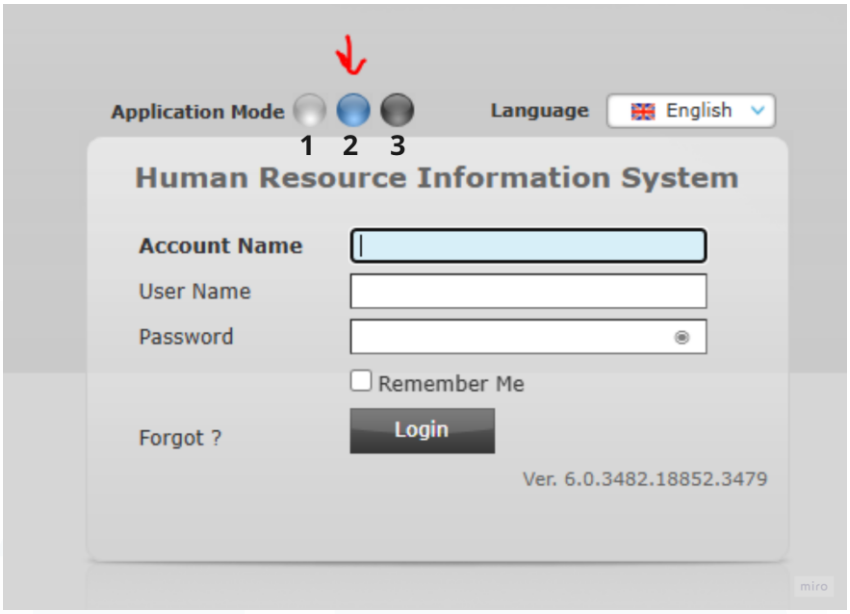
Tabel 3.1 Timeline Magang

Module	Pekerjaan	Minggu ke-	Tanggal Mulai	Tanggal Selesai
Training	*Onboarding* dan *Hands-on SF7* dan *SF6*	1-2	17-3-2025	28-3-2025
SF6 – Jamsostek Report / BPJS TK	BUG: *Error Preview* BPJS TK *Report*	3-4	4-4-2025	10-4-2025
SF6 – Setting / Global Parameter	BUG: *Date format did not follow global param*	5	14-4-2025	15-4-2025
SF6 – Payroll Report / Employee Payroll Data	ENC: *Enhance data entry for tax deduction [TH]*	5-6	16-4-2025	23-4-2025
SF6 – Jamsostek Report / BPJS TK	BUG: *CFQuery* BPJS TK	6	21-4-2025	24-4-2025

SF6 – Payroll Report / Employee Salary Card	BUG: *Error Preview Emp Salary Card Annually*	6-8	25-4-2025	5-5-2025
SF7 – Payroll Report / Data Changes Log	BUG: *No Record when preview payroll data changes log*	8	6-5-2025	9-5-2025
SF7 [Vietnam] – Payroll Report / Tax Report	BUG: *Error Preview PIT Finalization and Declaration*	8-9	7-5-2025	15-5-2025
SF7 – Payroll / HitungPajak	ENC: *Update coretax calculation for* hitungpajak.dataon.com	9-10	16-5-2025	23-5-2025
SF7 – Payroll / BackPay Process	ENC: *Add modified date when salary change*	10-12	23-5-2025	2-6-2025

SunFish 6 sendiri dikembangkan dengan CFML untuk *user interface* dan *server application* menggunakan *Lucee* atau *ColdFusion*. Keduanya, *Lucee* dan *ColdFusion*, dibangun diatas *Java* berjalan sebagai *Java Servlet*, dan dapat diintegrasikan dengan *Java libraries*. Sedangkan untuk *SunFish 7*, penerus dari *SunFish 6*, menggunakan *FrontEnd libraries* yang lebih *modern*, *React*, dengan *BackEnd application* menggunakan *Lucee*.

Sebelum melakukan *debugging SunFish 6* diperlukan akses ke jaringan internal untuk membuka aplikasi melalui *server* di kantor. Pada menu *login* terdapat tiga opsi yang bisa dipilih, lihat gambar 3.2, *development* (Opsi 2), *testing* (Opsi 3), dan *release* (Opsi 1). Proses *debugging* dilakukan pada opsi *development*.



Gambar 3.2 *SunFish 6 Login Page*

Sebelum proses *debugging SunFish 7* diperlukan akses jaringan internal untuk melakukan *clone repository* atau *update* dari *Gitea* atau *GitLab* dan dijalankan secara lokal. *Ortus CommandBox* digunakan untuk memulai *local server* yang sudah *include Lucee server*. Atau dapat mengakses *host* dikantor, yaitu *sf7dev* (*Development*), *sf7qa* (*Testing*), dan *sf7rel* (*Release*). Akses *DataBase* di *server* kantor perlu koneksi jaringan internal untuk *SunFish* 6 dan 7 baik lokal ataupun dari *host* yang telah disediakan.

3.2.1 *Onboarding* dan *Hands-on* SF7 dan SF6

Magang dimulai pada tanggal 17 Maret 2025 dengan proses *Onboarding* di Nissi Bintaro Campus. Proses *Onboarding* dimulai pukul 09.00 sampai dengan 15.00 WIB. Kegiatan meliputi perkenalan PT Indodev Niaga Internet, *jobdesc Software Developer Intern*, aturan kantor, perkenalan dengan aplikasi *SunFish*, penggunaan aplikasi *SunFish 7*, *office tour*, perkenalan dengan *supervisor* dan anggota tim, perkenalan gedung kantor (Nissi Bintaro Campus).

Untuk minggu pertama, dari tanggal 18 Maret 2025 sampai dengan 21 Maret 2025, setup akun *SFCola* dan akun *Gitea* dan *GitLab*, *connect* jaringan internal, *clone repository* dengan *command line app* internal, dan belajar CFML, dokumentasi SF6, mempersiapkan *setup BackEnd* dan *FrontEnd* SF7.

Minggu kedua dari tanggal 24 Maret 2025 sampai dengan 28 Maret 2025, *hands-on* aplikasi SF7 dan SF6. Mengerjakan beberapa pekerjaan yang diberikan. Perkenalan dengan *debugging tools* seperti *devtools* di *browser* dan *setup postman*, *setup database*.

Tiga *local* API untuk SF7 dan SF6 yang ada di perusahaan diperkenalkan untuk keperluan seperti akses akun *client*, akses akun *DataBase client*, dan memberikan informasi terkait jenis *DataBase* yang digunakan dan *runtime environment* yang digunakan. Berikut *list public* API:

- *Server_Host/index.cfm? code=generate account_name={X}* : Untuk mengakses akun *client / user* yang telah di *restore* ke lokal. Hasil dari API ini akan memberikan tiga *file* berupa *public key*, *private key*, dan *credential db* untuk *module BackEnd*. *X* variabel nama akun *client*. [SF7]
- *Server_Host/index.cfm? code=system & param=dbdriver* : API ini digunakan untuk memeriksa jenis DB yang digunakan. [SF6 & SF7]
- *IP_Local_SFCola/sf6cola/index.cfm?param=db&account_name={X}* : API ini digunakan untuk memeriksa *credential* dari *database* milik *client*. *X* variabel nama akun *client*. [SF6 & SF7]

3.2.2 BUG: *Error Preview BPJS TK Report*

Detail *module* dan *issue* yang dikerjakan dapat dilihat pada tabel 3.2.

Tabel 3.2 *Module* dan *Issue* BPJS TK

Module	*SF6 – Jamsostek Report / BPJS TK*
Issue	Isu dilaporkan oleh *client* AR kepada *Solution Delivery* dalam bentuk tiket kategori *bug*. Isu muncul pada menu BPJS TK *Report* untuk periode 2024 dari bulan januari sampai desember. Ketika *user* melakukan *preview* halaman *pop up* memerlukan *loading* yang lama (±10 menit), layar kosong, lalu terdapat *alert* bahwa data tidak ada. Setelah diperika data untuk periode tersebut tidak kosong.

Proses dimulai dengan login ke akun *client* yang telah di *restore* ke lokal *server* kantor dan masuk menu *BPJS TK Report*. Selanjutnya *debugging* dimulai dengan melihat API yang di *request* pada menu *BPJS TK Report*, menggunakan *devtools browser*. Dari API tersebut param *qlid* berisikan nilai dengan strucktur nilai *{nama_object}.{nama_method}* pada menu ini nilai *qlid* adalah *BPJS_TK_Report.getData*, nama *object* disesuaikan dengan nama *file* yang akan diperbaiki pada kasus ini nama *file*-nya adalah BPJSTKReport dengan *extension cfc* (*cold fusion component*). Setelah mendapatkan *object* dan *method* yang dipanggil, selanjutnya dilakukan analisis *method* tersebut dengan *dump* hasil *query* atau pemanggilan ke *method* lain. Untuk melihat hasil dump, proses yang sama, untuk opsi dan input yang sama, untuk menghasilkan *error* dijalankan kembali.

Setelah dianalisa hasil dari *dump* ditemukan bahwa *query* yang dijalankan tidak memberikan *output* dan terjadi *timeout*. Informasi ini lalu disampaikan kepada *Senior Dev*, setelah berdiskusi dengan *Senior Dev* akar dari permasalahan disebabkan oleh *custom tag* yang lambat dalam melakukan *fetching* dari ke *database*. *Custom Tag* dimaksudkan untuk melakukan validasi ke setiap data. *Query* yang dijalankan juga memiliki *response time* yang lama, terjadi *timeout* dengan *limit* satu menit, dikarenakan jumlah data lebih dari 1000 dan adanya *subquery* yang memakan cukup banyak waktu.

Solusi yang dikerjakan dengan melakukan *query* terpisah untuk data yang perlu divalidasi, yaitu data terkait karyawan. Setelah data karyawan sukses divalidasi maka dengan *query* yang sebelumnya juga perlu dipecah menjadi *query* tersendiri untuk menghindari *query* didalam *query*, terutama *subquery* yang memiliki *response time* yang lama. Hasil dari *query* yang telah dipecah per bagian tersebut digabung kembali menjadi satu juga bersama dengan *query* terkait data karyawan.

Solusi yang diterapkan dijalankan kembali dengan *input* yang sama, *output* berhasil didapatkan, hal yang sama berlaku jika *input* yang diterapkan berbeda-beda. Untuk memastikan apakah *query* yang diperbaiki benar dan tidak ada kesalahan hasil perubahan diterapkan untuk *testing*, konfirmasi perubahan

kepada QA. QA telah mengkonfirmasi dan selanjutnya lapor kepada *supervisor* untuk *apply* ke *production*.

3.2.3 BUG: *Date format did not follow global param*

Detail *module* dan *issue* yang dikerjakan dapat dilihat pada tabel 3.3.

Tabel 3.3 *Module* dan *Issue* format Penanggalan

Module	*SF6 – Setting / Global Parameter*
Issue	Isu dilaporkan oleh *Quality Assurance*. *Error* terjadi ketika memasukkan *input* pada *form* tanggal, untuk *start date* dan *end date*, format yang ditampilkan tidak sesuai dengan *setting* pada *global parameter* setelah memilih tanggal melalui *icon calendar*.
Format Penanggalan	*Start Date* atau *Input Date* memiliki dua tipe format penanggalan: • Format: mm/dd/yyyy. Contoh 03/25/2025 • Format: dd/mm/yyyy. Contoh 25/03/2025 *End Date* atau *Output Date* memiliki tiga tipe format penanggalan: • Format: mm/dd/yyyy. Contoh 03/25/2025 • Format: dd/mm/yyyy. Contoh 25/03/2025 • Format: dd MM yyyy. Contoh 25 Mar 2025

Proses *debugging* pada *Setting module* dapat dilakukan pada *custom script*. *Custom Script* terdapat dimenu *Setting / System Setting / Script Management*. *Debugging* dimulai dengan membuka dokumen *custom script*, yang ditulis menggunakan *JavaScript*. Pada *script* yang ditulis terdapat potongan *code* yang mengakses format dari *global parameter setting*, variabel ini kemudian di *dump/print* melalui *browser console*, didapatkan hasilnya kosong.

Temuan ini segera disampaikan ke *Senior Dev* untuk mencari tahu *global variable* yang digunakan untuk *formatting*. Setelah diperbaiki tes pun dilakukan, hasil tes variabel mendapatkan nilai sesuai dengan yang ada di *setting*. Setelah itu dilakukan tes dengan memasukkan *input* secara acak,

ditemukan tanggal yang ditampilkan secara *default* memiliki format *mm/dd/yyyy*, hal ini dapat dibuktikan dengan mengubah *global parameter* menjadi format selain *mm/dd/yyyy*. Ketika tanggal seperti 02/03/2025 (02 Maret 2025) dipilih melalui *calendar* UI *output* yang ditampilkan menjadi 03/02/2025, ketika *icon calendar* diklik maka tanggalnya pun mengikuti hasil yang ditampilkan, yaitu 03 Februari 2025 yang seharusnya 02 Maret 2025.

Solusi yang kerjakan dengan memanipulasi DOM pada elemen HTML yang memberikan *output*, dalam kasus ini tag input itu sendiri. Manipulasi dilakukan dengan mengambil ID untuk elemen tersebut, masing-masing dari *start date* dan *end date*. Nilai yang telah di *input* oleh *user* akan diambil dan dilakukan *formatting* berdasarkan format dari *global parameter*.

Setelah dicoba dengan *input* 02/03/2025 sampai 04 Mar 2025, untuk *start date* dan *end date* masing-masingnya, hasil yang ditampilkan sesuai dengan format. Langkah selanjutnya mengkonfirmasikan ini dengan QA dan sekaligus *supervisor* untuk dilakukan *patch*.

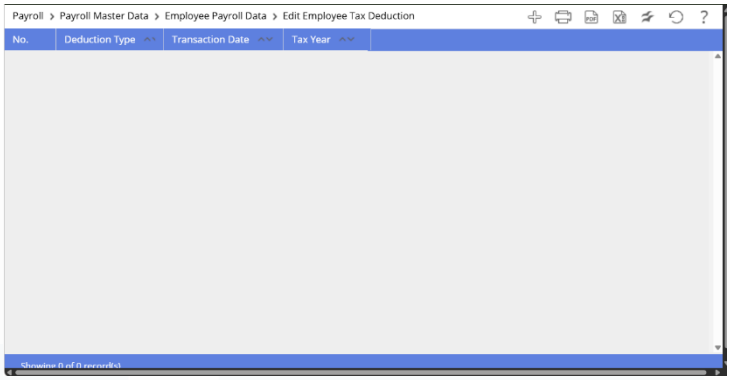
3.2.4 ENC: *Enhance data entry for tax deduction [TH]*

Detail *module* dan *feature description* dapat dilihat pada tabel 3.4.

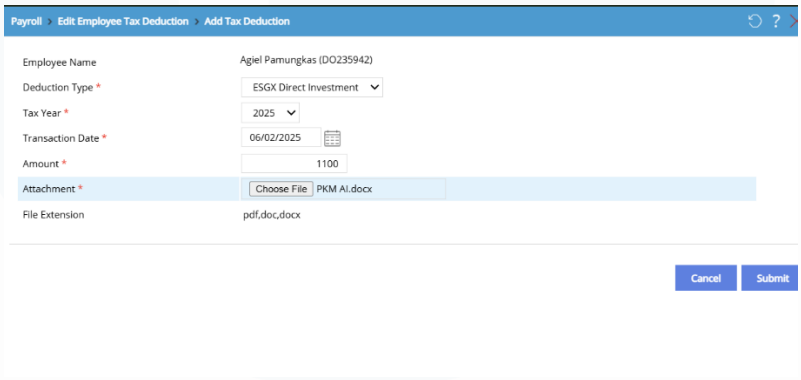
Tabel 3.4 *Module* dan *Tax Deduction Feature* untuk negara Thailand

Module	*SF6 – Payroll Report / Employee Payroll Data*
Feature Description	Penjelasan *feature* langsung disampaikan oleh *Project Manager*. Untuk *user* dari *Thailand* terdapat fitur tambahan dimana fitur ini berdasarkan pada regulasi pemerintahannya. Regulasinya terkait pengurangan pajak, dimana pendapatan akan dikurangin dengan nilai yang sudah ditetapkan. Pengurangan berdasarkan *item* yang dimiliki dan disesuaikan dengan aturan apakah termasuk kategori *deduction*. Hasil dari pengurangan yang akan masuk perhitungan pajak. Untuk perhitungan pajak *Thailand* terdapat tim pajak yang menangani hal ini. Penulis ditugaskan untuk menambahkan *pop up* menu dan *basic CRUD operation* untuk melakukan

	list, update, create, delete pada item yang masuk sebagai *tax deduction*.



Gambar 3.3 Tampilan *Listing Page*



Gambar 3.4 Tampilan *Create Page*

EMPSELFDEDUC	
PK	**deduction_id INT**
	emp_id VARCHAR(20)
	tax_year INT
	transaction_date DATE
	amount INT
	document_path VARCHAR(255)
	deduction_type VARCHAR(255)

Gambar 3.5 Tabel *EmpSelfDeduc*

Penambahan fitur dimulai dengan membaca deskripsi tiket yang diberikan beserta *module* yang akan ditambahkan fitur didalamnya. Penambahan elemen UI baru mengikuti contoh-contoh yang diberikan dan memakai *custom tag* dari *module* SF6. *Project Manager* juga melampirkan dokumen terkait *CRUD operation* dan *table* yang harus dibuat, *table* dapat dilihat pada gambar 3.5.

Fitur yang pertama *Adding/Create item* per karyawan, tampilan dapat dilihat pada gambar 3.4. Fitur ini dibuat dalam bentuk *form*. *User* dapat menambahkan *item* dan *upload* dokumen terkait pembelian *item* tersebut. Untuk bagian *backend* akan menerima data dari *user* ketika pengguna klik *submit button*. Untuk pembuatan atau penambahan API baru pada *coldfusion component* kita perlu menambahkan *method* baru, dengan nama *addNewDeduction* pada *file* atau *component* yang akan ditambahkan yaitu *EmployeePayrollData*, secara otomatis API baru akan ditambahkan. Lalu menambahkan *request* dengan *param qlid* dengan *value EmployeePayrollData.addNewDeduction* pada *submit button* dibagian *frontend* untuk *hit* API yang telah dibuat.

Fitur kedua *Listing item*, lihat pada gambar 3.3. Pada menu ini *item* yang berhasil di *submit* oleh *user* akan ditampilkan pada *pop up menu*, setiap *item* yang diklik akan menampilkan *pop up menu* kedua. Pada *pop up menu* pengguna dapat melakukan *Edit* dan juga *Delete item* tersebut ataupun *download* kembali dokumen yang di *upload*. *Edit*, *Delete*, dan *List* masing-

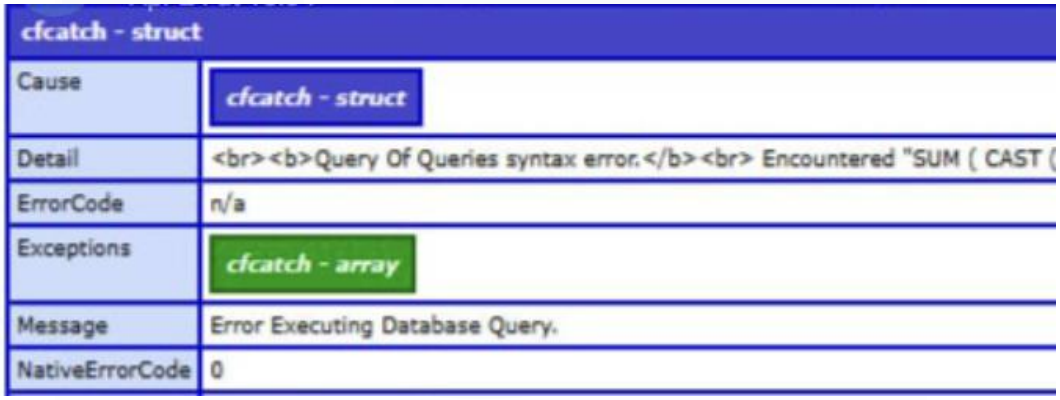
masing melakukan *request* ke *method editDeductionItem*, *deleteDeductionItem*, dan *listDeductionItem*.

3.2.5 BUG: *CFQuery* BPJS TK

Detail *module* dan *issue* yang dikerjakan dapat dilihat pada tabel 3.5.

Tabel 3.5 *Module* dan *Query Issue* BPJS TK

Module	*SF6 – Jamsostek Report / BPJS TK*
Issue	*Bug* ini dilaporkan dari *client* melalui *Solution Delivery*. *Error* timbul karena kesalahan *syntax* dan penggunaan fungsi yang hanya berjalan untuk MariaDB, MSSQL dan tidak untuk CFQuery.*Error message* dilampirkan dalam bentuk *file* gambar dan dapat dilihat pada gambar 3.6.



Gambar 3.6 Tampilan error message

Proses *debugging* dimulai dengan melakukan analisis pada *module* dan pengujian pada *mode development* SF6. *Debugging* dilakukan dengan *dump* hasil dari baris kode yang menghasilkan *error*. *Error* yang dihasilkan terjadi karena tipe dari kolom *Column_X* merupakan tipe *string* yang seharusnya tipe *number* untuk dilakukan operasi *sum*.

Solusi yang terapkan untuk *error* tersebut dengan menggunakan *CAST* secara langsung dari pada *query* tersebut. Dengan *query* seperti berikut:

```
SELECT SUM( CAST( column_x as NUMERIC ) )
```

Satu hari setelah *bug fixing* hal yang serupa terjadi dengan *error message* yang serupa pada akun (*user*) yang berbeda. Permasalahan ini didiskusikan dengan *supervisor*. *Supervisor* menyampaikan bahwa *Error* yang terjadi karena *Query of Queries Syntax Error*, kesalahan *syntax* pada *query* yang menggunakan *query built-in* dari *ColdFusion* atau *Lucee*.

Solusi yang diterapkan dengan melakukan *casting* sebelum kolom *colomn_x* digunakan untuk melakukan operasi pertambahan. Penerapan solusi sebagai berikut:

Query 1: SELECT CAST(column_x as NUMERIC) as column_y
Query 2: SELECT SUM(column_y)

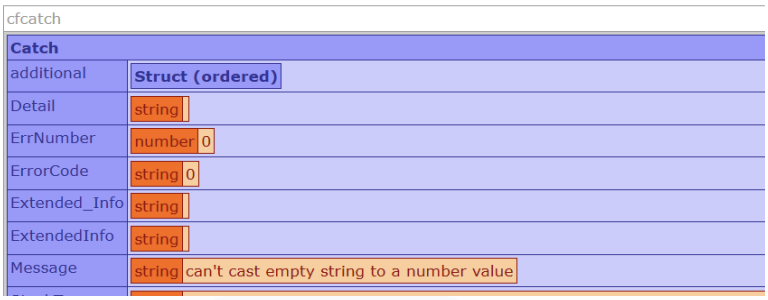
Query 2 dapat melakukan operasi *SUM* karena *column_y* sebelumnya telah dilakukan *casting* dari tipe *string* ke tipe *numeric*. Tiket kemudian dilakukan *testing* oleh QA dan *release* ke *production*, *user* tidak mendapatkan *error* setelah *bug fixing*.

3.2.6 BUG: *Error Preview Emp Salary Card Annually*

Detail *module* dan *issue* yang dikerjakan dapat dilihat pada tabel 3.6.

Tabel 3.6 Module dan Issue pada menu Emp Salary Card

Module	*SF6 – Payroll Report / Employee Salary Card*
Issue	*Bug* ini dilaporkan dari *client* melalui *Solution Delivery*. *Error* muncul ketika *client* ingin melakukan *preview* memilih periode Januari sampai Desember tahun 2024. *Error message* dapat dilihat pada gambar 3.7.



Gambar 3.7 *Error Message* pada *module EmpSalaryCard*

Proses *debugging* dimulai dengan *login* akun *development* dan melihat API yang di *request* melalui *dev tools*. Ketika melakukan hal yang sama untuk menghasilkan *error* dengan klik *preview button* dapat dilihat API melakukan *request* ke *param rpid* (*Report ID*) dengan *value* yaitu *EmpSalaryCard.getEmpSalary*. Setelah mengetahui nilai dari rpid maka dilakukan *dump* pada *method getEmpSalary* untuk setiap *query* yang dijalankan. Ditemukan bahwa terdapat beberapa *record* karyawan memiliki data yang kosong (*Empty String*), data ini kemudian akan dilakukan kalkulasi gaji karyawan. Namun permasalahan muncul ketika dilakukan kalkulasi pada data yang kosong dimana *empty string* di CFML tidak dapat langsung di *casting* ke tipe numerik. Juga ditemukan bahwa terdapat data karyawan yang seharusnya tidak terdaftar namun muncul ditabel.

Solusi yang digunakan dengan menambah *Query of Queries*. Dimana hasil dari *query* yang di *request* langsung ke *database* akan disimpan sebagai *recordset* didalam memori. *RecordSet* tersebut dapat kita lakukan operasi *query* seperti pada umumnya dengan limitasi klausa. Dapat dilihat pada gambar 3.8.

```
<cfquery name="LOCAL.qHeader" dbtype="query">
    SELECT *
    FROM qHeader
    WHERE (#PreserveSingleQuotes(filterqueryemp_auth_2)#)
</cfquery>
```

Gambar 3.8 Potongan kode *filter* hasil *query qHeader*

Pada gambar 3.8 solusi yang digunakan dengan melakukan *filter*, untuk hasil *query* yang melakukan pengambilan kolom data yang nantinya digunakan untuk kalkulasi, seperti gaji per bulan, pengurangan. *Filter* diterapkan pada

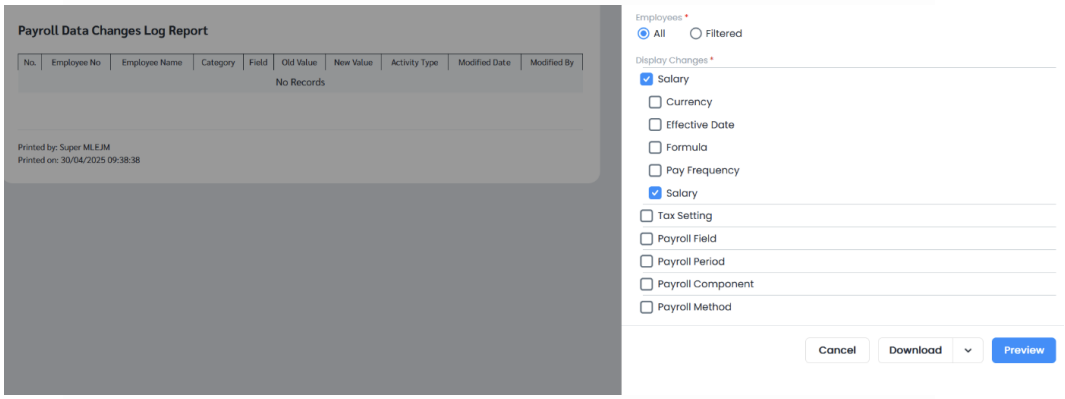
recordset qHeader dengan menggunakan *WHERE clause*. Klausa *WHERE* mencocokkan dengan *list* semua karyawan yang dipilih melalui *form* oleh *user*, baik itu yang juga dipilih langsung oleh *user* atau dengan memilih semuanya.

3.2.7 BUG: *No Record when preview payroll data changes log*

Detail *module* dan *issue* yang dikerjakan dapat dilihat pada tabel 3.7.

Tabel 3.7 *Module* dan *Issue* pada *menu payroll data changes log*

Module	*SF7 – Payroll Report / Data Changes Log*
Issue	*Bug* ini dilaporkan oleh *Solution Delivery*. *Issue* terjadi ketika *user* ingin *preview* data *payroll data changes log report* yang dimana hasilnya kosong. Dapat dilihat pada gambar 3.9. Namun ketika dilakukan pemeriksaan pada menu *data changes log* terdapat riwayat data.
Pre-Debugging	Proses *setup* dimulai dengan membuka *CommandBox* dan menjalankan *Lucee Server*. Kemudian melakukan *request* ke API *Local_Server_Host / index.cfm?sfid=generate & account_name=openML*. Setelah mendapatkan akun dari *client* dan berhasil login dengan *PostMan* (*Hit Login API*). Proses *debugging* dapat dimulai. API yang digunakan untuk login *server_host/?ofid=sfsystem.loginUser*. Juga *set environment* sebelum *login* untuk akun *client* seperti *user*, *account*, *password*, dan *company name*.



Gambar 3.9 Data tidak ada ketika di *submit*

Proses *debugging* pada SF7 dimulai dengan menganalisis API yang dituju ketika melakukan *preview*. Untuk proses *debug* di SF7 tidak jauh berbeda dengan SF6, namun terdapat tambahan *prefix* dari *param* dan nama *file* yang dituju. Nama *file* atau *module* ditambahkan SF (*SunFish*) dan dua huruf pertama dari *param* tersebut, contoh

Param qlid = *EmpSalary*.getData

Nama *module* = SFQLEmpSalary

Ditemukan bahwa ketika melakukan *request* ke *PayrollDataChangesLog.getData*, hasil dari *request* berupa data kosong. Kemudian kode untuk *method getData* dilakukan analisis dan *dump* hasil *query* yang dieksekusi dilokal. Pada kasus ini digunakan *PostMan* untuk melakukan *request* ke API yang sama dengan *payloads* yang sama. *Payloads* diambil pada saat melakukan pengujian dengan *input* yang sama untuk menghasilkan *error* melalui *sf7qa*.

CommandBox digunakan untuk menjalankan *Local Server Lucee*, melakukan perubahan pada *module* dan *debugging* secara lokal dengan *PostMan*.

Setelah menganalisa isi kode untuk *getData method*, ditemukan bahwa terdapat sebuah *key* pada *payloads* dengan format yang salah. Format untuk *key* A yang hanya diterima oleh *getData method* berbeda dengan yang didapatkan sebenarnya. Contoh, pada kode menerima *key* A sama dengan 'Salary', namun *paylaods* yang diterima pada *key* A sama dengan 'Salary-ID'

Solusi yang diterapkan dengan memperbaiki format *payloads* pada bagian *frontend* disesuaikan dengan format atau data yang diterima *backend*.

3.2.8 BUG: *Error Preview PIT Finalization and Declaration*

Detail *module* dan *issue* yang dikerjakan dapat dilihat pada tabel 3.8.

Tabel 3.8 *Module* dan *Issue* Laporan Pajak untuk negara Vietnam

Module	*SF7 [Vietnam] – Payroll Report / Tax Report*

Issue	*Bug* dilaporkan dari *client* asal *Vietnam*. *Issue* terjadi ketika *user* ingin melakukan *preview* pada menu *PIT Finalization Report* untuk *form 05 QTT TNCN* dan *05-1 BK QTT TNCN*.
Pre-Debugging	*Login* dengan akun *client*, untuk mendapatkan akses ke *database client* yang telah di *restore* ke *server* kantor. Dan jalankan *backend application* secara lokal untuk melakukan *debugging*.

Proses *debugging* dimulai dengan menghasilkan *issue* yang sama seperti yang dilaporkan dan dijalankan di *sf7dev* untuk mengetahui API yang dipanggil dan *payloads* yang dikirim. Setelah itu proses *dump* dapat dilakukan secara lokal dengan *payloads* dan *request* ke API yang sama.

Setelah melakukan *dump* ke *getReport method* untuk kedua *form* tersebut, ditemukan bahwa terdapat data dari hasil *query* yang kosong. Pengujian *query* langsung ke *database* juga menunjukkan data kosong, tabel ini berisi data salah satunya untuk perhitungan *PIT Finalization* (*Personal Income Tax*). Hasil temuan didiskusikan dengan *supervisor*, kemudian *supervisor* menambahkan langsung data tersebut ke *database client*.

Hari berikutnya *issue* lain terjadi pada *menu* yang sama. Setelah ditelusuri hasil dari perhitungan pada *calculateTax method* memberikan hasil *empty string*.

Solusi dengan ditambahkan *handling* untuk hasil ketika *empty string* menjadi nol dengan tipe numerik.

3.2.9 ENC: *Update coretax calculation for hitungpajak.dataon.com*

Detail *module* dan *feature description* dapat dilihat pada tabel 3.9.

Tabel 3.9 *Module* dan *Feature* hitungpajak.dataon.com

Module	*SF7 – Payroll / HitungPajak*
Feature Description	*Update* perhitungan pajak untuk *web* *hitungpajak.dataon.com*. Karena *web* *hitungpajak.dataon* terakhir *update* pada tahun 2019 dan tidak ada *update* setelah

	tahun berikutnya. Maka *project* ini diberikan untuk *redesign* tampilan dan *update* perhitungan sesuai dengan aturan PPh21 yang terbaru. *Web* ini selain digunakan untuk menghitung pajak perorangan juga digunakan untuk menghitung pajak untuk perusahaan, banyak karyawan.

Utamanya untuk *update* perhitungan PPh21 dibagi menjadi empat kategori, yaitu PPh21 Bulanan, PPh21 Final, PPh21 Tidak Final dan PPh21 Tahunan (A1/A2). Pembuatan *logic* perhitungan dibuat dengan *Lucee* dan mengikuti panduan PPh21 dari KemenKu (Kementrian Keuangan).

Berikut beberapa potongan kode untuk setiap perhitungan:

- Potongan kode PPh21 Bulanan, *gross up* lihat pada gambar 3.10.

```
case "gross_up":
    local.fn_cal_PPh_Gross_Up = (Number net_bruto, any tarif) => {
        return () => {
            return net_bruto * ( tarif / ( 100 - tarif ) );
        }
    }

    while (true) {
        local.func_calculate_PPh = local.fn_cal_PPh_Gross_Up(local.net_bruto_per_month, local.tarif);
        local.new_Net_Bruto = int(local.func_calculate_PPh()) + local.net_bruto_per_month;
        local.new_Tarif = get_PPh_21_PTKP(arguments.argEntries.ptkp, local.new_Net_Bruto);

        if( local.new_Tarif neq local.tarif ){
            local.tarif = local.new_Tarif;
        } else {
            local.net_bruto_per_month = local.func_calculate_PPh() + local.net_bruto_per_month
            break;
        }
    }
    local.pph21_total = local.func_calculate_PPh();
    break;
```

Gambar 3.10 Potongan kode PPh21 Bulanan

Pada kode ini dijalankan untuk pajak PPh21 Bulanan yang menggunakan *gross up*. Perhitungan tarif didasarkan pada TER Bulanan. *Looping* akan terus dijalankan dan melakukan kalkulasi berdasarkan tarif baru yang didapatkan dan akan berhenti ketika tarif yang sebelum dan sesudah nilainya sama.

- Potongan kode PPh21 *Final* lihat pada gambar 3.11.

```
// Define lapisan yang kena pajak dan tidak kena pajak (Rp 50.000.000 pertama tidak kena pajak, tarif = 0%)
local.fn_define_layer = (struct layer,Number net_bruto, function get_iteration, function get_tarif) => {
    local.nums_of_iteration = arguments.get_iteration(net_bruto);
    local.net_bruto = arguments.net_bruto; // Uang Pesangon
    local.temp_res = 0;
    for( i=1 ; i<=local.nums_of_iteration ; i++ ){
        if( (i eq 1) OR (i eq 2) ){
            local.temp_res = local.net_bruto - 50000000;
            layer["net"][i] = (local.temp_res <= 0 ? local.net_bruto : 50000000);
            local.net_bruto -= layer["net"][i];
        } else if( i eq 3 ){
            layer["net"][i] = (local.net_bruto > 400000000 ? 400000000 : (local.net_bruto) );
            local.net_bruto -= layer["net"][i];
        } else {
            layer["net"][i] = local.net_bruto;
        }

        layer["tarif"][i] = arguments.get_tarif(i);
        layer["tarifMax"] = arguments.get_tarif(i);
    }
};
```

Gambar 3.11 Potongan kode PPh21 *Final*

Kode ini akan dijalankan untuk PPh21 *Final* dengan kode objek pajak pesangon. Uang pesangon yang diterima akan dilakukan iterasi dan dihitung per lapisan, dimana setiap lapisan memiliki tarif progresif.

- Potongan kode PPh21 Tidak *Final* lihat pada gambar 3.12.

```
local.fn_define_layer = (struct layer, number net_bruto, function get_iteration, function get_tarif) => {
    local.nums_of_iteration = arguments.get_iteration(arguments.net_bruto);
    local.net_bruto = arguments.net_bruto;
    local.temp_res = 0;
    for( i=1 ; i<=local.nums_of_iteration ; i++ ){
        if( i eq 1 ){
            local.temp_res = local.net_bruto - 60000000;
            layer["net"][i] = local.temp_res <= 0 ? local.net_bruto : 60000000;
        } else if( i eq 2 ){
            layer["net"][i] = local.net_bruto > 190000000 ? 190000000 : local.net_bruto;
        } else if( i eq 3 ){
            layer["net"][i] = local.net_bruto > 250000000 ? 250000000 : local.net_bruto;
        } else if( i eq 4){
            layer["net"][i] = local.net_bruto > 4500000000 ? 4500000000 : local.net_bruto;
        } else {
            layer["net"][i] = local.net_bruto;
        }
        local.net_bruto -= layer["net"][i];
        layer["tarif"][i] = arguments.get_tarif(i);
        layer["tarifMax"] = arguments.get_tarif(i);
    }
};
```

Gambar 3.12 Potongan kode PPh21 Tidak *Final*

Kode ini dijalankan untuk PPh21 Tidak *Final*. Perhitungan dilakukan secara iterasi dengan tarif progresif.

- Potongan kode PPh21 Tahunan lihat pada gambar 3.13.

```
local.fn_cal_PPh_terutang = (struct layer, number net_bruto, function get_iteration) => {
    return () => {
        local.nums_of_iteration = get_iteration(net_bruto);
        local.total_pajak = 0;
        for( i=1 ; i<=local.nums_of_iteration ; i++ ){
            local.total_pajak += ( layer["tarif"][i] / 100 ) * layer["net"][i];
        }
        return local.total_pajak;
    };
};
```

Gambar 3.13 Potongan kode PPh21 Tahunan

Kode ini dijalankan untuk PPh21 untuk dua kategori, setahun dan disetahunkan. Kode ini akan melakukan iterasi dan melakukan operasi perkalian tarif dengan penghasilan neto dan kemudian dijumlahkan total PPh 21 terutang.

3.2.10 ENC: *Add modified date when salary change*

Detail *module* dan *feature description* dapat dilihat pada tabel 3.10.

Tabel 3.10 *Module* dan *Feature* pada *menu backpay process*

Module	*SF7 – Payroll / BackPay Process*
Feature Description	*Client* AA ingin menambahkan fitur yang nantinya mereka dapat panggil, melalui API dari *SunFish 7*, untuk melakukan kalkulasi ulang ketika *salary* satu atau lebih karyawan yang berubah nilainya, kalkulasi ulang untuk *backpay*. Untuk kalkulasi ulang dengan menggunakan kode dan *logic* yang sama dan API telah disediakan. Namun dibutuhkan sebuah *trigger* untuk mengetahui apakah *salary* terjadi perubahan atau tidak. Untuk itu tiket ini dibuat agar proses *trigger* dapat terjadi.

Proses penambahan *functionality* dimulai dengan *setup*, menjalankan *frontend* dan *backend* secara lokal, Memahami struktur kode dan *logic* yang dijalankan. Fitur yang pertama kali dibuat dalam proses pengerjaan dengan menambahkan kolom *modified_salary_date* pada tabel *backpay* untuk mencatat perubahan *salary* setiap karyawan ketika dilakukan *update*.

Langkah selanjutnya dengan menambahkan sebuah *trigger logic* ketika terjadi *update* pada *salary* untuk mengubah *value* kolom *modifieds_salary_date* dengan nilai pada waktu perubahan terjadi. *Trigger* juga ditambahkan *conditional statement* untuk menjaga efek *trigger* terjadi jika dan hanya jika nilai *salary* berbeda dengan sebelumnya. Nilai yang sebelumnya akan masuk ke tabel *backpay_history*. *Trigger Logic* juga berlaku untuk perubahan *salary* pada tabel *backpay_history*.

Penerapan *trigger logic* yang dibuat juga diimplementasikan ketika *upload file* untuk memasukkan data karyawan dan gajinya. *Logic* yang sama juga diterapkan untuk data gaji karyawan yang mengalami perubahan baik itu untuk data gaji pada tanggal yang terbaru (Pada tabel *backpay*) atau gaji untuk tanggal yang sebelumnya (Pada tabel *backpay_history*).

3.3 Kendala yang Ditemukan

Selama magang di PT Indodev Niaga Internet sebagai *Software Developer*, berikut beberapa kendala yang ditemukan, meliputi:

- **Deskripsi tugas kurang detail**

  Instruksi atau deskripsi tugas pada tiket yang diberikan terkadang kurang lengkap dan cukup sulit untuk dipahami secara langsung. Hal ini umumnya terjadi karena tiket-tiket tersebut, terutama yang berasal dari tim Solution Delivery, awalnya ditujukan untuk ditangani oleh senior developer atau project manager. Setelah itu, barulah tugas tersebut didistribusikan kepada intern untuk dikerjakan. Alur ini menyebabkan beberapa informasi teknis atau konteks penting tidak selalu tersampaikan secara menyeluruh kepada intern, sehingga dibutuhkan inisiatif lebih dalam memahami kebutuhan dan ruang lingkup pekerjaan dari tiket tersebut.

- **Kurangnya pengalaman *maintainability***

  Penulis menyadari bahwa kurangnya pengalaman dalam melakukan pemeliharaan kode, proses debugging, serta adaptasi terhadap berbagai *tools* yang digunakan dalam proses tersebut menjadi tantangan tersendiri selama

masa magang. Selain itu, keterbatasan praktik dalam menulis kode yang bersih, terbaca (*readable*), dan *maintainable* turut memengaruhi efektivitas dalam memahami alur sistem dan melakukan perbaikan pada modul yang kompleks.

3.4 Solusi atas Kendala yang Ditemukan

Solusi atas kendala yang dialami selama magang, yaitu:

- **Proaktif**

  Untuk memahami deskripsi tiket yang kurang jelas, penulis akan terlebih dahulu mengonfirmasi langsung kepada *supervisor* atau *senior developer* guna memperoleh penjelasan yang lebih rinci. Selain itu, apabila penulis menghadapi kendala dalam proses *debugging*, khususnya saat menemukan *bugs* atau *errors* yang sulit ditemukan, maka penulis juga akan meminta arahan dari *supervisor* atau *senior developer* agar dapat menyelesaikan permasalahan dengan pendekatan yang lebih tepat dan efisien.