# BAB III ANALISIS DAN PERANCANGAN

## Analisis

### Identifikasi Masalah

Identifikasi masalah yang memiliki kekurangan atau kelemahan sistem saat ini dapat dibagi menjadi pokok-pokok permasalahan antara lain:

1. Aplikasi Space Venturo saat ini belum memiliki sebuah fitur untuk membuat list competency sehingga Human Resource Development mencatat competency mengggunakan excel.
2. Aplikasi Space Venturo saat ini belum memiliki sebuah fitur untuk mengajukan Performance Development Review sehingga proses penilaian dan dokumentasi Performance Development Review dilakukan secara manual.
3. Aplikasi Space Venturo saat ini belum memiliki fitur untuk menampilkan hasil laporan dari Performance Development Review yang sudah dijalankan sehingga Human Resource Development masih mengggunakan dokumen fisik.

Masalah yang dihadapi tersebut juga dapat diidentifikasikan dan dianalisa menggunakan tabel sebab akibat yang dapat diuraikan pada tabel berikut

Tabel 3. 1 Tabel Sebab Akibat

|  |  |  |
| --- | --- | --- |
| Permasalahan | Solusi | Manfaat |
| Aplikasi Space Venturo saat ini belum memiliki sebuah fitur untuk mencatat *competency list* sehingga c*ompetency list* dicatat menggunakan excel | Membuat fitur untuk mencatat *Competency List* | Memudahkan *Human Resources Department* mencatat c*ompetency list* untuk memudahkan proses *Performance Development Review*. |
| Aplikasi Space Venturo saat ini belum memiliki sebuah fitur untuk mengajukan *Performance Development Review* sehingga proses review dilakukan dengan wawancara singkat. | Membuat fitur untuk mengajukan *Performance Development Review* | Memudahkan dan mempercepat proses pengajuan *Performance Development Review* untuk anggota tim. |
| Aplikasi Space Venturo saat ini belum memiliki fitur untuk mereview *Performance Development Review* yang telah diajukan anggota timsehingga pemimpin timmasih melakukan wawancara singkat. | Membuat fitur untuk mereview *Performance Development Review* | Memudahkan pemimpin timuntuk mereview *Performance Development Review* yang telah diajukan oleh anggota tim. |
| Aplikasi Space Venturo saat ini belum memiliki fitur untuk menampilkan hasil laporan dari *Performance Development Review* yang telah dilakukan sehingga *Human Resource Department* masih menggunakan excel. | Membuat fitur untuk menampilkan laporan dari *Performance Development Review* yang sudah dilakukan. | Memudahkan *Human Resource Department* untuk menyajikan informasi terkait *Performance Development Review* yang telah dilakukan. |

### Pemecahan Masalah

Berdasarkan Berdasarkan identifikasi sebelumnya, solusi untuk masalah tersebut adalah dengan menggunakan Sistem Performance Development Review (PDR). PDR merupakan sistem yang membantu dalam mengelola dan menyimpan informasi terkait evaluasi dan pengembangan karyawan. Dalam PDR, atasan memiliki peran penting sebagai evaluator kinerja anggota tim. Mereka akan melakukan evaluasi kinerja berdasarkan kompetensi yang telah ditetapkan dan memberikan umpan balik kepada karyawan. HRD bertanggung jawab dalam menyediakan dukungan dan bimbingan terkait pelaksanaan PDR, serta memastikan bahwa prosesnya berjalan sesuai dengan kebijakan perusahaan. Setelah evaluasi selesai, HRD akan mempresentasikan hasil PDR kepada atasan dan pihak terkait untuk dijadikan dasar dalam pengambilan keputusan terkait pengembangan karyawan. Dengan menggunakan PDR, proses evaluasi kinerja dapat dilakukan secara efektif dengan melibatkan atasan sebagai evaluator utama, dan HRD sebagai pendukung dan fasilitator dalam pelaksanaannya.

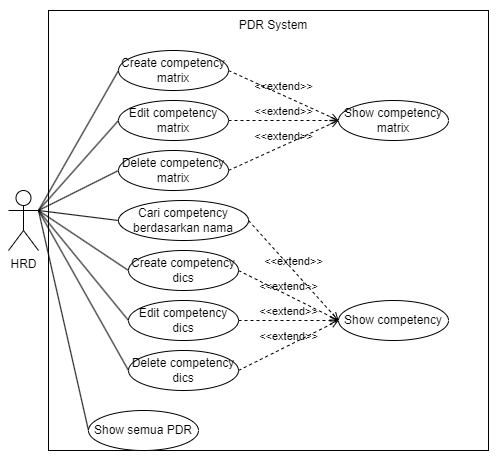
## Perancangan

### Perancangan Sistem

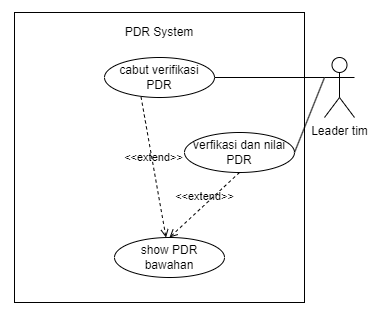
Perancangan sistem untuk PDR akan diimplementasikan dalam aplikasi PDR Space Venturo. Sistem ini akan didesain berdasarkan pendekatan proses dan dijelaskan melalui Use Case Diagram.

#### **Use Case Diagram**

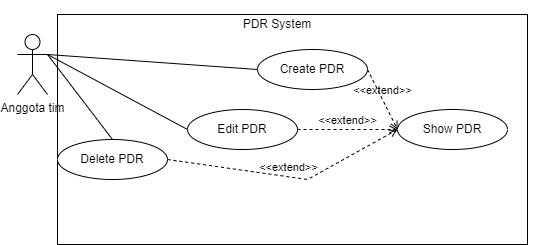
*Use Case* adalah diagram yang menjelaskan tentang siapa saja aktor yang menggunakan sistem. *Use Case* juga menjelaskan apa saja yang bisa dilakukan aktor tersebut di dalam sistem.



Gambar 3.1 Use Case Diagram HRD



Gambar 3.2 Use Case Diagram Leader tim



Gambar 3.3 Use Case Diagram Anggota tim

*Use Case diagram* diatas memperlihatkan terdapat tiga aktor dan 21 use case yang mana menggambarkan proses yang terjadi pada sistem. Dalam Use Case Diagram, terdapat fitur perpanjangan (extends) yang menunjukkan bahwa suatu use case merupakan perluasan dari use case sebelumnya. Selain itu, terdapat juga fitur inklusi (includes) yang menunjukkan bahwa suatu use case termasuk dalam proses use case sebelumnya.

1. **Definisi Aktor**

Berikut jabaran mengenai aktor-aktor yang terdapat didalam aplikasi PDR Venturo Space.

Tabel 3. 2 Tabel Definisi Aktor

|  |  |  |
| --- | --- | --- |
| No | Aktor | Deskripsi |
| 1. | *Human Resources Department* (HRD) | HRD bertanggung jawab dalam mengelola dan mengawasi proses *Performance Development Review* (PDR). HRD memiliki izin untuk membuat, memperbarui, dan menghapus daftar kompetensi serta matriks kompetensi. Selain itu, mereka dapat melihat semua PDR yang diajukan oleh karyawan dan memiliki kemampuan untuk menghapus PDR jika diperlukan. |
| 2. | Anggota Tim | Anggota Tim adalah karyawan yang berpartisipasi dalam proses PDR. Mereka memiliki kemampuan untuk mengajukan PDR mereka sendiri, serta menghapus dan memperbarui PDR yang belum diverifikasi. |
| 3. | Leader Tim | Pimpinan Tim bertanggung jawab untuk memverifikasi PDR yang diajukan oleh Anggota Tim dalam tim mereka. Leader memiliki kemampuan untuk memberikan penilaian terhadap PDR yang diverifikasi, serta mencabut status verifikasi jika diperlukan. |

1. **Use Case Scenario**

Tabel 3. 3 Tabel Use Case Create Competency Dics

| *Use Case Create Competency Dics* | |
| --- | --- |
| Objektif | Aktor dapat membuat *Competency Dics* baru |
| Aktor | HRD |
| Pre-Condition | Aktor berada diluar menu *Competency Dics* |
| Main Flow | 1. Aktor memilih menu *Competency Dics.* 2. Aktor mendapatkan *List* *Competency Dics.* 3. Aktor menekan tombol *Add Competency DIcs.* 4. Aktor mengisi field wajib yang ditandai dengan tanda bintang(\*) dan menekan tombol create. 5. Aktor kembali ke *List* *Competency Dics* dengan tambahan data baru. |
| Alternative Flow | Alternatif 1 (Membuat List baru dengan field wajib yang ditandai dengan tanda bintang(\*) kosong)   1. Aktor memilih menu *Competency Dics.* 2. Aktor mendapatkan *List Competency Dics*. 3. Aktor menekan tombol *Add Competency Dics*. 4. Aktor membiarkan *field* wajib yang ditandai dengan tanda bintang(\*) kosong dan menekan tombol create. 5. Muncul peringatan bahwa ada *field* yang kosong. |
| Post-Condition | *List Competency Dics* tampil beserta data baru |

Tabel 3. 4 Tabel Use Case Create Competency Matrix

| *Use Case Create Competency Matrix* | |
| --- | --- |
| Objektif | Aktor dapat menambahkan elemen baru pada *Competency Matrix* |
| Aktor | HRD |
| Pre-Condition | Aktor berada diluar menu *Competency Matrix* |
| Main Flow | 1. Aktor memilih menu *Competency Matrix.* 2. Aktor mendapatkan *Competency Matrix.* 3. Aktor menekan tombol *add* yang disimbolkan dengan tanda *plus*(+) *.* 4. Aktor menambahkan setidaknya satu *Competency* lalu menekan tombol *Save*. 5. Aktor kembali ke *Competency Matrix* dengan tambahan data baru. |
| Alternative Flow | Alternatif 1 (Menambahkan elemen baru tanpa memilih *Competency*)   1. Aktor memilih menu *Competency Matrix.* 2. Aktor mendapatkan *Competency Matrix*. 3. Aktor menekan tombol *add* yang disimbolkan dengan tanda *plus*(+). 4. Aktor tidak menambahkan *Competency* lalu menekan tombol *Save*. 5. Muncul peringatan bahwa tidak ada *Competency* yang terpilih. |
| Post-Condition | *Competency Matrix* tampil beserta data baru |

Tabel 3. 5 Tabel Use Case Edit Competency Dics

| *Use Case Edit Competency Dics* | |
| --- | --- |
| Objektif | Aktor dapat memodifikasi detail pada *Competency Dics* |
| Aktor | HRD |
| Pre-Condition | Aktor berada diluar menu *Competency Dics* |
| Main Flow | 1. Aktor memilih menu *Competency Dics.* 2. Aktor mendapatkan *Competency Dics.* 3. Aktor menekan tombol *edit* pada salah satu *Competency*. 4. Aktor mendapatkan detail dari *Competency* yang dipilih. 5. Aktor memodifikasi detail *Competency* menekan tombol *Save Changes*. 6. Aktor kembali ke *List* *Competency Dics* dengan data yang telah diperbarui baru. |
| Alternative Flow | Alternatif 1 (Memodifikasi field wajib yang ditandai dengan tanda bintang(\*) dengan data kosong)   1. Aktor memilih menu *Competency Dics.* 2. Aktor mendapatkan *Competency Dics.* 3. Aktor menekan tombol *edit* pada salah satu *Competency*. 4. Aktor mendapatkan detail dari *Competency* yang dipilih. 5. Aktor memodifikasi *field* wajib yang ditandai dengan tanda bintang(\*) dengan data kosong lalu menekan tombol *Save Changes*. 6. Muncul peringatan bahwa ada *field* kosong. |
| Post-Condition | *List Competency Dics* tampil dengan data yang telah diperbarui. |

Tabel 3. 6 Tabel Use Case Edit Competency Matrix

| *Use Case Edit Competency Matrix* | |
| --- | --- |
| Objektif | Aktor dapat memodifikasi elemen pada *Competency Matrix* |
| Aktor | HRD |
| Pre-Condition | Aktor berada diluar menu *Competency Matrix* |
| Main Flow | 1. Aktor memilih menu *Competency Matrix.* 2. Aktor mendapatkan *Competency Matrix.* 3. Aktor menekan tombol edit yang disimbolkan dengan tanda pensil*.* 4. Aktor memodifikasi elemen matrixlalu menekan tombol *Save*. 5. Aktor kembali ke *Competency Matrix* dengan tambahan data baru. |
| Alternative Flow | - |
| Post-Condition | *Competency Matrix* tampil dengan data yang telah diperbarui. |

Tabel 3. 7 Tabel Use Case Create PDR

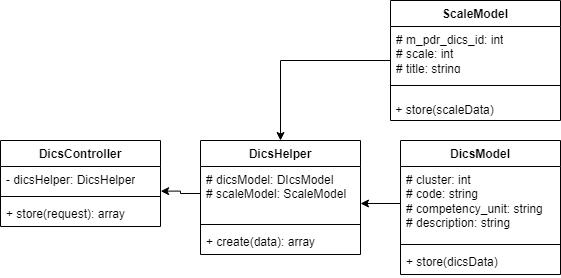
| *Use Case Create PDR* | |
| --- | --- |
| Objektif | Aktor dapat membuat dan *submit* PDR baru |
| Aktor | Anggota Tim |
| Pre-Condition | Aktor berada diluar menu *PDR* |
| Main Flow | 1. Aktor memilih menu *PDR.* 2. Aktor mendapatkan *List PDR* yang telah ter-*submit.* 3. Aktor menekan tombol *New Performance Development Review.* 4. Aktor memilih atasan dengan mengetik nama di kolom pencarian dan menekan tombol *Start PDR*. 5. Aktor mendapatkan tampilan 3 tahapan *PDR. Performance Score, Competency Score,* dan *Growth Plan.* 6. Aktor menekan tombol next atau tab *Competency Score* dan mendapatkan tampilan *Competency Score*. 7. Aktor memilih level untuk setiap *Competency* beserta *proof* jika ada lalu menekan tombol *Save*. 8. Aktor mendapatkan tampilan *Growth Plan.* 9. Aktor mengisi field yang tersedia dan menekan tombol *Submit for Approval.* 10. Aktor mendapatkan tampilan konfirmasi lalu menekan tombol oke. 11. Aktor kembali ke *List PDR* dengan tambahan data baru. |
| Alternative Flow | Alternatif 1 (Aktor mengosongkan *field* “Saran untuk Perusahaan” pada tampilan *Growth Plan*)   1. Aktor mendapatkan tampilan *Growth Plan.* 2. Aktor mengosongkan field “Saran untuk Perusahaan” dan menekan tombol *Submit for Approval.* 3. Aktor mendapatkan tampilan konfirmasi lalu menekan tombol oke. 4. Muncul peringatan bahwa ada field kosong. |
| Post-Condition | *List PDR* tampil beserta data baru. |

Tabel 3. 8 Tabel Use Case Verifikasi PDR

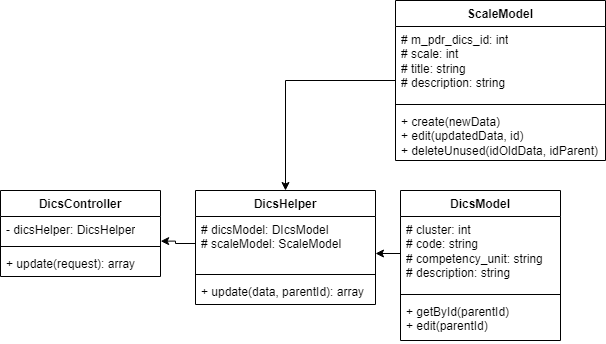
| *Use Case Verify PDR* | |
| --- | --- |
| Objektif | Aktor dapat menilai dan memverifikasi *PDR* anggota tim |
| Aktor | Leader Tim |
| Pre-Condition | Aktor berada diluar menu *PDR My Squad* |
| Main Flow | 1. Aktor memilih menu *PDR My Squad.* 2. Aktor mendapatkan tampilan *List PDR* anggotatim*.* 3. Aktor memilih salah satu *PDR* anggota berstatus *pending.* 4. Aktor mendapatkan tampilan detail *PDR* anggota. 5. Aktor menekan tombol *Start PDR.* 6. Aktor mendapatkan tampilan 3 tahapan *PDR*. *Performance Score*, *Competency Score*, dan *Growth Plan*. 7. Aktor menekan tombol next atau tab *Competency Score* dan mendapatkan tampilan *Competency Score* 8. Aktor memilih level untuk setiap *Competency* beserta *proof* jika ada lalu menekan tombol *Save*. 9. Aktor mendapatkan tampilan *Summary* yang berupa nilai dari aktor beserta *proof* untuk semua *Competency.* 10. Aktor menekan tombol *next* untuk menampilkan tab detail *Growth Plan PDR.* 11. Aktor menekan tombol *Verify* untuk menyelesaikan proses verifikasi. 12. Aktor kembali ke *List PDR* anggota timdengan status *PDR* yang telah diperbarui. |
| Alternative Flow | Alternatif 1 (Aktor menekan tombol *edit* untuk meninjau ulang proses verifikasi)   1. Aktor mendapatkan tampilan *Summary* yang berupa nilai dari aktor beserta *proof* untuk semua *Competency..* 2. Aktor menekan tombol edit untuk mendapatkan tampilan penilaian*.* 3. Aktor memilih level untuk setiap *Competency* beserta *proof* jika ada lalu menekan tombol *Save.* 4. Aktor mendapatkan tampilan *Summary* yang berupa nilai dari aktor beserta *proof* untuk semua *Competency.* |
| Post-Condition | *List PDR* anggota timtampil dengan status dan nilai yang telah diperbarui. |

#### **Class Diagram**

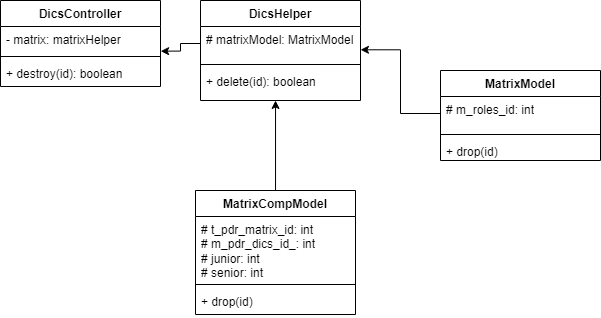
Class diagram merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. Class Diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan constraint yang berhubungan dengan objek yang dikoneksikan



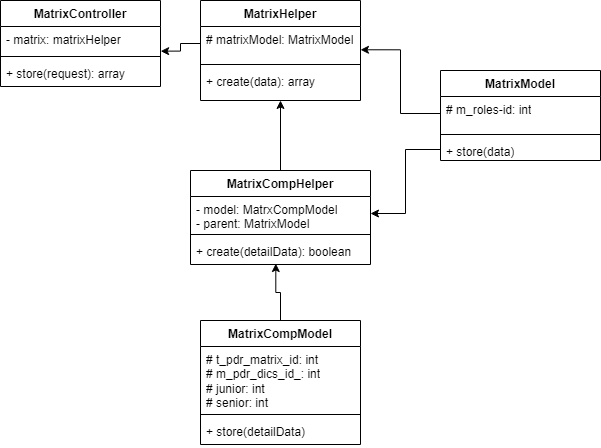
Gambar 3.4 Clas Diagram Create Dics



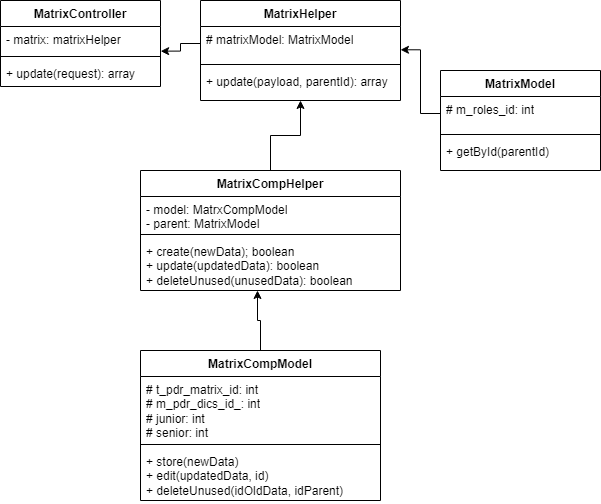
Gambar 3.5 Class Diagram Edit Dics



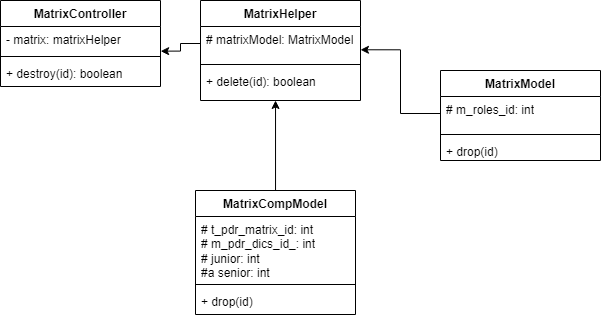
Gambar 3.6 Class Diagram Delete Dics



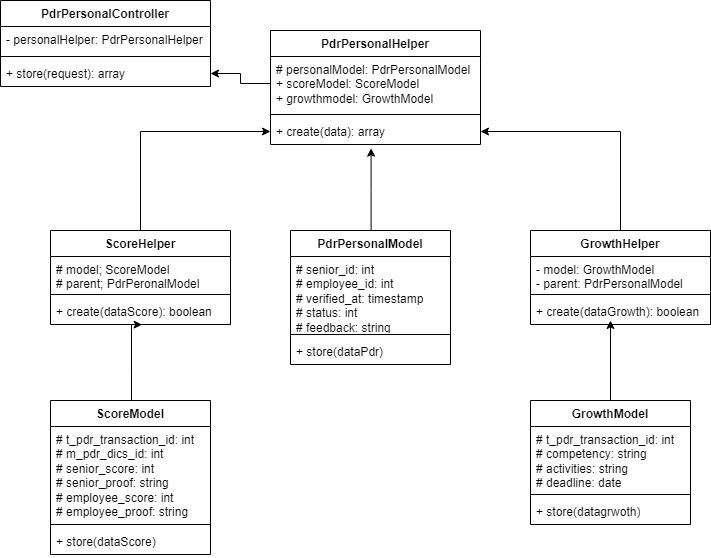
Gambar 3.7 Class Diagram Create Matrix



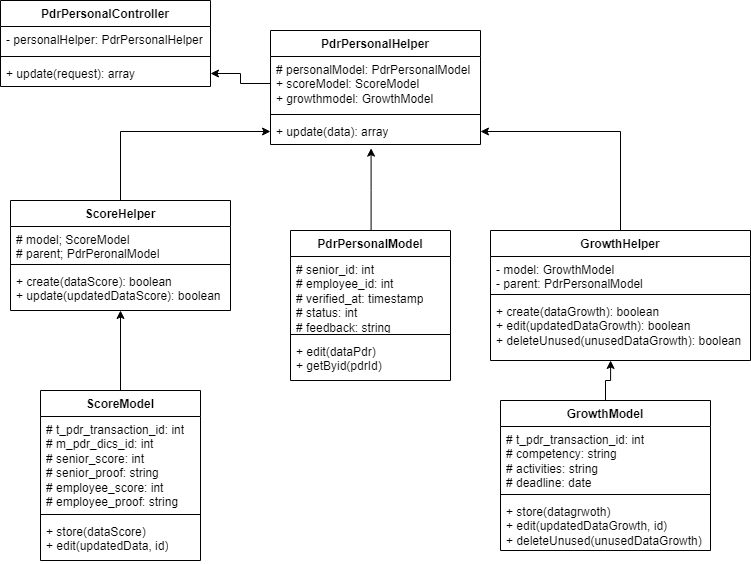
Gambar 3.8 Class Diagram Edit Matrix



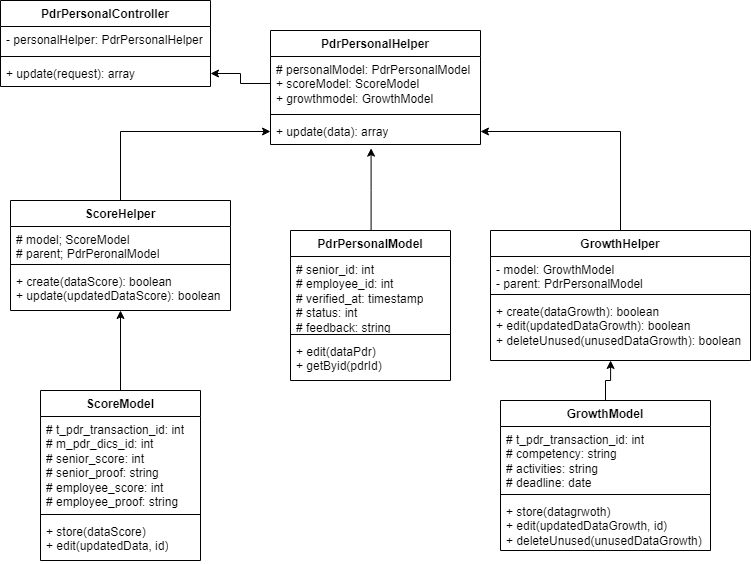
Gambar 3.9 Class Diagram Delete Matrix



Gambar 3.10 Class Diagram Create PDR



Gambar 3.11 Class Diagram Edit PDR



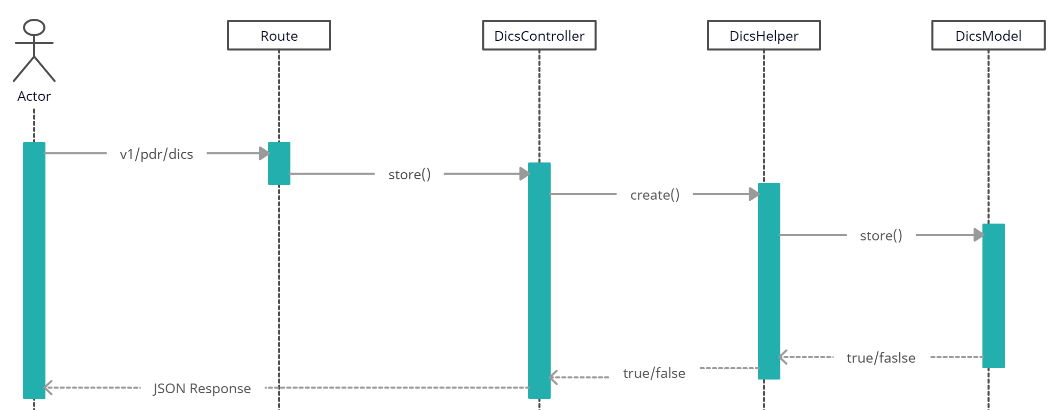
Gambar 3.12 Class Diagram Verifikasi PDR

#### **Sequence Diagram**

Sequence diagram menjelaskan interaksi object yang disusun dalam suatu urutan waktu. Diagram ini secara khusus berasosiasi dengan use case diagram, memperlihatkan tahap demi tahap apa yang seharusnya terjadi untuk menghasilkan sesuatu di dalam use case.

* + - 1. Create Dics

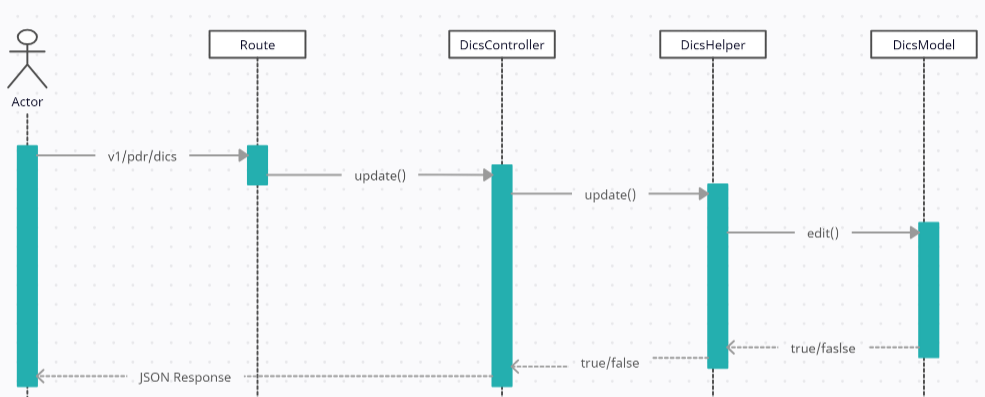
Pada aktifitas create dics, client melakukan *request* ke API melalui *route* yang memanggil fungsi yang sesuai pada *controller*, yang akan memanggil fungsi yang sesuai pada *helper* yang kemudian menggunakan model untuk pengaksesan layer data dan akan kembali ke *helper* lalu *controller* akan mengirim response ke *client*.



Gambar 3.13 Sequence Diagram Create Dics

* + - 1. Update Dics

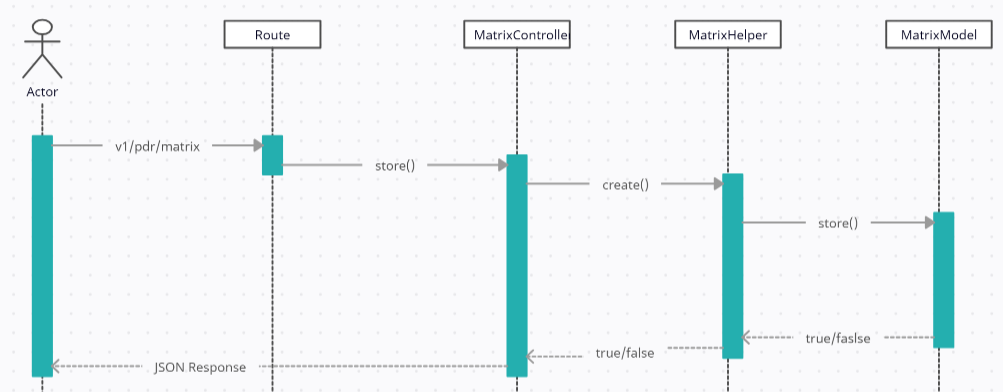
Pada aktifitas update dics, client melakukan request ke API melalui route yang memanggil fungsi yang sesuai pada controller, yang akan memanggil fungsi yang sesuai pada helper yang kemudian menggunakan model untuk pengaksesan layer data dan akan kembali ke helper lalu controller akan mengirim response ke client.



Gambar 3.13 Sequence Diagram Update Dics

* + - 1. Create Matrix

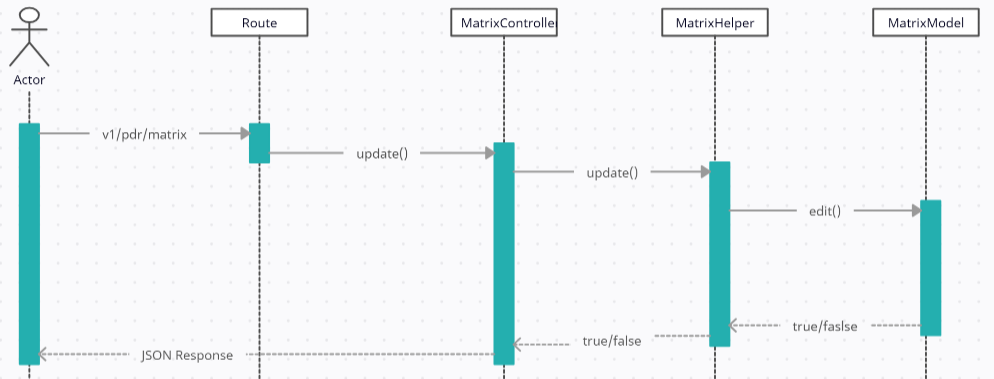
Pada aktifitas create matrix, client melakukan request ke API melalui route yang memanggil fungsi yang sesuai pada controller, yang akan memanggil fungsi yang sesuai pada helper yang kemudian menggunakan model untuk pengaksesan layer data dan akan kembali ke helper lalu controller akan mengirim response ke client.



Gambar 3.13 Sequence Diagram Create Matrix

* + - 1. Update Matrix

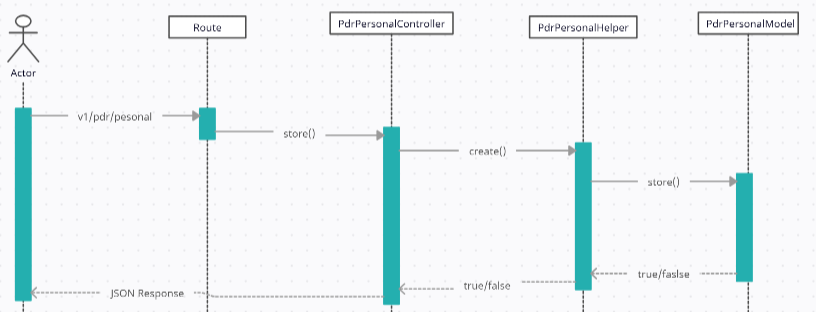
Pada aktifitas update matrix, client melakukan request ke API melalui route yang memanggil fungsi yang sesuai pada controller, yang akan memanggil fungsi yang sesuai pada helper yang kemudian menggunakan model untuk pengaksesan layer data dan akan kembali ke helper lalu controller akan mengirim response ke client.



Gambar 3.13 Sequence Diagram Update Dics

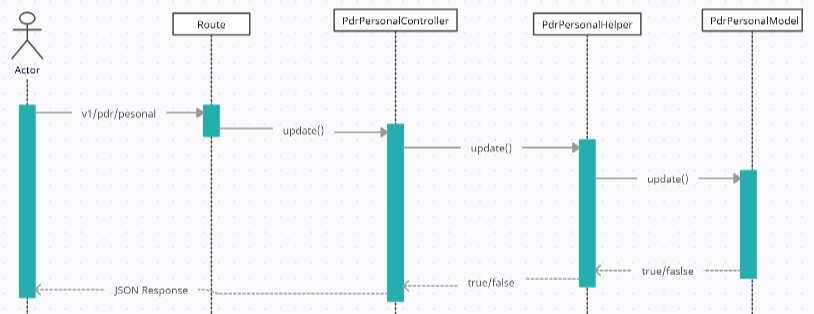
* + - 1. Create PDR

Pada aktifitas create PDR, client melakukan request ke API melalui route yang memanggil fungsi yang sesuai pada controller, yang akan memanggil fungsi yang sesuai pada helper yang kemudian menggunakan model untuk pengaksesan layer data dan akan kembali ke helper lalu controller akan mengirim response ke client.



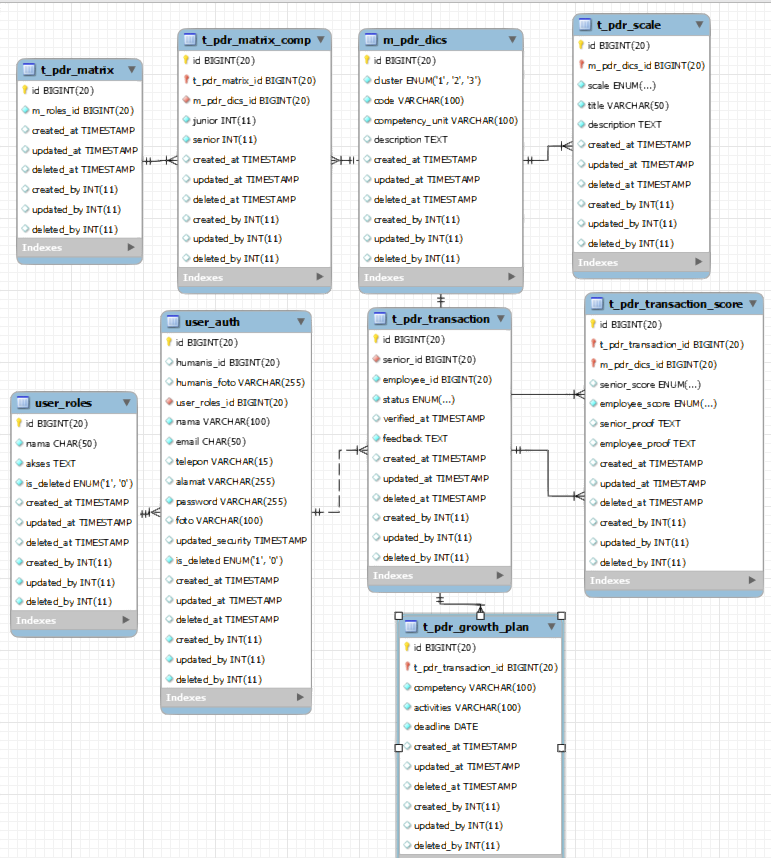
* + - 1. Verifikasi PDR

Pada aktifitas create PDR, client melakukan request ke API melalui route yang memanggil fungsi yang sesuai pada controller, yang akan memanggil fungsi yang sesuai pada helper yang kemudian menggunakan model untuk pengaksesan layer data dan akan kembali ke helper lalu controller akan mengirim response ke client.



### Perancangan Data

Berikut merupakan gambaran perancangan kerangka database yang digunakan dalam pembuatan sistem ini. Data yang dimasukkan pengguna disimpan dalam database melalui Restful API. Gambar terlampir apabila gambar kurang jelas.



### Perancangan Dokumentasi *API*

Dokumentasi API sangat penting untuk memberikan pemahaman kepada pengguna tentang fungsi dan cara mengakses setiap API yang tersedia. Dokumentasi yang baik harus lengkap dengan informasi autentikasi, parameter yang dibutuhkan dalam permintaan, pesan yang jelas, dan contoh respon. Dokumentasi API secara umum terdiri dari HTTP Request dan HTTP Response. Dengan adanya dokumentasi yang komprehensif, pengguna dapat memanfaatkan API secara efektif dan mengintegrasikannya ke dalam aplikasi atau sistem mereka.

* + - 1. HTTP Request

Bagian ini menjelaskan secara detail bagaimana contoh request yang dikirimkan yang terdiri dari:

* + - * 1. *Environment*

Environment menjelaskan HTTP *method* apa yang digunakan dalam melakukan *request* seperti: GET, POST, PUT, DELETE, dan alamat endpoint yang akan di akses

* + - * 1. *Header*

*Header* menjelaskan parameter yang harus disertakan pada *header* setiap *request*.

* + - * 1. *Body*

Biasanya bagian *body* diperlukan ketika kita menggunakan HTTP *method* POST / PUT, karena dalam menggunakan *method* tersebut disertakan data - data yang akan dikirim bersama dengan *request*.

* + - 1. HTTP *Response*

Bagian ini menjelaskan tentang response yang diberikan oleh aplikasi setelah melakukan request yang tediri dari:

* + - * 1. *Status*

Bagian status memberikan informasi kepada client tentang status dari sebuah response yang terdiri dari HTTP code, contoh sederhana adalah kode 200 untuk sukses dan kode 422 untuk kesalahan input.

* + - * 1. *Message*

Bagian *Message* biasanya memberikan informasi kepada client tentang pesan dari status, dapat berupa pesan sukses atau kesalahan dalam melakukan HTTP method POST, PUT, atau DELETE.

* + - * 1. *Data*

Bagian data biasanya memberikan kembalian data yang diminta pada request GET, dapat berupa sebuah object dalam bentuk json.

* + - 1. *Response* *Example*

Bagian ini menjelaskan contoh dari response dalam bentuk json agar lebih mudah dipahami.

### Rumus Penilaian

Berikut merupakan rumus penilaian yang digunakan dalam sistem Performance Development Review (PDR):

Jika PDR belum diverifikasi oleh senior:

Rumus di atas digunakan untuk menghitung nilai persentase dari penilaian kinerja anggota tim dalam PDR ketika PDR belum diverifikasi oleh senior. Proses perhitungan dilakukan dengan menambahkan semua nilai yang diberikan oleh anggota tim pada setiap kompetensi, kemudian hasil penjumlahan nilai tersebut dibagi dengan total skala kompetensi yang ada pada semua kompetensi yang dinilai. Akhirnya, nilai persentase diperoleh dengan mengalikan hasil pembagian tersebut dengan faktor 1/100.

Jika PDR telah diverifikasi oleh senior:

Rumus di atas digunakan untuk menghitung nilai persentase dari penilaian kinerja anggota tim dalam PDR setelah PDR diverifikasi oleh senior. Proses perhitungan dilakukan dengan menambahkan nilai-nilai yang diberikan oleh anggota tim dan atasan, kemudian hasil penjumlahan tersebut dibagi dengan dua untuk memperoleh rata-rata nilai antara anggota tim dan atasan. Selanjutnya, hasil rata-rata tersebut dibagi dengan total skala kompetensi yang ada pada semua kompetensi yang dinilai. Akhirnya, nilai persentase diperoleh dengan mengalikan hasil pembagian tersebut dengan faktor 1/100.

Rumus penilaian ini memungkinkan sistem PDR untuk mengakomodasi penilaian dari senior yang telah diverifikasi dalam perhitungan nilai persentase. Hal ini memberikan bobot yang seimbang antara penilaian dari anggota tim dan penilaian dari senior. Dengan adanya rumus ini, sistem PDR dapat menghasilkan nilai persentase yang mencerminkan kinerja anggota tim dengan mempertimbangkan penilaian dari kedua belah pihak secara proporsional terhadap skala kompetensi yang digunakan.

## Rancangan Pengujian

Dalam Metodologi pengujian yang digunakan dalam penelitian ini adalah pengujian black box. Metode pengujian ini fokus pada fungsionalitas dan kinerja backend aplikasi berbasis Laravel.

Dengan menggunakan metode black box, penelitian ini dapat menguji aplikasi secara efektif untuk memastikan bahwa backend berfungsi dengan baik dan memenuhi kebutuhan pengguna.

Tabel 3. 9 Tabel Rancangan Pengujian

|  |  |  |  |
| --- | --- | --- | --- |
| **Fitur** | **Skenario** | **Kasus** | **Hasil yang diharapkan** |
| *Competency Dics* | Tambah c*ompetency* baru | Menambahkan *competency.* | Data *competency* berhasil tersimpan. |
| Menambahkan dengan data kosong. | Muncul peringatan bahwa ada *field* kosong. |
| Ubah *competency* | Mengubah detail competency seperti *dics name, description*, dan *levels.* | Detail berhasil diubah. |
| Mengubah detail dengan data kosong. | Muncul peringatan bahwa ada *field* yang masih kosong. |
| Hapus *competency* | Menghapus *competency* dengan *soft delete.* | Data *competency* tidak tampil lagi walapun data masih ada di *database*. |
| Mencari *competency* | Mencari *competency* dengan nama sebagai kata kunci. | Muncul *competency* yang dicari. |
| *Competency Matrix* | Tambah elemen *competency matrix* | Menambahkan elemen dengan memilih satu atau lebih kompetensi. | Berhasil menambah elemen. |
| Menambahkan elemen tanpa memilih *competency.* | Muncul peringatan untuk memilih setidaknya satu *competency.* |
| Menambahkan elemen dengan dua atau lebih *competency* yang sama. | Muncul peringatan bahwa ada duplikat *competency.* |
| Ubah elemen *competency matrix* | Mengubah elemen seperti *competency,* junior, dan senior*.* | Elemen berhasil diubah. |
| Mengubah elemen dengan dua atau lebih *competency* yang sama. | Muncul peringatan bahwa ada duplikat *competency.* |
| Hapus seluruh elemen *competency matrix* | Menghapus *elemen* dengan *soft delete.* | Data *elemen* tidak tampil lagi walapun data masih ada di database. |
| *PDR* *Board* | Hapus *PDR* | Menghapus *PDR* dengan *soft delete.* | Data *PDR* tidak tampil lagi walapun data masih ada di database. |
| *PDR Submittal* | Tambah *PDR* baru | Menambahkan *PDR* baru dengan lengkap (*level competency*, *growth plan*, *saran perusahaan*). | Data *PDR* berhasil tersimpan dan diajukan. |
| Menambahkan *PDR* tanpa mengisi “saran untuk perusahaan” pada *growth plan*. | Muncul peringatan bahwa ada field yang masih kosong. |
| Ubah detail *PDR* yang belum diverifikasi | Mengubah c*ompetency score* atau g*rowth plan PDR* yang belum diverifikasi oleh *leader*. | *PDR* berhasil diubah. |
| Mengubah *PDR* dengan mengosongkan nama kompetensi pada *growth plan*. | *PDR* diubah dengan dan kegiatan *growth plan* yang dikosongi dihapus dengan *soft delete* sehingga tidak tampil namun masih ada di database. |
| Mengubah *PDR* dengan mengosongkan nama kompetensi pada *growth plan*. | Muncul peringatan bahwa ada field yang masih kosong. |
| Hapus *PDR* yang belum diverifikasi | Menghapus *PDR* dengan *soft delete.* | Data *PDR* tidak tampil lagi walapun data masih ada di database. |
| *PDR Verification* | Verifikasi *PDR* | Menambahkan nilai dari sisi *leader* dan ubah status *PDR* dari *pending* ke *verified*. | *PDR* berhasil diubah. |
| Cabut Verifikasi *PDR* | Mencabut verifikasi *PDR* dan mengembalikan status dari *verified* ke *pending*. | *PDR* berhasil diubah. |