**BAB 5**

IMPLEMENTASI SISTEM

**BAB 5**

**IMPLEMENTASI SISTEM**

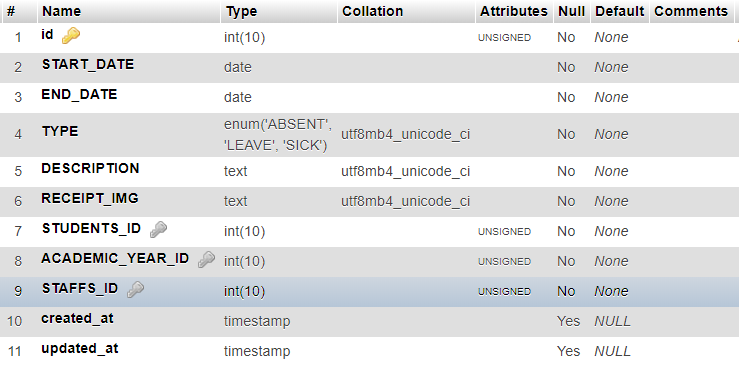
Bab ini membahas tentang implementasi sistem dari rancangan solusi yang sudah direncakan dari bab-bab sebelumnya. Implementasi solusi ini dibangun meliputi database dan bagaimana program ditulis.

**5.1 Implementasi Basis Data**

Rancangan *Entity Relationship Diagram* (ERD) yang telah dibuat diimplementasikan dengan database manager MySql. Pada bagian ini akan dijelaskan setiap bagian dari tabel yang terbentuk dalam database. Selain itu, pada bagian ini juga akan dijelaskan kegunaan dari setiap tabel yang telah dibentuk.

1. Tabel Absents

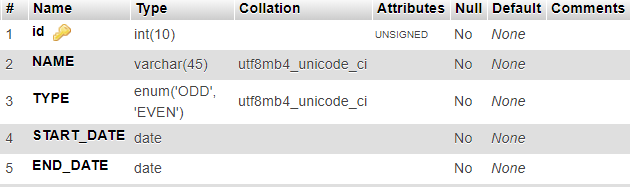
Tabel absents berguna untuk menyimpan catatan absen siswa. Data yang disimpan yaitu tanggal awal absen, tanggal berakhir absen, tipe absen, deskripsi/keterangan absen, bukti keterangan tidak hadir, id siswa, id staff/karyawan yang mencatat, dan akan tersimpan secara otomatis oleh bawaan sistem yaitu *created\_at* dan *updated\_at*.Struktur tabel absents bisa dilihat seperti pada gambar 5.1.



**Gambar 5.1. Struktur tabel absents**

2. Tabel Academic\_Years

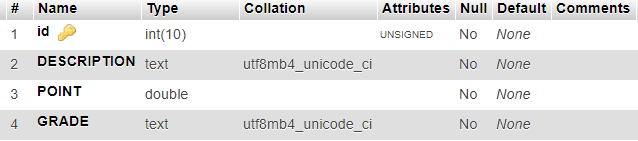
Tabel academic\_years adalah tabel master yang digunakan untuk menyimpan daftar tahun ajaran. Data yang akan disimpan dalam tabel ini adalah nama tahun ajaran, tipe tahun ajaran yaitu ganjil atau genap, tanggal awal dimulai tahun ajaran, tanggal berakhirnya tahun ajaran. Struktur tabel academic\_years dapat dilihat pada tabel 5.2.



**Gambar 5.2. Struktur tabel academic years**

3. Table Achievements

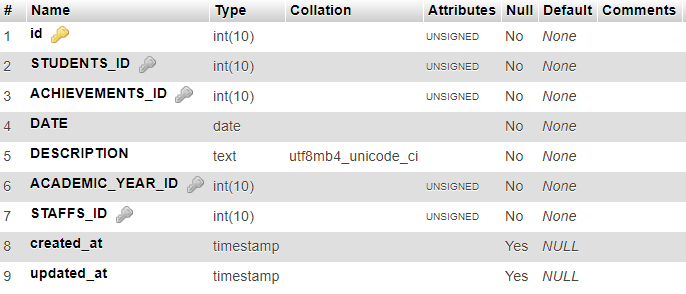
Tabel achievements adalah tabel master yang digunakan untuk menyimpan daftar penghargaan yang berlaku di sekolah. Data yang akan disimpan dalam tabel ini adalah deskripsi, poin, tingkat penghargaan, dan tipe penghargaan. Struktur tabel academic\_years dapat dilihat pada tabel 5.3.



**Gambar 5.3 Struktur tabel achievements**

4. Table Achievement\_Records

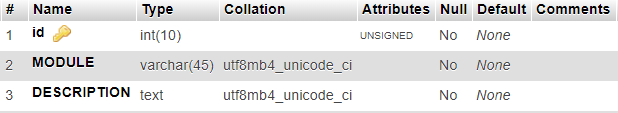
Tabel achievement\_records berguna untuk menyimpan catatan penghargaan yang diperoleh siswa dan detail catatannya. Data yang disimpan pada tabel ini adalah id siswa, id penghargaan, tanggal, deskripsi, id tahun ajaran, id staff/karyawan, *created\_at* dan *updated\_at.* Struktur tabel achievement\_records bisa dilihat pada gambar 5.4.



**Gambar 5.4 Struktur tabel achievement records**

5. Tabel Activities

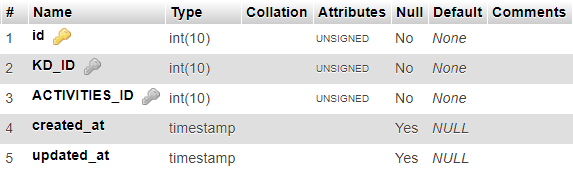
Tabel activities adalah tabel master yang digunakan untuk menyimpan daftar kegiatan/aktivitas pembelajaran yang berlaku di sekolah. Data yang akan disimpan dalam tabel ini adalah module dan deskripsi. Struktur tabel academic\_years dapat dilihat pada tabel 5.4.



**Gambar 5.4 Stuktur tabel activites**

6. Tabel Activities\_KD

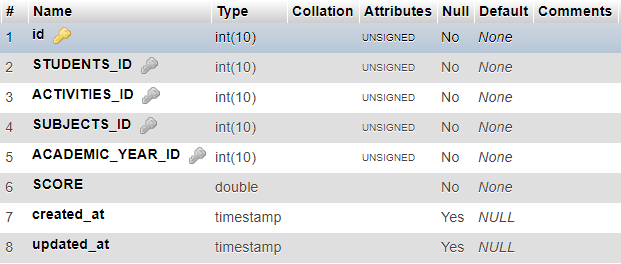
Tabel activites\_kd adalah hubungan antara tabel activities dan tabel kd. Tabel ini berguna untuk menyimpan aktivitas yang dilakukan yang berkaitan dengan satu kompetensi dasar (kd) tertentu. Di dalam tabel ini akan menyimpan data yaitu, id kd, id aktivitas, *created\_at* dan *updated\_at.* Struktur tabel activites kd digambarkan pada gambar 5.5.



**Gambar 5.5 Struktur tabel activities kd**

7. Tabel Activities\_Students

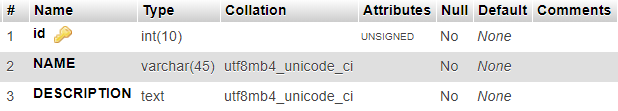
Tabel activites students digunakan untuk menyimpan detail transaksi data pada saat melakukan peng-inputan nilai suatu mata pelajaran oleh *user.* Di dalam tabel ini akan tersimpan data yaitu id siswa sebagai *foreign key,* id aktivitas sebagai *foreign key,* id mata pelajaran sebagai *foreign key,* id tahun ajaran sebagai *foreign key, created\_at* dan *updated\_at.* Struktur tabel activities\_students ini dapat digambarkan seperti pada gambar 5.7.



**Gambar 5.7 Struktur tabel activites students**

8. Tabel Banks

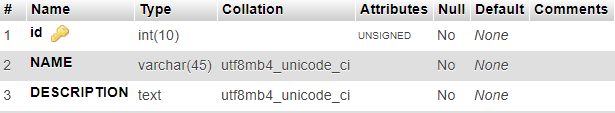
Tabel banks digunakan untuk menampung data master bank. Data yang disimpan dalam tabel ini yaitu nama bank, keterangan bank. Untuk membedakan setiap bank digunakan primary key. Struktur tabel bank dapat dilihat pada tabel 5.8.



**Gambar 5.8 Struktur tabel banks**

9. Tabel Departments

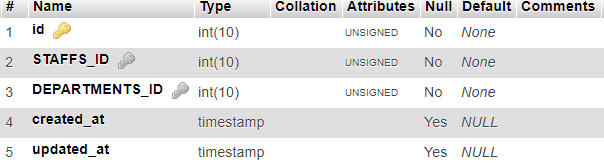
Tabel departments adalah tabel master yang digunakan untuk menyimpan daftar departemen yang berada di sekolah. Tabel ini akan menyimpan data yaitu nama departemen, dan deskripsi dari departemen. Struktur tabel ini bisa dilihat pada gambar 5.9.



**Gambar 5.9 Struktur tabel departments**

10. Tabel Departments\_Staffs

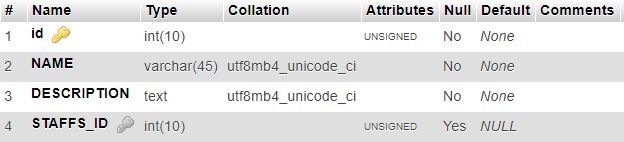
Tabel departments\_staffs adalah tabel hasil relasi *many to many* antara tabel departmens dengan tabel staffs. Tabel ini digunakan sebagai penyimpan data id staff dan id departemen. Selain itu juga disimpan tanggal transaksi kapan hubungan antara satu id staff dan satu id departemen terbentuk yaitu dengan menyimpan *created\_at* dan *updated\_at.* Struktur tabel departments\_staffs digambarkan seperti pada gambar 5.10.



**Gambar 5.10 Struktur tabel departments staffs**

11. Tabel Extracurriculars

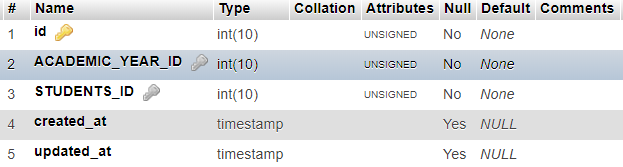
Tabel extracurriculars adalah tabel master yang menyimpan daftar ekstrakurikuler yang berada di sekolah. Tabel ini menyimpan data seperti nama ekskul, deskripsi ekskul, dan id staff sebagai *foreign key* yang menandakan guru pengajar ekskul tersebut. Struktur tabel extracurriculars digambarkan seperti pada gambar 5.11.



**Gambar 5.11 Struktur tabel extracurriculars**

12. Tabel Extracurricular\_Records

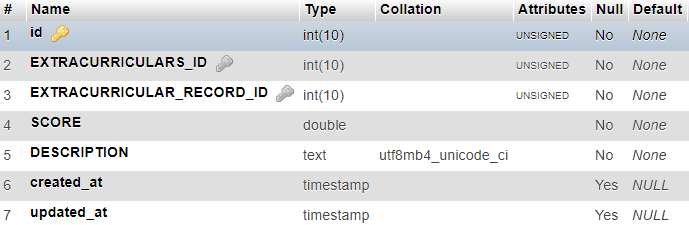
Tabel extracurricular\_records adalah tabel yang digunakan untuk menyimpan catatan ekstrakurikuler setiap siswa pada tahun ajaran tertentu. Tabel ini akan menyimpan data yakni id tahun ajaran sebagai *foreign key¸*dan juga id siswa sebagai *foreign key, created\_at* dan *updated\_at.* Struktur tabel extracurricular\_records digambarkan seperti pada gambar 5.12.



**Gambar 5.12 Struktur tabel extracurricular records**

13. Tabel Extracurricular\_Reports

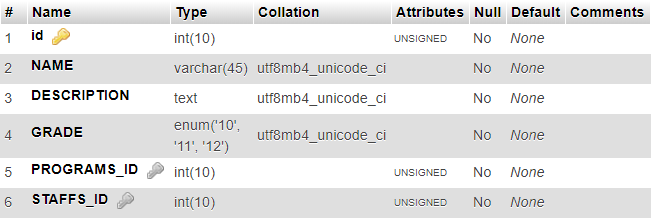
Tabel extracurricular\_reports merupakan tabel hasil relasi *many to many* antara tabel extracurricular\_records dan tabel extracurriculars. Tabel ini digunakan untuk menyimpan transaksi detail setiap mata ekskul yang terdapat dalam satu id catatan ekstrakurikuler pada tabel extracurricular\_records. Data yang akan disimpan pada tabel ini adalah id extracurricular sebagai *foreign key,* id dari tabel extracurricular\_records sebagai *foreign key*, nilai, deskripsi, *created\_at* dan *updated\_at.* Tabel ini digambarkan seperti pada gambar 5.13.



**Gambar 5.13 Struktur tabel extracurricular reports**

14. Tabel Grades

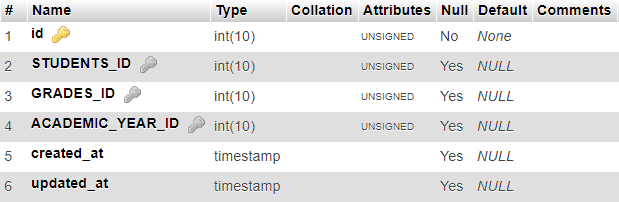
Tabel grades digunakan untuk menyimpan daftar nama kelas. Data yang dicatat yaitu nama kelas, keterangan kelas, tingkat kelas, id unik dari tabel program sebagai *foreign key*, id unik dari tabel staff sebagai *foreign key*. Struktur tabel grades digambarkan seperti pada gambar 5.14.



**Gambar 5.14 Struktur tabel grades**

15. Tabel Grades\_Students

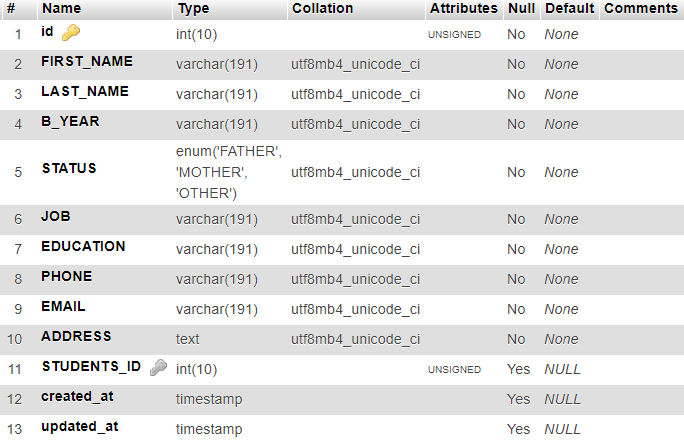
Tabel grades\_students adalah tabel hasil relasi antara tabel grades dan tabel students. Tabel ini untuk mencatat transaksi detail antara siswa dan kelas. Hal ini digunakan untuk memetakan bahwa satu siswa memiliki satu kelas disetiap tahun ajaranya. Artinya siswa tidak terus menerus menetap pada kelas itu, dalam kasus nyata setiap siswa pasti ada peningkatan kelas. Oleh itu data yang dicatat dalam tabel ini adalah id unik siswa yang menjadi *foreign key*, id unik kelas yang menjadi *foreign key*, dan id unik tahun ajaran sebagai *foreign key, created\_at* dan *updated\_at.* Struktur data tabel grades\_students dapat dilihat pada tabel 5.15.



**Gambar 5.15 Struktur tabel grades students**

16. Tabel Guardians

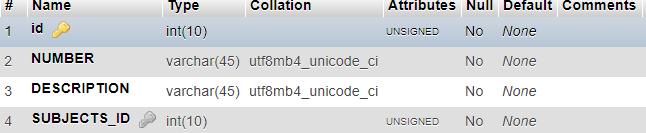
Tabel guardians adalah tabel yang digunakan untuk menyimpan data wali siswa. Setiap siswa memiliki perwakilan yang datanya harus disimpan. Adapun data yang disimpan pada tabel ini adalah nama depan wali siswa, nama akhir wali siswa, tanggal lahir wali siswa, status wali siswa apakah ayah/ibu/lainnya, pekerjaan wali siswa, pendidikan terakhir, nomor telepon, alamat email, alamat tempat tinggal, id siswa sebagai *foreign key, created\_at* dan *updated\_at.* Struktur tabel ini dapat dilihat seperti pada gambar 5.16.



**Gambar 5.16 Struktur tabel guardians**

17. Tabel KD

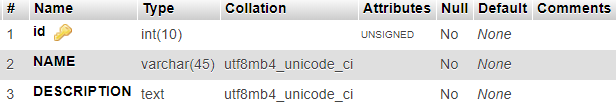
Tabel kd adalah tabel master yang berguna untuk menyimpan daftar kompetensi dasar. Tabel ini akan menyimpan nomor kompetensi dasar, deskripsi kd, dan juga id mata pelajaran sebagai *foreign key.* Struktur tabel kd digambarkan seperti pada gambar 5.17.



**Gambar 5.17 Struktur tabel kd**

18. Tabel Programs

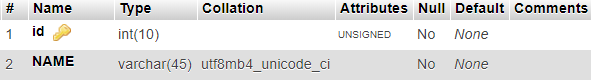
Tabel programs digunakan untuk menampung data mengenai program studi disekolah. Data yang disimpan yaitu nama program studi, keterangan program studi. Struktur tabel programs dapat dilihat pada tabel 5.9.



**Gambar 5.9 Struktur tabel programs**

19. Tabel Religions

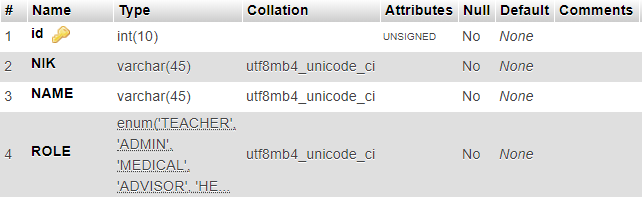
Tabel religion merupakan tabel master untuk menampung data mengenai agama. Data yang akan dicatat dalam tabel ini adalah nama agama. Struktur tabel religions dapat dilihat pada tabel 5.3.



**Gambar 5.19 Struktur tabel religions**

20. Tabel Staffs

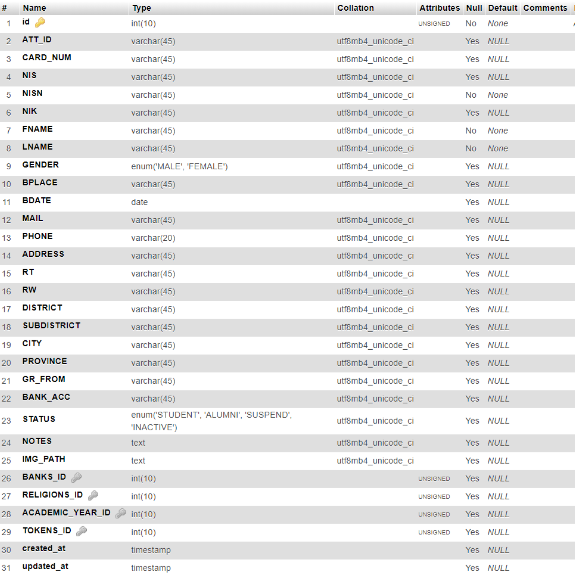
Tabel staffs adalah tabel yang digunakan untuk menyimpan data karyawan/staff yang bekerja di sekolah. Tabel ini akan menyimpan data seperti nomor induk kepegawaian (NIK), nama lengkap staff, hak akses/*role* staff yaitu apakah terdaftar sebagai guru atau admin atau petugas uks atau guru BK atau kepala sekolah. Struktur tabel staffs digambarkan seperti pada gambar 5.20.



**Gambar 5.20 Struktur tabel staffs**

21. Tabel Students

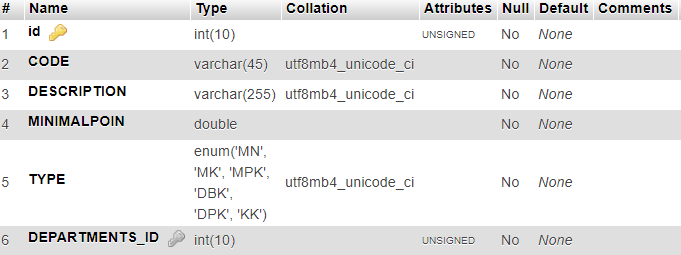
Tabel students adalah tabel yang digunakan untuk menyimpan data siswa. Data siswa yang akan disimpan yaitu atribut id siswa, nomor kartu pelajar, nomor induk sekolah (NIS), nomor induk siswa nasional (NISN), nomor induk kependudukan jika ada (NIK), nama depan siswa, nama belakang siswa, jenis kelamin, tempat dan tanggal lahir, email, nomor telepon, alamat tempat tinggal, rt, rw, kelurahan, kecamatan, kota tempat tinggal, provinsi, data lulusan dari sekolah menengah pertama mana sebelumnya, nomor rek bank, status siswa, catatan biodata, foto siswa, id bank sebagai *foreign key,* id agama sebagai *foreign key,* id tahun ajaran sebagai *foreign key* yang merupakan atribut penjelas pada tahun ajaran keberapa siswa tersebut terdaftar, id token sebagai *foreign key*, *created\_at* dan *updated\_at.* Struktur tabel siswa digambarkan seperti pada gambar 5.21.



**Gambar 5.21 Stuktur tabel students**

22. Tabel Subjects

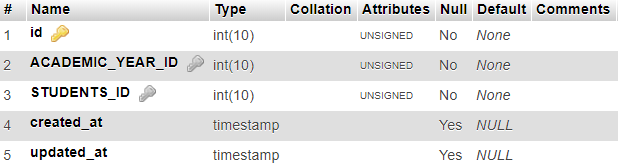
Tabel subjects adalah tabel yang digunakan untuk menyimpan daftar mata pelajaran sesuai departemen yang ada disekolah. Data yang akan dicatat dalam tabel ini yaitu kode mata pelajaran, deskripsi mata pelajaran, nilai kkm, tipe mata pelajaran, id departemen sebagai *foreign key*. Struktur data tabel subjects dapat dilihat pada tabel 5.22.



**Gambar 5.22 Struktur tabel subjects**

23. Tabel Subject\_Records

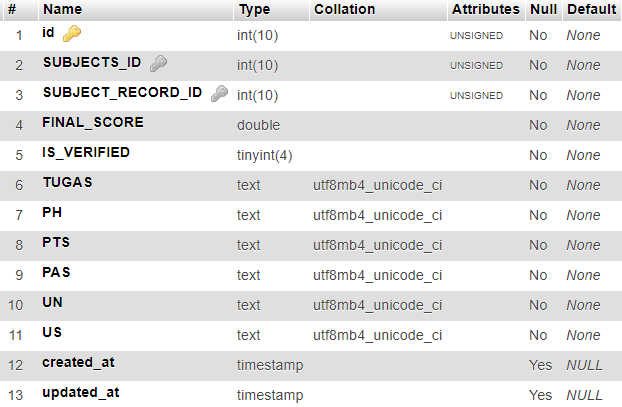
Tabel subject\_records adalah tabel yang menyimpan catatan mata pelajaran setiap satu id siswa dengan id tahun ajaran tertentu. Data yang tersimpan yaitu id tahun ajaran sebagai *foreign key,* id siswa sebagai *foreign key, created at* dan *updated at*. Struktur tabel ini digambarkan seperti pada gambar 5.23.



**Gambar 5.23 Stuktur tabel subject records**

24. Tabel Subject\_Reports

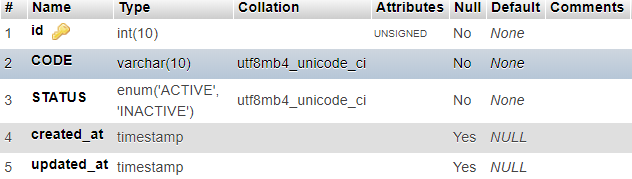
Tabel subject\_reports merupakan tabel hasil relasi *many to many* antara tabel subject\_records dan tabel subjects. Tabel ini digunakan untuk menyimpan transaksi detail setiap mata pelajaran yang terdapat dalam satu id catatan mata pelajaran. Data yang akan dicatat dalam tabel ini adalah id unik dari tabel subject\_record sebagai *foreign key*, id unik subjects sebagai *foreign key*, nilai akhir setiap mata pelajaran atau akumulasi nilai tiap-tiap mata pelajaran, verifikasi terhadap nilai tersebut apakah sudah diverifikasi atau belum. Selain itu tabel ini juga menyimpan data dalam format *json string* yang berguna untuk menyimpan rincian nilai dalam satu kolom berbentuk string yaitu tugas, ph, pts, pas, un, us, dan tentunya tanggal transaksi dicatat, dan tanggal transaksi diubah. Struktur tabel subjects\_reports dapat dilihat pada tabel 5.24.



**Gambar 5.24 Struktur tabel subject reports**

25. Tabel Tokens

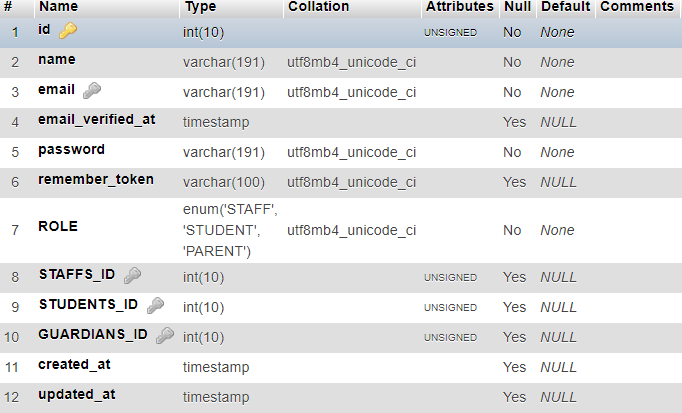
Tabel tokens digunakan untuk menyimpan nomor token setiap kali siswa terdaftar. Data yang disimpan adalah kode token, status token apakah aktif atau tidak, *created\_at* dan *updated\_at.* Struktur tabel tokens digambarkan seperti pada gambar 5.25.



**Gambar 5.25 Struktur tabel tokens**

26. Tabel Users

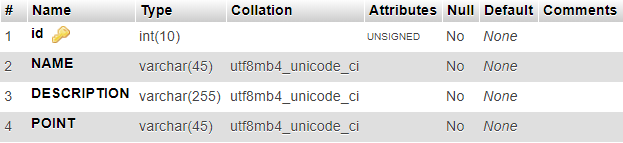
Tabel users berguna untuk menyimpan data-data user yang terdaftar pada sistem. Tabel users menyimpan data nama lengkap user, email user sebagai *key* yang unik agar user satu dengan yang lain tidak bisa memiliki email yang sama untuk login. Kemudian juga menyimpan tanggal email terverifikasi, *password* user, remember token, *role* atau hak akses user, id staff sebagai *foreign key,* id siswa sebagai *foreign key,* id wali siswa sebagai *foreign key.* Struktur tabel users digambarkan seperti pada gambar 5.26.



**Gambar 5.26 Struktur tabel users**

27. Tabel Violations

Tabel violations merupakan tabel master untuk menyimpan daftar pelanggaran yang berlaku disekolah. Data yang dicatat dalam tabel ini adalah nama pelanggaran, deskripsi pelanggaran, poin pelanggaran. Tabel ini berelasi *many to many* dengan tabel students. Struktur data tabel violations dapat dilihat pada tabel 5.27.

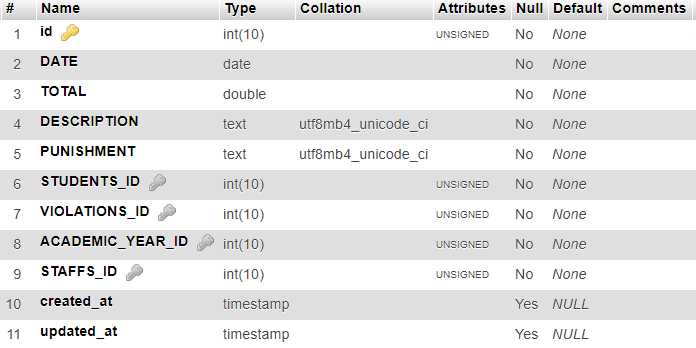


**Gambar 5.27 Struktur tabel violations**

28. Tabel Violation\_Records

Tabel violation records merupakan tabel hasil relasi antara tabel violation dengan tabel students. Tabel ini digunakan untuk mencatat transaksi detail pelanggaran yang dibuat siswa.

Data yang dicatat dan akan disimpan dalam tabel ini yaitu tanggal terjadi pelanggaran, total poin pelanggaran, deskripsi pelanggaran, bentuk sanski, id unik siswa sebagai *foreign key*, id unik pelanggaran sebagai *foreign key,* id unik tahun ajaran sebagai *foreign key*, id unik staff sebagai *foreign key,* tanggal transaksi dicatat, dan tanggal transaksi diubah. Struktur data tabel violationrecords dapat dilihat pada tabel 5.28.



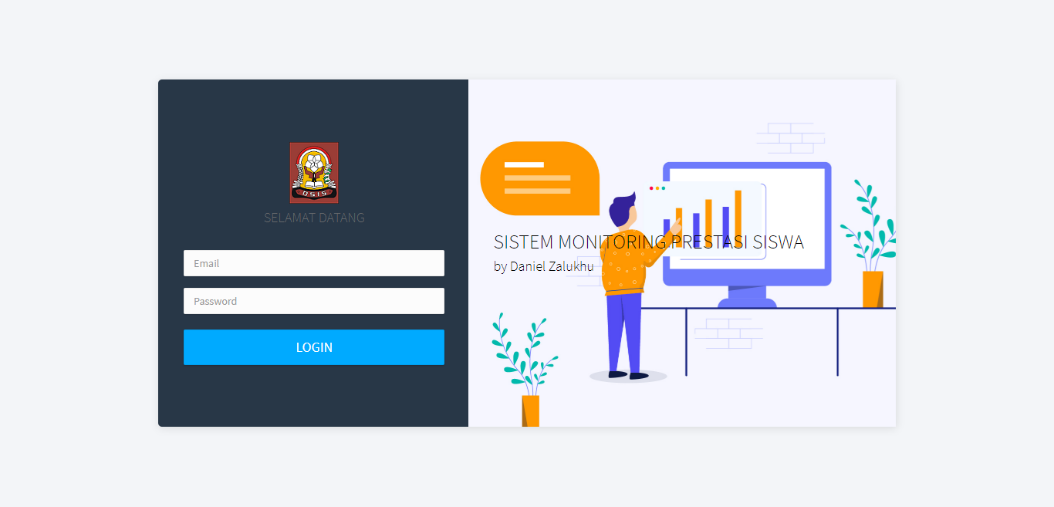
**Gambar 5.28 Struktur tabel violation records**

**5.2 Implementasi Tampilan**

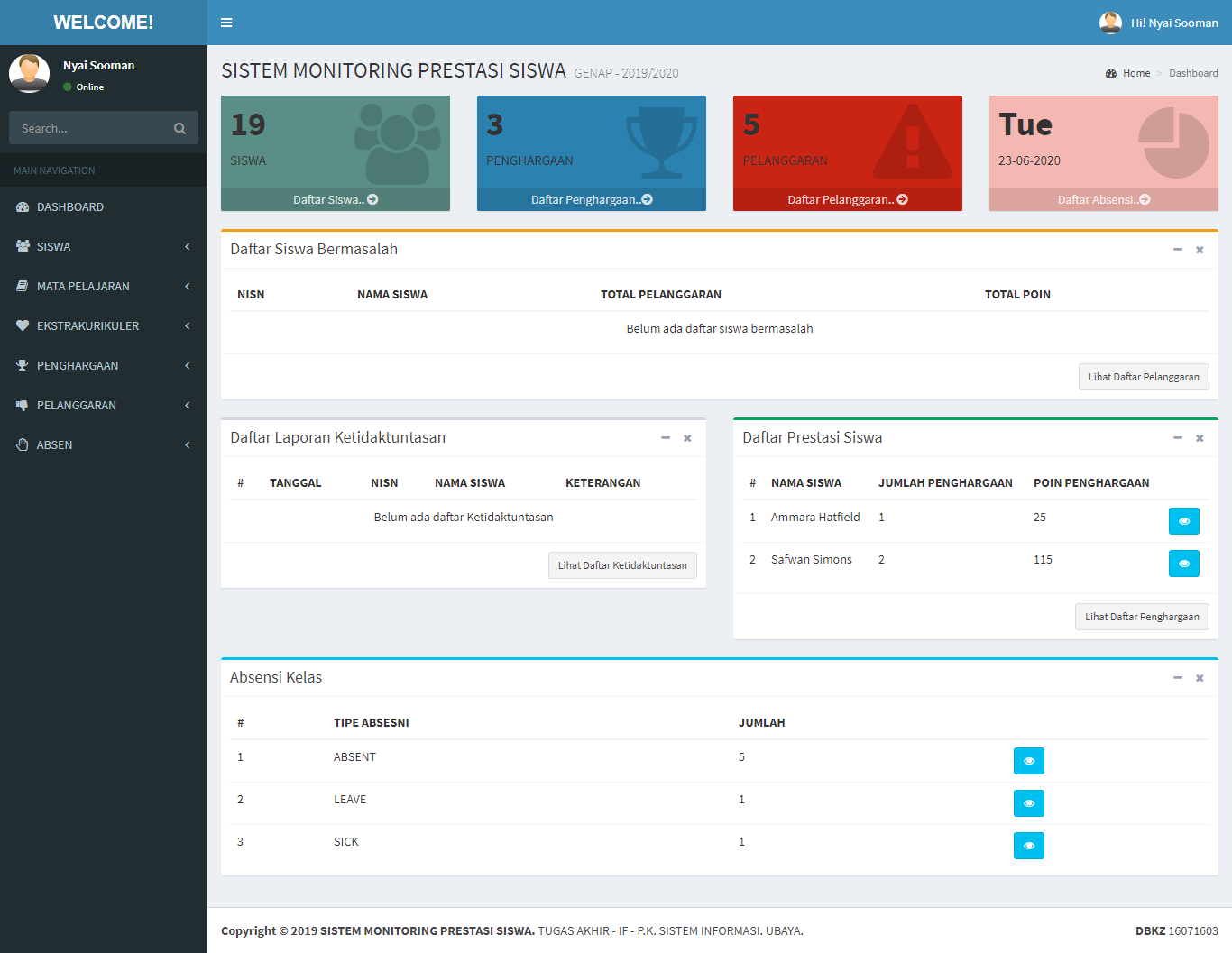
Implementasi tampilan akan menjelaskan setiap desain *user interface* yang telah dirancang untuk diimplementasikan. Pada subbab ini juga akan menjelaskan kegunaan dari setiap halaman yang ada, bagaimana proses tampilan berjalan.

* **Halaman Login**

Halaman login adalah halaman pertama yang akan dijumpai *user* ketika menggunakan sistem. Terdapat dua *field* yang harus diisi oleh *user* yaitu email dan password. Tampilan halaman *dashboard* dapat dilihat pada gambar 5.29.



* **Halaman Dashboard**



**Gambar 5.30 Tampilan halaman dashboard**

* **Halaman Daftar Siswa**

A screenshot of a computer

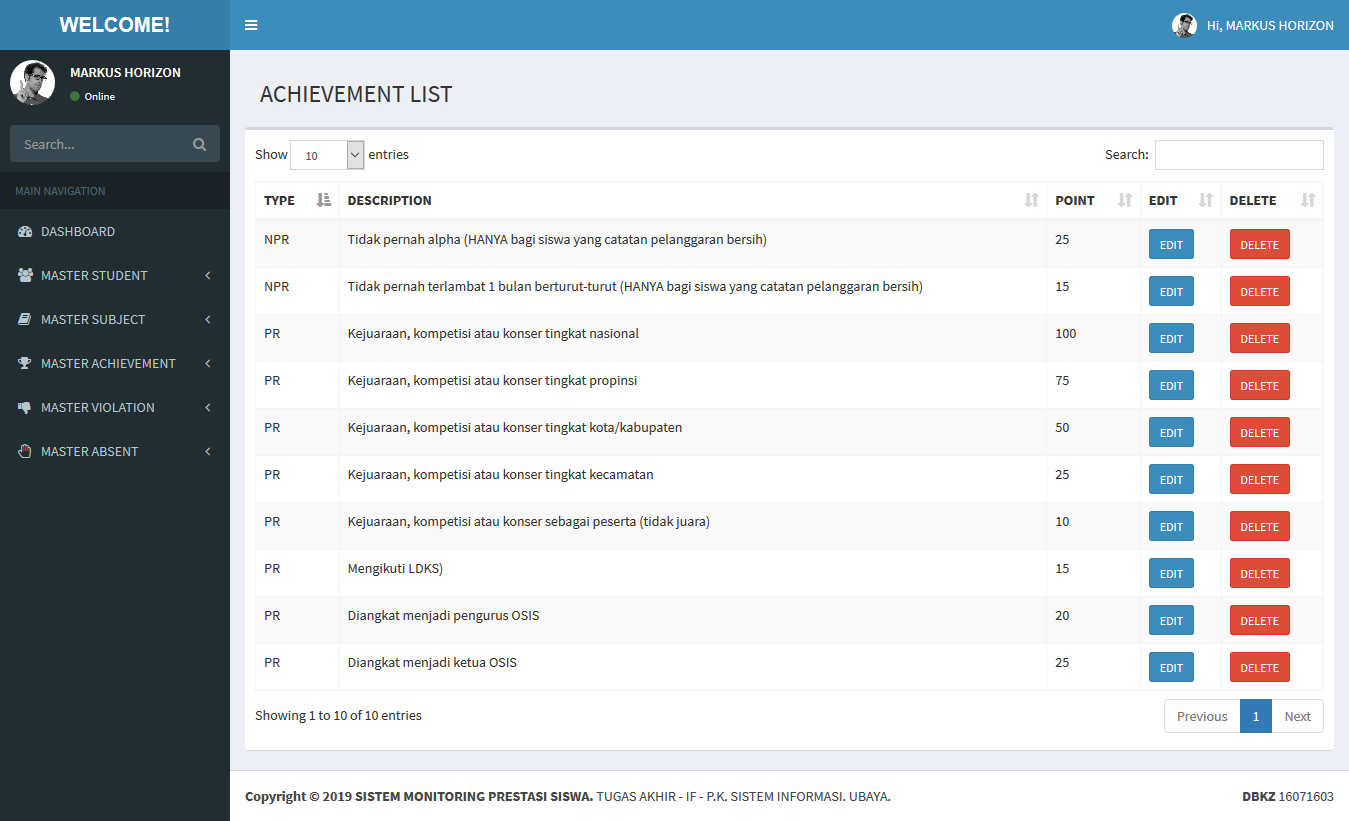
Description automatically generated

* **Halaman Profil Siswa**

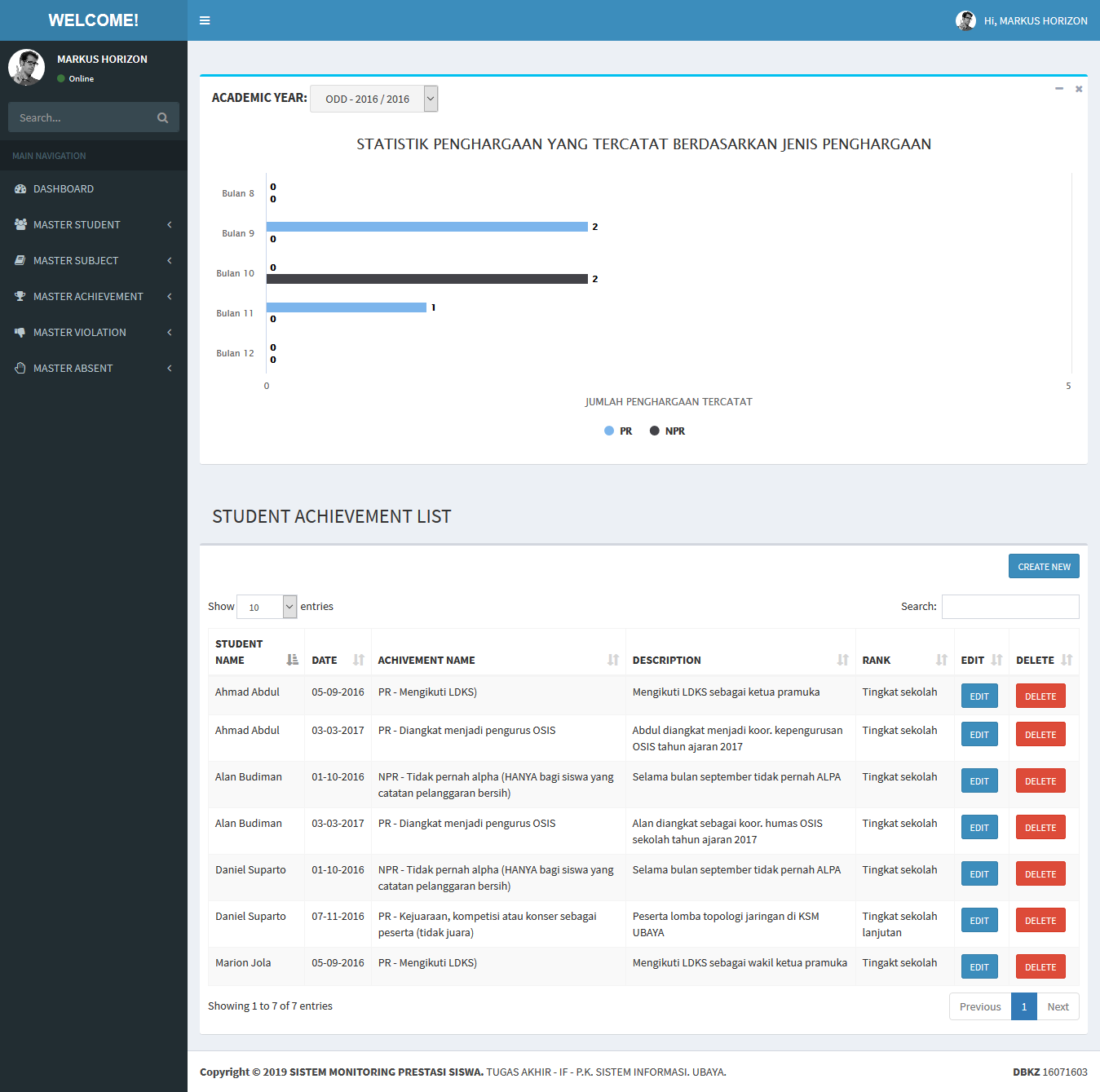
A screenshot of a computer

Description automatically generated

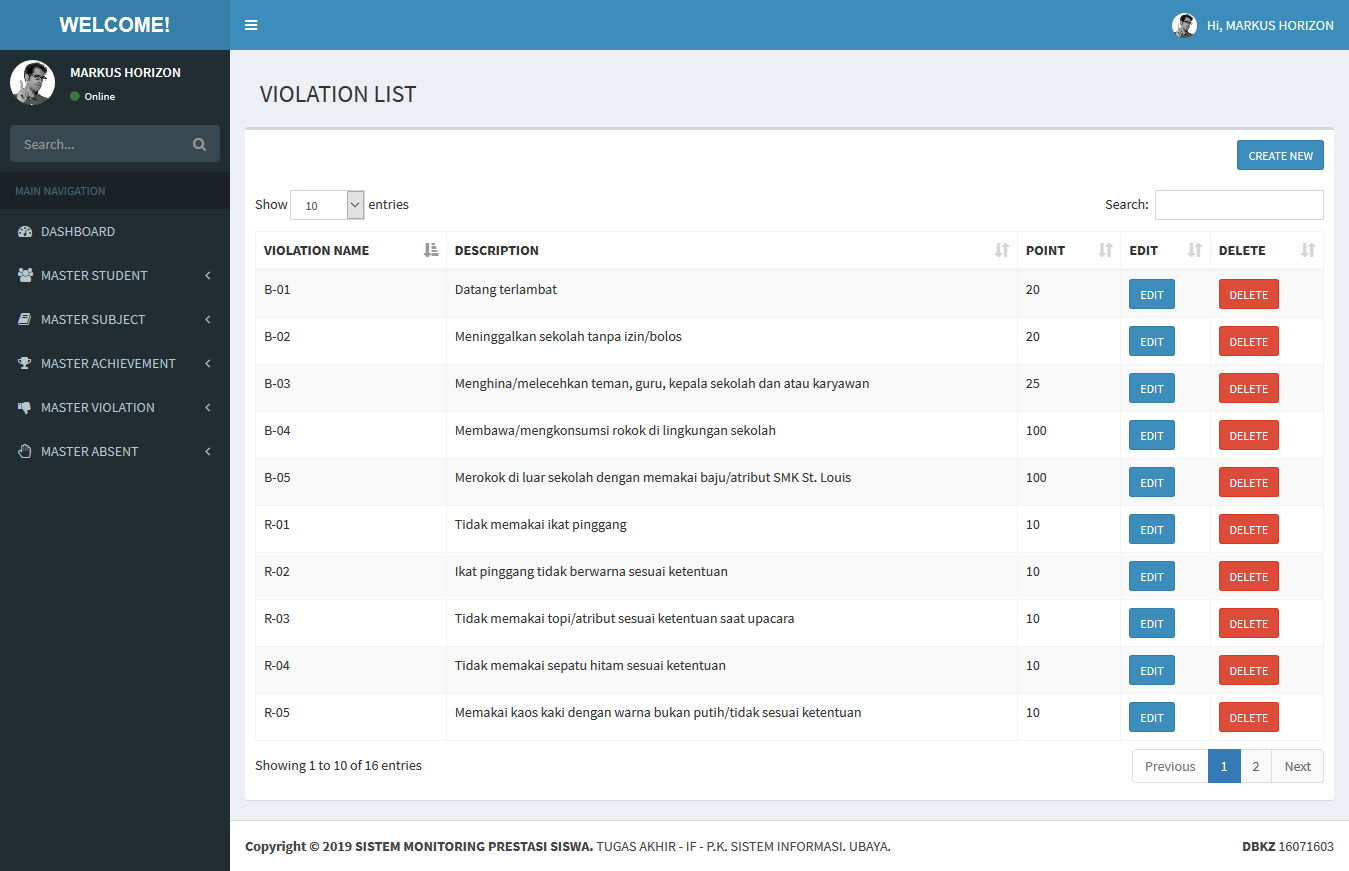
* **Halaman Daftar Penghargaan**



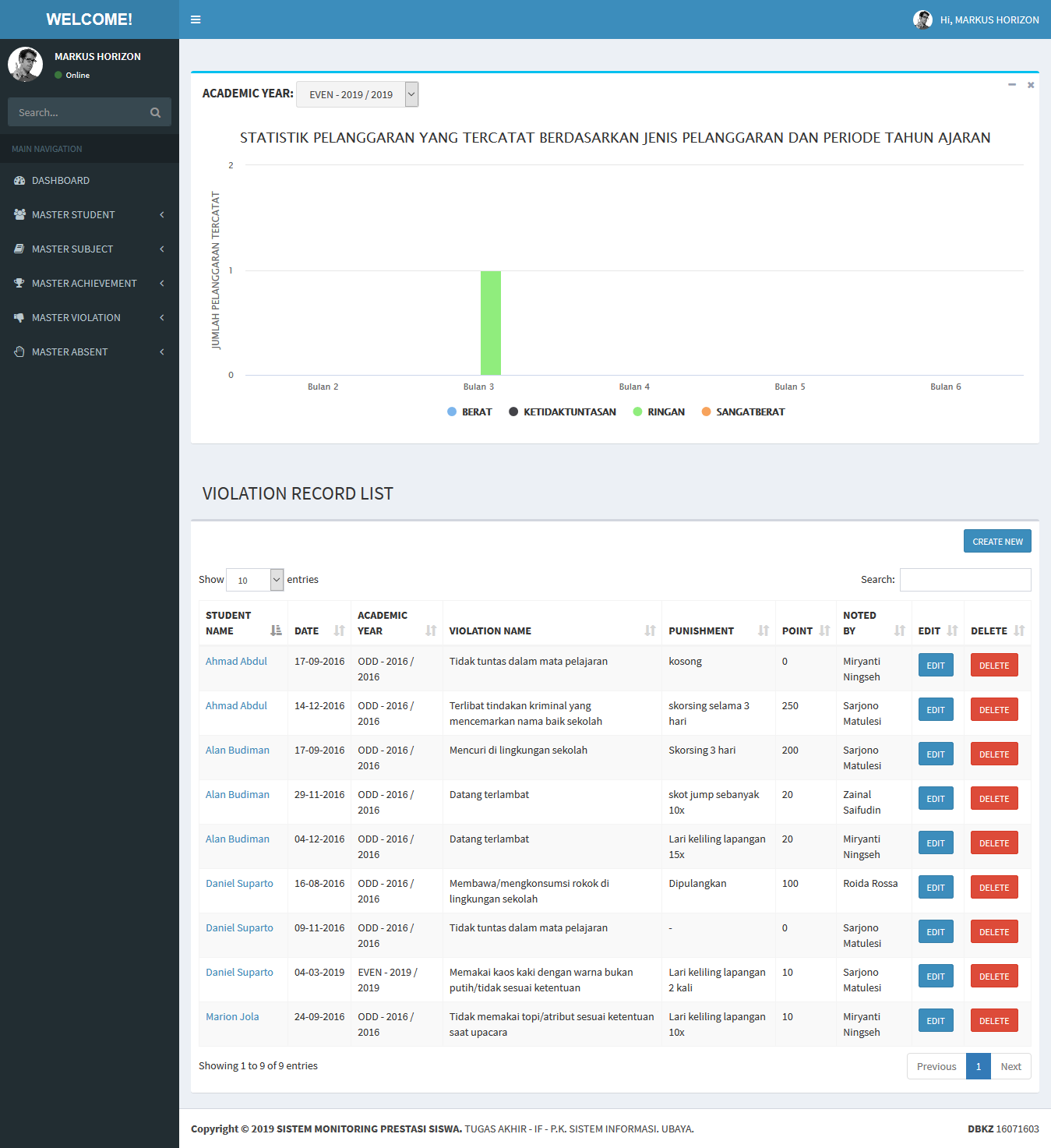
* **Halaman Daftar Catatan Penghargaan**



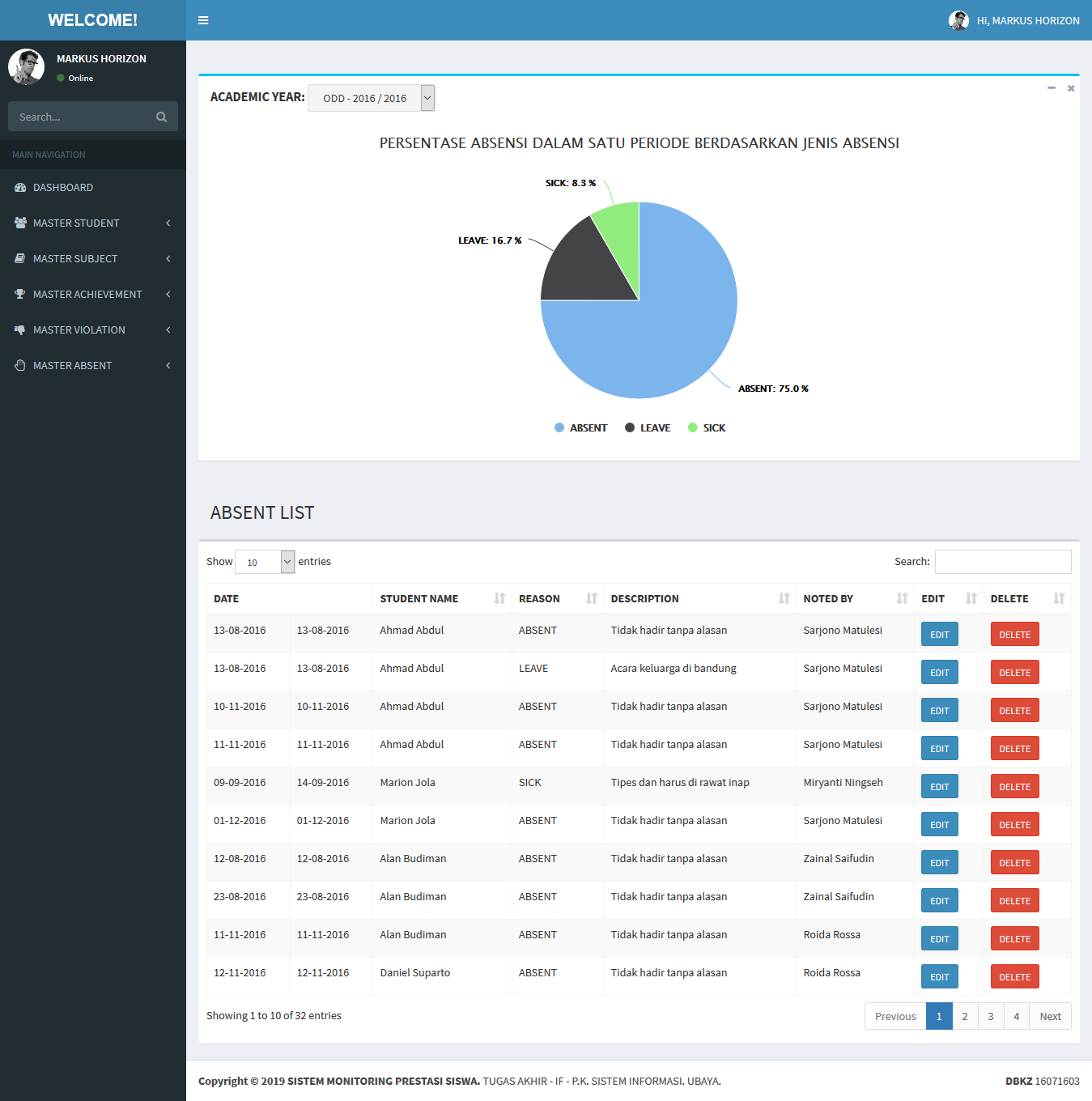
* **Halaman Daftar Pelanggaran**



* **Halaman Daftar Catatan Pelanggaran**



* **Halaman Daftar Absen**



**5.3 Implementasi Program**

Subbab implementasi program akan menjelaskan potongan kode untuk fungsi utama pada sistem. Tampilan sistem ini menggunakan *framework* yang berbasis HTML dan CSS. Pada bagian pengolahan data, sistem ini menggunakan bantuan *framework* PHP yaitu Laravel untuk memudahkan dalam membuat, mengubah, dan memporses data. Demikian juga pada implementasi program diperlukan potongan kode JQuery dan Javascript dalam menerima data.

**5.3.1 Halaman Dashboard**

Halaman *dashboard* adalah halaman utama ketika *user* masuk ke dalam sistem setelah melakukan login. Terdapat perbedaan data yang ditampilkan pada halaman ini antara *role* kepala sekolah, guru/wali kelas, dan siswa.

Kepala sekolah akan mendapatkan informasi mengenai keseluruhan data yang tercatat di database seperti total jumlah siswa, total jumlah penghargaan, total pelanggaran, daftar siswa bermasalah, daftar laporan ketidaktuntasan, daftar prestasi siswa, dan daftar absen siswa. Sementara untuk guru/wali kelas akan mendapatkan informasi data-data siswa yang dimiliki oleh kelasnya saja. Terakhir, untuk siswa hanya akan mendapatkan informasi mengenai datanya sendiri yang tercatat dalam *database.* Pada halaman ini terjadi banyak proses pengolahan data, seperti dibawah ini.

**1. Daftar Siswa Bermasalah**

Proses pemberian informasi daftar siswa bermasalah kepada kepala sekolah dapat dilihat pada potongan baris program listing 5.1. Pengolahan data untuk menampilkan siswa bermasalah dijalankan dengan *function “*getTroubleStudent”. Fungsi ini akan menghitung banyaknya jumlah pelanggaran tiap-tiap siswa, dan menjumlahkan poin pelanggaran. Hasil dari fungsi ini akan dilemparkan ke halaman *dashboard* oleh fungsi index oleh *controller dashboard.* Demikian pula untuk role guru/wali kelas, baris program dapat dilihat pada listing 5.2.

**Listing 5.1 Baris program daftar siswa bermasalah role kepala sekolah**

|  |
| --- |
| public function getTroubleStudent($iduser, $selected\_student){  $siswa = ViolationRecord::join('students', 'violation\_records.ST UDENTS\_ID', 'students.id')  ->select('students.\*',  DB::raw('COUNT(\*) as BANYAKPELANGGARAN'),  DB::raw('SUM(violation\_records.TOTAL) as TOTALPOIN'))  ->groupBy('students.id')  ->orderBy('violation\_records.id', 'DESC')->take(5)  ->having('TOTALPOIN', ">=", 50 )  ->get(); } |

**Listing 5.2 Baris program daftar siswa bermasalah role guru/wali kelas**

|  |
| --- |
| public function getTroubleStudent($iduser, $selected\_student)  {  $kelas\_guru = Grade::where('STAFFS\_ID', $iduser)->first()  ->id;    $siswa = GradeStudent::where('GRADES\_ID', $kelas\_guru)  ->where('ACADEMIC\_YEAR\_ID', $selected\_student)  ->get();    $arr\_siswa = [];    foreach($siswa as $s){         array\_push($arr\_siswa, $s->STUDENTS\_ID);    }    $siswa = ViolationRecord::join('students', 'violation\_records.STUDENTS\_ID', 'students.id')     ->select('students.\*',             DB::raw('COUNT(\*) as BANYAKPELANGGARAN'),             DB::raw('SUM(violation\_records.TOTAL) as TOTALPOIN'))     ->whereIn('violation\_records.STUDENTS\_ID', $arr\_siswa)     ->groupBy('students.id')     ->orderBy('violation\_records.id', 'DESC')->take(5)     ->having('TOTALPOIN', ">=", 50 )    ->get();  } |

**2. Daftar Laporan Ketidaktuntasan**

Pada halaman *dashboard* akan diberikan informasi mengenai laporan ketidaktuntasan. Daftar laporan ini akan ditampilkan dengan menjalankan fungsi “showStudentIncompleteness” dan “myIncompleteness” yang membutuhkan parameter id *user.* Fungsi ini akan berjalan apabila di dalam tabel *database* terdapat daftar ketidaktuntasan dan akan membatasi jumlah daftar yang ditampilkan pada halaman ini. Fungsi ini dapat dilihat pada listing 5.3 dan 5.4 untuk fungsi bagi *role* siswa.

**Listing 5.3 Baris program daftar laporan ketidaktuntasan**

|  |
| --- |
| public function showStudentIncompleteness($iduser, $selected\_student)  {  if(Auth::guard('web')->user()->staff->ROLE == "TEACHER"){    $kelas\_guru = Grade::where('STAFFS\_ID', $iduser)->first()  ->id;        $siswa = GradeStudent::where('GRADES\_ID', $kelas\_guru)  ->where('ACADEMIC\_YEAR\_ID', $selected\_student)  ->get();          $arr\_siswa = [];        foreach($siswa as $s){             array\_push($arr\_siswa, $s->STUDENTS\_ID);        }         $ketidaktuntasan = ViolationRecord::join('violations',  'violation\_records.VIOLATIONS\_ID','=','violations.id')        ->select('violation\_records.\*')         ->where('violations.NAME','TTS')         ->whereIn('violation\_records.STUDENTS\_ID', $arr\_siswa)         ->orderBy('violation\_records.id', 'DESC')->take(5)         ->get();  }    elseif(Auth::guard('web')->user()->staff  ->ROLE == "HEADMASTER")  {       $ketidaktuntasan = ViolationRecord::join('violations',  'violation\_records.VIOLATIONS\_ID','=','violations.id')             ->select('violation\_records.\*')             ->where('violations.NAME','TTS')             ->orderBy('violation\_records.id', 'DESC')->take(5)             ->get();    }    return $ketidaktuntasan;  } |

**Listing 5.3 Baris program daftar laporan ketidaktuntasan role siswa**

|  |
| --- |
| public function myIncompleteness($idsiswa)  {  $academic\_year\_id = AcademicYear::select(DB::raw('MAX(id) asid'))->get()[0]->id; |

**Listing 5.3 Baris program daftar laporan ketidaktuntasan role siswa**

|  |
| --- |
| $ketidaktuntasan = ViolationRecord::join('violations',  'violation\_records.VIOLATIONS\_ID','=','violations.id')         ->select('violation\_records.\*')         ->where('violations.NAME','TTS')         ->where('violation\_records.STUDENTS\_ID', $idsiswa)  ->where('violation\_records.ACADEMIC\_YEAR\_ID', $academic\_year\_id)         ->orderBy('violation\_records.id', 'DESC')->take(5)         ->get();    return $ketidaktuntasan;  } |

**3. Daftar Prestasi Siswa**

Halaman *dashboard* memberikan informasi mengenai prestasi/penghargaan yang dimiliki siswa. Daftar prestasi yang ditampilkan berupa nama siswa beserta dengan jumlah penghargaan yang tercatat dalam *database* dengan menjalankan fungsi “studentAchievements” dan “myAchievement” untuk *role* siswa. Potongan baris program fungsi daftar prestasi siswa dapat dilihat pada listing 5.4 dan 5.5.

**Listing 5.4 Baris program daftar prestasi siswa**

|  |
| --- |
| public function studentAchievements($iduser)  {    if(Auth::guard('web')->user()->staff->ROLE == "TEACHER")  {  $selected\_student = GradeStudent::select(DB::raw('MAX(ACADEMIC\_YEAR\_ID) AS id'))->limit(1)->first()->id;      $kelas\_guru = Grade::where('STAFFS\_ID', $iduser)->first()  ->id;     $siswa = GradeStudent::where('GRADES\_ID', $kelas\_guru)  ->where('ACADEMIC\_YEAR\_ID', $selected\_student)->get();       $arr\_siswa = [];     foreach($siswa as $s){         array\_push($arr\_siswa, $s->STUDENTS\_ID);     } |

**Listing 5.4 Baris program daftar prestasi siswa (lanjutan)**

|  |
| --- |
| $penghargaan\_siswa = AchievementRecord::join('achievements',  'achievement\_records.ACHIEVEMENTS\_ID', 'achievements.id')       ->select('achievement\_records.\*',  DB::raw('COUNT(\*) AS BANYAKPENGHARGAAN'),                 DB::raw('SUM(POINT) AS POINPENGHARGAAN'))       ->whereIn('achievement\_records.STUDENTS\_ID', $arr\_siswa)       ->groupBy('achievement\_records.STUDENTS\_ID')       ->orderBy('achievement\_records.id', 'DESC')->take(5)       ->get();  }   elseif(Auth::guard('web')->user()->staff->ROLE == “HEADMASTER”)  {  $penghargaan\_siswa = AchievementRecord::join('achievements',  'achievement\_records.ACHIEVEMENTS\_ID', 'achievements.id')  ->select('achievement\_records.\*',  DB::raw('COUNT(\*) AS BANYAKPENGHARGAAN'),                 DB::raw('SUM(POINT) AS POINPENGHARGAAN'))       ->groupBy('achievement\_records.STUDENTS\_ID')       ->orderBy('achievement\_records.id', 'DESC')->take(5)       ->get();   }   return $penghargaan\_siswa;  } |

**Listing 5.5 Baris program daftar prestasi siswa role siswa**

|  |
| --- |
| public function myAchievement($idsiswa)  {   $academic\_year\_id = AcademicYear::select(  DB::raw('MAX(id) as id'))->get()[0]->id;     $penghargaan\_siswa = AchievementRecord::join('achievements',   'achievement\_records.ACHIEVEMENTS\_ID', 'achievements.id')        ->select('achievement\_records.\*',  DB::raw('COUNT(\*) AS BANYAKPENGHARGAAN'),                  DB::raw('SUM(POINT) AS POINPENGHARGAAN'))       ->where('achievement\_records.STUDENTS\_ID', $idsiswa)        ->where('achievement\_records.ACADEMIC\_YEAR\_ID',  $academic\_year\_id)        ->groupBy('achievement\_records.STUDENTS\_ID')        ->orderBy('achievement\_records.id', 'DESC')->take(5)        ->get();          return $penghargaan\_siswa;  } |

**4. Daftar Absen**

*User* role kepala sekolah, wali kelas, dan siswaakan mendapatkan informasi tentang daftar absen. Daftar absen ini menjalankan perintah fungsi yang diatur pada *controller dashboard* yaitu “showAbsent” untuk *role* kepala sekolah dan guru/wali kelas, sedangkan funsi “showAbsentKu” untuk *role* siswa. Potongan baris program fungsi yang menampilkan daftar absen dapat dilihat pada listing 5.6 dan 5.7.

**Listing 5.6 Baris program daftar absen**

|  |
| --- |
| public function showAbsent($iduser)  {     $academic\_year\_id = AcademicYear::select(DB::raw('MAX(id) as  Id'))->get()[0]->id;      $count\_total\_day\_each\_ay = AcademicYear::select(  DB::raw('DATEDIFF(END\_DATE, START\_DATE) AS TOTALHARI'))          ->where('id', $academic\_year\_id)->first()->TOTALHARI;      if(Auth::guard('web')->user()->staff->ROLE == "TEACHER")  {        $type = Absent::select(DB::raw('TYPE AS TIPE'))  ->groupBy('TYPE')->get();          $selected\_student = GradeStudent::select(DB::raw('MAX(  ACADEMIC\_YEAR\_ID) AS id'))->limit(1)->first()->id;        $kelas\_guru = Grade::where('STAFFS\_ID', $iduser)  ->first()->id;          $siswa = GradeStudent::join('students', 'grades\_students  .STUDENTS\_ID', 'students.id')                ->select('students.id')                ->where('grades\_students.GRADES\_ID', $kelas\_guru)           ->where('grades\_students.ACADEMIC\_YEAR\_ID',  $selected\_student)  ->get();         $arr\_siswa = [];        foreach($siswa as $s){             array\_push($arr\_siswa, $s->id);        } |

**Listing 5.6 Baris program daftar absen (lanjutan)**

|  |
| --- |
| $data = Absent::select(DB::raw('TYPE AS TIPE'),  DB::raw('ACADEMIC\_YEAR\_ID AS TAHUNAJARAN'),  DB::raw('COUNT(\*) AS JUMLAH')             ->where('ACADEMIC\_YEAR\_ID', $academic\_year\_id)            ->whereIn('STUDENTS\_ID', $arr\_siswa)               ->groupBy('TIPE', 'TAHUNAJARAN')               ->get();          $absen = Absent::where('ACADEMIC\_YEAR\_ID',  $academic\_year\_id)  ->whereIn('STUDENTS\_ID', $arr\_siswa)->get();     }     elseif(Auth::guard('web')->user()->staff->ROLE ===   "HEADMASTER")  {        $data = Absent::select(DB::raw('TYPE AS TIPE'),  DB::raw('ACADEMIC\_YEAR\_ID AS TAHUNAJARAN'), DB::raw('COUNT(\*) AS JUMLAH'))                            ->where('ACADEMIC\_YEAR\_ID',  $academic\_year\_id)        ->groupBy('TIPE', 'TAHUNAJARAN')        ->get();     }       $value["data"] = $data;       return $value;  } |

**Listing 5.7 Baris program daftar absen role siswa**

**5.3.2 Daftar Siswa**

Daftar siswa merupakan halaman pada hak akses untuk kepala sekolah dan guru/wali kelas yang menampilkan daftar siswa yang terdaftar di sekolah. Jika *user* login sebagai kepala sekolah maka dapat melihat keseluruhan daftar siswa disekolah. Jika *user* login sebagai guru/wali kelas dapat melihat daftar siswa kelasnya saja. Baris program daftar siswa dapat dilihat pada listing 5.1.

**Lisitng 5.1 Baris program function daftar siswa**

|  |
| --- |
| public function index(Request $request)  {  $kelas = Grade::all();  $min\_grade\_id = Grade::select(DB::raw('MIN(id) as id'))  ->get()[0]->id;  if($request->has('gradeId')){  $grade\_id = $request->gradeId;  }  else{  $grade\_id = $min\_grade\_id;  }  $selected\_student = GradeStudent::select(  DB::raw('MAX(ACADEMIC\_YEAR\_ID) AS id'))  ->limit(1)->first()->id;  if(Auth::guard('web')->user()->staff->ROLE === "TEACHER"){  $kelas\_guru = Grade::where('STAFFS\_ID',  $request->session()->get('session\_user\_id'))  ->first()->id;  $siswa = GradeStudent::where('GRADES\_ID',$kelas\_guru)  ->where('ACADEMIC\_YEAR\_ID',  $selected\_student)->get();  }  else{  $siswa = GradeStudent::where('GRADES\_ID', $grade\_id)  ->where('ACADEMIC\_YEAR\_ID', $selected\_student)->get();  }  return view('student.index', compact('kelas', 'siswa', 'grade\_id'));  } |

**5.3.3 Input Daftar Siswa**

Untuk dapat melakukan input daftar siswa hanya bisa dilakukan oleh user dengan *role* sebagai teacher atau wali kelas. Fitur input daftar siswa terdapat di halaman daftar siswa dengan menekan tombol dibagian *header* table. Proses menginputkan data siswa dilakukan pada *function* “importStudent” pada import *controller* yang kemudian melemparkan data ke *function* berupa collection di dalam *student import collection* tersendiri. Listing 5.2 menunjukan baris program import *controller*, sedangkan listing 5.3 menunjukan baris program *function* collection.

**Listing 5.2 Baris program melempar data ke function collection**

|  |
| --- |
| public function importStudent(Request $request)  {  $this->validate($request, [  'student\_import' => 'required|mimes:csv,xls,xlsx'  ]);  Excel::import(new StudentImport($request), $request->file('student\_import'));  Session::flash('sukses','Import data siswa berhasil ditambahkan');  return redirect('student');  **}** |

**Listing 5.3 Baris program function collection input data ke database**

|  |
| --- |
| public function collection(Collection $collection)  {  foreach($collection as $key => $row)  {  if($key >= 3)  {  $grade\_id = $row[4];  $tmp\_id\_grade = $this->selectGradeId($grade\_id);    foreach($tmp\_id\_grade as $ig)  {  $id\_kelas = $ig->id;  }    if($key >= 5)  {  $student\_full\_name = $row[5];  $studentName = explode(" ", $row[5], 2);  $siswa = new Student();  $siswa->NISN = $row[4];  $siswa->FNAME = $studentName[0];  $siswa->LNAME = $studentName[1]  $siswa->ACADEMIC\_YEAR\_ID = $this->request  ->session()  ->get("session\_academic\_year\_id");  $siswa->save();  $kelas\_siswa = GradeStudent::create([  'STUDENTS\_ID' => $siswa->id,  'GRADES\_ID' => $id\_kelas,  'ACADEMIC\_YEAR\_ID' => $this->request  ->session()->get("session\_academic\_year\_id")  ]); |

**Listing 5.3 Baris program function collection input data ke database (lanjutan)**

|  |
| --- |
| $user = User::create([  'name' => $student\_full\_name,  'email' => "s".$row[4].”@student.ac.id",  'password' =>  Hash::make($studentName[0]),  'ROLE' => 'STUDENT',  'STUDENTS\_ID' => $siswa->id  ]);  }  }  }} |

**5.3.4 Halaman Profil Siswa**

Halaman profile siswa bisa diakses dari *button* yang terdapat di kolom tabel aksi tabel daftar siswa. Pada halaman profile, user bisa melihat informasi data yang menjadi tolak ukur dalam sistem ini yaitu catatan pelanggaran, catatan penghargaan, dan catatan absen masing-masing siswa. Untuk menampilkan halaman ini, data diproses pada *controller* siswa menggunakan *function* “profile”. Baris program halaman siswa dapat dilihat pada listing 5.4.

**Listing 5.4 Baris program function profile**

|  |
| --- |
| public function profile(Request $request, $id)  {    $siswa = Student::find($id);  $student\_grade\_name = $siswa->getGradeName();    $siswaku = Student::where('STUDENTS\_ID', $id);    $absen = Absent::where('STUDENTS\_ID', $id)  ->orderBy('START\_DATE', 'END\_DATE', 'ASC')     ->get();   $catatan\_absen = Absent::select('TYPE', DB::raw('COUNT(TYPE)  AS TOTAL'))  ->where('STUDENTS\_ID', $id)  ->where('ACADEMIC\_YEAR\_ID',  $request->session()->get('session\_academic\_year\_id'))    ->groupBy('TYPE')    ->get(); |

**Listing 5.4 Baris program function profile (lanjutan)**

|  |
| --- |
| $x = new DashboardController();    $kehadiran =  $x->showAbsentKu($siswa->id);    $selected\_student\_ay = Student::select('ACADEMIC\_YEAR\_ID AS  AY\_ID')->where('id', $siswa->id) ->first()  ->AY\_ID;    $tahun\_ajaran = AcademicYear::where('id', '>=', $selected\_stu dent\_ay)->get();  $catatan\_pelanggaran = ViolationRecord::join('violations',  'violation\_records.VIOLATIONS\_ID', 'violations.id')          ->select('violation\_records.\*', 'violations.\*')  ->where('STUDENTS\_ID', '=', $id)  ->where('ACADEMIC\_YEAR\_ID', '=', $request->session()  ->get('session\_academic\_year\_id'))  ->where('violations.NAME', 'NOT LIKE', 'TTS%')  ->get();  $violation\_point = ViolationRecord::select(DB::raw('SUM(TOTAL ) AS POINT'))    ->where('STUDENTS\_ID', $id)    ->first()->POINT;  $achievement\_point = AchievementRecord::join('achievements', 'achievement\_records.ACHIEVEMENTS\_ID', 'achievements.id')     ->select(DB::raw('SUM(POINT) AS POINT'))     ->where('achievement\_records.STUDENTS\_ID', $id)      ->first()->POINT;    $catatan\_penghargaan = AchievementRecord::join('achievements,'achievement\_records.ACHIEVEMENTS\_ID', 'achievements.id')     ->select('achievements.\*', 'achievement\_records.\*')     ->where('STUDENTS\_ID', '=', $id)     ->where('ACADEMIC\_YEAR\_ID', '=', $request->session()  ->get('session\_academic\_year\_id'))     ->get();  $maxId = AcademicYear::select(DB::raw('MAX(id) as id'))  ->get()[0]->id;  if($request->has('academicYearId')){     $academic\_year\_id = $request->academicYearId;  }  else{     $academic\_year\_id = $maxId;  }  $selected\_tahun\_ajaran = AcademicYear::select(DB::raw('MONTH(START\_DATE) AS STARTMONTH'),                                 DB::raw('MONTH(END\_DATE) AS ENDMONTH'))  ->where('id', $academic\_year\_id)->get()[0]; |

**Listing 5.4 Baris program function profile (lanjutan)**

|  |
| --- |
| *// GRAFIK TAB VIOLATION*  $kategori = DB::select("SELECT (CASE  WHEN v.NAME LIKE 'R%' THEN 'RINGAN'  WHEN v.NAME LIKE 'B%' THEN 'BERAT'  WHEN v.NAME LIKE 'SB%' THEN 'SANGATBERAT'  END) AS KATEGORI  FROM violation\_records vr INNER JOIN violations v ON vr.VIOLATIONS\_ID = v.id  WHERE v.NAME NOT LIKE 'TTS%'  GROUP BY KATEGORI ");    $data = ViolationRecord::join('violations', 'violation\_records.VIOLATIONS\_ID', 'violations.id')   ->select(DB::raw('(CASE  WHEN violations.NAME LIKE "R%" THEN "RINGAN"             WHEN violations.NAME LIKE "B%" THEN "BERAT"             WHEN violations.NAME LIKE "SB%" THEN "SANGATBERAT"  END) AS KATEGORI'),  DB::raw('MONTH(violation\_records.DATE) AS BULAN'),  DB::raw('COUNT(\*) AS JUMLAH')) -where('violation\_records.ACADEMIC\_YEAR\_ID', $academic\_year\_id)  ->where('violation\_records.STUDENTS\_ID', $id)  ->where('violations.NAME', 'NOT LIKE', 'TTS%')  ->groupBy('KATEGORI')->groupBy('BULAN')  ->orderBy('violation\_records.id', 'DESC')->get();  *// GRAFIK TAB ACHIEVEMENT*  $type = Achievement::select(DB::raw('GRADE AS TINGKAT'))  ->groupBy('TINGKAT')  ->get();  $dataAchievement = DB::select("SELECT a.GRADE AS TINGKAT, MONTH (ass.DATE) AS BULAN , COUNT(\*) AS JUMLAH   FROM achievements a INNER JOIN achievement\_records ass ON a.id = ass.ACHIEVEMENTS\_ID  WHERE ACADEMIC\_YEAR\_ID = " . $academic\_year\_id ."  AND STUDENTS\_ID = " . $id . "  GROUP BY TINGKAT, BULAN  ORDER BY BULAN ASC");  *// GRAFIK TAB ABSENT*  $tipeAbsen = Absent::select(DB::raw('TYPE AS TIPE'))      ->groupBy('TIPE')                    ->get();  $dataAbsen = Absent::select(DB::raw('TYPE AS TIPE'), DB::raw('ACADEMIC\_YEAR\_ID AS TAHUNAJARAN'), DB::raw('COUNT(\*) AS JUMLAH'))  ->where('ACADEMIC\_YEAR\_ID', $academic\_year\_id)  ->where('STUDENTS\_ID', $id) |

**Listing 5.4 Baris program function profile (lanjutan)**

|  |
| --- |
| ->groupBy('TIPE', 'TAHUNAJARAN')  ->get();  $count\_total\_day\_each\_ay = AcademicYear::select(DB::raw('DATEDIFF(END\_DATE, START\_DATE) AS TOTALHARI'))  ->where('id', $academic\_year\_id)->first()  ->TOTALHARI;  *// RETURN VIEW*  if(Auth::guard('web')->user()->ROLE === "STAFF")  return view('student.profile', compact('siswa', 'student\_grad e\_name', 'absen', 'catatan\_absen', 'kehadiran', 'tahun\_ajaran',  'catatan\_pelanggaran', 'violation\_point',  'catatan\_penghargaan', 'achievement\_point',  'kategori', 'data', 'selected\_tahun\_ajaran', 'academic\_year\_id',  'type', 'dataAchievement','tipeAbsen', 'dataAbsen', 'count\_total\_day\_each\_ay'));  else  return view('student.profile', compact('siswa', 'absen', 'catatan\_absen', 'kehadiran', 'tahun\_ajaran',  'catatan\_pelanggaran', 'violation\_point',  'catatan\_penghargaan', 'achievement\_point',  'kategori', 'data', 'selected\_tahun\_ajaran', 'academic\_year\_id', 'type', 'dataAchievement',  'tipeAbsen', 'dataAbsen', 'count\_total\_day\_each\_ay')); |

**5.3.5 Halaman Mapel-Guru**

Halaman mapel guru adalah salah satu sub-menu dari menu mata pelajaran. Halaman ini bisa diakses hanya oleh *user* sebagai wali kelas. Untuk menampilkan halaman ini diperlukan session id user sebagai parameter dari *function* “mapelguru”. Fungsi ini memproses data user terlebih dahulu *user* berada pada departemen id berapa kemudian mencari id kelas yang diajar sehingga baru di proses untuk menemukan kode-kode mata pelajaran yang berada dalam kelas *user.* Baris program daftar mapel-guru dapat dilihat pada listing 5.5.

**Lisiting 5.5 Baris program function mapelguru**

|  |
| --- |
| public function mapelguru(Request $request)  {  $get\_department\_id = DepartmentStaff::select(  'DEPARTMENTS\_ID')  ->where('STAFFS\_ID', $request->session()  ->get('session\_user\_id'))  ->get();    $get\_grade\_id = Grade::select('NAME')  ->where('STAFFS\_ID', $request->session()  ->get('session\_user\_id'))  ->get();    $a = array($get\_department\_id, $get\_grade\_id);  $tmp\_subject = [];  foreach($get\_department\_id as $deparment\_id){  foreach($get\_grade\_id as $grade\_id){  $selected\_mapel = Subject::select('id', 'CODE'  ,'DESCRIPTION')  ->where('DEPARTMENTS\_ID',$deparment\_id->DEPARTMENTS\_ID)  ->where('CODE', 'LIKE', '%' . $grade\_id->NAME . '%')  ->get();  array\_push($tmp\_subject, $selected\_mapel);  }  }  $kelas = Grade::all();  $gname = Grade::select('NAME')  ->where('id', Grade::select(DB::raw('MIN(id) as id'))  ->first()->id)->get()[0]->NAME;    $gid = Grade::select(DB::raw('MIN(id) as id'))->first()->id;  if($request->has('gradeName')){  $grade\_name = $request->gradeName;  }  else{  $grade\_name = $gname;  }  if(Auth::guard('web')->user()->staff->ROLE === "TEACHER")  $subject = $tmp\_subject[0];  elseif(Auth::guard('web')->user()->staff->ROLE  === "HEADMASTER")  $subject = Subject::where('CODE', 'LIKE', '%' .  $grade\_name . '%')->get();    return view('student.mapel-guru', compact('subject', 'kelas'  ,'grade\_name', 'gid'));  } |

**5.3.5 Halaman Detail Mata Pelajaran**

Halaman detail mata pelajaran adalah halaman dimana *user* bisa melihat rincian setiap nilai yang tercatat dalam *database.* Dalam halaman ini terdapat 2 buah kolom informasi yang diberikan kepada user yaitu kolom grafik yang sebagai informasi yang menujukan ada atau tidaknya kenaikan/menurunnya nilai siswa, dan tabel rinciannya. Data pada halaman ini diproses menggunakan *function* “subjectDetail” yang dibedakan cara memproses data untuk setiap *role.*

1. **Halaman detail mata pelajaran untuk *user role* wali kelas**

Halaman detail mata pelajaran untuk *user role* wali kelas akan menampilkan data nilai seluruh siswanya pada tabel. Sedangkan informasi pada bagian grafik memberikan data nilai rata-rata kelas, jumlah siswa yang nilai akhirnya berada dibawah nilai ketuntasan minimal/KKM dan KKM itu sendiri. Baris program detail mata pelajaran pada tabel dapat dilihat pada listing 5.6. Sedangkan baris program data nilai dalam bentuk grafik dapat dilihat pada listing 5.7.

**Listing 5.6 Baris program detail mata pelajaran pada tabel**

|  |
| --- |
| public function subjectDetail(Request $request, $id)  {      $mapel = Subject::find($id);      $max\_academic\_year\_id = AcademicYear::select(DB::raw  ('MAX(id)as id'))->get()[0]->id;      if($request->has('academicYearId')){          $academic\_year\_id = $request->academicYearId;      }      else{          $academic\_year\_id = $max\_academic\_year\_id;      }        $kkm = Subject::select('MINIMALPOIN')->where('id', $mapel  ->id)->get();      $x = explode("-", $mapel->CODE, 3);      $y = $x[0] . "-" . $x[1]; |

**Listing 5.6 Baris program detail mata pelajaran pada tabel (lanjutan)**

|  |
| --- |
| $url = parse\_url(url()->previous(), PHP\_URL\_QUERY);      $get\_url\_param = explode("=", $url);      $siswa\_dibawah\_rata = [];  if(Auth::guard('web')->user()->ROLE === "STAFF"){         $select\_grade\_id = Grade::select('id')  ->where('NAME', $y)->first()->id;           $selected\_student\_in\_ay = GradeStudent::select(  DB::raw('MAX(ACADEMIC\_YEAR\_ID) AS id'))->limit(1)  ->first()->id;         $selected\_student\_ay = Student::select('students  .ACADEMIC\_YEAR\_ID AS AY\_ID')  ->join('grades\_students', 'students.id',  'grades\_students.STUDENTS\_ID')    ->where('grades\_students.GRADES\_ID', $select\_grade\_id)  ->limit(1)->first()->AY\_ID;           $tahun\_ajaran = AcademicYear::where('id', '>=',  $selected\_student\_ay)->get();           $detail\_mapel = SubjectReport::join('subject\_records',  'subject\_reports.SUBJECT\_RECORD\_ID',  'subject\_records.id')         ->join('subjects', 'subject\_reports.SUBJECTS\_ID',  'subjects.id')         ->join('students', 'subject\_records.STUDENTS\_ID',  'students.id')         ->join('grades\_students', 'students.id',  'grades\_students.STUDENTS\_ID')        ->select('subjects.\*', 'subject\_records.\*',  'subject\_reports.\*')       ->where('subject\_records.ACADEMIC\_YEAR\_ID',  $academic\_year\_id)         ->where('subject\_reports.SUBJECTS\_ID', $mapel->id)         ->where('grades\_students.GRADES\_ID', '=',  $select\_grade\_id)  ->where('grades\_students.ACADEMIC\_YEAR\_ID',  $selected\_student\_in\_ay)  ->get();           $rata\_kelas = SubjectReport::join('subject\_records',  'subject\_reports.SUBJECT\_RECORD\_ID',  'subject\_records.id')        ->join('subjects', 'subject\_reports.SUBJECTS\_ID',  'subjects.id')  ->join('students', 'subject\_records.STUDENTS\_ID',  'students.id')        ->join('grades\_students', 'students.id',  'grades\_students.STUDENTS\_ID') |

**Listing 5.6 Baris program detail mata pelajaran pada tabel (lanjutan)**

|  |
| --- |
| ->select(DB::raw('ROUND(SUM(subject\_reports.FINAL\_SCORE)/  COUNT(subject\_records.STUDENTS\_ID), 2) AS RATAKELAS'),  'subject\_records.\*')        ->where('subject\_reports.SUBJECTS\_ID', $mapel->id)        ->where('grades\_students.GRADES\_ID', '=',  $select\_grade\_id)        ->where('grades\_students.ACADEMIC\_YEAR\_ID',  $selected\_student\_in\_ay)        ->groupBy('subject\_records.ACADEMIC\_YEAR\_ID')        ->get();          foreach($rata\_kelas as $rk)  {         if(!empty($rk))           $siswa\_dibawah\_rata = SubjectReport::join  ('subject\_records',  'subject\_reports.SUBJECT\_RECORD\_ID',  'subject\_records.id')            ->join('students', 'subject\_records.STUDENTS\_ID',  'students.id')           ->join('subjects', 'subject\_reports.SUBJECTS\_ID',    'subjects.id')            ->join('grades\_students', 'students.id',  'grades\_students.STUDENTS\_ID')          ->select('subject\_records.\*', DB::raw('COUNT(\*)  AS JUMLAH\_SISWA\_DIBAWAH\_RATA'))            ->where('subject\_reports.SUBJECTS\_ID', $mapel->id)            ->where('subject\_reports.FINAL\_SCORE', '<',  $rk->RATAKELAS)            ->where('grades\_students.GRADES\_ID', '=',  $select\_grade\_id)            ->where('grades\_students.ACADEMIC\_YEAR\_ID', $selected\_student\_in\_ay)                                                 ->groupBy('subject\_records.ACADEMIC\_YEAR\_ID')            ->get();          else              $siswa\_dibawah\_rata = [];        }     }  } |

**Listing 5.7 Baris program data nilai dalam bentuk grafik**

|  |
| --- |
| var categories = {!! json\_encode($tahun\_ajaran) !!}      var kkm = {!! json\_encode($kkm) !!}      var averageScorePerClass = {!! json\_encode($rata\_kelas) !!}      var badStudent = {!! json\_encode($siswa\_dibawah\_rata) !!}      var category = []  var series1 = []      var series2 = []      var series3 = [] |

**Listing 5.7 Baris program data nilai dalam bentuk grafik (lanjutan)**

|  |
| --- |
| categories.forEach(function(item){          badStudent.forEach(function(obj\_x){              if(obj\_x.ACADEMIC\_YEAR\_ID == item.id){                  var catName = item.TYPE + " " + item.NAME                  category.push(catName)                  averageScorePerClass.forEach(function(obj\_rata){                      if(obj\_rata.ACADEMIC\_YEAR\_ID == item.id){                          var tmp\_rata = obj\_rata.RATAKELAS                          series3.push(tmp\_rata)                           kkm.forEach(function(obj\_kkm){                              var tmp\_kkm = obj\_kkm.MINIMALPOIN                              series2.push(tmp\_kkm)                          })                      }                  })   var tmp\_jumlah = obj\_x.JUMLAH\_SISWA\_DIBAWAH\_RATA                  series1.push(tmp\_jumlah)              }          })       })       Highcharts.chart('gradeOfSubjectChart', {          chart: {              type: 'line'          },          title: {              text: 'Statistik Nilai Akhir Kelas Per Semester'          },          xAxis: {              categories: category          },  yAxis: {              title: {                  text: 'Nilai'              }          },          plotOptions: {              line: {                  dataLabels: {                      enabled: true                  },                  enableMouseTracking: true              }},          credits: {              enabled: false  }, series: [{              name: 'Jumlah siswa nilai akhir dibawah rata kelas',              data: series1          }, |

**Listing 5.7 Baris program data nilai dalam bentuk grafik (lanjutan)**

|  |
| --- |
| {    name: 'KKM',              data: series2          },{              name: 'Nilai rata-rata kelas',              data: series3 }]}); |

1. **Filter tahun ajaran untuk tabel detail mata pelajaran wali kelas**

Pada halaman detail mata pelajaran bagi wali kelas juga disediakan satu *combobox* yang berisikan daftar tahun ajaran. *Combobox* filter tahun ajaran ini berguna bagi wali kelas melihat/mencari detail-detail nilai mata pelajaran pada tahun ajaran tertentu. Adapun baris program filter tahun ajaran untuk tabel detail mata pelajaran dapat dilihat pada listing 5.8.

**Listing 5.8 Baris program filter tahun ajaran untuk tabel detail mata pelajaran**

|  |
| --- |
| $('#dropdown-detail-subject-academicyear’).val( {{$academic\_year\_id}})  $('#dropdown-detail-subject-academic-year').change(function(){     var academicYearId = $(this).val()     var route = "{{ route('subject.detail', ':id') }}"     route = route.replace(':id', '{{ $mapel->id }}')     window.location = route+"?academicYearId="+academicYearId;  }) |

1. **Halaman detail mata pelajaran untuk *user role* siswa**

Halaman detail mata pelajaran untuk *user role* siswa akan menampilkan data nilai siswa itu sendiri. Rincian nilai setiap tahun ajaran permata pelajaran akan ditampilkan pada bagian grafik maupun tabel. Baris program detail mata pelajaran pada tabel dapat dilihat pada listing 5.9. Sedangkan baris program data nilai dalam bentuk grafik dapat dilihat pada listing 5.7 sama seperti untuk bagian wali kelas.

**Listing 5.9 Baris program detail nilai mata pelajaran siswa**

|  |
| --- |
| public function subjectDetail(Request $request, $id)  {    $siswa = Student::find($request->session()  ->get('session\_student\_id'));    $selected\_student\_ay = Student::select('ACADEMIC\_YEAR\_ID AS  AY\_ID')  ->where('id', $request->session()->get('session\_student\_id'))  ->first()->AY\_ID;    $tahun\_ajaran = AcademicYear::where('id', '>=',  $selected\_student\_ay)->get();    $detail\_mapel\_ku = SubjectReport::join('subject\_records',  'subject\_reports.SUBJECT\_RECORD\_ID',  'subject\_records.id')    ->join('subjects', 'subject\_reports.SUBJECTS\_ID',  'subjects.id')    ->select('subjects.\*', 'subject\_records.\*',  'subject\_reports.\*')    ->where('subjects.DESCRIPTION', $mapel->DESCRIPTION)  *->where('subject\_records.ACADEMIC\_YEAR\_ID', $academic\_year\_id)*    ->where('subject\_records.STUDENTS\_ID', $request->session()  ->get('session\_student\_id'))    ->get();      $data\_final\_score = SubjectReport::join('subject\_records',  'subject\_reports.SUBJECT\_RECORD\_ID', 'subject\_records.id')    ->join('subjects', 'subject\_reports.SUBJECTS\_ID',  'subjects.id')   ->select('subject\_records.\*', 'subject\_reports.\*')  ->where('subjects.DESCRIPTION', $mapel- >DESCRIPTION)    ->where('subject\_records.STUDENTS\_ID', $request->session()- >get('session\_student\_id'))->get();    if(isset($url)){       $select\_grade\_id = Grade::select('id')  ->where('NAME', $get\_url\_param[1])->first()->id;       $selected\_student\_in\_ay = GradeStudent::select  ('ACADEMIC\_YEAR\_ID')       ->where('STUDENTS\_ID', $request->session()  ->get('session\_student\_id'))  ->where('GRADES\_ID', $select\_grade\_id)  ->first()->ACADEMIC\_YEAR\_ID;    } |

**Listing 5.9 Baris program detail nilai mata pelajaran siswa (lanjutan)**

|  |
| --- |
| else{        $select\_grade\_id = Grade::select('id')  ->where('NAME', $y)->first()->id;        $selected\_student\_in\_ay = GradeStudent::select(DB::raw  ('MAX(ACADEMIC\_YEAR\_ID) AS id'))->limit(1)  ->first()->id;    }      $rata\_kelas = SubjectReport::join('subject\_records',  'subject\_reports.SUBJECT\_RECORD\_ID',  'subject\_records.id')    ->join('subjects', 'subject\_reports.SUBJECTS\_ID',  'subjects.id')    ->join('students', 'subject\_records.STUDENTS\_ID',   'students.id')    ->join('grades\_students', 'students.id',  'grades\_students.STUDENTS\_ID')    ->select(DB::raw('ROUND(SUM(subject\_reports.FINAL\_SCORE)/  COUNT(subject\_records.STUDENTS\_ID), 2) AS RATAKELAS'),  'subject\_records.\*')    ->where('subjects.DESCRIPTION', $mapel->DESCRIPTION)    ->where('grades\_students.GRADES\_ID', '=', $select\_grade\_id)    ->where('grades\_students.ACADEMIC\_YEAR\_ID',  $selected\_student\_in\_ay)    ->groupBy('subject\_records.ACADEMIC\_YEAR\_ID')    ->get();  } |

**5.3.6 Halaman Daftar Laporan Ketidaktuntasan**

Halaman daftar laporan ketidaktuntasan adalah halaman yang bisa diakses oleh *user* dengan *role* sebagai kepala sekolah, guru, dan siswa. Jika *user* sebagai kepala sekolah maka hanya bisa melihat isi data yang ditampilkan. Sedangkan *user* sebagai guru/wali kelas bisa melakukan penambahan, pengubahan, atau menghapus daftar laporan. Begitu juga siswa, siswa hanya dapat laporan mengenai dirinya sendiri. Adapun proses yang terdapat pada halaman ini seperti berikut.

**1. Menginputkan daftar laporan ketidaktuntasan**

Proses menginputkan daftar laporan ketidaktuntasan hanya bisa dilakukan oleh guru/wali kelas. Proses ini terjadi ketika *user* menekan tombol “buat daftar ketidaktuntasan” pada bagian atas dari daftar tabel yang disediakan. Untuk melakukan proses penginputan catatan tersebut diperlukan beberapa *field* yang sebagai *request* data dari form input yang kemudian ditangkap oleh *controller* dan menciptakan baris baru dalam tabel *database.* Listing 5.10 dibawah ini menunjukan bari program get *request* data dari form input pada halaman daftar laporan ketidaktuntasan.

**Listing 5.10 Baris program input daftar laporan ketidaktuntasan**

|  |
| --- |
| public function storeIncomplete(Request $request)  {    $point = 0;    $hukuman = "-";    $date = $request->get('vr\_date');    $request\_date = Carbon::parse($date);    $session\_start\_ay = $request->session()  ->get('session\_start\_ay');    $start\_ay = Carbon::parse($session\_start\_ay);   $session\_end\_ay = $request->session()->get('session\_end\_ay');    $end\_ay = Carbon::parse($session\_end\_ay);    $check = $request\_date->between($start\_ay,$end\_ay);    if($check == false)    {      return redirect('incomplete')->with('error',  'Input tanggal tidak sesuai dengan tahun ajaran  yang berlaku');    }    else    {      $ketidaktuntasan = new ViolationRecord([                  'DATE' => $request->get('vr\_date'),                  'STUDENTS\_ID' => $request  ->get('vr\_student\_name'),                  'DESCRIPTION' => $request  ->get('vr\_description'),                  'PUNISHMENT' => $hukuman,                  'VIOLATIONS\_ID' => $request  ->get('vr\_violation\_name'), |

**Listing 5.10 Baris program input daftar laporan ketidaktuntasan (lanjutan)**

|  |
| --- |
| 'STAFFS\_ID' => $request->session()  ->get('session\_user\_id'),                  'TOTAL' => $point,                  'ACADEMIC\_YEAR\_ID' => $request->session()  ->get('session\_academic\_year\_id')              ]);              $ketidaktuntasan->save();              return redirect('incomplete')  ->with('sukses',  'Incomplete Report has been created');          }      } |

**2. Menginputkan daftar laporan ketidaktuntasan**

Proses mengubah daftar laporan ketidaktuntasan hanya bisa dilakukan oleh *user* sebagai guru/wali kelas. Proses ini terjadi ketika guru menekan tombol “edit” pada kolom aksi pada tabel tersedia. Pengubahan data ini berdasarkan setiap id dari daftar ketidaktuntasan. Setiap id yang ada memiliki isi data yang akan ditampilkan dalam form halaman edit laporan ketidaktuntasan. Baris program edit laporan ketidaktuntasan seperti pada listing 5.11.

**Listing 5.11 Baris program ubah daftar laporan ketidaktuntasan**

|  |
| --- |
| public function updateIncomplete(Request $request, $id){          $ketidaktuntasan = ViolationRecord::find($id);          $point = 0;          $hukuman = "-";          $date = $request->get('vr\_date');          $request\_date = Carbon::parse($date);          $session\_start\_ay = $request->session()  ->get('session\_start\_ay');          $start\_ay = Carbon::parse($session\_start\_ay);          $session\_end\_ay = $request->session()  ->get('session\_end\_ay');          $end\_ay = Carbon::parse($session\_end\_ay);          $check = $request\_date->between($start\_ay,$end\_ay);       if($check == false){ |

**Listing 5.11 Baris program ubah daftar laporan ketidaktuntasan (lanjutan)**

|  |
| --- |
| return redirect(action('SubjectController@editIncomplete',  $ketidaktuntasan->id))  ->with('error',  'Input tanggal tidak sesuai dengan tahun ajaran yang berlaku');      }   else{              $ketidaktuntasan->DATE = $request->get('vr\_date');              $ketidaktuntasan->DESCRIPTION = $request  ->get('vr\_desc');              $ketidaktuntasan->STUDENTS\_ID = $request  ->get('vr\_student\_name');              $ketidaktuntasan->VIOLATIONS\_ID = $request  ->get('vr\_violation\_name');              $ketidaktuntasan->ACADEMIC\_YEAR\_ID = $request  ->session()->get('session\_academic\_year\_id');              $ketidaktuntasan->STAFFS\_ID = $request->session()  ->get('session\_user\_id');              $ketidaktuntasan->PUNISHMENT = $hukuman;              $ketidaktuntasan->TOTAL = $point;          }  $ketidaktuntasan->save();  return redirect(action('SubjectController@incomplete',  $ketidaktuntasan->id))->with('sukses',  'Laporan ketidaktuntasan berhasil diubah');  } |

**3. Menghapus daftar laporan ketidaktuntasan**

Proses menghapus daftar laporan ketidaktuntasan hanya bisa dilakukan oleh *user* sebagai guru/wali kelas. Proses ini terjadi ketika guru menekan tombol “delete” pada kolom aksi pada tabel tersedia. Menghapus data ini berdasarkan setiap id dari daftar ketidaktuntasan. Baris program hapus laporan ketidaktuntasan seperti pada listing 5.12.

**Listing 5.12 Baris program hapus laporan ketidaktuntasan**

|  |
| --- |
| public function destroyIncomplete($id)  {     $ketidaktuntasan = ViolationRecord::whereId($id)  ->firstOrFail();     $ketidaktuntasan->delete(); |

**Listing 5.12 Baris program hapus laporan ketidaktuntasan (lanjutan)**

|  |
| --- |
| return redirect(action('SubjectController@incomplete'))  ->with('sukses', 'Laporan ketidaktuntasan berhasil dibuat');  } |

**5.3.7 Input Nilai Mata Pelajaran**

Proses input nilai setiap mata pelajaran hanya dapat dilakukan oleh *user* guru/wali kelas. Penginputan nilai dengan cara mengimport file *excel* yang telah disediakan kedalam sistem. Pengolahan data terjadi ketika *user* menginputkan file yang ditangkap oleh *controller* import dan dilemparkan ke dalam fungsi *collection* pada *controller* import subject. Baris program import file bisa dilihat pada listing 5.13. Sedangkan untuk menampung semua data yang diimport dapat dilihat pada listing 5.20.

Pada tahap pengolahan nilai, sistem akan membaca terlebih dahulu setiap baris pada file *excel* untuk mendapatkan id dari data tersebut seperti nama mata pelajaran, kelas, nisn siswa, jenis aktivitas, nomor kd. Untuk mendapatkan id setiap data maka dibutuhkan fungsi untuk mengambil id sesuai kondisi data. Potongan kecil fungsi akan dipakai dalam fungsi pengolahan nilai. Potongan fungsi-fungsi kecil tersebut dapat dilihat pada listing 5.14 untuk mengambil deskripsi mata pelajaran, 5.15 untuk mendapatkan nama kelas, 5.16 untuk mendapatkan id siswa, 5.17 untuk mendatapkan id aktivitas, 5.18 untuk mendapatkan id nomor kd, dan 5.19 untuk mendapatkan id mata pelajaran. Baris program proses pengolahan nilai dapat dilihat pada listing 5.20.

**Listing 5.13 Baris program import file excel**

|  |
| --- |
| public function importAssesment(Request $request)      {        $this->validate($request, [          'assesment\_import' => 'required|mimes:csv,xls,xlsx'        ]);          Excel::import(new SubjectImport($request), $request  ->file('assesment\_import'));        Session::flash('sukses','Import penilaian siswa berhasil  ditambahkan');        return redirect('subject/assesment');      } |

**Listing 5.14 Baris program fungsi get deksripsi mata pelajaran**

|  |
| --- |
| public function selectSubjectDescription($value)      {          $mapel = DB::SELECT('SELECT DESCRIPTION                               FROM subjects                               WHERE subjects.DESCRIPTION = "' .  $value . '"');          return $mapel;      } |

**Listing 5.15 Baris program fungsi get nama kelas**

|  |
| --- |
| public function selectGradeCode($value)      {          $kelas = DB::SELECT('SELECT NAME                               FROM grades                               WHERE grades.NAME = "' . $value . '"') ;          return $kelas;      } |

**Listing 5.16 Baris program fungsi get id siswa**

|  |
| --- |
| public function selectStudentId($value)      {          $siswa = DB::SELECT('SELECT id                               FROM students                               WHERE students.NISN =  "' . $value . '"');          return $siswa;      } |

**Listing 5.17 Baris program fungsi get id aktivitas**

|  |
| --- |
| public function selectActivityId($value)      {          $aktivitas = DB::SELECT('SELECT id                                   FROM activities                                   WHERE activities.MODULE =   "' . $value . '"');          return $aktivitas;      } |

**Listing 5.18 Baris program fungsi get id kompetensi dasar**

|  |
| --- |
| public function selectKdId($value)      {          $kd = DB::SELECT('SELECT id                            FROM KD                            WHERE NUMBER =  "' . $value . '"');          return $kd;      } |

**Listing 5.19 Baris program fungsi get id mata pelajaran**

|  |
| --- |
| public function selectSubjectId($subjectName, $gradeCode)      {          $mapel = Subject::select('id')                          ->where('DESCRIPTION', $subjectName)                          ->where('CODE',  'LIKE' , '%'. $gradeCode .'%')                          ->get();          return $mapel;      } |

**Listing 5.20 Baris program fungsi collection pengolahan data import**

|  |
| --- |
| public function collection(Collection $collection){          foreach($collection as $key => $row)          {              if($key >= 1)              {                  $subject\_name = $row[4];                  $tmp\_mapel = $this  ->selectSubjectDescription($subject\_name);                  foreach($tmp\_mapel as $tmp)                  {                      $mapel = $tmp->DESCRIPTION;                  } |

**Listing 5.20 Baris program fungsi collection pengolahan data import (lanjutan)**

|  |
| --- |
| if($key >= 3)                  {                      $grade\_code = $row[4];                      $tmp\_grade = $this  ->selectGradeCode($grade\_code);                      foreach($tmp\_grade as $tm)                      {                          $kodeKelas = $tm->NAME;                      }                      if($key >= 5)                      {                          $student\_id = $row[4];                          $id\_siswa = $this  ->selectStudentId($student\_id)[0]->id;                          $activity\_id = explode("; ", $row[6])[1];                          $id\_aktivitas = $this  ->selectActivityId($activity\_id)[0]->id;                          $kd\_id = $row[7];                          $id\_kd = $this->selectKdId($kd\_id)[0]  ->id;                          $nilai = $row[8];                          $tmp\_id\_mapel = $this  ->selectSubjectId($mapel, $kodeKelas);                          foreach($tmp\_id\_mapel as $tim)                          {                              $id\_mapel = $tim->id;                          }                          $a = ActivityStudent::create([                              'STUDENTS\_ID' => $id\_siswa,                              'ACTIVITIES\_ID' => $id\_aktivitas,                              'SUBJECTS\_ID' => $id\_mapel,                              'ACADEMIC\_YEAR\_ID' => $this->request  ->session()  ->get("session\_academic\_year\_id"),                              'SCORE' => $nilai,                          ]);                          ActivityKD::create([                              'KD\_ID' => $id\_kd,                              'ACTIVITIES\_ID' => $id\_aktivitas,                          ]);                      }                  }                }          }          $this->formula($a->SUBJECTS\_ID);      } |

**Listing 5.21 Baris program pengolahan nilai pada fungsi formula**

|  |
| --- |
| public function formula($mapel\_id)  {     $tmp\_nilai = ActivityStudent::join('students',  'activities\_students.STUDENTS\_ID', 'students.id')     ->select('students.\*', 'activities\_students.\*')     ->where('activities\_students.SUBJECTS\_ID', $mapel\_id)     ->where('activities\_students.ACADEMIC\_YEAR\_ID',  $this->request->session()->get("session\_academic\_year\_id"))     ->get();       $arrayOfStudent = ActivityStudent::join('students',  'activities\_students.STUDENTS\_ID', 'students.id')     ->select('students.\*', 'activities\_students.\*')     ->where('activities\_students.SUBJECTS\_ID', $mapel\_id)     ->where('activities\_students.ACADEMIC\_YEAR\_ID',  $this->request->session()->get("session\_academic\_year\_id"))     ->groupBy('activities\_students.STUDENTS\_ID')     ->get();       foreach ($arrayOfStudent as $student)     {        $tmp\_nama = $student->FNAME ." ". $student->LNAME;        $tmp\_nisn = $student->NISN;        $tmp\_tugas = [];        $tmp\_ph = [];        $tmp\_pts = [];        $tmp\_pas = [];        $tmp\_us = [];        $tmp\_un = [];        $subject\_record = new SubjectRecord();        $subject\_record->ACADEMIC\_YEAR\_ID = $this->request  ->session()->get("session\_academic\_year\_id");        $subject\_record->STUDENTS\_ID = $student->STUDENTS\_ID;        $subject\_record->save();          foreach ($tmp\_nilai as $nilai)        {          if($nilai->STUDENTS\_ID === $student->STUDENTS\_ID){              if($nilai->ACTIVITIES\_ID === 1){                  array\_push($tmp\_tugas, $nilai->SCORE);           }elseif($nilai>ACTIVITIES\_ID === 2){                                array\_push($tmp\_ph, $nilai->SCORE);              }else if($nilai>ACTIVITIES\_ID === 3){  array\_push($tmp\_pts, $nilai->SCORE);              }else if($nilai->ACTIVITIES\_ID === 4){                  array\_push($tmp\_pas, $nilai->SCORE);              }else if($nilai->ACTIVITIES\_ID === 5){                  array\_push($tmp\_us, $nilai->SCORE);              }else if($nilai->ACTIVITIES\_ID === 6){                  array\_push($tmp\_un, $nilai->SCORE);              }          }} |

**Listing 5.21 Baris program pengolahan nilai pada fungsi formula**

|  |
| --- |
| $count\_tugas = $this->countActivity($student  ->STUDENTS\_ID, ($student->ACTIVITIES\_ID = 1));        $sum\_score\_tugas = $this->sumTotalScore($student  ->STUDENTS\_ID, ($student->ACTIVITIES\_ID = 1));        $count\_ph = $this->countActivity($student  ->STUDENTS\_ID, ($student->ACTIVITIES\_ID = 2));        $sum\_score\_ph = $this->sumTotalScore($student  ->STUDENTS\_ID, ($student->ACTIVITIES\_ID = 2));        $count\_pts = $this->countActivity($student  ->STUDENTS\_ID, ($student->ACTIVITIES\_ID = 3));        $sum\_score\_pts = $this->sumTotalScore($student  ->STUDENTS\_ID, ($student->ACTIVITIES\_ID = 3));       $count\_pas = $this->countActivity($student  ->STUDENTS\_ID, ($student->ACTIVITIES\_ID = 4));        $sum\_score\_pas = $this->sumTotalScore($student  ->STUDENTS\_ID, ($student->ACTIVITIES\_ID = 4));        $count\_us = $this->countActivity($student  ->STUDENTS\_ID, ($student->ACTIVITIES\_ID = 5));        $sum\_score\_us = $this->sumTotalScore($student  ->STUDENTS\_ID, ($student->ACTIVITIES\_ID = 5));        $count\_un = $this->countActivity($student  ->STUDENTS\_ID, ($student->ACTIVITIES\_ID = 6));       $sum\_score\_un = $this->sumTotalScore($student  ->STUDENTS\_ID, ($student->ACTIVITIES\_ID = 6));        foreach($count\_tugas as $ct){            $tmp\_ct = $ct->BANYAKAKTIVITAS;            foreach($sum\_score\_tugas as $sct){                $tmp\_sct = $sct->TOTALNILAI;            }            $avarage\_tugas = ($tmp\_sct / $tmp\_ct) \* 0.1;        }        foreach($count\_ph as $cph){            $tmp\_cph = $cph->BANYAKAKTIVITAS;            foreach($sum\_score\_ph as $scph){                $tmp\_scph = $scph->TOTALNILAI;            }            $avarage\_ph = ($tmp\_scph / $tmp\_cph) \* 0.2;        }        foreach($count\_pts as $cpts){            $tmp\_cpts = $cpts->BANYAKAKTIVITAS;            foreach($sum\_score\_pts as $scpts){                $tmp\_scpts = $scpts->TOTALNILAI;  }        $avarage\_pts = ($tmp\_scpts / $tmp\_cpts) \* 0.3;       }  foreach($count\_pas as $cpas){            $tmp\_cpas = $cpas->BANYAKAKTIVITAS;            foreach($sum\_score\_pas as $scpas){                $tmp\_scpas = $scpas->TOTALNILAI;            }            $avarage\_pas = ($tmp\_scpas / $tmp\_cpas) \* 0.4;        } |

**Listing 5.21 Baris program pengolahan nilai pada fungsi formula**

|  |
| --- |
| $total\_score\_student = $avarage\_tugas + $avarage\_ph +  $avarage\_pts + $avarage\_pas;       $fix\_total\_score\_round = round($total\_score\_student, 2);         $subject\_report = SubjectReport::create([                  'SUBJECTS\_ID' => $student->SUBJECTS\_ID,                  'SUBJECT\_RECORD\_ID' => $subject\_record->id,                  'FINAL\_SCORE' => $fix\_total\_score\_round,                  'IS\_VERIFIED' => 0,                  'TUGAS' => json\_encode($tmp\_tugas),                  'PH' => json\_encode($tmp\_ph),                  'PTS' => json\_encode($tmp\_pts),                  'PAS' => json\_encode($tmp\_pas),                  'US' => json\_encode($tmp\_us),                  'UN' => json\_encode($tmp\_un),        ]);    }  } |

**5.3.8 Ubah Nilai Mata Pelajaran**

Setiap data yang telah berhasil diimport maka akan tersimpan. Namun, tidak menutupi adanya kesalahan penginputan data maka bisa dilakukan pengubahan data pada halaman *subject assessment* sebelum nilai verifikasi. Proses pengubahan nilai menjalankan satu fungsi “updateAssesment”yang terjadi pada *controller* mata pelajaran. Baris program pengubahan data nilai yang diinput bisa dilihat pada listing 5.22.

**Listing 5.22 Baris program pengubahan nilai yang diinputkan**

|  |
| --- |
| public function updateAssesment(Request $request, $id)    {          $laporan\_mapel = SubjectReport::find($id);          $array\_tugas = $request->get('a\_nilai\_tugas');          $array\_ph = $request->get('a\_nilai\_ph');          $array\_pts = $request->get('a\_nilai\_pts');          $array\_pas = $request->get('a\_nilai\_pas');          $array\_un = array();          $array\_us = array();         $final\_score = 0; |

**Listing 5.22 Baris program pengubahan nilai yang diinputkan (lanjutan)**

|  |
| --- |
| $sum\_nilai\_tugas = 0;          $rata\_nilai\_tugas = 0;          $final\_nilai\_tugas = 0;          foreach($array\_tugas as $tugas){              $sum\_nilai\_tugas = $sum\_nilai\_tugas + $tugas;          }          $rata\_nilai\_tugas = $sum\_nilai\_tugas /  count($array\_tugas);          $final\_nilai\_tugas = $rata\_nilai\_tugas \* 0.1;          $sum\_nilai\_ph = 0;          $rata\_nilai\_ph = 0;          $final\_nilai\_ph = 0;          foreach($array\_ph as $ph){              $sum\_nilai\_ph = $sum\_nilai\_ph + $ph;          }          $rata\_nilai\_ph = $sum\_nilai\_ph / count($array\_ph);          $final\_nilai\_ph = $rata\_nilai\_ph \* 0.2;          $sum\_nilai\_pts = 0;          $rata\_nilai\_pts = 0;          $final\_nilai\_pts = 0;          foreach($array\_pts as $pts){              $sum\_nilai\_pts = $sum\_nilai\_pts + $pts;          }          $rata\_nilai\_pts = $sum\_nilai\_pts / count($array\_pts);          $final\_nilai\_pts = $rata\_nilai\_pts \* 0.3;          $sum\_nilai\_pas = 0;          $rata\_nilai\_pas = 0;          $final\_nilai\_pas = 0;          foreach($array\_pas as $pas){              $sum\_nilai\_pas = $sum\_nilai\_pas + $pas;          }          $rata\_nilai\_pas = $sum\_nilai\_pas / count($array\_pas);          $final\_nilai\_pas = $rata\_nilai\_pas \* 0.4;          $final\_score = $final\_nilai\_tugas + $final\_nilai\_ph +  $final\_nilai\_pts + $final\_nilai\_pas;          $fix\_final\_score = round($final\_score, 2);          $laporan\_mapel->SUBJECTS\_ID = $request  ->get('a\_subject\_id');          $laporan\_mapel->SUBJECT\_RECORD\_ID = $laporan\_mapel  ->SUBJECT\_RECORD\_ID;          $laporan\_mapel->TUGAS = json\_encode($array\_tugas);          $laporan\_mapel->PH = json\_encode($array\_ph);          $laporan\_mapel->PTS = json\_encode($array\_pts);          $laporan\_mapel->PAS = json\_encode($array\_pas);          $laporan\_mapel->UN = json\_encode($array\_un);          $laporan\_mapel->US = json\_encode($array\_us);          $laporan\_mapel->FINAL\_SCORE = $fix\_final\_score;          $laporan\_mapel->save(); |

**Listing 5.22 Baris program pengubahan nilai yang diinputkan (lanjutan)**

|  |
| --- |
| return redirect(action  ('SubjectController@assesmentImport',  $laporan\_mapel->id))  ->with('sukses',  'Data nilai siswa berhasil diubah');      } |

**5.3.9 Hapus Nilai yang Diinputkan**

Nilai-nilai mata pelajaran siswa yang telah diinputkan, dapat dihapus sebelumnilai diverifikasi. Dengan menjalankan fungsi “destroyAssesment” untuk setiap id nilai siswa yang dipilih maka data akan terhapus. Baris program hapus nilai dapat dilihat pada listing 5.23.

**Listing 5.23 Baris program hapus nilai yang diinput**

|  |
| --- |
| public function destroyAssesment($id)  {  $laporan\_mapel = SubjectReport::whereId($id)  ->firstOrFail();   $laporan\_mapel->delete();   return redirect(action('SubjectController@assesmentImport'))  ->with('sukses', 'Data nilai siswa berhasil dihapus');  } |

**5.3.10 Verifikasi Nilai yang Diinputkan**

Nilai-nilai mata pelajaran siswa yang telah diinputkan, dapat diubah dan dihapus sebelumnilai diverifikasi. Proses verifikasi terjadi ketika *user* menekan tombol hijau pada laman *subject assessment* dan menjalankan fungsi “setStatus” seperti pada potongan kode yang dapat dilihat di listing 5.24.

**Listing 5.24 Baris program verifikasi nilai**

|  |
| --- |
| public function setStatus(Request $request)  {     $laporan\_mapel = SubjectReport::where('IS\_VERIFIED', 0)  ->get();     foreach($laporan\_mapel as $lm){        $lm->IS\_VERIFIED = $request->status;        $lm->save();     }     return redirect(action('SubjectController@assesmentImport'))  ->with('sukses', 'Daftar nilai berhasil  di verifikasi dan tersimpan!');  } |

**5.3.11 Daftar Catatan Penghargaan**

Daftar catatan penghargaan akan menampilkan daftar catatannya seperti nama siswa, kelas, jumlah penghargaan dan total poin penghargaan dengan menggunakan sebuah fungsi pada *controller* penghargaan yaitu “showAchievement” seperti pada listing 5.25. Daftar catatan penghargaan juga menampilkan grafik jumlah catatan setiap tahun ajaran dengan menggunakan fungsi *javascript* yang mendapat data dari *contoller* untuk data grafik seperti pada listing 5.26.

**Listing 5.25 Baris program fungsi show achievement**

|  |
| --- |
| public function showAchievement($user\_id, $ay)  {    $selected\_student = GradeStudent::select(DB::raw('MAX  (ACADEMIC\_YEAR\_ID) AS id'))->limit(1)->first()->id;     $nasional = Achievement::select('id')  ->where('GRADE', 'Nasional')->get();     $propinsi = Achievement::select('id')  ->where('GRADE', 'Propinsi')->get();     $kota = Achievement::select('id')->where('GRADE', 'Kota')  ->get();     $sekolah = Achievement::select('id')  ->where('GRADE', 'Sekolah')->get();     if(Auth::guard('web')->user()->staff->ROLE == "TEACHER"){ |

**Listing 5.25 Baris program fungsi show achievement (lanjutan)**

|  |
| --- |
| $kelas\_guru = Grade::where('STAFFS\_ID', $user\_id)  ->first()->id;        $penghargaan = AchievementRecord::join('achievements',  'achievement\_records.ACHIEVEMENTS\_ID', 'achievements.id')        ->join('students', 'achievement\_records.STUDENTS\_ID',  'students.id')        ->join('grades\_students', 'students.id',  'grades\_students.STUDENTS\_ID')        ->select('achievement\_records.\*',          DB::raw('COUNT(\*) AS JUMLAHPENGHARGAAN'),         DB::raw('SUM(achievements.POINT) AS POINPENGHARGAAN'))      ->where('grades\_students.GRADES\_ID', $kelas\_guru)        ->where('achievement\_records.ACADEMIC\_YEAR\_ID', $ay)        ->where('grades\_students.ACADEMIC\_YEAR\_ID',  $selected\_student)        ->groupBy('achievement\_records.STUDENTS\_ID')        ->get();          $count\_nasional = $this->countAchievementPerGrade  ($nasional, $ay, $kelas\_guru, $selected\_student)  ->first()->JMLH;        $count\_propinsi = $this->countAchievementPerGrade  ($propinsi, $ay, $kelas\_guru, $selected\_student)  ->first()->JMLH;        $count\_kota = $this->countAchievementPerGrade  ($kota, $ay, $kelas\_guru, $selected\_student)  ->first()->JMLH;        $count\_sekolah = $this->countAchievementPerGrade  ($sekolah, $ay, $kelas\_guru, $selected\_student)  ->first()->JMLH;          $count\_achievement\_list = Achievement::select  (DB::raw('COUNT(\*) AS JUMLAH'))->first()->JUMLAH;          $res\_nasional = ROUND(($count\_nasional /  $count\_achievement\_list) \* 100);       $res\_propinsi = ROUND(($count\_propinsi /  $count\_achievement\_list) \* 100);        $res\_kota = ROUND(($count\_kota /  $count\_achievement\_list) \* 100);        $res\_sekolah = ROUND(($count\_sekolah /  $count\_achievement\_list) \* 100);  $data = DB::select("SELECT GRADE AS TINGKAT,  COUNT(\*) AS JUMLAH,        (SELECT (CASE WHEN TINGKAT = 'Kota' THEN  $res\_kota                      WHEN TINGKAT = 'Nasional' THEN $res\_nasional                      WHEN TINGKAT = 'Propinsi' THEN $res\_propinsi                      WHEN TINGKAT = 'Sekolah' THEN $res\_sekolah  END))                 AS PERSENTASE |

**Listing 5.25 Baris program fungsi show achievement (lanjutan)**

|  |
| --- |
| FROM achievement\_records ar INNER JOIN achievements a ON  ar.ACHIEVEMENTS\_ID = a.id         INNER JOIN students s ON ar.STUDENTS\_ID = s.id         INNER JOIN grades\_students gs ON gs.STUDENTS\_ID = s.id          WHERE ar.ACADEMIC\_YEAR\_ID = " . $ay . "  AND gs.GRADES\_ID  = " . $kelas\_guru . " AND  gs.ACADEMIC\_YEAR\_ID = ". $selected\_student ."         GROUP BY TINGKAT         ORDER BY TINGKAT DESC");  }     elseif(Auth::guard('web')->user()->staff  ->ROLE == "HEADMASTER")  {    $penghargaan = AchievementRecord::join('achievements',  'achievement\_records.ACHIEVEMENTS\_ID', 'achievements.id')         ->join('students', 'achievement\_records.STUDENTS\_ID',  'students.id')        ->join('grades\_students', 'students.id',  'grades\_students.STUDENTS\_ID')         ->select('achievement\_records.\*',           DB::raw('COUNT(\*) AS JUMLAHPENGHARGAAN'),           DB::raw('SUM(achievements.POINT) AS POINPENGHARGAAN'))         ->where('achievement\_records.ACADEMIC\_YEAR\_ID', $ay)         ->where('grades\_students.ACADEMIC\_YEAR\_ID',  $selected\_student)         ->groupBy('achievement\_records.STUDENTS\_ID')         ->get();           $count\_nasional = $this->countAchievementPerGradeKepsek  ($nasional, $ay, $selected\_student)->first()->JMLH;         $count\_propinsi = $this->countAchievementPerGradeKepsek  ($propinsi, $ay, $selected\_student)->first()->JMLH;         $count\_kota = $this->countAchievementPerGradeKepsek  ($kota, $ay, $selected\_student)->first()->JMLH;  $count\_sekolah = $this->countAchievementPerGradeKepsek  ($sekolah, $ay, $selected\_student)->first()->JMLH;    $count\_achievement\_list = Achievement::select  (DB::raw('COUNT(\*) AS JUMLAH'))->first()->JUMLAH;           $res\_nasional = ROUND(($count\_nasional /  $count\_achievement\_list) \* 100);         $res\_propinsi = ROUND(($count\_propinsi /  $count\_achievement\_list) \* 100);         $res\_kota = ROUND(($count\_kota /  $count\_achievement\_list) \* 100);         $res\_sekolah = ROUND(($count\_sekolah /  $count\_achievement\_list) \* 100); |

**Listing 5.25 Baris program fungsi show achievement (lanjutan)**

|  |
| --- |
| $data = DB::select("SELECT GRADE AS TINGKAT,  COUNT(\*) AS JUMLAH,  (SELECT(CASE WHEN TINGKAT = 'Kota' THEN  $res\_kota                     WHEN TINGKAT = 'Nasional' THEN $res\_nasional                     WHEN TINGKAT = 'Propinsi' THEN $res\_propinsi                     WHEN TINGKAT = 'Sekolah' THEN $res\_sekolah                END))                AS PERSENTASE       FROM achievement\_records ar INNER JOIN achievements a ON  ar.ACHIEVEMENTS\_ID = a.id        INNER JOIN students s ON ar.STUDENTS\_ID = s.id        INNER JOIN grades\_students gs ON gs.STUDENTS\_ID = s.id          WHERE ar.ACADEMIC\_YEAR\_ID = " . $ay . "  AND gs.ACADEMIC\_YEAR\_ID = ". $selected\_student ."        GROUP BY TINGKAT        ORDER BY TINGKAT DESC");     }     else{        $penghargaan = AchievementRecord::join('achievements',  'achievement\_records.ACHIEVEMENTS\_ID', 'achievements.id')        ->join('students', 'achievement\_records.STUDENTS\_ID',  'students.id')        ->join('grades\_students', 'students.id',  'grades\_students.STUDENTS\_ID')        ->select('achievement\_records.\*',          DB::raw('COUNT(\*) AS JUMLAHPENGHARGAAN'),          DB::raw('SUM(achievements.POINT) AS POINPENGHARGAAN'))        ->where('achievement\_records.ACADEMIC\_YEAR\_ID', $ay)        ->where('grades\_students.ACADEMIC\_YEAR\_ID',  $selected\_student)        ->groupBy('achievement\_records.STUDENTS\_ID')        ->get();     }       $value["penghargaan"] = $penghargaan;     $value["data"] = $data;     return $value;  } |

**Listing 5.26 Baris program informasi penghargaan dengan grafik**

|  |
| --- |
| var types = {!!json\_encode($type)!!}      var dataGraph = {!!json\_encode($data)!!}      var selectedTahunAjaran = {!!json\_encode  ($selected\_tahun\_ajaran)!!}      var startMonth = selectedTahunAjaran.STARTMONTH;      var endMonth = selectedTahunAjaran.ENDMONTH;      var dataSeries = []      var dataCategory = [] |

**Listing 5.26 Baris program informasi penghargaan dengan grafik (lanjutan)**

|  |
| --- |
| types.forEach(function(item){       var values = []  for(var i=startMonth; i <= endMonth; i++){              values[i-startMonth] = 0              dataCategory.push('Bulan ' + i)          }          dataGraph.forEach(function(obj){              if(obj.TINGKAT == item.TINGKAT){                  values[obj.BULAN - startMonth] = obj.JUMLAH              }          })          dataSeries.push({              name: item.TINGKAT,              data: values          })      })      Highcharts.chart('achievementChartStatistic', {          chart: {              type: 'column'          },          title: {              text: 'STATISTIK PENGHARGAAN YANG TERCATAT  BERDASARKAN JENIS PENGHARGAAN DAN PERIODE TAHUN AJARAN'          },          xAxis: {              categories: dataCategory,              crosshair: true          },          yAxis: {              min: 0,              allowDecimals: false,              title: {                  text: 'JUMLAH PENGHARGAAN TERCATAT'              }          },          credits: {              enabled: false,          },          plotOptions: {              bar: {                  dataLabels: {                      enabled: true                  },                  enableMouseTracking: true,                  pointPadding: 0.08                }          },          series: dataSeries      }); |

**5.3.12 Persentase Penghargaan**

Persentase penghargaan adalah tabel yang muncul ketika *user* menekan tombol persentase penghargaan. Data yang akan ditampilkann adalah kategori, jumlah tercatat dan nilai persentase setiap catatan penghargaan per tingkatnya. Data-data yang ditampilkan terproses pada fungsi “showAchievement” seperti pada listing 5.25 diatas. Namun untuk menampilkan tabel ini menggunakan fungsi *jQuery* seperti pada listing 5.27 dibawah ini.

**Listing 5.27 Baris program menampilkan tabel persentase penghargaan**

|  |
| --- |
| $('#button-persen-penghargaan').on('click', (function(){          $('#persentase-penghargaan').show();  })) |

**5.3.13 Detail Penghargaan Berdasarkan Tingkat**

Detail penghargaan berdasarkan tingkat menampilkan sebuah tabel ketika menekan tombol dengan icon mata pada tabel persentase penghargaan dimana fungsi tersebut menggunakan fungsi *ajax*. Fungsi *ajax* tersebut melakukan *request* get data dari fungsi “showAchievementItemByGrade” dari *controller.* Baris program fungsi “showAchievementItemByGrade” bisa dilihat pada listing 5.28. Baris program ajax catatan penghargaan berdasarkan tingkat dapat dilihat pada listing 5.29.

**Listing 5.28 Baris program fungsi showAchievementItemByGrade**

|  |
| --- |
| public function showAchievementItemByGrade(Request $request)  {     $selected\_student = GradeStudent::select(DB::raw('MAX  (ACADEMIC\_YEAR\_ID) AS id'))->limit(1)->first()->id;     if(Auth::guard('web')->user()->staff->ROLE == "TEACHER"){ |

**Listing 5.28 Baris program fungsi showAchievementItemByGrade (lanjutan)**

|  |
| --- |
| $kelas\_guru = Grade::where('STAFFS\_ID', $request  ->session()->get('session\_user\_id'))->first()->id;         $penghargaan = Achievement::join('achievement\_records',   'achievements.id', 'achievement\_records.ACHIEVEMENTS\_ID')         ->join('students', 'achievement\_records.STUDENTS\_ID',  'students.id')        ->join('grades\_students', 'students.id',  'grades\_students.STUDENTS\_ID')         ->select('achievements.\*',  DB::raw('COUNT(\*) AS JUMLAHPERPENGHARGAAN'))         ->where('GRADE', $request->achievementGrade)         ->where('grades\_students.GRADES\_ID', $kelas\_guru)         ->where('achievement\_records.ACADEMIC\_YEAR\_ID', $request  ->academicYearId)         ->where('grades\_students.ACADEMIC\_YEAR\_ID',  $selected\_student)        ->groupBy('achievements.id')         ->get();     }     elseif(Auth::guard('web')->user()->staff  ->ROLE == "HEADMASTER")  {     $penghargaan = Achievement::join('achievement\_records',  'achievements.id', 'achievement\_records.ACHIEVEMENTS\_ID')         ->join('students', 'achievement\_records.STUDENTS\_ID',  'students.id')  ->join('grades\_students', 'students.id',  'grades\_students.STUDENTS\_ID')         ->select('achievements.\*',  DB::raw('COUNT(\*) AS JUMLAHPERPENGHARGAAN'))         ->where('GRADE', $request->achievementGrade)         ->where('achievement\_records.ACADEMIC\_YEAR\_ID', $request  ->academicYearId)         ->where('grades\_students.ACADEMIC\_YEAR\_ID',  $selected\_student)         ->groupBy('achievements.id')         ->get();     }     return $penghargaan;  } |

**Listing 5.29 Baris program fungsi ajax detail penghargaan berdasar tingkat**

|  |
| --- |
| $('#tbody-persentase-penghargaan').on('click', '.button-detail  -persentase', (function(){    var achievementGrade = $(this).attr('achievement-grade')    var academicYearId = $('#selector-dropdown  -achievementrecord-year').val();   $('#detail-penghargaan-berdasarkan-tingkat').show(); |

**Listing 5.29 Baris program fungsi ajax detail penghargaan berdasar tingkat (lanjutan)**

|  |
| --- |
| $.ajax({       url: '{{route("achievementrecord.achievementItem")}}',       type: 'get',       data: {achievementGrade: achievementGrade,  academicYearId: academicYearId},  success: function(result){          $('#tbody-detail-penghargaan-berdasarkan  -tingkat').empty()          result.forEach(function(obj){             $('#tbody-detail-penghargaan-berdasarkan  -tingkat').append(                          `                          <tr>                              <td>${obj.GRADE}</td>                              <td>${obj.DESCRIPTION}</td>                              <td>${obj.JUMLAHPERPENGHARGAAN}</td>                              <td>                   <button type="button" class="btn btn- primary btn-sm button-detail" achievement-id=${obj.id}><i class="fa fa-eye"></i></button>                              </td>                          </tr>  )           })          },          error: function(err){                console.log(err)          }       })    })) |

**5.3.14 Detail Penghargaan Setiap Siswa**

Detail pengharggan setiap siswa akan menampilkan data penghargaan setiap id siswa. Jika pada halaman daftar catatan penghargaan terdapat nama dan jumlah penghargaan, maka detail penghargaan ini menampilkan data-datanya. Tampilan detail akan berupa modal form. Untuk menampilkan modal tersebut maka perlu menekan tombol yang menjalan fungsi *ajax* yang melakukan *request*  data dari method “studentDetailAchievement” dari *ajax controller.* Baris program fungsi “studentDetailAchievement” bisa dilihat pada listing 5.30. Baris program ajax catatan penghargaan berdasarkan tingkat dapat dilihat pada listing 5.31.

**Listing 5.30 Baris program fungsi studentDetailAchievement**

|  |
| --- |
| public function studentDetailAchievement(Request $request)  {  $penghargaan\_tiap\_siswa = AchievementRecord::join  ('achievements', 'achievement\_records.ACHIEVEMENTS\_ID',  'achievements.id')     ->join('academic\_years',  'achievement\_records.ACADEMIC\_YEAR\_ID', 'academic\_years.id')      ->join('students', 'achievement\_records.STUDENTS\_ID',  'students.id')      ->select('achievement\_records.id',  'achievement\_records.DATE',  'achievement\_records.DESCRIPTION AS DESKRIPSI1',  'achievements.DESCRIPTION AS DESKRIPSI2' ,  'achievements.GRADE',               'achievements.POINT', 'students.FNAME',  'students.LNAME', 'academic\_years.TYPE',  'academic\_years.NAME')      ->where('STUDENTS\_ID', $request->studentId)      ->where('achievement\_records.ACADEMIC\_YEAR\_ID',  $request->academicYearId)      ->orderBy('achievement\_records.id', 'DESC')      ->get();        return $penghargaan\_tiap\_siswa;  } |

**Listing 5.31 Baris program fungsi ajax detail penghargaan setiap siswa**

|  |
| --- |
| $('#tbody-daftar-siswa-yang-ada-penghargaan')  .on('click', '.button-detail-penghargaan', (function(){      var studentId = $(this).attr('student-id');      var academicYearId = $('#selector-dropdown  -achievementrecord-year').val();        $.ajax({         url: '{{ route("achievementrecord  .studentDetailAchieve ment") }}',         type: 'get',         data: {studentId: studentId,  academicYearId: academicYearId},        success: function(result){ |

**Listing 5.31 Baris program fungsi ajax detail penghargaan setiap siswa (lanjutan)**

|  |
| --- |
| $('#tbody-student-achievement-list').empty()             result.forEach(function(obj){           var route =  "{{ route('achievementrecord.edit', ':id')  }}"  route = route.replace(':id', `${obj.id}`)           var route\_del = "{{ route ('achievementrecord.destroy',   ':id' )}}"           route\_del = route\_del.replace(':id', `${obj.id}`)           var $role\_teacher = "{{ Auth::guard('web')->user()  ->staff->ROLE === "TEACHER" }}"           var $role\_headmaster = "{{ Auth::guard('web')->user()  ->staff->ROLE === "HEADMASTER" }}"  if($role\_teacher){  $('#tbody-student-achievement  -list').append(                              `                              <tr>                                  <td style="width: 20px">${obj.id}</td>                                  <td style="width: 90px">${obj.FNAME} ${obj.LNAME}</td>                                  <td style="width: 20px">${obj.DATE}</td>                                  <td style="width: 60px">${obj.TYPE} - ${obj.NAME}</td>                                  <td style="width: 40px">${obj.GRADE}</td>                                  <td style="width: 200px">${obj.DESKRIPSI2}</td>                                  <td style="width: 200px">${obj.DESKRIPSI1}</td>                                  <td style="width: 40px">${obj.POINT}</td>                                  <td style="width: 80px">                                  <a href=${route}  class="btn btn-warning btn-sm">                                          <i class="fa fa-pencil"></i>                                  </a>  <form action=${route\_del} method="POST" class="inline">                 @method('delete')                 @csrf                 <button class="btn btn-danger btn  -sm" onclick="return confirm('Are you sure?')"  value="DELETE">                       <i class="fa fa-trash"></i>                 </button>                 </form>                 </td>            </tr>            `            )  } |

**Listing 5.31 Baris program fungsi ajax detail penghargaan setiap siswa (lanjutan)**

|  |
| --- |
| else if ($role\_headmaster) {            $('#tbody-student-achievement  -list').append(                              `                              <tr>                                  <td style="width: 20px">${obj.id}</td>                                  <td style="width: 90px">${obj.FNAME} ${obj.LNAME}</td>                                  <td style="width: 20px">${obj.DATE}</td>                                  <td style="width: 60px">${obj.TYPE} - ${obj.NAME}</td>                                  <td style="width: 40px">${obj.GRADE}</td>                                  <td style="width: 200px">${obj.DESKRIPSI2}</td>                                  <td style="width: 200px">${obj.DESKRIPSI1}</td>                                  <td style="width: 40px">${obj.POINT}</td>                              </tr>                     `          }        })        $('#modalDetailAchievementEachStudent').modal('show')      },      error: function(err){          console.log(err)      }    })  })); |

**5.3.15 Daftar Catatan Pelanggaran**

Daftar catatan pelanggaran akan menampilkan daftar catatannya seperti nama siswa, kelas, jumlah pelanggaran dan total poin pelanggaran dengan menggunakan sebuah fungsi pada *controller* pelanggaran yaitu “showViolation” seperti pada listing 5.32. Daftar catatan pelanggaran juga menampilkan grafik jumlah catatan setiap tahun ajaran dengan menggunakan fungsi *javascript* yang mendapat data dari *contoller* untuk data grafik seperti pada listing 5.33.

**Listing 5.32 Baris program fungsi showViolation**

|  |
| --- |
| public function showViolation($user\_id, $ay)  {    $selected\_student = GradeStudent::select(DB::raw('MAX  (ACADEMIC\_YEAR\_ID) AS id'))->limit(1)->first()->id;       $r = Violation::select('id')->where('NAME', 'LIKE', 'R%')  ->get();     $b = Violation::select('id')->where('NAME', 'LIKE', 'B%')  ->get();     $sb = Violation::select('id')->where('NAME', 'LIKE', 'SB%')  ->get();       if(Auth::guard('web')->user()->staff->ROLE == "TEACHER"){        $kelas\_guru = Grade::where('STAFFS\_ID', $user\_id)->first()  ->id;          $pelanggaran = ViolationRecord::join('violations',  'violation\_records.VIOLATIONS\_ID', 'violations.id')        ->join('students', 'violation\_records.STUDENTS\_ID',  'students.id')        ->join('grades\_students', 'students.id',  'grades\_students.STUDENTS\_ID')        ->select('violation\_records.\*',         DB::raw('COUNT(\*) AS JUMLAHPELANGGARAN'),         DB::raw('SUM(violation\_records.TOTAL) AS POINPELANGGARAN'))        ->where('violations.NAME', 'NOT LIKE', 'TTS%')        ->where('grades\_students.GRADES\_ID', $kelas\_guru)        ->where('violation\_records.ACADEMIC\_YEAR\_ID', $ay)        ->where('grades\_students.ACADEMIC\_YEAR\_ID',  $selected\_student)        ->groupBy('violation\_records.STUDENTS\_ID')        ->orderBy('JUMLAHPELANGGARAN', 'DESC')        ->get();        $count\_r = $this->countViolationPerCategory  ($r, $ay, $kelas\_guru, $selected\_student)->first()->JMLH;        $count\_b = $this->countViolationPerCategory  ($b, $ay, $kelas\_guru, $selected\_student)->first()->JMLH;        $count\_sb = $this->countViolationPerCategory  ($sb, $ay, $kelas\_guru, $selected\_student)->first()->JMLH;          $count\_violation\_list = Violation::select(DB::raw  ('COUNT(\*)AS JUMLAH'))->where('NAME', 'NOT LIKE', 'TTS%')  ->first()->JUMLAH;          $data = DB::select("SELECT (CASE WHEN v.NAME LIKE 'R%'  THEN 'RINGAN'                WHEN v.NAME LIKE 'B%' THEN 'BERAT'                     WHEN v.NAME LIKE 'SB%' THEN 'SANGATBERAT'                END) AS KATEGORI,                COUNT(\*) AS JUMLAH, |

**Listing 5.32 Baris program fungsi showViolation (lanjutan)**

|  |
| --- |
| (SELECT (CASE WHEN KATEGORI = 'RINGAN' THEN ROUND  ($count\_r / $count\_violation\_list \* 100)        WHEN KATEGORI = 'BERAT' THEN ROUND  ( $count\_b / $count\_violation\_list \* 100)        WHEN KATEGORI = 'SANGATBERAT' THEN ROUND  ( $count\_sb / $count\_violation\_list \* 100)        END))AS PERSENTASE        FROM violation\_records vr INNER JOIN violations v ON  vr.VIOLATIONS\_ID = v.id  INNER JOIN students s ON vr.STUDENTS\_ID = s.id        INNER JOIN grades\_students gs ON gs.STUDENTS\_ID = s.id          WHERE vr.ACADEMIC\_YEAR\_ID = " . $ay . "  AND  v.NAME NOT LIKE 'TTS%'  AND gs.GRADES\_ID = " . $kelas\_guru . "  AND gs.ACADEMIC\_YEAR\_ID = ". $selected\_student ."        GROUP BY KATEGORI        ORDER BY KATEGORI DESC");    }    elseif(Auth::guard('web')->user()->staff  ->ROLE == "HEADMASTER"){     $pelanggaran = ViolationRecord::join('violations',  'violation\_records.VIOLATIONS\_ID', 'violations.id')       ->join('students', 'violation\_records.STUDENTS\_ID',  'students.id')       ->join('grades\_students', 'students.id',  'grades\_students.STUDENTS\_ID')       ->select('violation\_records.\*',       DB::raw('COUNT(\*) AS JUMLAHPELANGGARAN'),       DB::raw('SUM(violation\_records.TOTAL) AS POINPELANGGARAN'))      ->where('violations.NAME', 'NOT LIKE', 'TTS%')      ->where('violation\_records.ACADEMIC\_YEAR\_ID', $ay)      ->where('grades\_students.ACADEMIC\_YEAR\_ID',  $selected\_student)     ->groupBy('violation\_records.STUDENTS\_ID')      ->orderBy('JUMLAHPELANGGARAN', 'DESC')->get();        $count\_r = $this->countViolationPerCategoryKepsek  ($r, $ay, $selected\_student)->first()->JMLH;  $count\_b = $this->countViolationPerCategoryKepsek  ($b, $ay, $selected\_student)->first()->JMLH;      $count\_sb = $this->countViolationPerCategoryKepsek  ($sb, $ay, $selected\_student)->first()->JMLH;     $data = DB::select("SELECT (CASE WHEN v.NAME LIKE 'R%'  THEN 'RINGAN'     WHEN v.NAME LIKE 'B%' THEN 'BERAT'      WHEN v.NAME LIKE 'SB%' THEN 'SANGATBERAT'      END) AS KATEGORI, COUNT(\*) AS JUMLAH, |

**Listing 5.32 Baris program fungsi showViolation (lanjutan)**

|  |
| --- |
| (SELECT (CASE WHEN KATEGORI = 'RINGAN' THEN ROUND  ( $count\_r / (SELECT COUNT(\*) FROM violations  WHERE violations.NAME LIKE 'R%') \* 100)      WHEN KATEGORI = 'BERAT' THEN ROUND( $count\_b /  (SELECT COUNT(\*) FROM violations  WHERE violations.NAME LIKE 'B%') \* 100)      WHEN KATEGORI = 'SANGATBERAT' THEN ROUND( $count\_sb /  (SELECT COUNT(\*) FROM violations WHERE  violations.NAME LIKE 'S%') \* 100)      END)) AS PERSENTASE      FROM violation\_records vr INNER JOIN violations v ON  vr.VIOLATIONS\_ID = v.id      INNER JOIN students s ON vr.STUDENTS\_ID = s.id      INNER JOIN grades\_students gs ON gs.STUDENTS\_ID = s.id          WHERE vr.ACADEMIC\_YEAR\_ID = " . $ay . "  AND  v.NAME NOT LIKE 'TTS%' AND  gs.ACADEMIC\_YEAR\_ID = ". $selected\_student ."      GROUP BY KATEGORI      ORDER BY KATEGORI DESC");    }    else{      $pelanggaran = ViolationRecord::where('violation\_records  .ACADEMIC\_YEAR\_ID', $ay)  ->orderBy('violation\_records.id', 'DESC')->get();    }    $value["pelanggaran"] = $pelanggaran;    $value["data"] = $data;    return $value;  } |

**Listing 5.33 Baris program informasi pelanggaran dengan grafik**

|  |
| --- |
| var categories = {!!json\_encode($kategori)!!}      var dataGraph = {!!json\_encode($data)!!}      var selectedTahunAjaran = {!!json\_encode  ($selected\_tahun\_ajaran)!!}      var startMonth = selectedTahunAjaran.STARTMONTH;      var endMonth = selectedTahunAjaran.ENDMONTH;      var dataSeries = [];      var dataCategory = []      categories.forEach(function(item){          var values = []          for(var i=startMonth; i <= endMonth; i++){              values[i-startMonth] = 0              dataCategory.push('Bulan '+ i)          }          dataGraph.forEach(function(obj){              if(obj.KATEGORI == item.KATEGORI){                  values[obj.BULAN - startMonth] = obj.JUMLAH |

**Listing 5.33 Baris program informasi pelanggaran dengan grafik (lanjutan)**

|  |
| --- |
| })          dataSeries.push({              name: item.KATEGORI,              data: values          })      })      Highcharts.setOptions({          colors: ['#F21402', '#028CF2 ', '#E4F202 ', '028CF2']      });      Highcharts.chart('violationChartStatistic', {          chart: {              type: 'column'          },          title: {              text: 'STATISTIK PELANGGARAN YANG TERCATAT  BERDASARKAN JENIS PELANGGARAN DAN PERIODE TAHUN AJARAN'          },          xAxis: {              categories: dataCategory,              crosshair: true          },          yAxis: {              min: 0,              allowDecimals: false,              title: {                  text: 'JUMLAH PELANGGARAN TERCATAT'              }          },          credits: {              enabled: false,          },          plotOptions: {              column: {                  pointPadding: 0.08,                  borderWidth: 0              }          },          series: dataSeries      }); |

**5.3.16 Persentase Pelanggaran**

Persentase pelanggaran adalah tabel yang muncul ketika *user* menekan tombol persentase pelanggaran. Data yang akan ditampilkann adalah kategori, jumlah tercatat dan nilai persentase setiap catatan pelanggaran per kategorinya. Data-data yang ditampilkan terproses pada fungsi “showViolation” seperti pada listing 5.32 diatas. Namun untuk menampilkan tabel ini menggunakan fungsi *jQuery* seperti pada listing 5.34 dibawah ini.

**Listing 5.34 Baris program menampilkan tabel persentase pelanggaran**

|  |
| --- |
| $('#button-persen-pelanggaran').on('click', (function(){     $('#persentase-pelanggaran').show();  })) |

**5.3.17 Detail Pelanggaran Berdasarkan Kategori**

Detail pelanggaran berdasarkan tingkat menampilkan sebuah tabel ketika menekan tombol dengan icon mata pada tabel persentase pelanggaran dimana fungsi tersebut menggunakan fungsi *ajax*. Fungsi *ajax* tersebut melakukan *request* get data dari fungsi “showViolationItemByCategory” dari *controller.* Baris program fungsi “showViolationItemByCategory” bisa dilihat pada listing 5.35. Baris program ajax catatan penghargaan berdasarkan tingkat dapat dilihat pada listing 5.36.

**Listing 5.35 Baris program fungsi showViolationItemByCategory**

|  |
| --- |
| public function showViolationItemByCategory(Request $request)  {     $selected\_student = GradeStudent::select(DB::raw('MAX  (ACADEMIC\_YEAR\_ID) AS id'))->limit(1)->first()->id;     if(Auth::guard('web')->user()->staff->ROLE == "TEACHER"){        $kelas\_guru = Grade::where('STAFFS\_ID',  $request->session()  ->get('session\_user\_id'))->first()->id;          $pelanggaran = Violation::join('violation\_records',  'violations.id', 'violation\_records.VIOLATIONS\_ID')        ->join('students', 'violation\_records.STUDENTS\_ID',  'students.id')        ->join('grades\_students', 'students.id',  'grades\_students.STUDENTS\_ID') |

**Listing 5.35 Baris program fungsi showViolationItemByCategory (lanjutan)**

|  |
| --- |
| ->select('violations.\*',  DB::raw('COUNT(\*) AS JUMLAHPERPELANGGARAN'))        ->where('NAME', 'LIKE', $request->violationName . '%' )        ->where('grades\_students.GRADES\_ID', $kelas\_guru)        ->where('violation\_records.ACADEMIC\_YEAR\_ID',  $request->academicYearId)        ->where('grades\_students.ACADEMIC\_YEAR\_ID',  $selected\_student)        ->groupBy('violations.id')        ->get();     }     elseif(Auth::guard('web')->user()->staff  ->ROLE == "HEADMASTER")  {        $pelanggaran = Violation::join('violation\_records',  'violations.id', 'violation\_records.VIOLATIONS\_ID')        ->join('students', 'violation\_records.STUDENTS\_ID',  'students.id')        ->join('grades\_students', 'students.id',  'grades\_students.STUDENTS\_ID')       ->select('violations.\*',  DB::raw('COUNT(\*) AS JUMLAHPERPELANGGARAN'))        ->where('NAME', 'LIKE', $request->violationName . '%' )      ->where('violation\_records.ACADEMIC\_YEAR\_ID',  $request->academicYearId)        ->where('grades\_students.ACADEMIC\_YEAR\_ID',  $selected\_student)        ->groupBy('violations.id')        ->get();     }     return $pelanggaran;  } |

**Listing 5.36 Baris program fungsi ajax detail pelanggaran berdasar kategori**

|  |
| --- |
| $('#tbody-persentase-pelanggaran').on('click', '.button-detail-persentase', (function(){          var x = $(this).attr('violation-category')          var res = x.split("")          var violationName = res[0]          var academicYearId = $('#selector-dropdown  -violationrecord-year').val()            $('#detail-pelanggaran-berdasarkan  -kategori').show();            $.ajax({              url: '{{route("violationrecord.violationItem")}}',              type: 'get', |

**Listing 5.36 Baris program fungsi ajax detail pelanggaran berdasar kategori (lanjutan)**

|  |
| --- |
| data: {violationName: violationName,  academicYearId: academicYearId},            success: function(result){                  $('#tbody-detail-pelanggaran-berdasarkan  -kategori').empty()                    result.forEach(function(obj){                      $('#tbody-detail-pelanggaran-berdasarkan  -kategori').append(                          `                          <tr>                              <td>${obj.NAME}</td>                              <td>${obj.DESCRIPTION}</td>                              <td>${obj.JUMLAHPERPELANGGARAN}</td>                              <td>                                  <button type="button" class="btn  btn-primary btn-sm button  -detail"  violation-id=${obj.id}>  <i class="fa fa-eye"></i>  </button>                              </td>                          </tr>                          `                      )                  })              },              error: function(err){                  console.log(err)              }          })      })) |

**5.3.18 Detail Pelanggaran Setiap Siswa**

Detail pelanggaran setiap siswa akan menampilkan data pelanggaran setiap id siswa. Jika pada halaman daftar catatan pelanggaran terdapat nama dan jumlah pelanggaran, maka detail pelanggaran ini menampilkan data-datanya. Tampilan detail akan berupa modal form. Untuk menampilkan modal tersebut maka perlu menekan tombol yang menjalan fungsi *ajax* yang melakukan *request* data dari method “studentDetailViolation” dari *ajax controller.* Baris program fungsi “studentDetailViolation” bisa dilihat pada listing 5.37. Baris program ajax catatan pelanggaran berdasarkan tingkat dapat dilihat pada listing 5.38.

**Listing 5.37 Baris program fungsi studentDetailViolation**

|  |
| --- |
| public function studentDetailViolation(Request $request)  {     $pelanggaran\_tiap\_siswa = ViolationRecord::join  ('violations', 'violation\_records.VIOLATIONS\_ID',  'violations.id')     ->join('academic\_years',  'violation\_records.ACADEMIC\_YEAR\_ID',  'academic\_years.id')     ->join('students', 'violation\_records.STUDENTS\_ID',  'students.id')     ->select('violation\_records.id', 'violations.POINT' ,  'violation\_records.DATE' ,  'violation\_records.PUNISHMENT',  'violations.DESCRIPTION',              'students.FNAME', 'students.LNAME',              'academic\_years.TYPE', 'academic\_years.NAME')  ->where('STUDENTS\_ID', $request->studentId)     ->where('violation\_records.ACADEMIC\_YEAR\_ID',  $request->academicYearId)     ->orderBy('violation\_records.id', 'DESC')    ->get();     return $pelanggaran\_tiap\_siswa;  } |

**Listing 5.38 Baris program fungsi ajax detail pelanggaran setiap siswa**

|  |
| --- |
| $('#tbody-daftar-siswa-yang-ada-pelanggaran')  .on('click', '.button-detail-pelanggaran', (function(){       var studentId = $(this).attr('student-id');       var academicYearId = $('#selector-dropdown  -violationrecord-year').val()         $.ajax({       url: '{{ route("violationrecord  .studentDetailViolation ") }}',         type: 'get',         data: {studentId: studentId,  academicYearId: academicYearId}, |

**Listing 5.38 Baris program fungsi ajax detail pelanggaran setiap siswa (lanjutan)**

|  |
| --- |
| success: function(result){        $('#tbody-student-violation-list').empty()         result.forEach(function(obj){          var route =  "{{ route('violationrecord.edit', ':id')  }}"           route = route.replace(':id', `${obj.id}`)          var route\_del = "{{ route ('violationrecord.destroy',  ':id' )}}"           route\_del = route\_del.replace(':id', `${obj.id}`)           var $role\_teacher = "{{ Auth::guard('web')->user()  ->staff->ROLE === "TEACHER" }}"           var $role\_headmaster = "{{ Auth::guard('web')->user()  ->staff->ROLE === "HEADMASTER" }}"           if($role\_teacher)  {              $('#tbody-student-violation  -list').append(                              `                              <tr>                                  <td style="width: 20px">${obj.id}</td>                                  <td style="width: 90px">${obj.FNAME}  ${obj.LNAME}</td>                                  <td style="width: 20px">${obj.DATE}</td>                                  <td style="width: 60px">${obj.TYPE} –  ${obj.NAME}</td>                                  <td style="width: 110px">${obj.DESCRIPTION}  </td>                                  <td style="width: 110px">${obj.PUNISHMENT}  </td>                                  <td style="width: 40px">${obj.POINT}</td>                                  <td style="width: 80px">                                             <a href=${route}  class="btn btn-warning  btn-sm">                                          <i class="fa fa-pencil"></i>                                      </a>                                      <form action=${route\_del} method="POST"  class="inline">                                          @method('delete')                                          @csrf                                          <button class="btn btn-danger  btn-sm" onclick=  "return confirm('Are you sure?')"  value="DELETE">                                              <i class="fa fa-trash"></i>                                          </button>                                      </form>                                  </td>                              </tr> |

**Listing 5.38 Baris program fungsi ajax detail pelanggaran setiap siswa (lanjutan)**

|  |
| --- |
| `             )           }  else if ($role\_headmaster)  {              $('#tbody-student-violation  -list').append(                              `                              <tr>                                  <td style="width: 20px">${obj.id}</td>                                  <td style="width: 90px">${obj.FNAME}  ${obj.LNAME}</td>                                  <td style="width: 20px">${obj.DATE}</td>                                  <td style="width: 60px">${obj.TYPE} –  ${obj.NAME}</td>                                  <td style="width: 110px">${obj.DESCRIPTION}  </td>                                  <td style="width: 110px">${obj.PUNISHMENT}  </td>                                  <td style="width: 40px">${obj.POINT}</td>                              </tr>                              `               )            }           })         $('#modalDetailViolationEachStudent').modal('show')       },       error: function(err){          console.log(err)       }     })  })); |

**5.3.19 Daftar Catatan Absen**

Halaman daftar catatan absen adalah halaman yang meproses catatan-catatan absen. Halaman ini hanya bisa diakses oleh *user* sebagai kepala sekolah dan guru/wali kelas.

**1. Daftar Catatan Absen Kepala Sekolah**

Data yang akan diberikan kepada sekolah adalah informasi grafik yaitu persentase setiap absensi dan daftar tabel. Daftar tabel untuk kepala sekolah akan memberikan informasi nama kelas, dan total kehadiran setiap kelas per tahun ajarannya ada berapa. Kepala sekolah bisa melihatdetail absen dengan menakan tombol *icon* mata yang akan menampilkan modal daftar absen per kelas. Modal daftar absen per kelas ini menggunakan satu fungsi *request* data menggunakan *ajax.* Fungsi ajax akan meminta data dari *controller* pada method “absentEachGrade”.

Kepala sekolah juga bisa melihat detail lebih lagi daftar absen per tipe absennya. Untuk menampilkan daftar absen per tipe absen digunakan suatu fungsi *ajax* untuk lakukan *request* data kepada fungsi “detailAbsentEachType”. Baris program fungsi “absentEachGrade” dapat dilihat pada listing 5.39 dan untuk fungsi *ajax* sendiri meminta data daftar absen per kelas dapat dilihat pada listing 5.40. Sedangkan baris program fungsi “detailAbsentEachType” dapat dilihat pada listing 5.41 dan untuk fungsi *ajax* meminta data detail absen per tipe dapat dilihat pada listing 5.42.

**Listing 5.39 Baris program fungsi “absentEachGrade”**

|  |
| --- |
| public function absentEachGrade(Request $request)  {     $selected\_student = GradeStudent::select(DB::raw('MAX  (ACADEMIC\_YEAR\_ID) AS id'))->limit(1)->first()->id;       $absen\_per\_kelas = Absent::join('students',  'absents.STUDENTS\_ID', 'students.id')  ->join('grades\_students', 'grades\_students.STUDENTS\_ID',  'students.id')     ->join('grades', 'grades\_students.GRADES\_ID', 'grades.id') |

**Listing 5.39 Baris program fungsi “absentEachGrade” (lanjutan)**

|  |
| --- |
| ->select('absents.TYPE', 'grades.NAME',  DB::raw('COUNT(\*) ABSENPERTIPE'),  'grades\_students.GRADES\_ID')     ->where('grades\_students.GRADES\_ID', $request->gradeId)     ->where('grades\_students.ACADEMIC\_YEAR\_ID',  $selected\_student)     ->where('absents.ACADEMIC\_YEAR\_ID', $request->academicYearId)     ->groupBy('absents.TYPE')     ->get();     return $absen\_per\_kelas;  } |

**Listing 5.40 Baris program ajax request daftar absen per kelas**

|  |
| --- |
| $('#tbody-daftar-absen').on('click', '.button-detail-absen-per-kelas', (function(){          var gradeId = $(this).attr('grade-id');          var academicYearId = $('#selector-dropdown-absentrecord  -year').val()          $.ajax({              url: '{{ route("absent.absentEachGrade") }}',              type: 'get',              data: {gradeId: gradeId,  academicYearId: academicYearId},              success: function(result){                  $('#tbody-absent-each-grade-list').empty()                  result.forEach(function(obj){                      $('#tbody-absent-each-grade-list').append(                          `                          <tr>  <td style="width: 50px">${obj.TYPE} </td>  <td style="width: 50px">  ${obj.ABSENPERTIPE}</td>                              <td style="width: 50px">                                  <button type="button"                                          class="btn  btn-primary btn-sm  button-detail"                                          absent-type=${obj.TYPE}                                  grade-id=${obj.GRADES\_ID}>                                  <i class="fa fa-eye"></i>                                  </button>                              </td>                          </tr>                          `                      ) |

**Listing 5.40 Baris program ajax request daftar absen per kelas (lanutan)**

|  |
| --- |
| })                  $('#modalAbsentEachGrade').modal('show')              },              error: function(err){                  console.log(err)              }          })      })) |

**Listing 5.41 Baris program fungsi “detailAbsentEachType”**

|  |
| --- |
| public function detailAbsentEachType(Request $request)  {     $selected\_student = GradeStudent::select(DB::raw('MAX  (ACADEMIC\_YEAR\_ID) AS id'))->limit(1)->first()->id;      $detail\_absen\_per\_tipe = Absent::join('academic\_years',  'absents.ACADEMIC\_YEAR\_ID', 'academic\_years.id')      ->join('students', 'absents.STUDENTS\_ID', 'students.id')     ->join('grades\_students', 'grades\_students.STUDENTS\_ID',  'students.id')     ->select('absents.\*', 'students.FNAME',  'students.LNAME',  'academic\_years.TYPE AS TIPETHNAJARAN',  'academic\_years.NAME')  ->where('absents.TYPE', $request->absentType)      ->where('absents.ACADEMIC\_YEAR\_ID',  $request->academicYearId)      ->where('grades\_students.GRADES\_ID', $request->gradeId)      ->where('grades\_students.ACADEMIC\_YEAR\_ID',  $selected\_student)      ->get();      return $detail\_absen\_per\_tipe;  } |

**Listing 5.42 Baris program ajax request detail absen per tipe absen**

|  |
| --- |
| $('#tbody-absent-each-grade-list').on('click', '.button-detail', (function(){          var absentType = $(this).attr('absent-type')          var gradeId = $(this).attr('grade-id')          var academicYearId = $('#selector-dropdown-absentrecord  -year').val() |

**Listing 5.42 Baris program ajax request detail absen per tipe absen (lanjutan)**

|  |
| --- |
| $.ajax({              url: '{{ route("absent.detailAbsentEachType") }}',              type: 'get',              data: {absentType: absentType, gradeId: gradeId,  academicYearId: academicYearId},              success: function(result){                  $('#tbody-detail-absent-each-type  -list').empty()                    result.forEach(function(obj){                      $('#tbody-detail-absent-each-type  -list').append(                          `                          <tr>                              <td style="width: 30px">${obj.id}  </td>  <td style="width: 110px">  ${obj.FNAME}  ${obj.LNAME}</td>  <td style="width: 80px">${obj.TYPE}</td>  <td style="width: 100px">  ${obj.TIPETHNAJARAN}-${obj.NAME}</td>  <td style="width: 75px">  ${obj.START\_DATE}</td>  <td style="width: 75px">  ${obj.END\_DATE}</td>  <td style="width: 150px">  ${obj.DESCRIPTION}</td>                          </tr>                          `                      )                  })                  $('#modalDetailAbsentEachType').modal('show')              },              error: function(err){                  console.log(err)              }          })      })) |

**1. Daftar Catatan Absen Guru/Wali Kelas**

Data yang akan diberikan wali kelas adalah informasi grafik yaitu persentase setiap absensi yang mendapat data dari *controller* dan daftar tabel. Daftar absen akan menampilkan catatan ketidakhadiran miliki siswa kelasnya sendiri.

Ada proses yang digunakan untuk melihat detail daftar absen per tipe yang dimiliki wali kelas. Detail daftar absen per tipe akan ditampilkan dalam sebuah modal dengan menggunakan fungsi *ajax* yang melakukanrequest detail absen per tipe absen kepada fungsi “detailAbsentEachType” pada absen *controller.* Subbab ini akan memberikan tiga potongan baris program yaitu, listing 5.43 merupakan baris program data pada grafik, listing 5.44 menjelaskan baris program fungsi “detailAbsentEachType” dan listing 5.45 adalah baris program ajax request detail absen per tipe absen.

**Listing 5.43 Baris program informasi catatan absen dengan grafik**

|  |
| --- |
| var types = {!! json\_encode($type) !!}      var dataGraph = {!! json\_encode($data) !!}      var totalDayEachAcademicYear =  {!! json\_encode($count\_total\_day\_each\_ay) !!}      var totalAmountExceptPresent = 0      var present\_percentage = 0      var dataSeries = []      var obj\_present = {}      types.forEach(function(item){          dataGraph.forEach(function(obj){              if(obj.TIPE == item.TIPE){                  dataSeries.push({                      name: item.TIPE,                      y: obj.JUMLAH                  })                  totalAmountExceptPresent =  totalAmountExceptPresent + obj.JUMLAH                  present\_percentage = totalDayEachAcademicYear –  totalAmountExceptPresent |

**Listing 5.43 Baris program informasi catatan absen dengan grafik (lanjutan)**

|  |
| --- |
| }          })      })      if(Array.isArray(dataGraph) == dataGraph.length){          obj\_present = {name: 'PRESENT', y: present\_percentage,  sliced: true, selected: true}          dataSeries.push(obj\_present)      }      else {          obj\_present = {name: 'PRESENT', y:  totalDayEachAcademicYear, sliced: true,  selected: true}          dataSeries.push(obj\_present)      }      Highcharts.setOptions({          colors: ['#F21402', '#2ECC71 ', '#E4F202 ', '#2874A6']      });      Highcharts.chart('absentChartStatistic', {          chart: {              plotBackgroundColor: null,              plotBorderWidth: null,              plotShadow: false,              type: 'pie'          },          credits: {              enabled: false          },          title: {              text: 'PERSENTASE ABSENSI DALAM SATU PERIODE  BERDASARKAN JENIS ABSENSI'          },          tooltip: {              pointFormat: '{series.name}: <b>  {point.percentage:.1f}%</b>'          },          accessibility: {              point: {valueSuffix: '%'}          },          plotOptions: {              pie: {                  allowPointSelect: true,                  cursor: 'pointer',                  dataLabels: {                      enabled: false },                  showInLegend: true}          },          series: [{              name: 'Persentase',              colorByPoint: true,              data: dataSeries          }]      }); |

**Listing 5.44 Baris program fungsi “detailAbsentEachType”**

|  |
| --- |
| public function detailAbsentEachType(Request $request)  {     $selected\_student = GradeStudent::select(DB::raw('MAX  (ACADEMIC\_YEAR\_ID) AS id'))->limit(1)->first()->id;     $detail\_absen\_per\_tipe = Absent::join('academic\_years',  'absents.ACADEMIC\_YEAR\_ID', 'academic\_years.id')     ->join('students', 'absents.STUDENTS\_ID', 'students.id')     ->join('grades\_students', 'grades\_students.STUDENTS\_ID',  'students.id')     ->select('absents.\*', 'students.FNAME',  'students.LNAME',  'academic\_years.TYPE AS TIPETHNAJARAN',  'academic\_years.NAME')  ->where('absents.TYPE', $request->absentType)  ->where('absents.ACADEMIC\_YEAR\_ID', $request->academicYearId)  ->where('grades\_students.GRADES\_ID', $request->gradeId)  ->where('grades\_students.ACADEMIC\_YEAR\_ID',  $selected\_student)  ->get();     return $detail\_absen\_per\_tipe;  } |

**Listing 5.45 Baris program ajax request detail absent per tipe absen**

|  |
| --- |
| $('#tbody-daftar-absen').on('click', '.button-detail-tipe-absen', (function(){          var absentType = $(this).attr('absent-type')          var gradeId = $(this).attr('grade-id')          var academicYearId = $('#selector-dropdown-absentrecord  -year').val()          $.ajax({              url: '{{ route("absent.detailAbsentEachType") }}',              type: 'get',              data: {absentType: absentType, gradeId: gradeId,  academicYearId: academicYearId},              success: function(result){                  $('#tbody-detail-absent-each-type  -list').empty()                  result.forEach(function(obj){                      var route =  "{{ route('absent.edit', ':id')  }}"                      route = route.replace(':id', `${obj.id}`)                      var route\_del = "{{ route ('absent.destroy',  ':id' )}}"                      route\_del = route\_del.replace(':id',  `${obj.id}`) |

**Listing 5.45 Baris program ajax request detail absent per tipe absen (lanjutan)**

|  |
| --- |
| $('#tbody-detail-absent-each-type  -list').append(                          `                          <tr>                              <td style="width: 30px">${obj.id}  </td>                              <td style="width: 110px">  ${obj.FNAME}    ${obj.LNAME}</td>                              <td style="width: 80px">${obj.TYPE}  </td>                              <td style="width: 100px">  ${obj.TIPETHNAJARAN}  - ${obj.NAME}</td>                              <td style="width: 75px">  ${obj.START\_DATE}</td>                              <td style="width: 75px">  ${obj.END\_DATE}</td>                              <td style="width: 150px">  ${obj.DESCRIPTION}</td>                              <td style="width: 70px">                                  <a href=${route}  class="btn  btn-warning btn-sm">                                      <i class="fa fa-pencil"></i>                                  </a>                                  <form action=${route\_del}  method="POST" class="inline">                                      @method('delete')                                      @csrf                                      <button class="btn  btn-danger btn-sm"  onclick="return  confirm('Are you sure?')"  value="DELETE">                                          <i class="fa  fa- trash"></i>                                      </button>                                  </form>                              </td>                          </tr>                          `                      )                  })                  $('#modalDetailAbsentEachType').modal('show')              },              error: function(err){                  console.log(err)              }          })      })) |