APLIKASI PENJADWALAN MATA PELAJARAN DI SMAN 31 MENGGUNAKAN ALGORITMA GENETIKA BERBASIS WEB

Ivan
Fakultas Teknologi dan Desain, Program Studi Teknik Informatika
Universitas Bunda Mulia
Email: leveldownn@gmail.com

Stephanus Raphael
Fakultas Teknologi dan Desain, Program Studi Teknik Informatika
Universitas Bunda Mulia
Email: raphael@stevinweb.com

Halim Agung
Fakultas Teknologi dan Desain, Program Studi Teknik Informatika
Universitas Bunda Mulia
Email: hagung@bundamulia.ac.id

ABSTRAK

Sekolah Menengah Atas Negeri 31 memiliki kelas sebanyak 29 kelas yang masing-masing memiliki 3 jurusan yaitu IPA, IPS dan Bahasa. Masing masing jurusan memiliki jumlah kelas yang cukup banyak. Sehingga pembuatan jadwal mata pelajaran oleh pihak sekolah memerlukan waktu yang cukup lama dan menggangu kelancaran proses belajar dan mengajar. aplikasi penjadwalan mata pelajaran menggunakan algoritma genetika merupakan solusi yang tepat untuk SMAN31. Algoritma Genetika adalah Proses seleksi alamiah yang melibatkan perubahan gen yang terjadi pada individu melalui proses perkembang-biakan yang menjadi proses dasar dan menjadi perhatian utama. Metode pengembangan sistem yang digunakan dalam penelitian ini adalah Metode *Waterfall*. Hasil dari penelitian ini adalah aplikasi penjadwalan SMAN 31 Jakarta yang dapat menyusun jadwal yang baik secara otomatis. Kesimpulan dari penelitian ini adalah algoritma genetika dapat diimplementasikan pada aplikasi penjadwalan dibuktikan dengan menguji 30 contoh jadwal yang tersusun oleh aplikasi tanpa ada yang bertabrakan satu sama lain.

Kata kunci: SMAN31, algoritma, penjadwalan, genetika, populasi.

ABSTRACT

Public High School 31 has 29 classes, each of which has three majors: Science, Sociology and Language. Each department has a sufficient number of classes. So that making the schedule of subjects the school takes a long time and disrupt the smooth process of learning and teaching. application of subject scheduling using genetic algorithm is the right solution for SMAN31. Genetic Algorithm is a natural selection process that involves the gene changes that occur in individuals through a process of proliferation that becomes a basic process and a major concern. System development method used in this research is Waterfall Method. The result of this research is SMAN 31 Jakarta scheduling application which can arrange a good schedule automatically. The conclusion of this study is that genetic algorithms can be implemented in scheduling applications evidenced by testing 30 sample schedules composed by applications without one colliding with each other.

Keywords: *SMAN31, algorithm, scheduling, genetics, population.*

1. PENDAHULUAN

Penjadwalan mata pelajaran di sekolah merupakan hal yang sangat penting dalam berlangsungnya kegiatan belajar mengajar di sekolah, jadwal ini bertujuan untuk mendukung, memperlancar, dan meningkatkan kualitas pendidikan. Secara umum jadwal mata pelajaran berfungsi untuk aktivitas akademik dalam meningkatkan kualitas mengajar dan kedisiplinan baik guru maupun siswa.

Sampai saat ini penjadwalan pelajaran di beberapa sekolah masih dilakukan secara manual oleh bagian kurikulum, dengan sebelumnya dilakukan rapat pembagian tugas bersama guru mata pelajaran. Dari

penentuan banyaknya kelas, banyaknya guru di sekolah, dan banyaknya jam mengajar untuk setiap guru masih dilakukan secara manual. Alokasi dan penentuan guru merupakan elemen yang penting dalam penyusunan jadwal mata pelajaran, namun juga menjadi permasalahan yang umum dalam proses penyusunan jadwal.

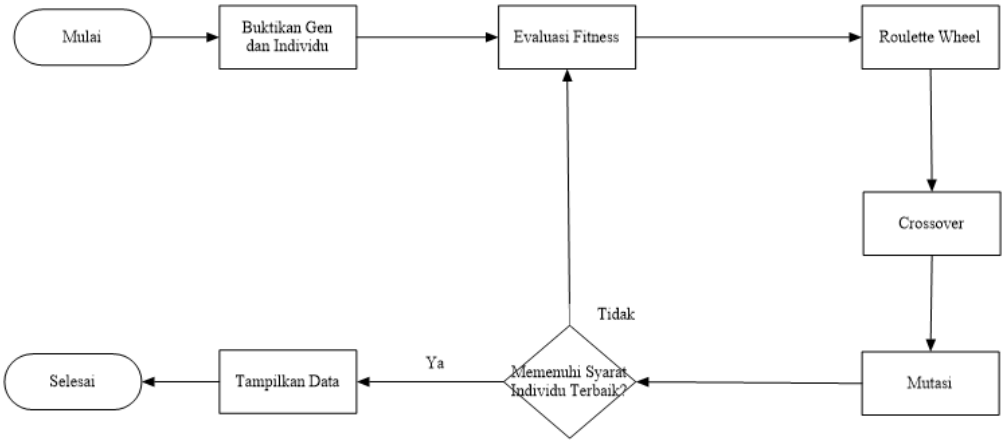
Penelitian ini menggunakan beberapa referensi penelitian yang dilakukan oleh peneliti sebelumnya. Penelitian terdahulu pertama yang menggunakan algoritma genetika [1] yang telah menghasilkan penjadwalan pesanan tanpa adanya pelanggaran pada penjadwalan. Penelitian terdahulu kedua [2] yang telah menghasilkan penjadwalan Mata kuliah tanpa adanya jadwal yang bertabrakan. Penelitian terdahulu ketiga [3] yang telah menghasilkan penjadwalan mata pelajaran tanpa adanya jadwal yang bertabrakan. Penelitian terdahulu keempat [4] yang dilakukan menghasilkan aplikasi yang dpat memprediksi waktu dan biaya pengerjaan proyek konstruksi yang dapat memberikan rekomendasi kepada kontraktor dalam pengerjaan proyek konstruksi. Penelitian terdahulu kelima membahas mengenai sistem rekomendasi wisata kuliner [5] dan disimpulkan bahwa dalam sistem rekomendasi wisata kuliner dengan menggunakan Algoritma Genetika, nilai *fitness* terbaik didapatkan dari metode *crossover* dengan satu titik potong dan mutasi dengan pergeseran gen. Kombinasi metode *crossover* dan mutasi ini menghasilkan nilai *fitness* rata-rata sebesar 924.

2. METODOLOGI PENELITIAN

2.1 Algoritma Genetika

Algoritma Genetika didasarkan pada proses genetik yang ada dalam makhluk hidup, yaitu perkembangan generasi dalam sebuah populasi yang alami, secara lambat laun mengikuti prinsip seleksi alam. Dengan meniru teori evolusi ini, Algoritma Genetika dapat digunakan untuk mencari solusi permasalahan-permasalahan dalam dunia nyata. Pada awal perkembangannya, metode Algoritma Genetika ini pertama kali diperkenalkan oleh John Holland dari Universitas *Michigan* pada tahun 1975 dalam bukunya yang berjudul " *Adaption in Natural and Artificial Systems*", dan pada akhirnya dipopulerkan oleh salah seorang muridnya, *David Goldberg*, yang mampu memecahkan masalah sulit yang melibatkan kontrol transmisi gas-pipa untuk disertasinya yang berjudul" *Computer-aided gas pipeline operation using genetic algorithms and rule learning*" [6].

Proses seleksi alamiah ini melibatkan perubahan gen yang terjadi pada individu melalui proses perkembang biakan. Dalam algoritma Genetika ini, proses perkembang-biakan ini menjadi proses dasar yang menjadi perhatian utama, dengan dasar berpikir "Bagaimana mendapatkan keturunan yang lebih baik". Algoritma Genetika ini ditemukan oleh *John Holland* dan dikembangkan oleh muridnya *David Goldberg*. Secara umum Algoritma Genetika menyelesaikan permasalahan dengan alur yang terdapat pada Gambar 1.



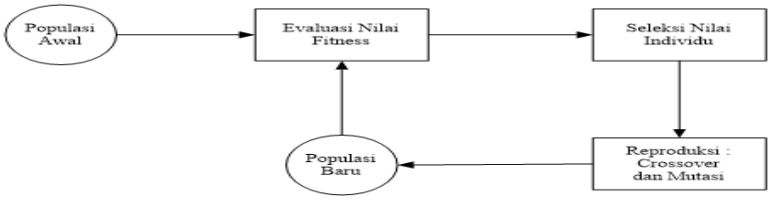
Gambar 1. Penyelesaian Algoritma Genetika Oleh David GoldBerg

2.2 Evaluasi Nilai Fitness

Untuk mengetahui baik tidaknya solusi yang ada pada suatu individu, setiap individu pada populasi harus memiliki nilai pembandingnya (*fitness cost*). Melalui nilai pembanding inilah akan didapatkan solusi terbaik dengan cara pengurutan nilai pembanding dari individu-individu dalam populasi.

Solusi terbaik ini akan dipertahankan, sementara solusi lain yang berubah untuk mendapatkan suatu solusi yang lain lagi, melalui tahap *cross-over* dan mutasi (*mutation*). Sebelum melakukan penempatan

jadwal kelas dilakukan dua buah pengecekan terlebih dahulu, yaitu pencarian hari dan jam yang masih kosong dan pengecekan prioritas yaitu pada hari dan jam mana yang paling tinggi prioritasnya.



Gambar 2. Evaluasi Nilai *Fitness* Oleh *David Goldberg*

Tabel.1 *Fitness cost*

Aturan	***Fitness cost***
Jadwal bertabrakan	Fitness cost – (jumlah mata kuliah bentrok *100)
Ruangan bertabrakan	Fitness cost – (jumlah responsi bentrok *100)
Mapel olahraga di atas jam ke 2	Fitness cost – (jumlah matakuliah semester *10)
Penumpukan jadwal yang sama dalam sehari	Fitness cost – (jumlah praktikum tidak sesuai *100)

Proses evaluasi yang terdapat pada Gambar 2 melibatkan fungsi objektif yang merupakan formula untuk menentukan jumlah nilai yang salah dan dikalkulasikan dengan nilai *fitness* yang disajikan pada Tabel 1. Adapun evaluasi nilai *fitness* dilakukan dengan parameter sebagai berikut:

a. Tidak boleh terjadi tabrakan jadwal.
b. Tidak boleh terjadi penumpukan ruangan.
c. Mata pelajaran olahraga tidak boleh di atas jam ke 2.
d. Tidak boleh terjadi penumpukan jadwal yang sama dalam satu hari.

Apabila terdapat aturan-aturan yang dilanggar maka nilai *fitness cost* akan dikurangi sehingga hasilnya akan menjadi lebih jelek

3. HASIL DAN PEMBAHASAN

3.1 Perencanaan Sistem

Setiap lembaga pendidikan sudah seharunya memiliki jadwal mata pelajaran agar kegiatan belajar mengajar dapat berjalan dengan lancar. saat ini pejadwalan di SMAN 31 masih dengan cara manual yaitu Tabel waktu (*Time Table)*, perancangan jadwal dengan cara ini akan memakan waktu yang cukup lama Karena di sekolah SMAN 31 ini memiliki jumlah kelas sebanyak 29 kelas dengan 3 jurusan yang berbeda sehingga dapat mengurangi produktivitas belajar mengajar. Dengan harapan meningkatkan produktivitas belajar mengajar maka di buatlah aplikasi penjadwalan mata pelajaran di SMAN 31 dengan menggunakan Algoritma Genetika sebagai algoritma pencarian untuk menentukan jadwal yang tepat, sehingga dapat mendukung proses belajar mengajar di sekolah, untuk membuat aplikasi ini di butuhkan data mata pelajaran di SMAN31 yang disajikan pada Tabel 2 (Mata Pelajaran Jurusan IPA), Tabel 3 (Mata Pelajaran Jurusan IPS), dan Tabel 4 (Mata Pelajaran Jurusan Bahasa).

Tabel 2. Mata pelajaran jurusan IPA

Kelas X,XI dan XII IPA	
Matematika	Kewarganegaraan
Bahasa Indonesia	Teknologi Informasi
Bahasa Inggris	Seni Budaya
Fisika	Olahraga
Biologi	Elektronika
Kimia	Sejarah Indonesia
Pendidikan Agama	

Tabel 3. Mata pelajaran jurusan IPS

Kelas X,XI dan XII IPS	
Matematika	Pendidikan Agama
Bahasa Indonesia	Kewarganegaraan
Bahasa Inggris	Teknologi Informasi
Ekonomi	Seni Budaya
Geografi	Olahraga
Sosiologi	Sejarah Indonesia
Prakarya	Sejarah Dunia

Tabel 4. Mata pelajaran jurusan bahasa

Kelas X,XI dan XII Bahasa	
Matematika	Pendidikan Agama
Bahasa Indonesia*	Kewarganegaraan
Bahasa Inggris	Teknologi Informasi
Bahasa Jerman	Seni Budaya
Antropologi	Olahraga
Sastra indonesia	Sastra Inggris
Prakarya	Sejarah Indonesia

3.2 Waterfall Model

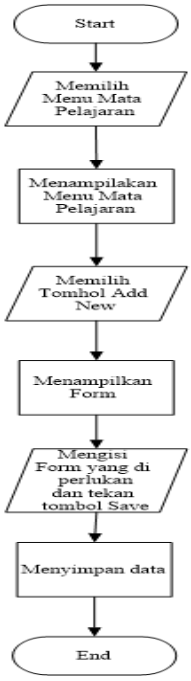
Pengembangan aplikasi ini menggunakan metode *waterfall* karena metode ini mewakili semua bagian penting dalam proses pengembangan aplikasi sistem informasi. Selain itu model ini memiliki fase yang berurut dalam merancang aplikasi Penjadwalan Mata Pelajaran di SMAN31. *Waterfall model* merupakan model *System Development Life Cycle* (SDLC) klasik yang bersifat sistematis. *Model* ini terdiri dari beberapa fase yang dikerjakan secara berurutan untuk membangun *software*. Penjelasan dari fase-fase *waterfall model* yang digunakan adalah :

a. *Requirements Definition*
   Menganalisis dan menentukan menu apa saja yang dibutuhkan dalam aplikasi dengan melakukan analisis spesifikasi kebutuhan pada aplikasi.
b. *System and Software Design*
   Membuat rencana yang akan dilakukan dalam aplikasi dengan merancang *user interface* dan desain untuk kebutuhan aplikasi yang akan dibuat dengan menggunakan arsitektur aplikasi, menggunakan metode *UML* serta *flowchart* sebagai desain alur kerja.
c. *Implementation and Unit Testing*
   Menerjemahkan hasil desain ke dalam bahasa komputer berupa baris *coding* dalam bahasa pemograman *PHP* dan kemudian dilakukan *testing* di setiap unit-unit program pada setiap fungsi.
d. *Integration and System Testing*
   Program unit yang tidak memiliki *error* atau *bugs* akan digabungkan menjadi sebuah kesatuan program atau sistem. Setelah digabungkan maka akan diuji secara keseluruhan menggunakan *blackbox testing*.
e. *Operation and Maintenance*
   Hasil dari uji *testing* yang telah bebas dari *error* atau *bugs* dapat dioperasikan. Apabila ada penambahan fitur atau perbaikan maka dapat dilakukan pengembangan atau perbaikan sistem.

3.3 Flowchart

Flowchart merupakan gambar atau bagan yang memperlihatkan urutan dan hubungan antar proses beserta instruksinya. Dengan adanya *flowchart* urutan proses kegiatan menjadi lebih jelas. Jika ada penambahan proses maka dapat dilakukan lebih mudah. Setelah *flowchart* selesai disusun, selanjutnya pemrogram (*programmer*) menerjemahkannya ke bentuk program dengan bahasa pemograman.

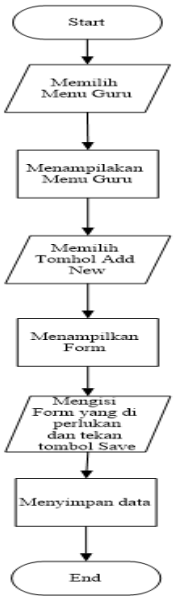
a. *Flowchart Menu* Mata Pelajaran



Gambar 3. *Flowchart* Mata Pelajaran

Gambar 3 adalah langkah-langkah yang di lakukan pada bagian *menu* mata pelajaran. Pertama *user* akan memilih *menu* mata pelajaran dan sistem akan menampilkan tampilan *menu* mata pelajaran, kemudian *user* dapat memilih tombol *add new* untuk menambahkan mata pelajaran, kemudian sistem akan menampilkan *Form* untuk mengisi data mata pelajaran kemudian *user* menyimpan data.

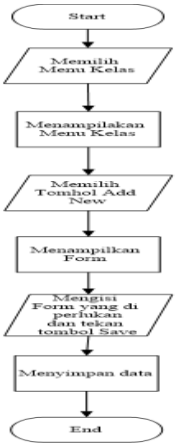
b. *Flowchart* menu Guru



Gambar 4. *Flowchart* Guru

Gambar 4 adalah langkah-langkah yang di lakukan *User* saat sudah memilih *menu* Guru. Pertama *user* memilih *menu* guru kemudian sistem akan menampilkan tampilan *menu* Guru kemudian *user* dapat memilih tombol *add new* dan kemudian sistem akan menampilkan *Form* untuk mengisi data Guru yang di perlukan.

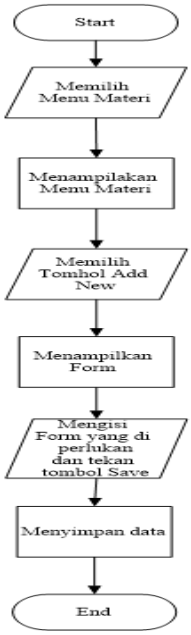
c. *Flowchart menu* Kelas



Gambar 5. *Flowchart* Kelas

Gambar 5 menunjukan langkah-langkah yang dapat di lakukan oleh *User* di dalam *menu* Kelas, Setelah *User* memilih *menu* kelas, maka sistem akan menampilkan *menu* kelas sehingga *User* dapat menambahkan data Kelas.

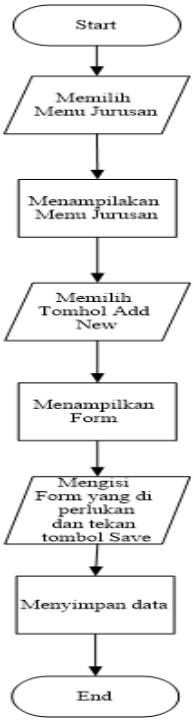
d. *Flowchart Menu* Materi



Gambar 6. *Flowchart* Materi

Gambar 6 adalah langkah-langkah *User* dalam menambahkan Materi-Materi Baru. Jika *User* memilih *menu* materi maka *User* dapat menambahkan Materi-Materi yang kurang kedalam aplikasi.

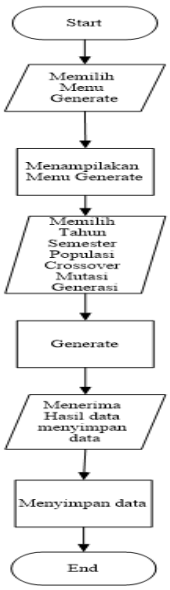
e. *Flowchart Menu* Jurusan



Gambar 7. *Flowchart* Jurusan

Gambar 7 adalah Langkah-langkah yang dapat di lakukan oleh *User* dalam menambahkan Jurusan. saat *User* Memilih *menu* jurusan, maka sistem akan menampilkan *menu* jurusan, kemudian *User* dapat menambahkan Jurusan yang lain.

f. *Flowchart Menu Generate*



Gambar 8. *Flowchart Generate*

Gambar 8 adalah langkah-langkah dalam menampilkan data yang sudah di ada di *menu-menu* sebelumnya dan di lakukan *generate* sehingga *User* mendapatkan hasil Jadwal dari Algoritma Genetika.

3.4 Analisis Kebutuhan

Dalam merancang sebuah aplikasi dibutuhkan banyak kebutuhan yang menunjang kesuksesan pembuatan aplikasi. Oleh karena itu diperlukannya analisis kebutuhan yang akan digunakan sebagai acuan terkait dengan kebutuhan yang diperlukan nantinya. Analisa kebutuhan bertujuan untuk memperoleh informasi mengenai kebutuhan *input* yang dibutuhkan untuk proses Penjadwalan Mata Pelajaran sehingga menjadi data untuk bisa mendapatkan hasil yang maksimal. Kebutuhan proses yaitu kebutuhan akan proses yang terjadi sebelum dilakukannya proses algoritma, dan kebutuhan *output* yang dibutuhkan dari sistem aplikasi yang ada. Kebutuhan masukan atau *input* yang dimaksud adalah kebutuhan *input* apa saja yang dibutuhkan pada saat proses pengerjaan algoritma *Genetika* dan sistem

3.4.1 Analisis Kebutuhan Fungsional

Sistem yang di rancang memiliki kebutuhan fungsional seperti pada umumnya yaitu :

a. Dapat memberikan sebuah hasil yaitu penjadwalan mata pelajaran di sekolah SMAN31.
b. Dapat memberikan kemudahan dalam pememeriksaan kurikulum / mata pelajaran di sekolah SMAN31.

3.4.2 Analisis Kebutuhan Non Fungsional

Dalam Perancangan Aplikasi ini, diperlukan spesifikasi sebagai berikut:

a. Sistem Operasi: Windows 7 SP1 atau lebih tinggi / Windows Server 2012 R2 (dengan *update* 2919355) atau lebih tinggi
b. *Hardware*: 1.8 GHz *or faster processor, Dual-core* atau lebih baik / 2 GB of RAM atau lebih tinggi / Penyimpanan 1GB – 40GB / *Video Card* WXGA (1366 by 768) atau lebih baik
c. Lainnya: .NET *framework* 4.5 / Windows 10 *Enterprise* LTSB *edition* dan Windows 10 S hanya bisa Visual Studio 2017 / *Internet Explorer* 11 atau *Edge* / SQL *Server* 2014 atau SQL *Server* 2016

3.5 Fungsi Fitness dan seleksi

Individu-individu dalam populasi telah terbentuk, maka langkah selanjutnya adalah menghitung nilai *fitness* setiap individu. Penghitungan dilakukan dengan memberikan pinalti untuk setiap aturan yang digunakan dalam penjadwalan. Semakin wajib aturan dilaksanakan, maka akan semakin besar nilai pinalti yang diberikan. Aturan penghitungan fungsi *fitness* yang dapat dilihat pada persamaan 1.

$$f(g) = 1/\,(1 + \sum P_1\, v_1(g)) \quad (1)$$

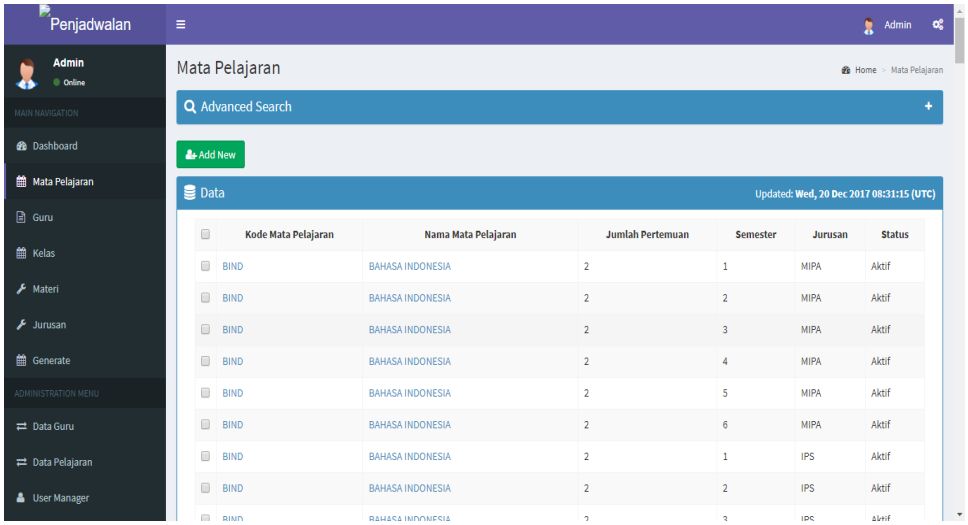
Dimana p_1 adalah pinalti yang diberikan untuk aturan *i*, dan v_1(g) = 1 jika jadwal *g* melanggar aturan *i*, bernilai 0 jika sebaliknya.

Tabel 5.Aturan dan seleksi *fitness*

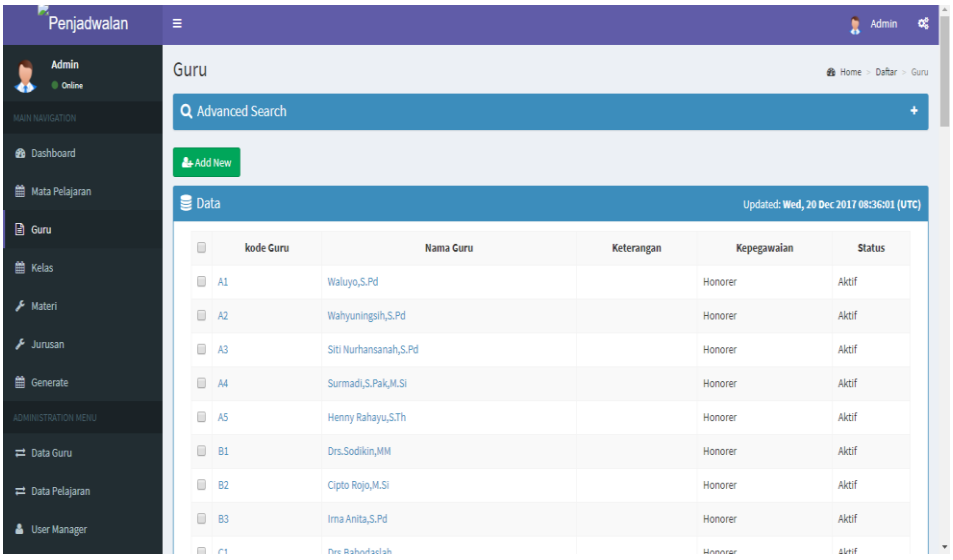
Aturan	*Nilai Pinalti*
Kesediaan waktu guru	1
Bentrok mata pelajaran wajib dan pilihan	2
Bentok waktu guru	3
Bentrok mata pelajaran satu semester	3

Apabila penghitungan *fitness* setiap indivudu telah dilakukan, maka langkah selanjutnya adalah seleksi induk. Seleksi yang digunakan adalah seleksi roda *roulette* yang terdapat pada Tabel 5. Pada seleksi roda *roulette*, semakin tinggi nilai *fitness* maka semakin besar kemungkinan untuk terpilih menjadi induk.

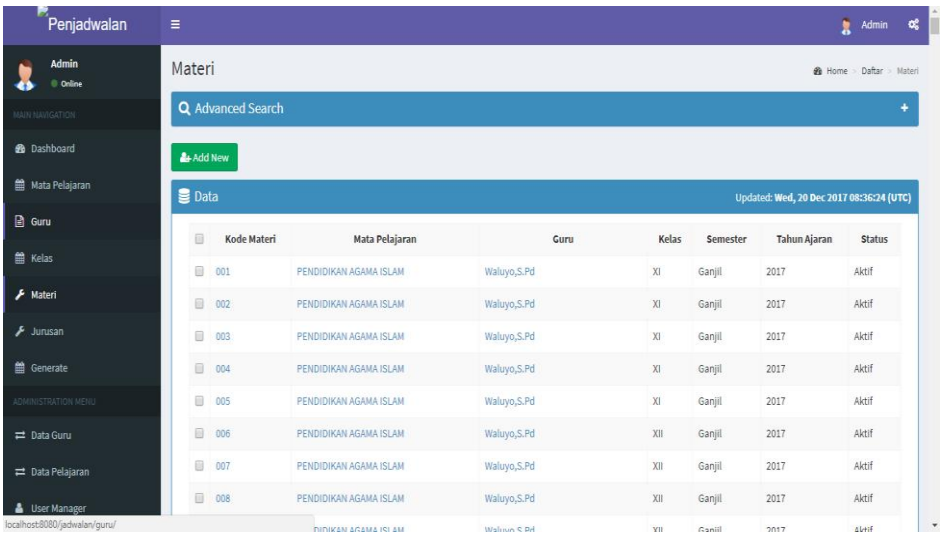
3.6 Hasil



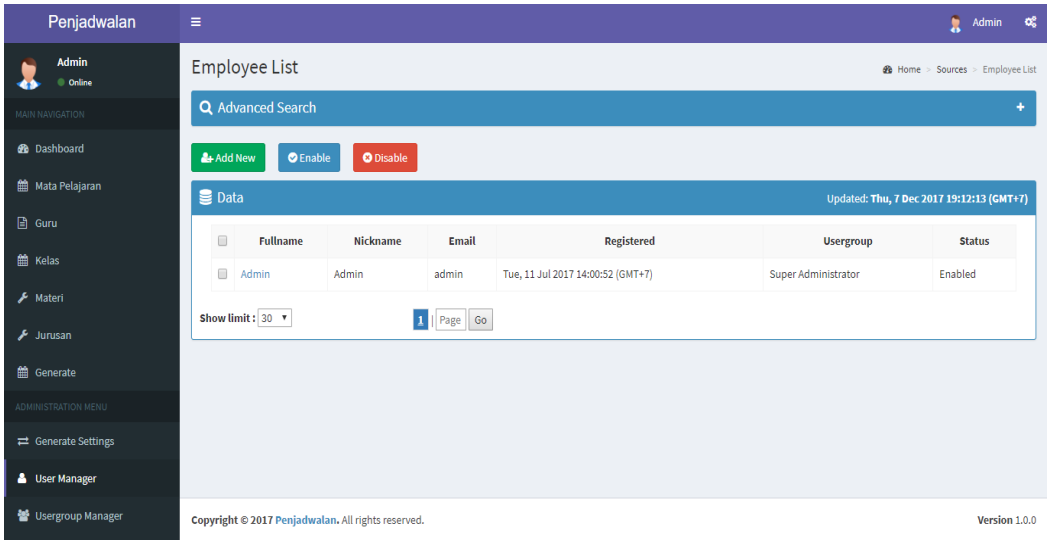
Gambar 10. Isi Data Mata Pelajaran Dari Aplikasi Penjadwalan



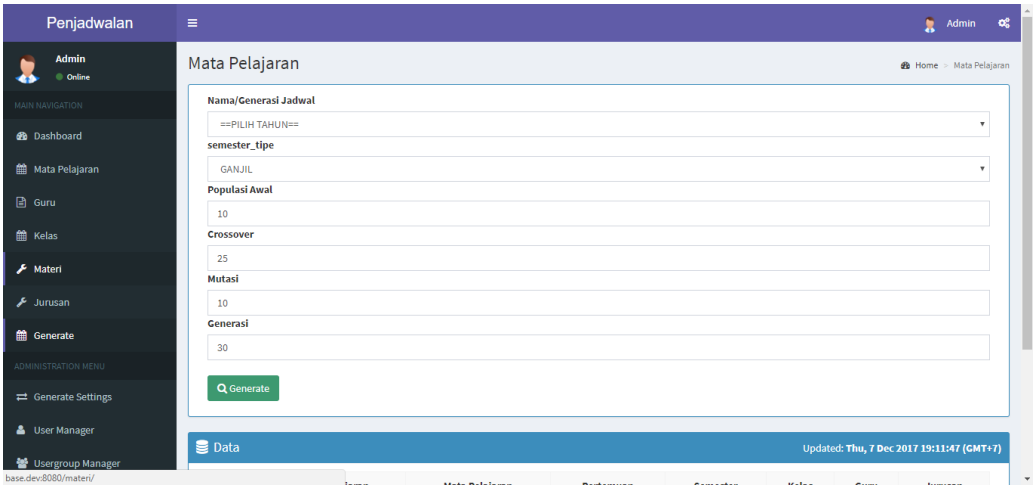
Gambar 11. Tampilan Data Guru Dari Aplikasi Penjadwalan



Gambar 12. Isi Data Materi Dari Aplikasi Penjadwalan



Gambar 13. Isi Data *User* Dari Aplikasi Penjadwalan.



Gambar 14. Tampilan *Menu* Dari *Generate*

Berikut ini adalah cara perhitungan dari *Generate* yang telah di terapkan kedalam algoritma genetika:

3.6.1 Pembentukan Chromosome

Karena Pembentukan *Chromosome* yang di cari adalah jadwal maka variabel A1, A2, B1, B2, ... di jadikan sebagai gen-gen pembentuk kromosom. Batasan nilai variabel untuk hari adalah bilangan *integer* 1 sampai 5 dan batasan nilai variabel untuk jam adalah bilangan *integer* 1 sampai 10.

3.6.2 Inisialisasi

Proses inisialisasi dilakukan dengan cara memberikan nilai awal gen-gen dengan nilai acak sesuai batasan yang telah ditentukan.Misalkan kita tentukan jumlah populasi adalah 6, maka:

Chromosome[1] = [A1;A2;B1;B2;...] = [[1][10];[4][4];[2][4];[1][8];...]
Chromosome[2] = [A1;A2;B1;B2;...] = [[4][9];[3][10];[5][2];[1][5];...]
Chromosome[3] = [A1;A2;B1;B2;...] = [[3][8];[5][9];[5][6];[2][5];...]
Chromosome[4] = [A1;A2;B1;B2;...] = [[5][7];[1][2];[2][7];[3][8];...]
Chromosome[5] = [A1;A2;B1;B2;...] = [[1][6];[2][6];[3][5];[5][5];...]
Chromosome[6] = [A1;A2;B1;B2;...] = [[3][2];[5][8];[5][2];[2][3];...]

3.6.3 Evaluasi Chromosome

Permasalahan yang ingin diselesaikan adalah nilai variabel A1, A2, B1, B2 , ... yang tidak bertabrakan, maka fungsi_objektif yang dapat digunakan untuk mendapatkan solusi adalah fungsi_objektif(*chromosome*) = | if([i][j]==[i+x][j+n]) *fitness* + 1 Kita hitung fungsi_objektif dari chromosome yang telah dibangkitkan:

fungsi_objektif(chromosome[1]) = 0 + 0 + 1 + 1 + 0 + 1 + 1 = 4
fungsi_objektif(chromosome[2]) = 0 + 1 + 0 + 0 + 0 + 1 + 0 = 2
fungsi_objektif(chromosome[3]) = 1 + 0 + 0 + 1 + 1 + 0 + 1 = 4
fungsi_objektif(chromosome[4]) = 1 + 1 + 0 + 1 + 0 + 1 + 1 = 5
fungsi_objektif(chromosome[5]) = 1 + 0 + 1 + 0 + 0 + 1 + 0 = 3
fungsi_objektif(chromosome[6]) = 0 + 1 + 1 + 0 + 0 + 0 + 1 = 3
Rata-rata dari fungsi objektif adalah:
rata-rata = (4+2+4+5+3+3)/6 = 21 / 6 = 3,5

3.6.4 Seleksi Chromosome

Proses seleksi dilakukan dengan cara membuat *chromosome* yang mempunyai fungsi_objektif kecil mempunyai kemungkinan terpilih yang besar atau mempunyai nilai probabilitas yang tinggi. Untuk itu dapat digunakan fungsi *fitness* = (1/(1+fungsi_objektif)), fungsi_objektif perlu ditambah 1 untuk menghindari kesalahan program yang diakibatkan pembagian oleh 0.

fitness[1] = 1 / (fungsi_objektif[1]+1) = 1 / 5 = 0.2
fitness[2] = 1 / (fungsi_objektif[2]+1) = 1 / 3 = 0.3333
fitness[3] = 1 / (fungsi_objektif[3]+1) = 1 / 5 = 0.2
fitness[4] = 1 / (fungsi_objektif[4]+1) = 1 / 6 = 0.1667
fitness[5] = 1 / (fungsi_objektif[5]+1) = 1 / 4 = 0.25
fitness[6] = 1 / (fungsi_objektif[6]+1) = 1 / 4 = 0.25
total_fitness = 0.2 + 0.3333 + 0.2 + 0.1667 + 0.25 + 0.25 = 1.4

Rumus untuk mencari probabilitas: P[i] = fitness[i] / total_fitness :

P[1] = 0.2 / 1.4 = 0.1428
P[2] = 0. 3333 / 1.4 = 0.2381
P[3] = 0. 2 / 1.4 = 0.1428
P[4] = 0. 1667 / 1.4 = 0.1191
P[5] = 0.25 / 1.4 = 0.1786
P[6] = 0.25 / 1.4 = 0.1786

Dari probabilitas diatas dapat kita lihat kalau *chromosome* ke 2 yang mempunyai *fitness* paling besar maka *chromosome* tersebut mempunyai probabilitas untuk terpilih pada generasi selanjutnya lebih besar dari *chromosome* lainnya. Untuk proses seleksi kita gunakan *roulete wheel*, untuk itu kita harus mencari dahulu nilai kumulatif probabilitasnya:
C[1] = 0.1428
C[2] = 0.1428+ 0.2381 = 0.3809
C[3] = 0.1428+ 0.2381 + 0.1428 = 0.5237
C[4] = 0.1428+ 0.2381 + 0.1428 + 0.1191 = 0.6428
C[5] = 0.1428+ 0.2381 + 0.1428 + 0.1191 + 0.1786 = 0.8214
C[6] = 0.1428+ 0.2381 + 0.1428 + 0.1191 + 0.1786 + 0.1786 = 1

Setelah dihitung cumulative probabilitasnya maka proses seleksi menggunakan *roulete-wheel* dapat dilakukan. Prosesnya adalah dengan membangkitkan bilangan acak R dalam *range* 0-1.Jika R[k] < C[1] maka pilih *chromosome* 1 sebagai induk, selain itu pilih *chromosome* ke-k sebagai induk dengan syarat C[k-1] < R < C[k]. Kita putar *roulete wheel* sebanyak jumlah populasi yaitu 6 kali (bangkitkan bilangan acak R) dan pada tiap putaran, kita pilih satu *chromosome* untuk populasi baru. Misal:

R[1] = 0.201, R[2] = 0.284, R[3] = 0.009, R[4] = 0.822, R[5] = 0.398, R[6] = 0.501,

Angka acak pertama R[1] adalah lebih besar dari C[1] dan lebih kecil daripada C[2] maka pilih chromosome[2] sebagai *chromosome* pada populasi baru, dari bilangan acak yang telah dibangkitkan diatas maka populasi *chromosome* baru hasil proses seleksi adalah:

chromosome[1] = chromosome[2]
chromosome[2] = chromosome[2]
chromosome[3] = chromosome[1]

chromosome[4] = chromosome[5]
chromosome[5] = chromosome[2]
chromosome[6] = chromosome[3]
Chromosome baru hasil proses seleksi:
chromosome[1] = [[4][9];[3][10];[5][2];[1][5];...]
chromosome[2] = [[4][9];[3][10];[5][2];[1][5];...]
chromosome[3] = [[1][10];[4][4];[2][4];[1][8];...]
chromosome[4] = [[3][2];[5][8];[5][2];[2][3];...]
chromosome[5] = chromosome[2]
chromosome[6] = chromosome[3]
Chromosome baru hasil proses seleksi:
chromosome[1] = [[4][9];[3][10];[5][2];[1][5];...]
chromosome[2] = [[4][9];[3][10];[5][2];[1][5];...]
chromosome[3] = [[1][10];[4][4];[2][4];[1][8];...]
chromosome[4] = [[3][2];[5][8];[5][2];[2][3];...]
chromosome[5] = [[4][9];[3][10];[5][2];[1][5];...]
chromosome[6] = [[3][2];[5][8];[5][2];[2][3];...]

3.6.5 Crossover

Setelah proses seleksi maka proses selanjutnya adalah proses *crossover*. Metode yang digunakan salah satunya adalah *one-cut point*, yaitu memilih secara acak satu posisi dalam *chromosome* induk kemudian saling menukar gen. *Chromosome* yang dijadikan induk dipilih secara acak dan jumlah *chromosome* yang mengalami *crossover* dipengaruhi oleh parameter crossover_rate (ρc). *Pseudocode* untuk proses *crossover* adalah sebagai berikut:

```
begin
       k <- 0;
       while(k < populasi) do
R[k] <- random(0-1);
       if (R[k] < ρc ) then
       select Chromosome[k] as parent;
       end;
       k = k + 1;
end;
end;
```

Misal kita tentukan *crossover probability* adalah sebesar 25%, maka diharapkan dalam satu generasi ada 50% *Chromosome* (3 *chromosome*) dari satu generasi mengalami proses *crossover*. Prosesnya adalah sebagai berikut:

Pertama kita bangkitkan bilangan acak R sebanyak jumlah populasi

R[1] = 0.191, R[2] = 0.259, R[3] = 0.760, R[4] = 0.006, R[5] = 0.159, R[6] = 0.340

Maka *Chromosome* ke k akan dipilih sebagai induk jika R[k] < ρc, dari bilangan acak R diatas maka yang dijadikan induk adalah chromosome[1], chromosome[4] dan chromosome[5]. Setelah melakukan pemilihan induk proses selanjutnya adalah menentukan posisi *crossover*. Ini dilakukan dengan cara membangkitkan bilangan acak dengan batasan 1 sampai (panjang chromosome-1), dalam kasus ini bilangan acak yang dibangkitkan adalah 1 – 3. Misalkan didapatkan posisi *crossover* adalah 1 maka *chromosome* induk akan dipotong mulai gen ke 1 kemudian potongan gen tersebut saling ditukarkan antar induk.

chromosome[1] >< chromosome[4], chromosome[4] >< chromosome[5], chromosome[5] >< chromosome[1]

Posisi *cut-point crossover* dipilih menggunakan bilangan acak 1-3 sebanyak jumlah *crossover* yang terjadi, misal

C[1] = 1, C[2] = 1, C[3] = 2
offspring[1] = chromosome[1] >< chromosome[4]= [[4][9];[3][10];[5][2];[1][5];...] >< [[3][2];[5][8];[5][2];[2][3];...] = [[4][9];[3][10];[5][2];[2][3];...]
offspring[4] = Chromosome[4] >< Chromosome[5] = [[3][2];[5][8];[5][2];[2][3];...] >< [[4][9];[3][10];[5][2];[1][5];...] = [[3][2];[5][8];[5][2];[1][5];...]
offspring[5] = Chromosome[5] >< Chromosome[1] = [[4][9];[3][10];[5][2];[1][5];...] >< [[4][9];[3][10];[5][2];[1][5];...]= [[4][9];[3][10];[5][2];[1][5];...]

Dengan demikian populasi *Chromosome* setelah mengalami proses *crossover* menjadi:

chromosome[1] = [[4][9];[3][10];[5][2];[2][3];...]
chromosome[2] = [[4][9];[3][10];[5][2];[1][5];...]
chromosome[3] = [[3][2];[5][8];[5][2];[1][5];...]
chromosome[4] = [[4][9];[3][10];[5][2];[1][5];...]
chromosome[5] = [[3][2];[5][8];[5][2];[1][5];...]
chromosome[6] = [[4][9];[3][10];[5][2];[1][5];...]

3.6.6 Mutasi

Jumlah *chromosome* yang mengalami mutasi dalam satu populasi ditentukan oleh parameter mutation_rate. Proses mutasi dilakukan dengan cara mengganti satu gen yang terpilih secara acak dengan suatu nilai baru yang didapat secara acak. Prosesnya adalah sebagai berikut. Pertama kita hitung dahulu panjang total gen yang ada dalam satu populasi. Dalam kasus ini panjang total gen adalah total_gen = (jumlah gen dalam *chromosome*) * jumlah populasi = 29 * 10 * 5 * 6 = 8700, Untuk memilih posisi gen yang mengalami mutasi dilakukan dengan cara membangkitkan bilangan integer acak antara 1 sampai total_gen, yaitu 1 sampai 24. Jika bilangan acak yang kita bangkitkan lebih kecil daripada variabel mutation_rate (ρm) maka pilih posisi tersebut sebagai *sub-chromosome* yang mengalami mutasi. Misal ρm kita tentukan 10% maka diharapkan ada 10% dari total_gen yang mengalami populasi: jumlah mutasi = 0.1 * 8700 = 870, Misalkan setelah kita bangkitkan bilangan acak terpilih posisi gen 12 dan 18 yang mengalami mutasi. Dengan demikian yang akan mengalami mutasi adalah *chromosome* ke-3 gen nomor 4 dan *Chromosome* ke-5 gen nomor 2. Maka nilai gen pada posisi tersebut kita ganti dengan bilangan acak 0-30. Misalkan bilangan acak yang terbangkitkan adalah 2 dan 5. Maka populasi *chromosome* setelah mengalami proses mutasi adalah:

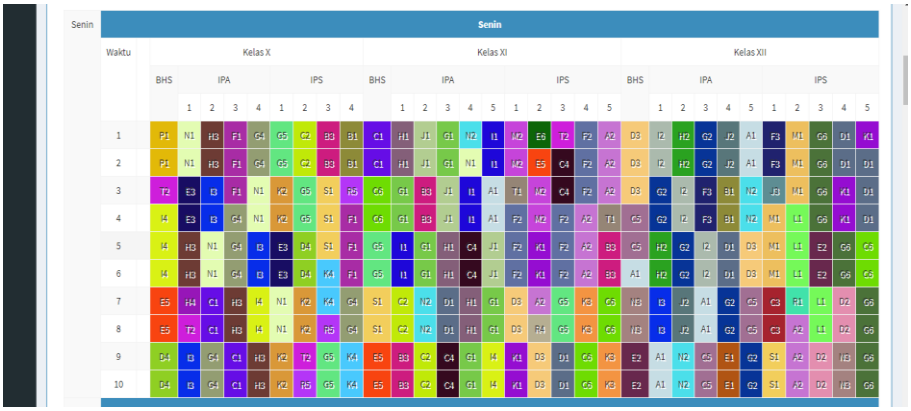
chromosome[1] = [[4][9];[4][10];[5][2];[2][3];...]
chromosome[2] = [[4][9];[1][9];[5][2];[1][5];...]
chromosome[3] = [[3][2];[5][8];[5][2];[4][7];...]
chromosome[4] = [[4][9];[3][5];[5][2];[1][5];...]
chromosome[5] = [[3][2];[2][8];[5][2];[5][5];...]
chromosome[6] = [[4][9];[3][1];[5][2];[1][6];...]

Setelah proses mutasi maka kita telah menyelesaikan satu iterasi dalam algoritma genetika atau disebut dengan satu generasi. Maka fungsi_objective setelah satu generasi adalah:

fungsi_objektif(chromosome[1]) = 0 + 0 + 1 + 1 + 0 + 1 + 1 = 4
fungsi_objektif(chromosome[2]) = 0 + 1 + 0 + 0 + 0 + 1 + 0 = 2
fungsi_objektif(chromosome[3]) = 1 + 0 + 0 + 1 + 1 + 0 + 1 = 4
fungsi_objektif(chromosome[4]) = 1 + 1 + 0 + 1 + 0 + 1 + 1 = 5
fungsi_objektif(chromosome[5]) = 1 + 0 + 1 + 0 + 0 + 1 + 0 = 3
fungsi_objektif(chromosome[6]) = 0 + 1 + 1 + 0 + 0 + 0 + 1 = 3

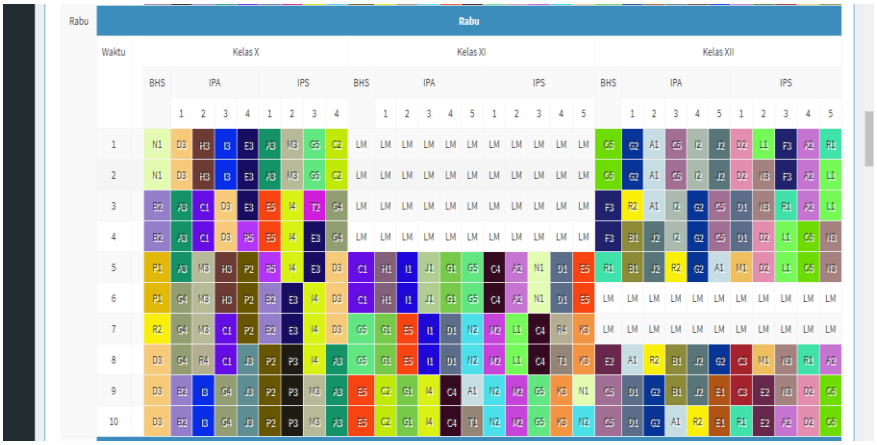
Chromosome ini akan mengalami proses yang sama seperti generasi sebelumnya yaitu proses evaluasi, seleksi, *crossover* dan mutasi yang kemudian akan menghasilkan *chromosome* baru untuk generasi yang selanjutnya. Proses ini akan berulang sampai sejumlah generasi yang telah ditetapkan sebelumnya. Setelah 50 generasi didapatkan chromosome yang terbaik adalah Chromosome = [[3][2];[2][8];[5][2];[5][5];...] yang tidak memiliki bentrok pelajaran dan guru

Setelah mengimplementasikan Algoritma kedalam aplikasi pejadwalan maka di uji aplikasi tersebut sebanyak 30 kali, di bawah ini adalah salah satu contoh hasil dari aplikasi penjadwalan.

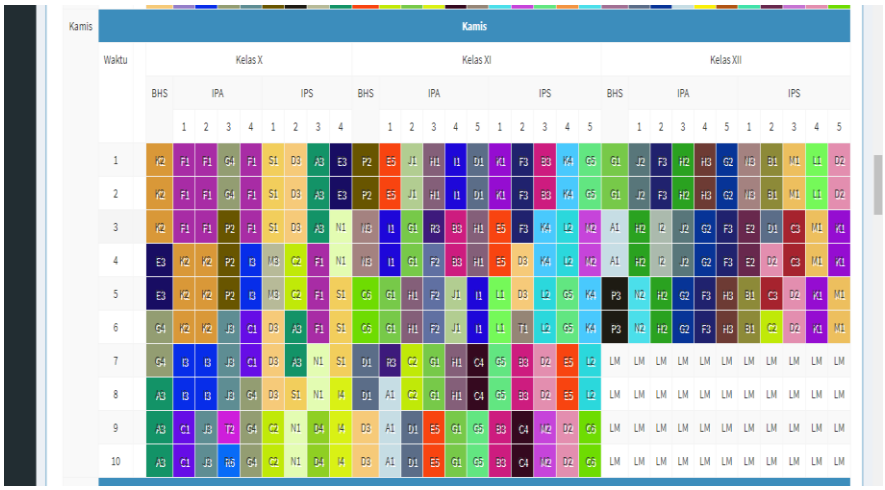


Gambar 16. Hasil Pengujian Jadwal A (Hari Senin)

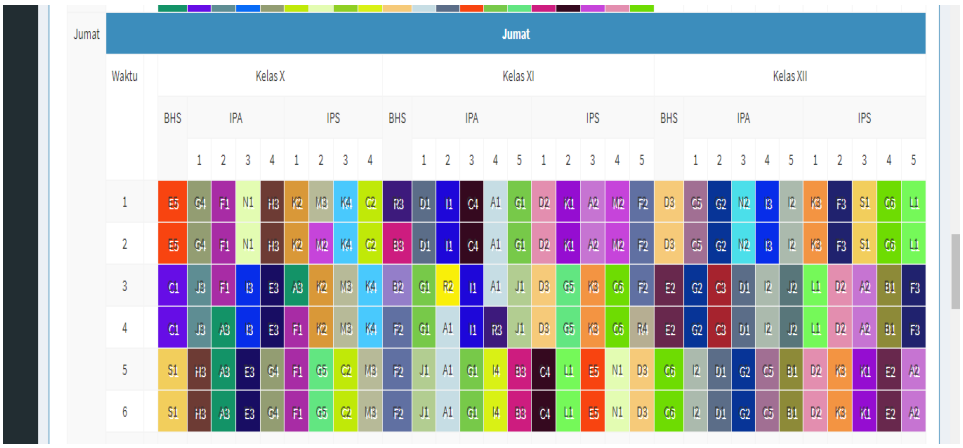
Gambar 17. Hasil Pengujian Jadwal B (Hari Selasa)



Gambar 18. Hasil Pengujian Jadwal C (Hari Rabu)



Gambar 19. Hasil Pengujian Jadwal D (Hari Kamis)



Gambar 20. Hasil Pengujian Jadwal E (Hari Jumat)

Tabel 6 adalah rangkuman hasil pengujian sebanyak 30 kali pengujian.

Tabel 6. Hasil pengujian aplikasi

Pengujian Ke	*Deskripsi*	*Pengujian ke*	*Deskripsi*	*Pengujian ke*	*Deskripsi*
1	Tidak ada yang bertabrakan	11	Tidak ada yang bertabrakan	21	Tidak ada yang bertabrakan
2	Tidak ada yang bertabrakan	12	Tidak ada yang bertabrakan	22	Tidak ada yang bertabrakan
3	Tidak ada yang bertabrakan	13	Tidak ada yang bertabrakan	23	Tidak ada yang bertabrakan
4	Tidak ada yang bertabrakan	14	Tidak ada yang bertabrakan	24	Tidak ada yang bertabrakan
5	Tidak ada Yang bertabrakan	15	Tidak ada yang bertabrakan	25	Tidak ada yang bertabrakan
6	Tidak ada yang bertabrakan	16	Tidak ada yang bertabrakan	26	Tidak ada yang bertabrakan
7	Tidak ada yang bertabrakan	17	Tidak ada yang bertabrakan	27	Tidak ada yang bertabrakan
8	Tidak ada yang bertabrakan	18	Tidak ada yang bertabrakan	28	Tidak ada yang bertabrakan
9	Tidak ada yang bertabrakan	19	Tidak ada yang bertabrakan	29	Tidak ada yang bertabrakan
10	Tidak ada yang bertabrakan	20	Tidak ada yang bertabrakan	30	Tidak ada yang bertabrakan

4. KESIMPULAN

Berdasarkan hasil Penelitian ini, aplikasi Penjadwalan Mata Pelajaran Di SMAN31 berbasis Web menggunakan Algoritma *Genetika*, dapat disimpulkan bahwa Aplikasi ini dapat memberikan Hasil berupa Informasi Jadwal Mata Pelajaran di SMAN 31 beserta informasi mengenai nama Guru, Kelas, Jurusan, dan *Grade*. Algoritma Genetika yang berhasil diimplementasikan pada sistem dapat dibuktikan berdasarkan hasil Informasi Jadwal mata pelajaran yang merupakan hasil pengujian nya tidak memiliki bentrok antara

jadwal 1 dan jadwal lainya, dengan mengimplementasikan algoritma Genetika proses penjadwalan mata pelajaran di SMAN31 menjadi lebih cepat dari proses manual.

DAFTAR PUSTAKA

[1] A. Nora, et al. 2012. "Penjadwalan Pesanan Menggunakan Algoritma Genetika Untuk Tipe Produksi Hybrid And Flexible Flowshop Pada Industri Kemasan Karton". Jurnal Teknik Industri ISSN: 1411-6380, pp. 176-188.
[2] W. A. Puspaningrum, 2013. "Penjadwalan Mata Kuliah Menggunakan Algoritma Genetika di Jurusan Sistem Informasi," ISSN: 2337-3539, vol. II, pp. 127-131.
[3] R. Destia, 2013. "Perancangan Aplikasi Prnjadwalan Mata Pelajaran Menggunakan Algoritma Genetika," Pelita Informatika Budi Darma, ISSN: 2301-9425, vol. V, pp. 148-151.
[4] Krisnandi, K. and Agung, H. 2017. “Implementasi Algoritma Genetika untuk Memprediksi Waktu dan Biaya Pengerjaan Proyek Konstruksi”. Jurnal FIFO.
[5] Widodo, A. W. and W. F. Mahmudy. 2010, "Penerapan Algoritma Genetika Pada Sistem Rekomendasi Wisata Kuliner", Jurnal Ilmiah KURSOR, Vol.5. No.4.
[6] Goldberg, D. 1987. “Computer-aided gas pipeline operation using genetic algorithm and rule learning”. e-ISSN: 1435-5663, e-ISSN 0177-0667.