BAB III

METODOLOGI DAN PERANCANGAN SISTEM

3.1 Metodologi

Metode yang digunakan dalam penelitian ini adalah sebagai berikut.

1. Studi pustaka

   Pada tahap ini dilakukan studi terhadap jurnal yang telah ditulis oleh peneliti lain, artikel atau buku yang berkaitan dengan arsitektur atau rancangan rumah yang ideal, artikel dari internet mengenai tahapan algoritma genetika, mempelajari *source code* aplikasi yang menggunakan algoritma genetika, dan mempelajari HTML5 untuk menghasilkan gambar *layout* yang sudah dirancang.

2. Analisis masalah

   Pada tahap ini dilakukan analisis dan observasi data dari hasil penelitian sebelumnya untuk menetapkan kebutuhan dan batasan dalam membangun aplikasi perancang *layout* rumah.

3. Perancangan sistem

   Pada tahap ini dilakukan perancangan algoritma dan *database* yang akan diterapkan dalam aplikasi perancang *layout* rumah berdasarkan tujuan, batasan, dan kebutuhan yang sudah ditetapkan.

4. Implementasi

   Pada tahap ini dilakukan implementasi algoritma dan *database* yang telah dirancang untuk membangun aplikasi perancang *layout* rumah berbasis *website* menggunakan bahasa pemrograman PHP, Javascript, dan HTML5.

5. Pengujian

   Pada tahap ini dilakukan pengujian terhadap aplikasi yang sudah dibuat dengan melakukan percobaan sebanyak *n*-kali dan menganalisis hasil dari pengujian yang dilakukan. Pengujian yang dilakukan yaitu pengujian skenario, pengujian performa, dan pengujian berdasarkan *feedback* dari pengguna.

6. Penulisan Laporan

   Pada tahap ini dilakukan penulisan laporan secara bertahap, mulai dari tahap studi pustaka sampai dengan pengujian aplikasi.

Beberapa variabel yang diteliti dalam penelitian ini, antara lain sebagai berikut.

1. Algoritma genetika dalam menentukan populasi baru dan fungsi *fitness* pada sistem untuk memenuhi kriteria dari pengguna, sehingga sistem dapat menghasilkan rancangan *layout* rumah yang ideal.
2. Data yang diperlukan sebagai indikator, yaitu aturan rancangan *layout* rumah yang baik, proses pengecekan tiap ruangan, dan ukuran jenis ruangan (kamar, ruang tamu, ruang makan, kamar mandi, dapur, garasi, gudang, ruang *laundry*, teras, dan halaman).

3.2 Perancangan Aplikasi

Secara garis besar, aplikasi dapat menampilkan hasil rancangan *layout* rumah dengan menggunakan algoritma genetika setelah minimal diberikan parameter atau *input* ukuran lahan. Parameter tiap ruangan dapat di-*generate* secara acak oleh aplikasi, kecuali jika pengguna memasukkan *input* jumlah dan ukuran dari tiap jenis ruangan.

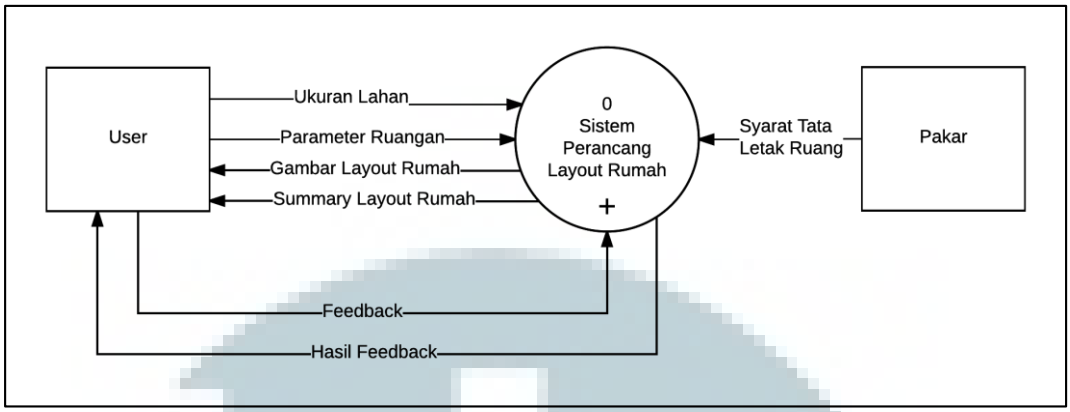
Gambar *layout* rumah yang dihasilkan memiliki arah bangunan yang sama, yaitu menghadap ke atas. Parameter jumlah tiap jenis ruangan memiliki batas minimal dan batas maksimal sehingga aplikasi dapat merancang *layout* rumah yang memiliki kebutuhan ruang paling sederhana dan *layout* rumah yang beragam kebutuhannya. Tabel 3.1 menunjukkan batas minimal dan maksimal dari jumlah tiap jenis ruangan. Prioritas menunjukkan urutan ruangan yang akan dirancang, sehingga jika solusi tidak ditemukan, ruangan akan dihilangkan dari prioritas terendah.

Tabel 3.1 Tabel Batas Minimal dan Maksimal Jumlah Tiap Jenis Ruangan

Prioritas	**Jenis Ruangan**	**Jumlah Minimal**	**Jumlah Maksimal**	**Keterangan**
1	Kamar	1	5	Harus ada
2	Ruang tamu	1	1	Harus ada
3	Kamar mandi	1	2	Harus ada
4	Dapur	1	1	Harus ada
5	Ruang makan	1	1	Harus ada
6	Teras	0	2	Optional
7	Garasi	0	1	Optional
8	Gudang	0	1	Optional
9	Ruang *laundry*	0	1	Optional
10	Halaman	0	2	Optional

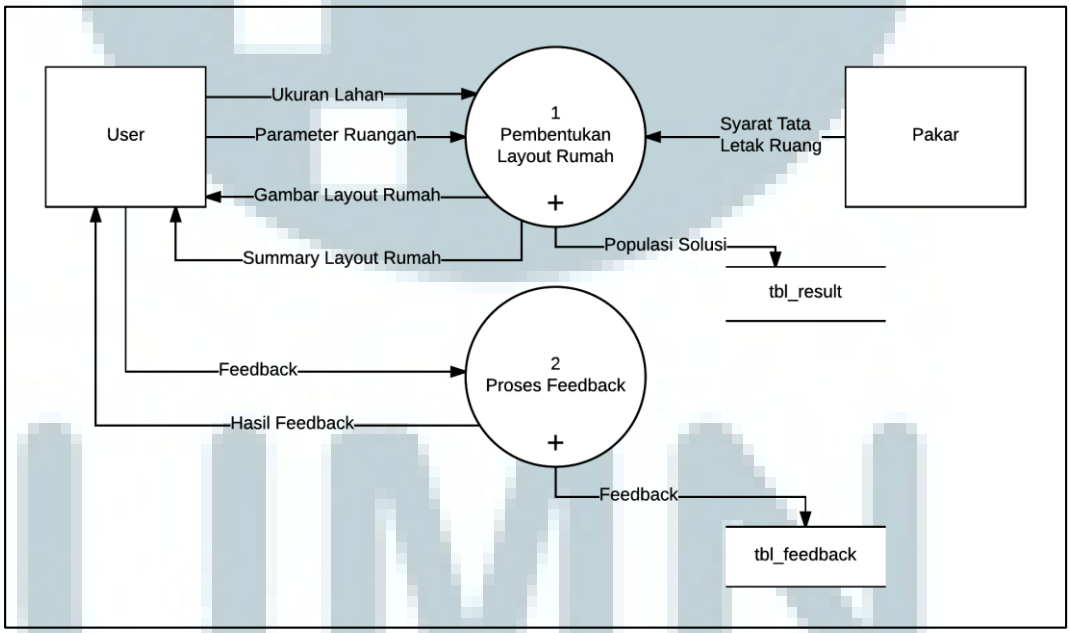
3.2.1 Data Flow Diagram

Berdasarkan data *input* dari pengguna dan *output* dari sistem, dapat digambarkan *Data Flow Diagram* dengan dua entitas, yaitu *user* sebagai pengguna aplikasi dan pakar sebagai penyedia syarat-syarat tata letak ruang. Syarat tata letak ruang yang digunakan sebagai data dari entitas pakar berdasarkan teori Sastra (2012) dan Liliana (2006). Gambar 3.1 menunjukkan diagram *level* 0 atau diagram konteks dari sistem yang dibangun.



Gambar 3.1 Diagram *Level* 0 (Diagram Konteks)

Dari diagram konteks pada gambar 3.1, dapat dijabarkan menjadi diagram *level* 1 dengan dua proses, yaitu proses pembentukan *layout* rumah dan proses *feedback*. Proses pembentukan *layout* rumah terhubung dengan tabel tbl_result, sedangkan proses *feedback* terhubung dengan tabel tbl_feedback. Gambar 3.2 menunjukkan diagram *level* 1 dari aplikasi yang dibangun.

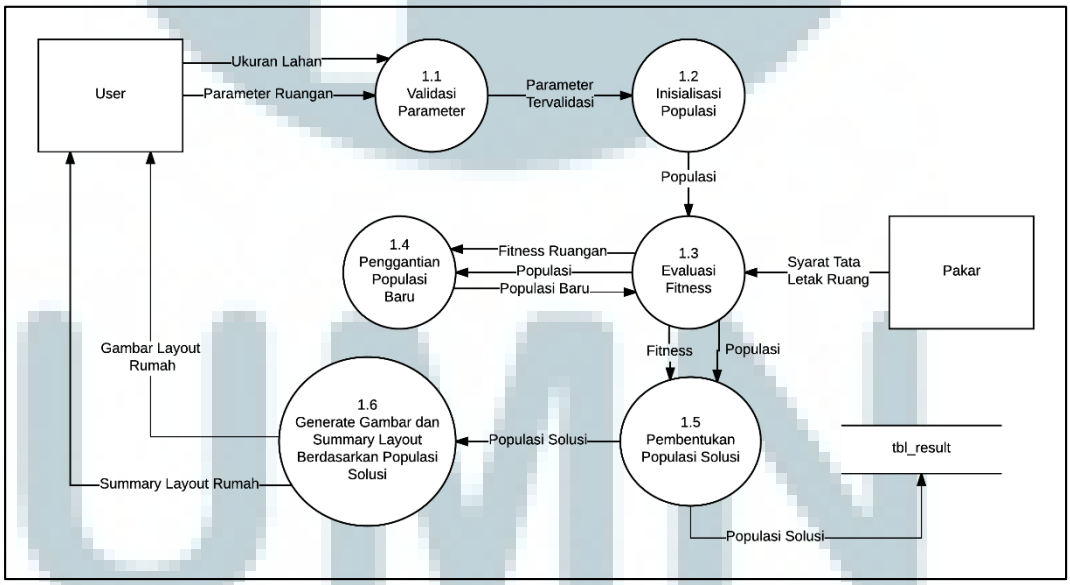


Gambar 3.2 Diagram *Level* 1

Pada proses pembentukan *layout* rumah, *user* memberikan *input* kepada sistem berupa ukuran lahan dan parameter ruangan. Parameter ruangan terdiri atas pilihan untuk menentukan ukuran ruangan secara acak dan ukuran tiap ruangan

yang diinginkan. Jika *user* memilih ukuran ruangan secara acak, aplikasi akan menentukan sendiri ukuran tiap ruangan secara acak, sedangkan jika tidak, ukuran ruangan yang digunakan adalah *input* yang diberikan oleh *user*. Sebelum memberikan *output* kepada *user*, sistem akan menyimpan hasil tata letak rumah (yang disimpan pada *array* populasi solusi) pada tabel tbl_result, agar semua populasi solusi yang telah dihasilkan dapat dilihat kembali. Setelah itu, sistem akan memberikan *output* berupa gambar *layout* rumah dan *summary* dari hasil *layout* rumah. *Summary layout* rumah berupa jumlah mutasi yang terjadi, *fitness* total, dan waktu eksekusi proses pembentukan *layout* rumah.

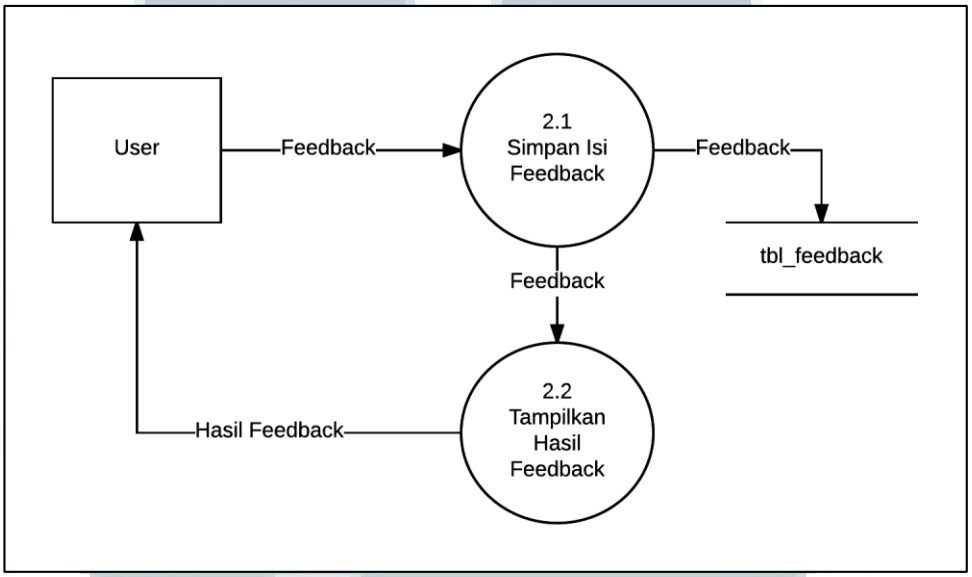
Isi dari populasi solusi adalah tata letak rumah yang sudah diproses, yang terdiri dari jumlah tiap ruangan, ukuran tiap ruangan (panjang dan lebar), dan posisi tiap ruangan (posisi x dan posisi y). Gambar 3.3 menunjukkan diagram *level* 2 untuk proses pembentukan *layout* rumah.



Gambar 3.3 Diagram *Level* 2 Pembentukan *Layout* Rumah

Proses *feedback* terjadi jika sistem sudah melakukan proses pembentukan *layout* rumah. Proses *feedback* akan menerima *input* dari *user* berupa *feedback* yang berisi nama depan pengguna dan jawaban dari setiap pertanyaan yang diberikan.

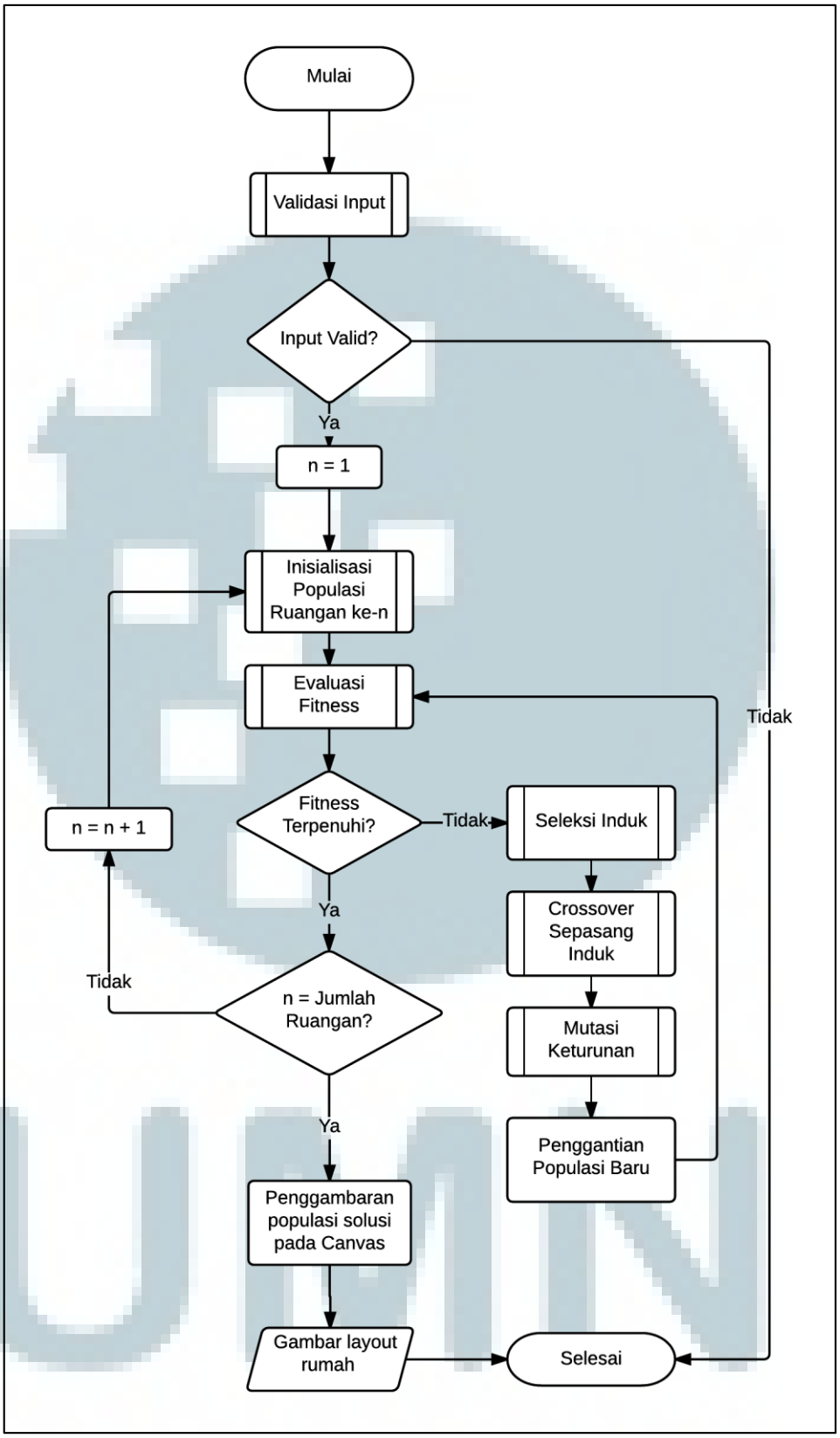
Aplikasi akan menyimpan *feedback* yang diberikan pada tabel tbl_feedback, setelah itu aplikasi akan menampilkan hasil *feedback* yang diberikan oleh *user*. Hasil *feedback* disimpan dalam tbl_feedback untuk penelitian mengenai tingkat kepuasan penggunaterhadap aplikasi. Gambar 3.4 menunjukkan diagram *level* 2 dari proses *feedback*.



Gambar 3.4 Diagram *Level* 2 Proses *Feedback*

3.2.2 Flowchart Program

Proses yang terjadi dalam sistem terdiri dari validasi *input*, inisialisasi populasi tiap ruangan, evaluasi *fitness* dari populasi, proses genetika, dan *generate layout* rumah (dari proses algoritma genetika). Gambar 3.5 menunjukkan *flowchart* aplikasi secara garis besar.

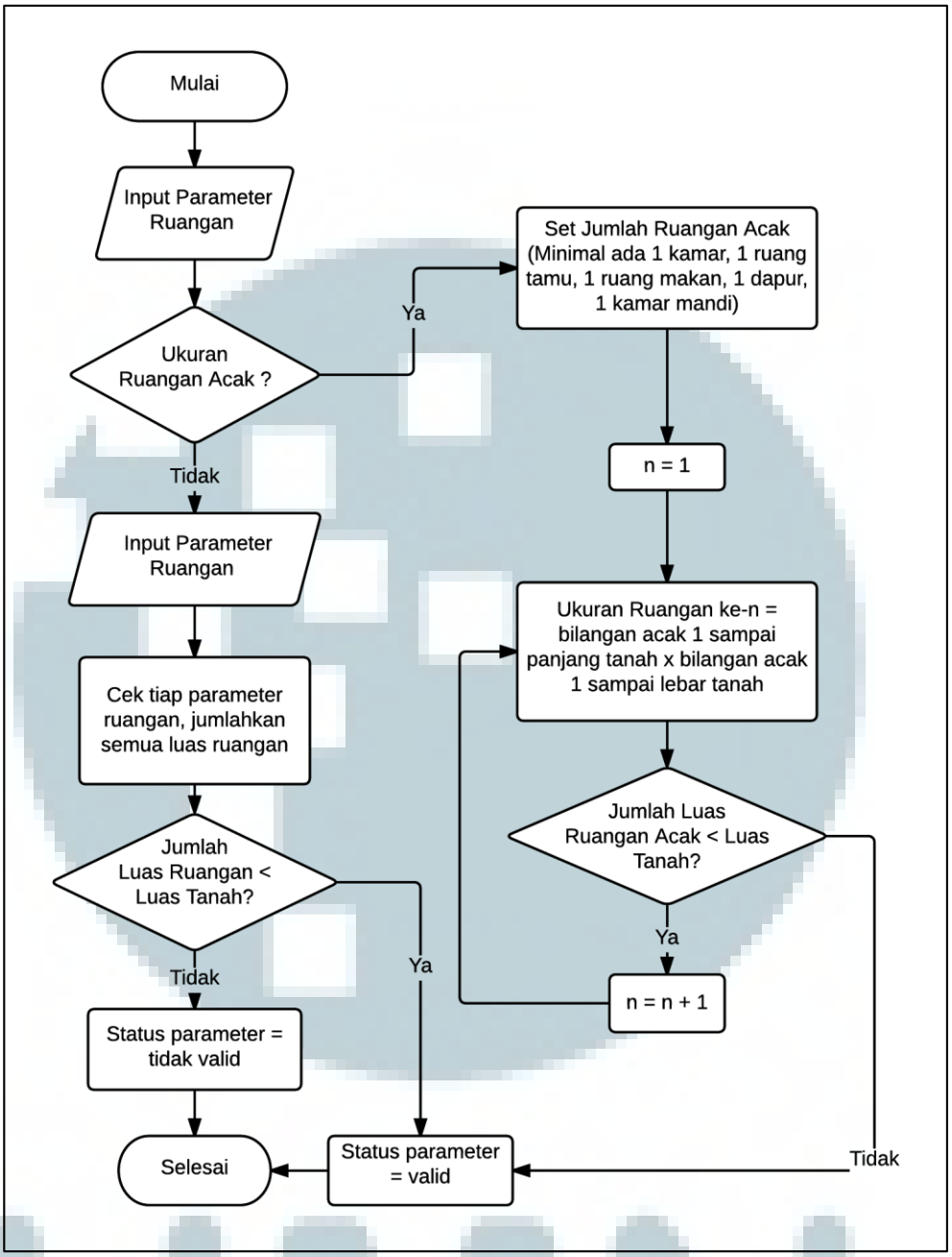


Gambar 3.5 *Flowchart* Aplikasi Secara Garis Besar

Aplikasi akan melakukan validasi terhadap *input*, dan melakukan inisialisasi populasi pertama jika *input* valid. Validasi dilakukan untuk menentukan ukuran ruangan yang akan di-*generate*, apakah ukuran tiap ruangan ditentukan oleh *user* atau tidak. Jika ukuran ruangan ditentukan *user*, proses validasi memastikan ukuran ruangan yang diberikan tidak melebihi ukuran lahan yang tersedia. Jika ukuran ruangan tidak ditentukan *user*, proses validasi akan menentukan ukuran tiap ruangan secara acak.

Aplikasi akan melakukan iterasi inisialisasi populasi dan evaluasi ruangan sesuai dengan jumlah ruangan yang diminta, apabila *fitness* dari populasi ruangan sebelumnya terpenuhi, sampai mendapatkan populasi solusi. Populasi solusi ini berisi ukuran dan letak ruangan dalam bentuk *array*, sehingga penggambaran tiap ruangan dapat direpresentasikan dengan meletakkan blok-blok persegi sesuai posisi ruangan (*x* dan *y*) pada *array* populasi solusi dan mengelompokkan blok-blok tersebut sesuai dengan ukuran ruangan. Populasi solusi ini kemudian akan direpresentasikan dalam bentuk gambar *floorplan* berdasarkan ukuran dan posisi yang sudah ditentukan.

Setelah menerima *input* dari pengguna, sistem akan melakukan validasi terhadap *input* sehingga semua parameter yang dibutuhkan dalam proses yang tersedia. Gambar 3.6 menunjukkan *flowchart* validasi *input* dari pengguna.

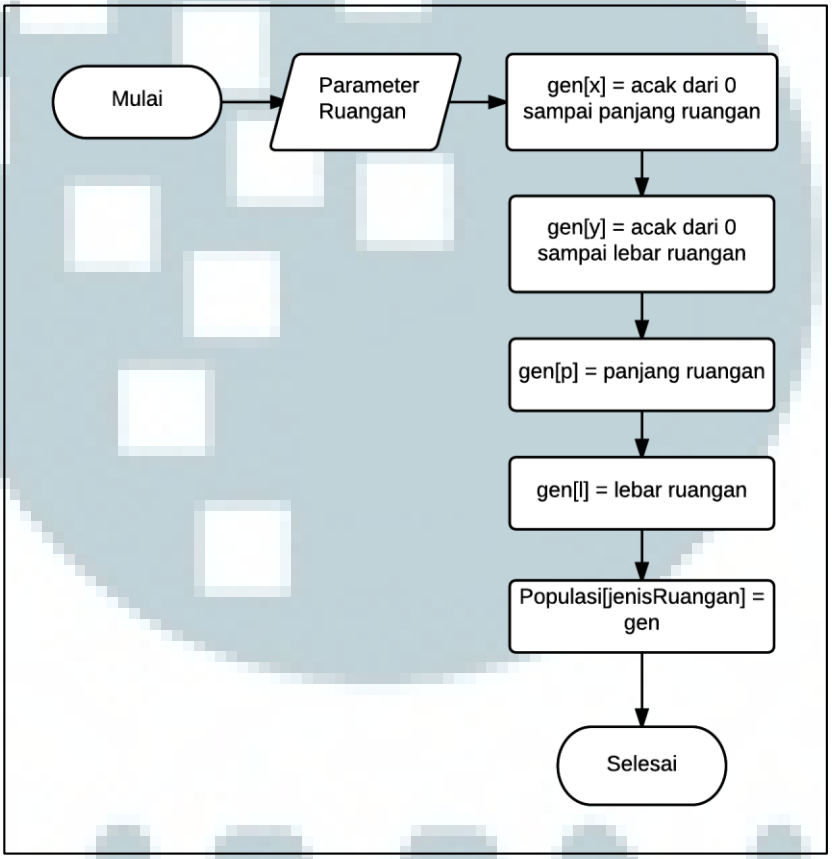


Gambar 3.6 *Flowchart* Validasi *Input* dari Pengguna

Pada validasi input, jika *user* tidak memasukkan parameter jumlah ruangan atau ukuran ruangan, jumlah atau ukuran tiap ruangan akan diacak oleh sistem. Jumlah ruangan minimal adalah satu kamar, satu ruang tamu, satu ruang makan, satu dapur, dan satu kamar mandi, seperti pada tabel 3.1. Ukuran ruangan acak adalah bilangan acak dari satu sampai panjang atau lebar tanah. Setelah semua parameter didapatkan, dilakukan pengecekan apakah jumlah luas seluruh ruangan

lebih kecil dari luas lahan. Jika ya, *input* dinyatakan valid. Jika tidak, *input* dinyatakan tidak valid. Proses ini dilakukan agar ukuran tiap ruangan tidak melebihi luas lahan yang tersedia.

Setelah *input* ter-validasi sebagai parameter ruangan, sistem akan melakukan inisialisasi populasi baru. Gambar 3.7 menunjukkan *flowchart* inisialisasi populasi baru untuk tiap jenis ruangan.

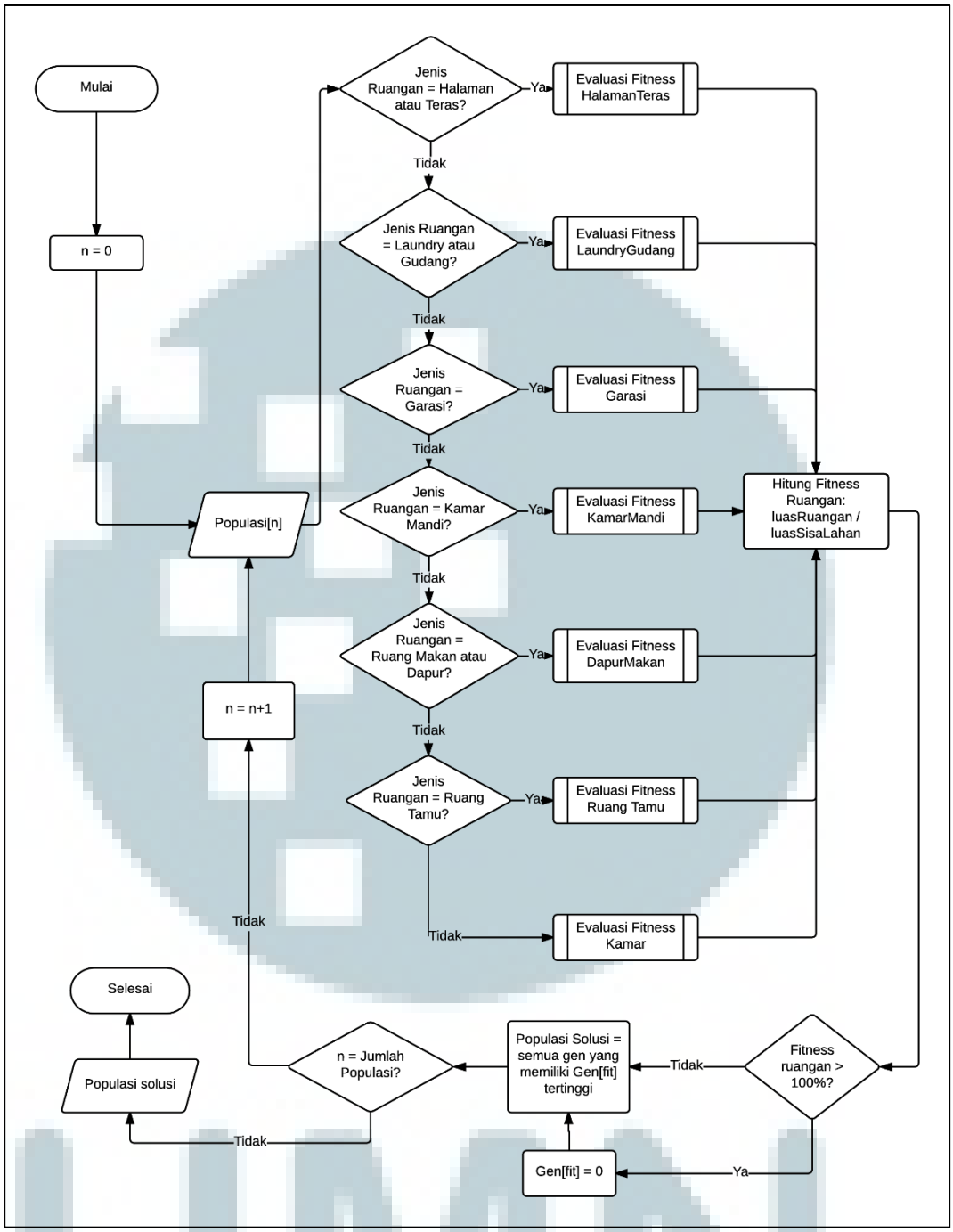


Gambar 3.7 *Flowchart* Inisialisasi Populasi Baru Ruangan Tertentu

Pada tahap inisialisasi populasi baru, parameter ruangan yang terdiri atas jumlah ruangan, jenis ruangan, panjang tiap ruangan, dan lebar tiap ruangan, merupakan sifat dari gen (ruangan) dalam populasi, dimana gen direpresentasikan dalam bentuk *array* untuk mempermudah pengolahan parameter tiap ruangan. Tahap inisialisasi populasi menghasilkan keluaran (*output*) berupa populasi yang merepresentasikan letak ruangan (koordinat *x* dan *y*), panjang ruangan, lebar

ruangan, dan jenis ruangan. Koordinat *x* dan *y* ini merupakan posisi tiap ruangan yang akan diletakkan pada gambar. Tiap gen koordinat dibentuk secara acak dan sebanyak tiga puluh gen dibentuk untuk memperbesar kemungkinan peletakan ruangan yang sesuai (tanpa tubrukan, *collision*, atau *overlap*) pada saat evaluasi *fitness*. Posisi ruangan dapat disesuaikan dengan *constraint* yang ada pada saat proses penggantian populasi baru.

Untuk setiap gen dalam populasi yang telah terbentuk akan melalui proses evaluasi *fitness*, sehingga dapat diketahui apakah posisi dan ukuran ruangan sesuai dengan syarat-syarat (*constraint*) tata letak ruangan seperti yang telah dijelaskan pada bab 2.2. Setiap jenis ruangan memiliki *constraint* yang berbeda, sehingga evaluasi *fitness* dilakukan secara berurutan berdasarkan letaknya dari depan rumah, dimulai dari evaluasi *fitness* halaman dan teras yang terletak di luar rumah. Proses evaluasi akan dilakukan pada semua gen dalam populasi, sehingga semua jenis ruangan dalam gen dapat ditentukan nilai *fitness*-nya. Nilai *fitness* ruangan menentukan apakah ruangan yang telah dievaluasi layak untuk menjadi gen solusi yang mewakili jenis ruangannya. Gambar 3.8 menunjukkan *flowchart* proses evaluasi *fitness* tiap ruangan.



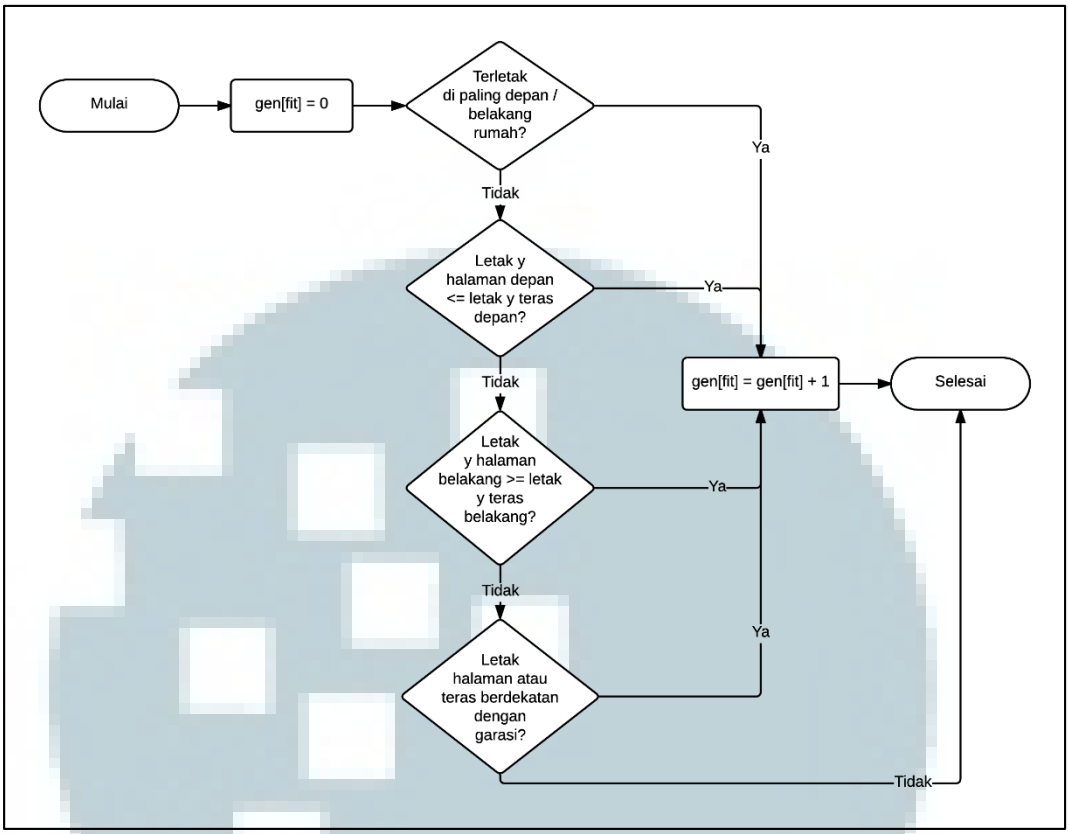
Gambar 3.8 *Flowchart* Evaluasi *Fitness*

Evaluasi *fitness* dilakukan pada tiap gen pada populasi berdasarkan jenis ruangan yang diminta. Evaluasi *fitness* tiap ruangan merupakan pengecekan gen pada populasi terhadap syarat-syarat (*constraint*) perancangan tata letak ruangan dalam rumah. Pengecekan *constraint* bertujuan untuk memberikan pertimbangan letak ruangan. Semakin besar poin evaluasi *fitness* jenis ruangan (nilai *fit* pada gen),

maka kemungkinan terpilihnya kromosom tersebut akan semakin besar. Jika *fitness* ruangan lebih dari 100%, ruangan tersebut tidak dapat diletakkan pada lahan karena luas ruangan tersebut jika ditambahkan dengan jumlah luas ruangan yang telah tersedia akan melebihi luas tanah yang tersedia dan menyebabkan kemungkinan *overlap* yang tinggi, sehingga nilai *fit* pada gen dikembalikan ke nilai nol untuk memperkecil kemungkinan ruangan tersebut tidak terpilih.

Dalam melakukan evaluasi untuk tiap jenis ruangan, program dapat mengetahui ukuran dan letak dari gen-gen unggul yang sudah diproses berdasarkan populasi solusi yang sudah ada, karena dalam gen sudah terdapat parameter *x* dan *y* sebagai indikator posisi ruangan. Jika pada populasi solusi belum ditemukan jenis ruangan yang sedang diproses, program akan memilih gen unggul (gen dengan *fitness* tertinggi) dalam populasi, kemudian dimasukkan dalam populasi solusi. Karena syarat-syarat peletakan tiap jenis ruangan berbeda, kondisi *fitness* tiap jenis ruangan menjadi berbeda, seperti yang telah dijelaskan pada bab 2.2.

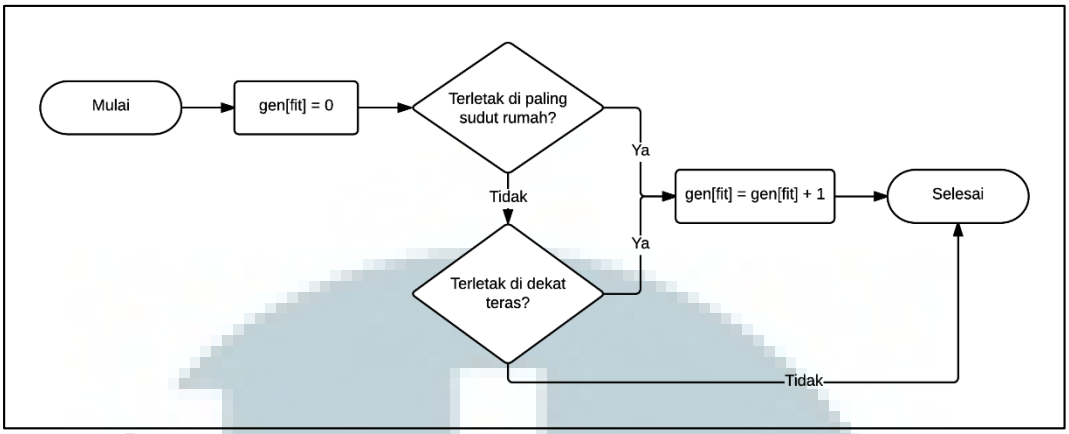
Syarat peletakan teras dan halaman dalam rumah, baik di depan rumah maupun di belakang rumah hampir sama, sehingga teras dan halaman dapat melalui tahap evaluasi *fitness* yang sama. Gambar 3.9 menunjukkan *flowchart* evaluasi *fitness* untuk jenis ruangan teras atau halaman.



Gambar 3.9 *Flowchart* Evaluasi *Fitness* untuk Teras atau Halaman

Evaluasi terhadap teras atau halaman dilakukan jika terdapat parameter teras atau halaman (ukuran ruangan tersedia), karena teras dan halaman merupakan jenis ruangan yang opsional. Pengecekan letak halaman atau teras terhadap ruangan lain dilakukan paling pertama, karena halaman atau teras harus berada di luar ruangan lain, sehingga posisi yang memungkinkan adalah di paling depan atau paling belakang rumah (y = 0 atau y = panjang tanah). Selain itu, halaman dapat dipisah dengan ruangan lain, namun teras harus berdekatan dengan ruangan bagian dalam rumah sehingga perlu dicek letak y dari halaman dan teras. Kondisi peletakan ruangan ini berpengaruh pada *fitness* ruangan yang dievaluasi.

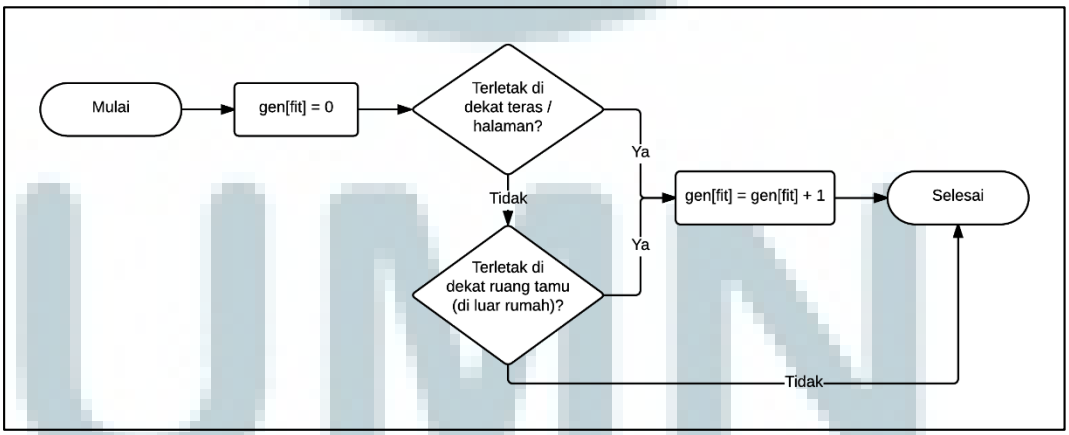
Gambar 3.10 menunjukkan *flowchart* evaluasi *fitness* untuk jenis ruangan gudang atau ruang *laundry*.



Gambar 3.10 *Flowchart* Evaluasi *Fitness* untuk Ruang *Laundry* atau Gudang

Evaluasi terhadap gudang atau ruang *laundry* dilakukan jika parameter ruangan tersedia, karena gudang dan ruang *laundry* bersifat opsional. Karena kedua jenis ruangan ini tidak terlalu berpengaruh terhadap kegiatan sehari-hari, sebaiknya diletakkan di sudut rumah (contohnya $x = 0$ dan $y = 0$). Ruang *laundry* juga sebaiknya diletakkan di dekat teras (jika tersedia pada populasi solusi), misalnya di sebelah kiri atau kanan teras, tergantung letak teras pada populasi solusi.

Gambar 3.11 menunjukkan *flowchart* evaluasi *fitness* untuk jenis ruangan garasi.

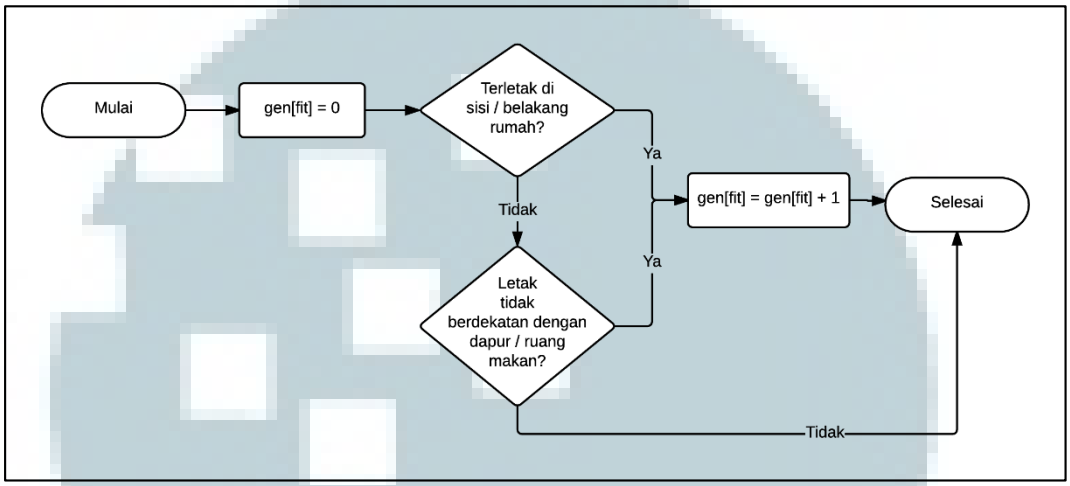


Gambar 3.11 *Flowchart* Evaluasi *Fitness* untuk Garasi

Evaluasi terhadap garasi dilakukan jika parameter ruangan jenis garasi tersedia. Garasi sebaiknya diletakkan di dekat teras, halaman, atau ruang tamu, untuk mempermudah keluar masuknya penghuni rumah. Pengecekan dilakukan

dengan membandingkan letak teras dan halaman terhadap garasi, sehingga dapat diletakkan bersebelahan.

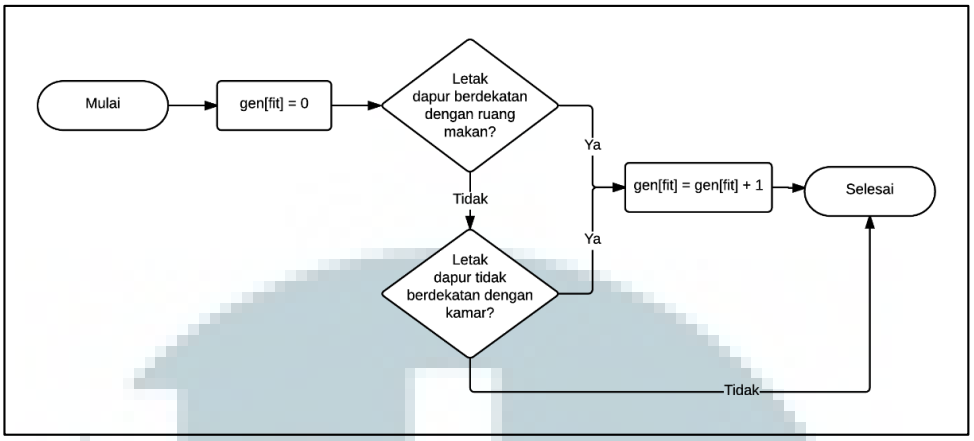
Gambar 3.12 menunjukkan *flowchart* evaluasi *fitness* untuk jenis ruangan kamar mandi.



Gambar 3.12 *Flowchart* Evaluasi *Fitness* untuk Kamar Mandi

Evaluasi terhadap jenis ruangan kamar mandi yaitu mengecek apakah letak *x* dan *y* yang dievaluasi berada pada sisi atau belakang rumah. Selain itu, perlu dibandingkan dengan letak dapur atau ruang makan pada populasi solusi (jika tersedia).

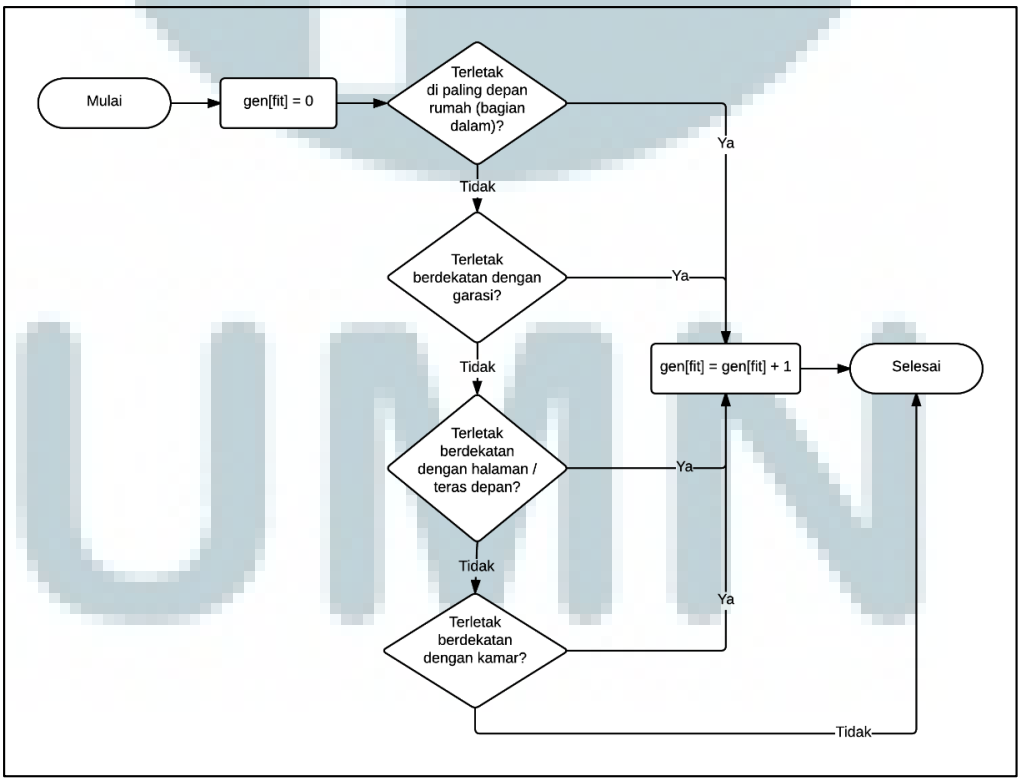
Syarat peletakan dapur dan ruang makan dalam rumah hampir sama, sehingga dapur dan ruang makan dapat melalui tahap evaluasi *fitness* yang sama. Gambar 3.13 menunjukkan *flowchart* evaluasi *fitness* untuk jenis ruangan dapur atau ruang makan.



Gambar 3.13 *Flowchart* Evaluasi *Fitness* untuk Dapur atau Ruang Makan

Evaluasi terhadap dapur atau ruang makan yaitu mengecek apakah letak *x* dan *y* antara dapur dan ruang makan berdekatan (selama tidak terjadi tubrukan). Jika pada populasi solusi sudah tersedia kamar, perlu dibandingkan letaknya terhadap kamar sehingga tidak berdekatan.

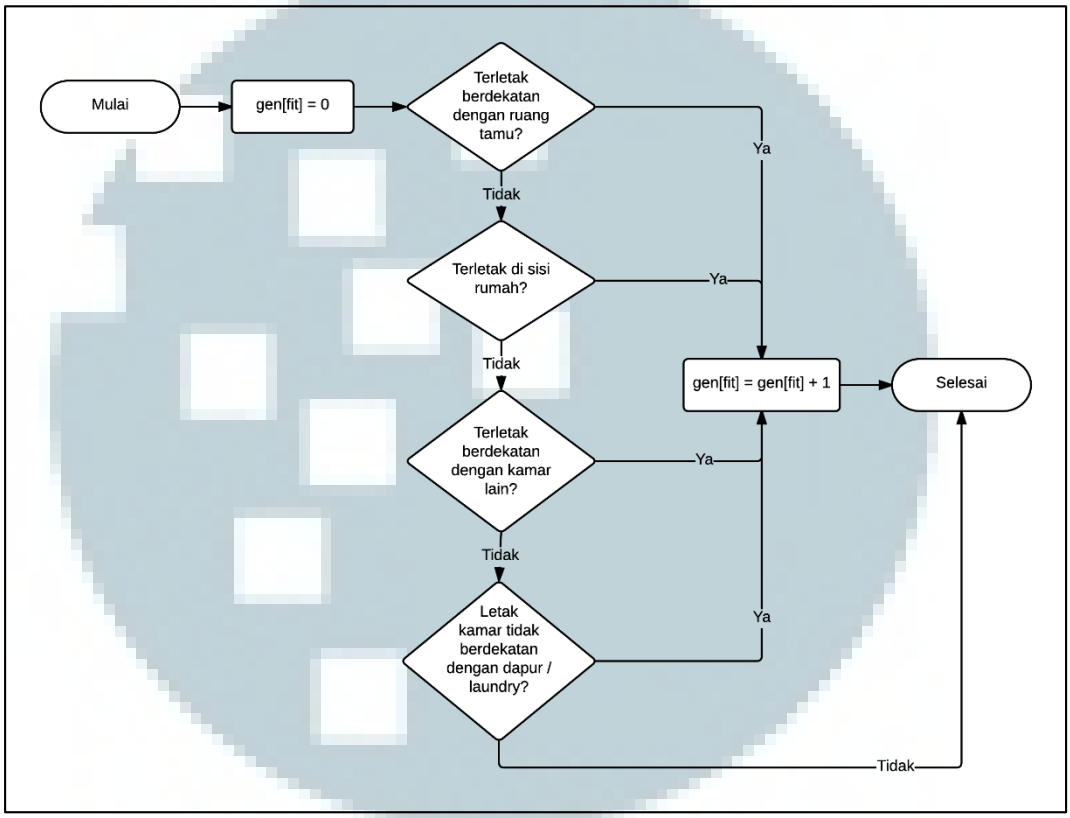
Gambar 3.14 menunjukkan *flowchart* evaluasi *fitness* untuk jenis ruangan ruang tamu.



Gambar 3.14 *Flowchart* Evaluasi *Fitness* untuk Ruang Tamu

Evaluasi terhadap ruang tamu yaitu mengecek apakah letak ruang tamu berdekatan dengan garasi, teras depan, dan kamar.

Gambar 3.15 menunjukkan *flowchart* evaluasi *fitness* untuk jenis ruangan kamar.

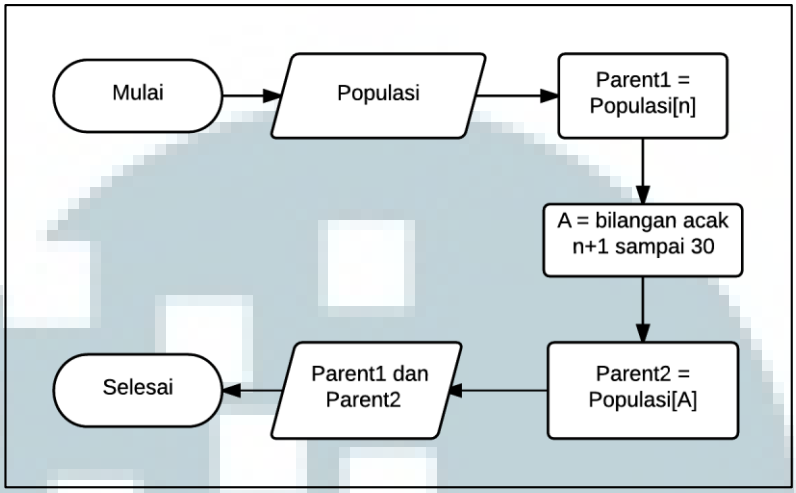


Gambar 3.15 *Flowchart* Evaluasi *Fitness* untuk Kamar

Evaluasi terhadap kamar yaitu mengecek apakah letak kamar berada di sisi rumah bagian dalam, dan tiap kamar saling berdekatan. Hal ini dapat dibandingkan dengan letak dan ukuran ruangan lain yang sudah ada dalam populasi solusi.

Setelah dilakukan pencocokan *constraint*, *fitness* ruangan akan dihitung berdasarkan rumus 2.1. Nilai *fitness* ini bertujuan untuk mencegah tata letak dan total luas ruangan tidak melebihi luas tanah yang tersedia. Apabila *fitness* ruangan tidak terpenuhi, maka perlu dibentuk populasi baru untuk generasi selanjutnya. Dalam pembentukan generasi baru, terdapat beberapa proses, yaitu seleksi induk,

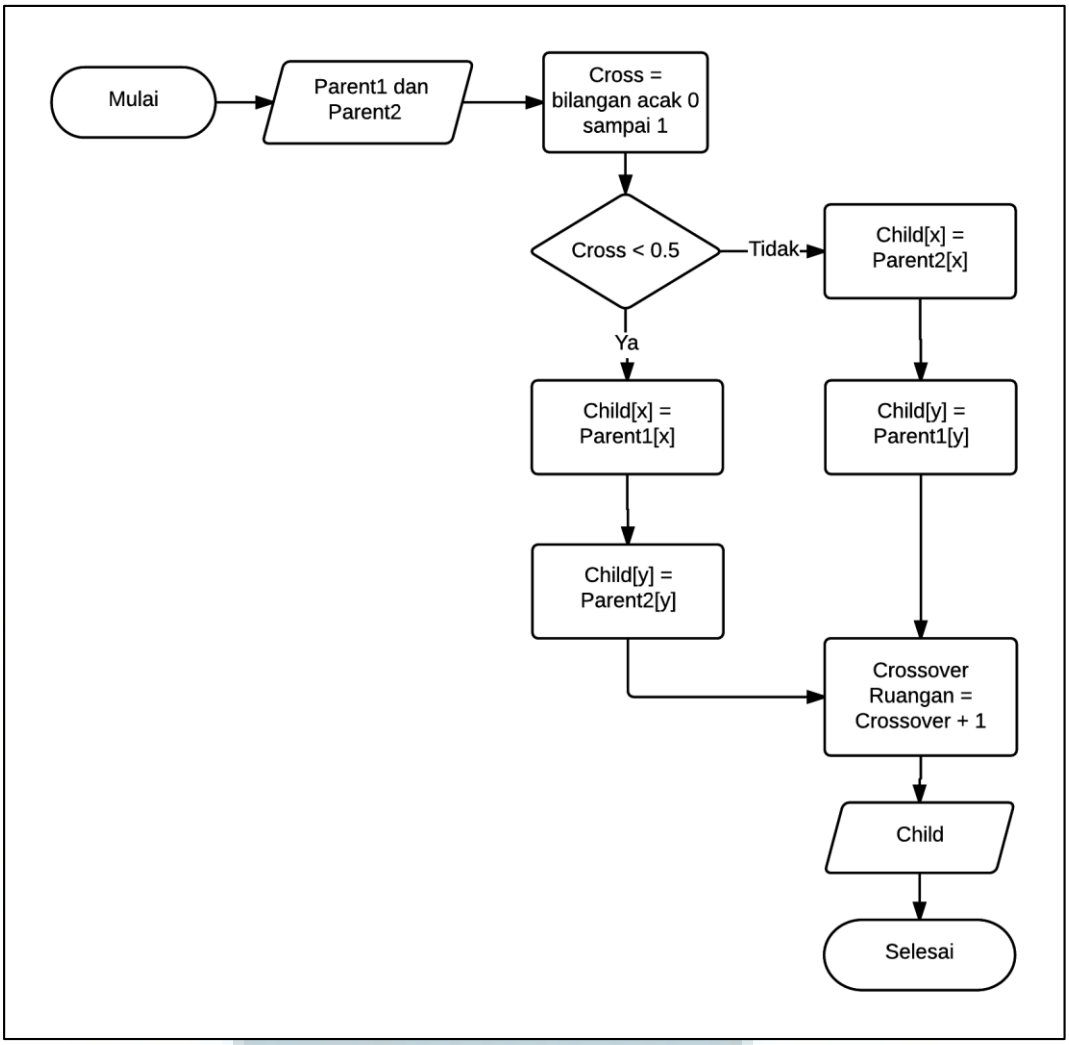
crossover sepasang induk, mutasi keturunan, dan penggantian populasi baru. Gambar 3.16 menunjukkan *flowchart* untuk seleksi induk.



Gambar 3.16 *Flowchart* Seleksi Induk

Jika nilai *fitness* (rata-rata kecocokan populasi dengan *constraint*) masih minimum, kecil, atau tidak mencapai setengahnya, berarti populasi yang didapat belum optimal, sehingga perlu dilakukan seleksi induk untuk menciptakan populasi baru dari populasi yang sudah ada. Dari populasi yang ada, akan diambil gen pertama dan satu gen lainnya secara acak sebagai sepasang induk. Kedua gen yang terpilih akan disilangkan (*crossover*).

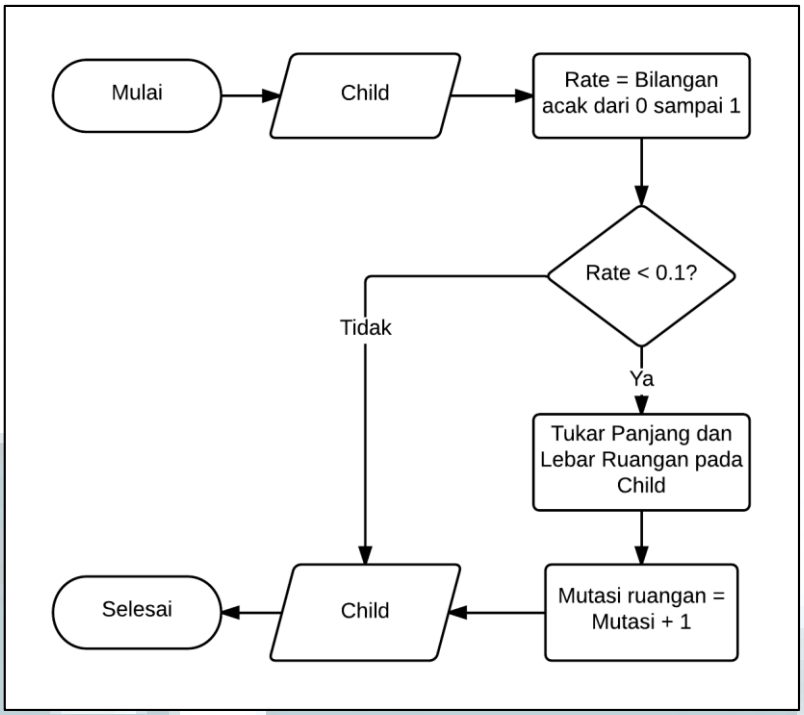
Gambar 3.17 menunjukkan *flowchart crossover* sepasang induk yang telah terpilih.



Gambar 3.17 *Flowchart Crossover* Sepasang Induk

Crossover dilakukan agar gen atau individu unggul pada populasi tidak hilang begitu saja, sehingga *crossover* terjadi dengan kemungkinan 0,5 dari bilangan acak. Jika bilangan acak kurang dari 0,5, maka populasi baru akan menggunakan posisi *x* dari induk pertama dan posisi *y* dari induk kedua, dan sebaliknya. Jumlah *crossover* yang dialami oleh populasi akan disimpan sebagai parameter genetika untuk membandingkan hasil tiap *layout* rumah yang dibuat.

Gambar 3.18 menunjukkan *flowchart* mutasi keturunan yang didapat dari proses *crossover*.



Gambar 3.18 *Flowchart* Mutasi Keturunan

Mutasi dilakukan pada gen panjang dan lebar ruangan dari keturunan. Tujuan dilakukannya mutasi yaitu memperoleh populasi keturunan yang lebih baik dari keturunan hasil *crossover*, sehingga kemungkinan terjadinya tubrukan antar ruangan semakin kecil. Mutasi terjadi apabila *rate* (bilangan acak dari nol sampai satu) kurang dari 0,1. Hal ini dilakukan untuk mengurangi kemungkinan mutasi terjadi. Jumlah mutasi yang dialami oleh populasi akan disimpan sebagai parameter genetika untuk membandingkan hasil tiap *layout* rumah yang dibuat.

3.2.3 Struktur Tabel

Program yang dibuat menggunakan dua tabel untuk menyimpan hasil *layout* rumah dan hasil *feedback* yang diberikan. Struktur tabel untuk menyimpan hasil *layout* rumah (tbl_result) ditunjukkan pada tabel 3.2, sedangkan struktur tabel untuk menyimpan hasil *feedback* (tbl_feedback) ditunjukkan pada tabel 3.3.

Nama tabel : tbl_result

Fungsi : menyimpan hasil rancangan tata letak rumah

Primary key : id

Foreign key : -

Tabel 3.2 Struktur Tabel tbl_result

Nama *Field*	Tipe Data	*Constraint*	Keterangan
id	int(11)	*Primary Key* *Not Null* *Auto Increment*	ID dari setiap hasil *layout* rumah
panjang_lahan	int(11)	*Not Null*	Panjang dari lahan yang tersedia
lebar_lahan	int(11)	*Not Null*	Lebar dari lahan yang tersedia
parameter_ruangan	text	*Not Null*	Semua parameter ruangan yang diproses (jumlah tiap jenis ruangan, ukuran ruangan, posisi ruangan, dan *fitness* tiap ruangan), disimpan dalam format JSON
fitness_lahan	int(11)	*Not Nul*	*Fitness* lahan yang sudah diproses
mutation_rate	int(11)	*Not Null*	Kemungkinan mutasi yang terjadi dalam proses
result_timestamp	timestamp	*Not Null* *Default Now*	*Timestamp* saat proses dijalankan

Nama tabel : tbl_feedback

Fungsi : menyimpan hasil *feedback* yang telah diberikan oleh pengguna

Primary key : id

Foreign key : -

Tabel 3.3 Struktur Tabel tbl_feedback

Nama *Field*	Tipe Data	*Constraint*	Keterangan
id	int(11)	*Primary Key* *Not Null* *Auto Increment*	ID dari setiap hasil *feedback*
nama_depan	varchar(20)	*Not Null*	Nama depan dari *user* yang memberikan *feedback*
feedback_timestamp	int(11)	*Not Null* *Default Now*	*Timestamp* saat proses dijalankan
hasil_feedback	text	*Not Null*	Semua jawaban dari pertanyaan yang diberikan, disimpan dalam format JSON
saran	text		Saran dari *user*

3.3 Perancangan Antarmuka Aplikasi

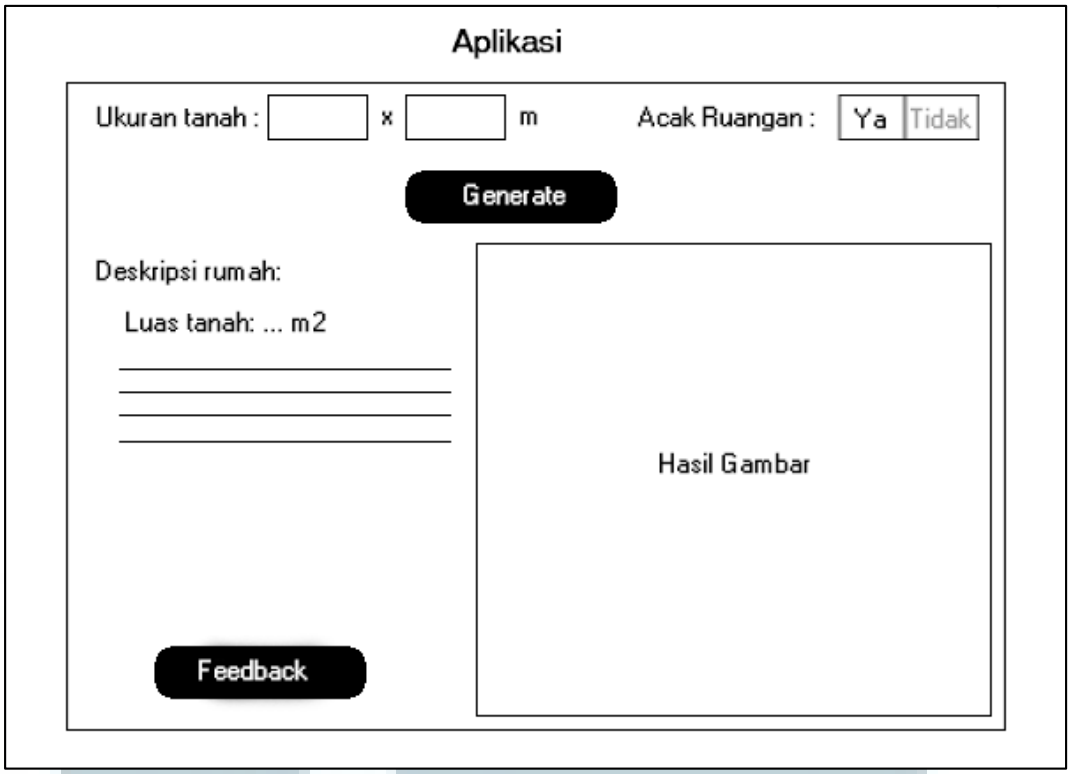
Aplikasi dibuat berbasis *website*, sehingga antarmuka aplikasi dipresentasikan dalam halaman *web* (*webpage*). Gambar 3.19 menunjukkan rancangan halaman awal dari aplikasi.

Gambar 3.19 Rancangan Antarmuka Halaman Awal Aplikasi

Dalam rancangan antarmuka aplikasi, terdapat menu ("Home", "Apps", dan "Help"), sebagai navigasi sehingga pengguna dapat menuju ke segmen yang diinginkan. Segmen "Home" berisi penjelasan singkat dan tujuan aplikasi dibuat, disertai dengan *button* "Start Apps" untuk memulai aplikasi. Segmen "Apps" berisi *form* untuk men-*generate layout* rumah. Segmen "Help" berisi manual penggunaan dan metode aplikasi.

Dalam aplikasi ini, segmen paling penting adalah segmen "Apps", karena pada bagian inilah *layout* rumah direpresentasikan. Terdapat *form* ukuran tanah untuk menentukan lahan yang tersedia, *toogle* acak ruangan untuk menentukan apakah pengguna akan memasukkan parameter ruangan atau tidak, serta *button* "Generate" untuk melakukan *generate layout* rumah berdasarkan isi dari *form*.

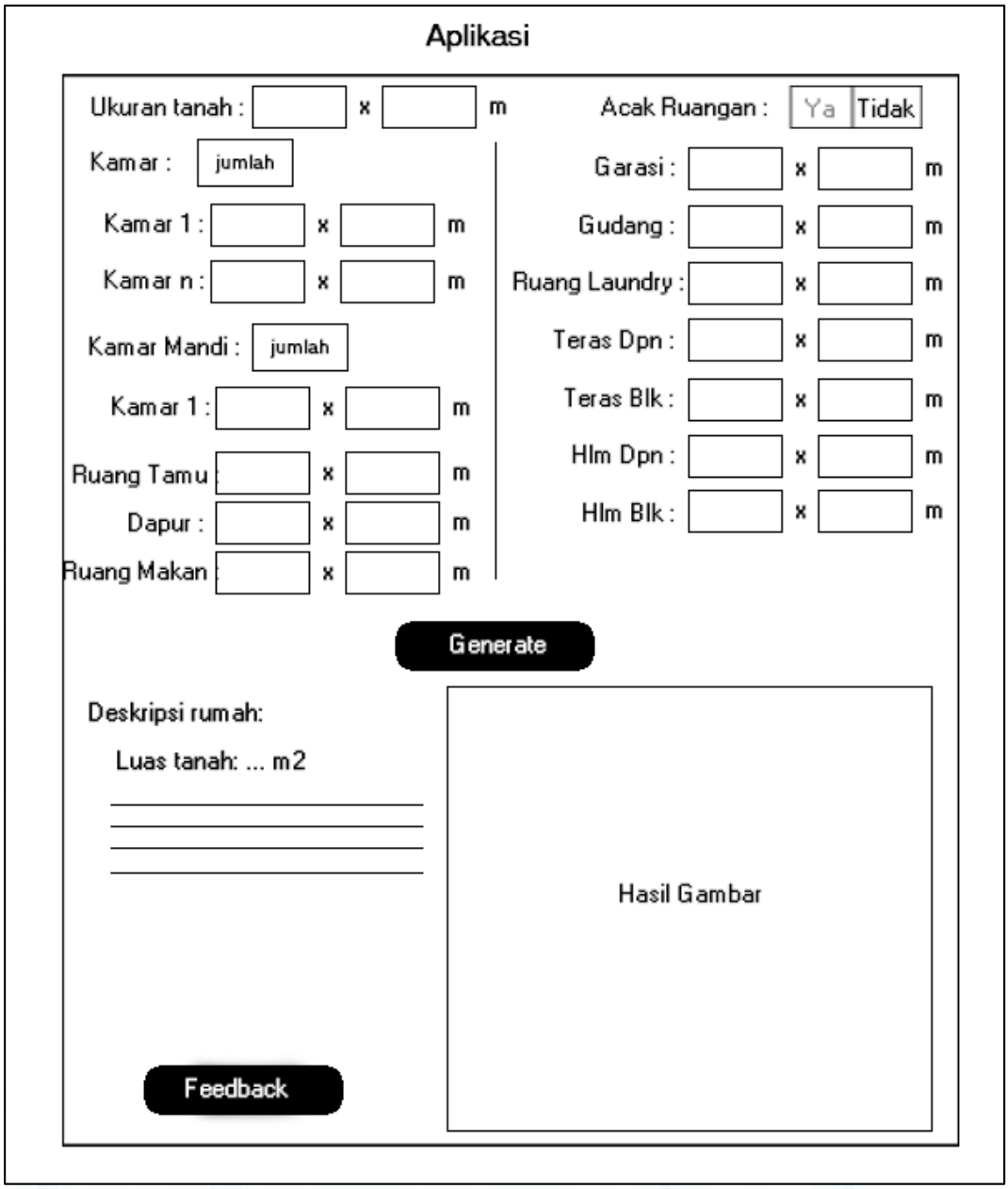
Rancangan antarmuka untuk aplikasi ini secara garis besar ada dua, yaitu antarmuka aplikasi yang ukuran dan jumlah ruangannya acak dan yang ditentukan. Gambar 3.20 menunjukkan rancangan antarmuka aplikasi jika ukuran dan jumlah ruangan tidak ditentukan (acak).



Gambar 3.20 Rancangan Antarmuka Aplikasi dengan Ukuran Ruangan Acak

Jika parameter ruangan tidak ditentukan, program akan men-*generate layout* rumah secara acak, disertai dengan deskripsi rumah (seperti luas tanah, dan sebagainya). Selain itu, terdapat *button* "Feedback" agar pengguna dapat memberikan pendapatnya mengenai kualitas aplikasi ini.

Gambar 3.21 menunjukkan rancangan antarmuka aplikasi jika jumlah dan ukuran ruangan ditentukan oleh pengguna.

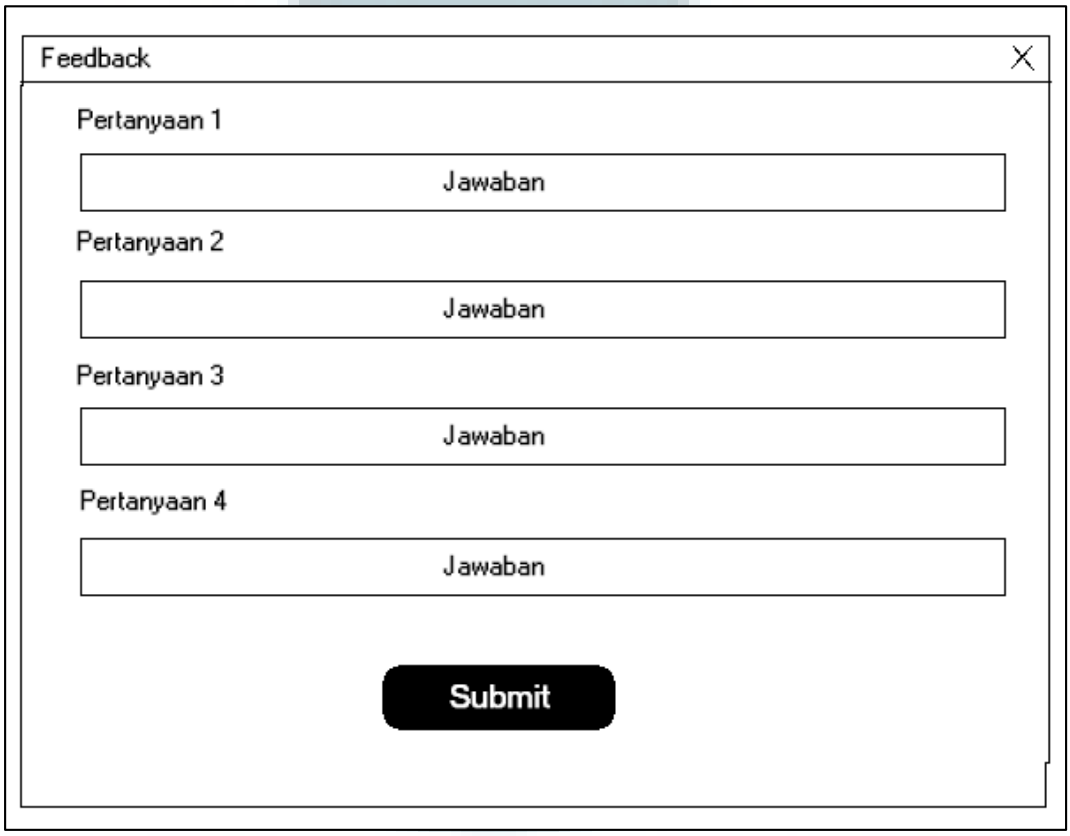


Gambar 3.21 Rancangan Antarmuka Aplikasi dengan Ukuran Ruangan yang Ditentukan oleh Pengguna

Hasil yang dikeluarkan aplikasi adalah gambar *layout* rumah dan deskripsi rumah, namun pengguna harus mengisi terlebih dahulu beberapa parameter ruangan yang diinginkan. *Input* yang perlu diberikan penguna adalah ukuran dari tiap ruangan, dimana kolom sebelah kiri merupakan ruangan yang wajib ada (harus diisi), sedangkan kolom sebelah kanan merupakan ruangan opsional. Jika *input* dikosongkan, misalnya pada kolom sebelah kanan, berarti parameter ruangan

dinyatakan tidak tersedia, sehingga jenis ruangan tersebut tidak disertakan dalam proses.

Gambar 3.22 menunjukkan rancangan antarmuka *feedback* aplikasi berupa kotak dialog (*pop-up*) yang berisi pertanyaan mengenai kualitas aplikasi.



Gambar 3.22 Rancangan Antarmuka Aplikasi untuk Kotak Dialog *Feedback*

Jika pengguna menekan *button* "Feedback" pada aplikasi, maka akan ditampilkan kotak dialog *feedback*, yang berisi pertanyaan mengenai kualitas aplikasi, dan pengguna diharapkan untuk men-*submit* jawaban atas pertanyaan yang tersedia.