BAB III

METODOLOGI DAN PERANCANGAN PROGRAM

3.1 Metodologi Penelitian

Penelitian akan dilakukan dengan metodologi sebagai berikut.

1. Studi Literatur

Studi literatur dilakukan dengan tujuan mendapatkan pengetahuan yang dibutuhkan untuk melakukan penelitian. Studi literatur dilakukan dengan membaca jurnal, artikel, dan penelitian-penelitian sebelumnya mengenai implementasi algoritma C4.5 untuk melakukan prediksi dan rancang bangun sistem pakar pendeteksi gangguan kesehatan.

2. Wawancara

Tahap ini dilakukan dengan mewawancarai dan berkonsultasi dengan pakar yang dipilih, yaitu Angeline Kartika Sosrodjojo, M.Psi., Psikolog. Tujuan dilakukannya wawancara yaitu memperoleh pengetahuan mengenai *Attention Deficit Hyperactivity Disorder*, serta gejala masalah konsentrasi dan hiperaktivitas yang dibutuhkan dalam penelitian.

3. Perancangan pohon keputusan

Pohon keputusan dibangun berdasarkan data *training* yang ada. Data *training* yang bertipe kontinu akan dimanipulasi dengan cara diskretisasi. Algoritma C4.5 akan diimplementasikan dalam membangun pohon keputusan.

4. Perancangan sistem

Perancangan sistem akan dilakukan dengan menyusun *flowchart* untuk menunjukkan alur aplikasi. Perancangan *database* dibuat dengan menggambarkan

ERD dan *database schema*. Desain antarmuka juga dibuat untuk gambaran visualisasi antarmuka aplikasi.

5. Implementasi

Implementasi akan dilakukan dengan merealisasikan perancangan ke dalam dua buah aplikasi, yaitu aplikasi *desktop* dan aplikasi *web*. Aplikasi *desktop* akan dibangun menggunakan bahasa pemrograman Java berupa aplikasi *command line* Windows. Aplikasi *command line* Windows berfungsi untuk membangun pohon keputusan. Sedangkan, aplikasi *web* dibangun menggunakan HTML, CSS, JavaScript, PHP, serta menggunakan *framework* CodeIgniter (CI). Aplikasi *web* berfungsi sebagai sistem pakar untuk melakukan prediksi masalah konsentrasi dan hiperaktivitas berdasarkan *input* dari pengguna. *Database* akan dibangun menggunakan MySQL.

6. Pengujian

Penggunaan algoritma C4.5 dalam membuat pohon keputusan akan diuji dengan membandingkan hasil pohon keputusan yang dibuat oleh program dengan hasil pohon keputusan yang dihasilkan dengan perhitungan manual, berdasarkan langkah-langkah membangun pohon keputusan menggunakan algoritma C4.5. Sistem yang telah dibangun akan diuji tingkat akurasinya dengan membandingkan diagnosa yang dihasilkan oleh sistem dengan hasil diagnosa oleh pakar sebanyak 40 *data testing*.

3.2 Perancangan Pohon Keputusan

Sistem pakar memanfaatkan pohon keputusan yang dibuat dengan menggunakan data *training* sebanyak 120 data. Data *training* adalah data hasil

wawancara dengan pengasuh atau orangtua dari anak berusia lima sampai sebelas tahun. Pertanyaan yang diajukan untuk wawancara disusun berdasarkan Child Behaviour Checklist (CBCL) untuk mendeteksi masalah konsentrasi dan Conners' Parent Rating Scale (CPRS) untuk mendeteksi masalah hiperaktif-impulsif. Selain menggunakan kuesioner CBCL dan CPRS, data usia dan jenis kelamin dari anak juga ditanyakan pada saat wawancara. Pengasuh atau orangtua dari anak akan mendeskripsikan perilaku anak pada saat wawancara dilakukan.

Tabel 3. 1 Child Behaviour Checklist (CBCL) untuk Mendeteksi Masalah Konsentrasi

0	1	2	No.	Pertanyaan
			1.	Bertingkah laku kekanak-kanakkan untuk usianya
			8.	Sulit berkonsentrasi atau memusatkan perhatian untuk jangka waktu yang lama
			10.	Tidak dapat duduk tenang, tidak bisa diam, hiperaktif
			13.	Terlihat bingung
			17.	Melamun atau banyak pikiran
			41.	Impulsif atau bertingkah laku tanpa berpikir terlebih dahulu
			45.	Cemas, tegang
			46.	Menampilkan tingkah laku cemas atau gelisah
			61.	Pekerjaan sekolah tergolong buruk
			62.	Koordinasi gerak yang buruk atau ceroboh
			80.	Pandangan mata kosong

Tabel 3.1 menunjukkan kuesioner Child Behaviour Checklist (CBCL) untuk mendeteksi masalah konsentrasi. Pertanyaan yang digunakan adalah pertanyaan ke-1, 8, 10, 13, 17, 41, 45, 46, 61, 62, dan 80 dari kuesioner CBCL.

Tabel 3.2 menunjukkan kuesioner Conners' Parent Rating Scale (CPRS) untuk mendeteksi masalah hiperaktif-impulsif. Pertanyaan yang digunakan adalah pertanyaan ke-4, 7, 11, 13, 14, 25, 31, 33, 37, dan 38 pada kuesioner CPRS.

Tabel 3.2 Conners' Parent Rating Scale (CPRS) untuk Mendeteksi Masalah Hiperaktif-impulsif

No.	Pertanyaan	Tidak sama sekali	Hanya sedikit	Seringkali	Sangat sering
4.	Bersemangat, impulsif.				
7.	Mudah atau seringkali menangis.				
11.	Gelisah seperti 'cacing' kepanasan.				
13.	Tidak bisa diam, selalu bersemangat dan bergerak.				
14.	Menghancurkan barang.				
25.	Gagal dalam menyelesaikan tugas.				
31.	Mudah terganggu dan sulit memusatkan perhatian.				
33.	Suasana hati mudah berubah dengan cepat dan drastis.				
37.	Mudah merasa frustasi dalam berusaha.				
38.	Mengganggu anak-anak lain.				

Setelah hasil wawancara didapatkan, pakar akan melakukan diagnosis untuk menentukan kategori masalah yang dialami. Terdapat empat kategori yang dapat didiagnosis, yaitu cenderung mengalami masalah konsentrasi, cenderung mengalami masalah hiperaktif-impulsif, cenderung mengalami masalah konsentrasi dan hiperaktif-impulsif, dan tidak terdeteksi adanya masalah.

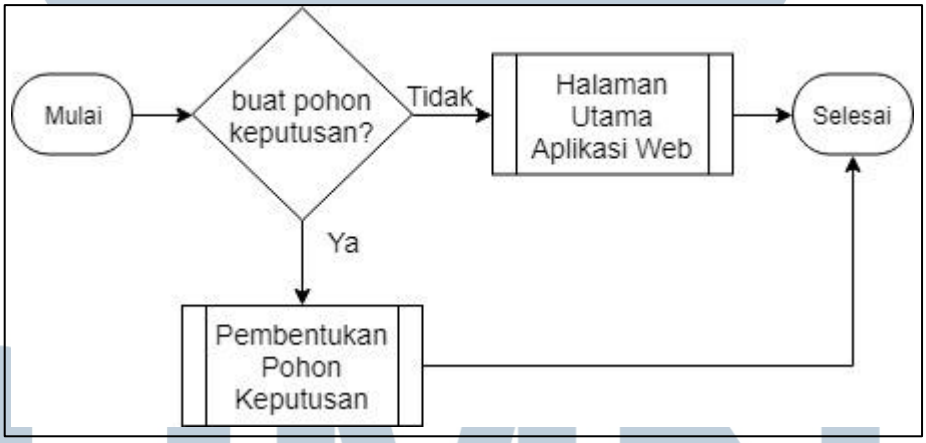
Pada tahap pembentukan pohon keputusan, manipulasi data dilakukan terhadap data *training*. Manipulasi data dilakukan dengan mengkonversi data yang bersifat kontinu menjadi sejumlah kategori dengan melakukan diskretisasi. Diskretisasi berfungsi untuk menentukan *split criteria* bagi data numerik atau kontinu. Data yang bersifat numerik pada data *training* adalah data usia.

3.3 Perancangan Sistem

Perancangan sistem akan dilakukan dengan menyusun *flowchart* aplikasi, *Entity Relationship Diagram*, *Database Schema*, struktur tabel, dan membuat desain antarmuka aplikasi.

3.3.1 Flowchart Sistem

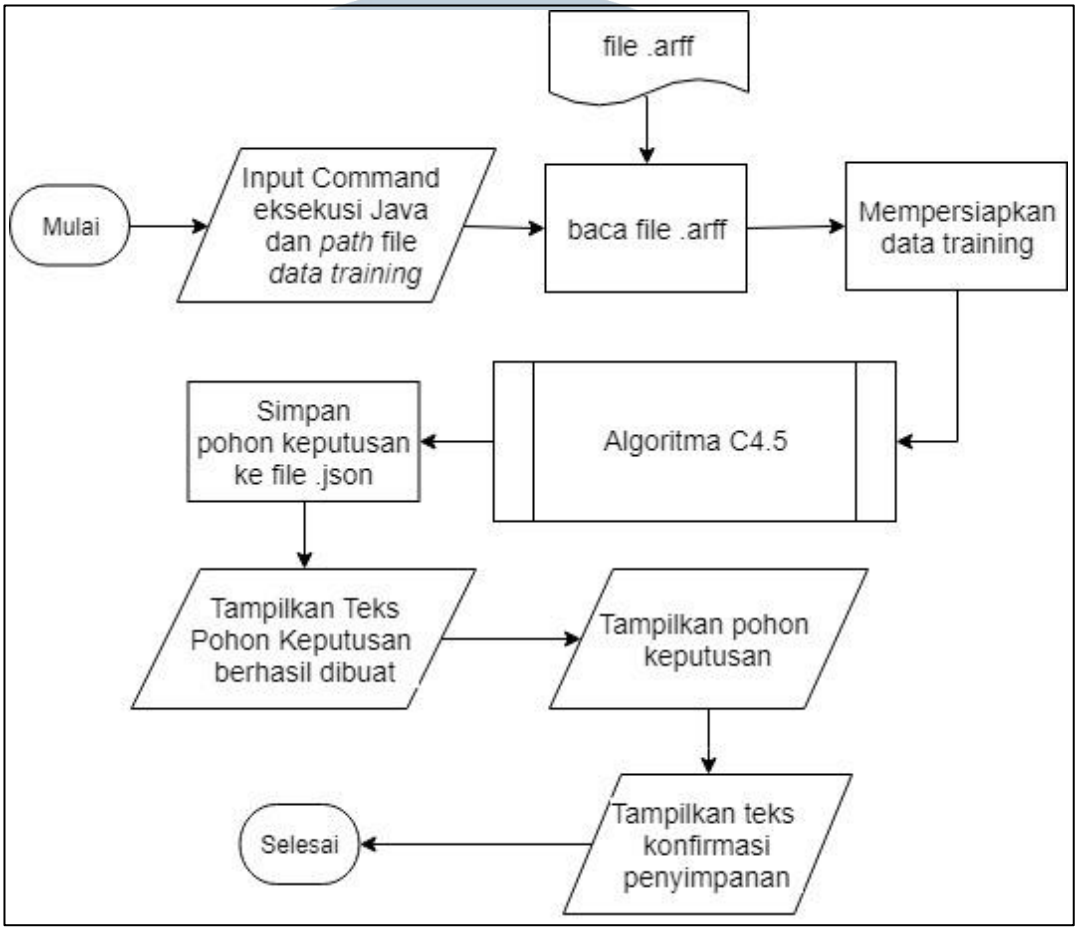
Terdapat dua aplikasi utama dalam penelitian ini, yaitu aplikasi yang berjalan pada *command line interface* pada sistem operasi Windows untuk membangun pohon keputusan, dan aplikasi *web* untuk mengimplementasikan pohon keputusan yang telah dibuat. Gambar 3.1 menunjukkan alur sistem aplikasi.



Gambar 3.1 Flowchart Sistem

Pengguna dapat memilih untuk membuat pohon keputusan atau melihat aplikasi *web*. Apabila pengguna memilih untuk membuat pohon keputusan, maka program akan menjalankan aplikasi *command line Windows*. Jika pengguna memilih untuk melihat aplikasi *web*, maka pengguna harus membuka *browser* dan membuka halaman utama aplikasi web.

3.3.2 Flowchart Aplikasi Desktop



Gambar 3.2 Flowchart Pembentukan Pohon Keputusan

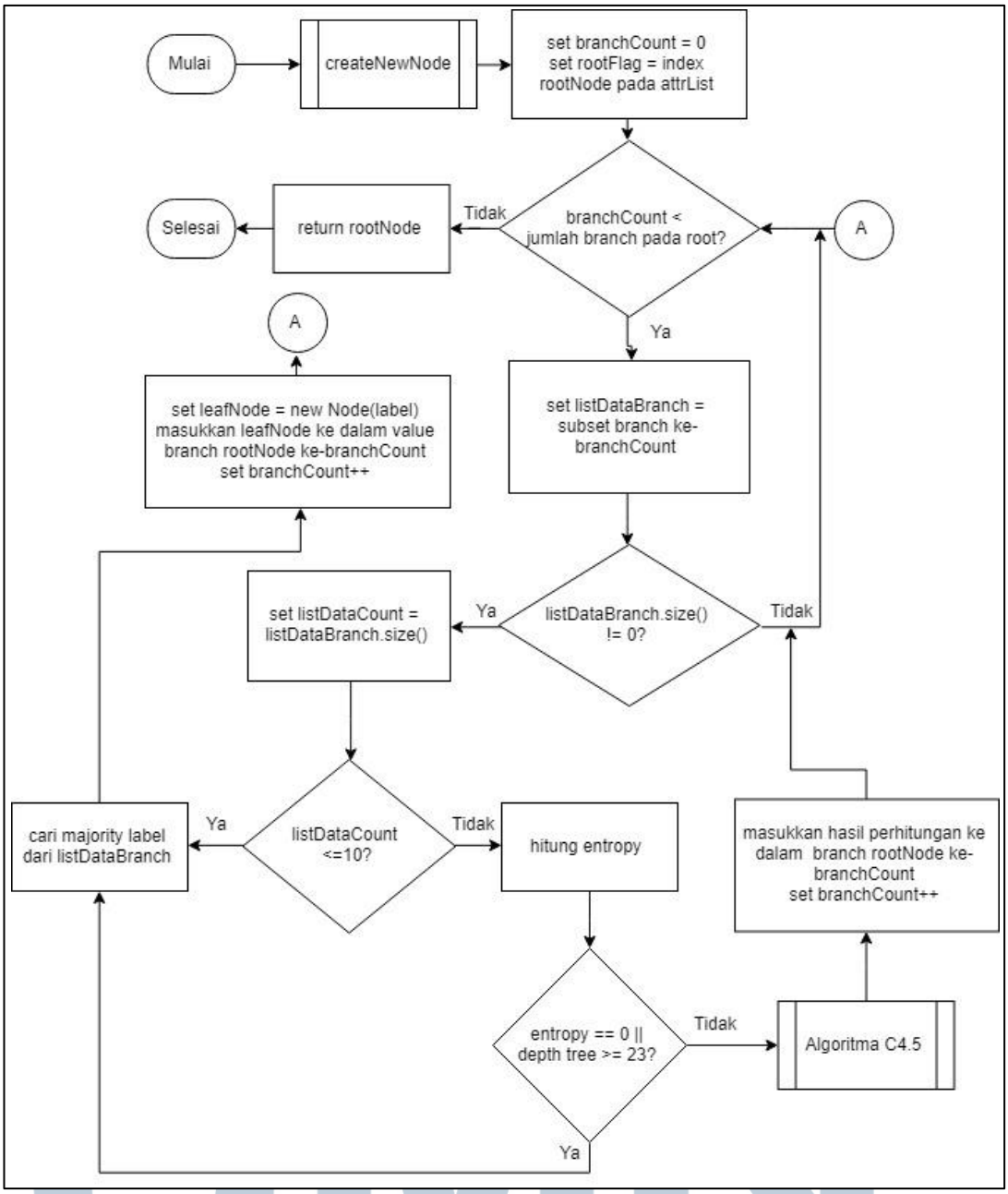
Gambar 3.2 menunjukkan alur dari aplikasi *command line* Windows. Pengguna memberi *input* berupa *command* untuk menjalankan *file* .jar beserta *path* dari *data training*. Sebelum memberikan *command*, pengguna harus memastikan bahwa posisi *path command line* sudah berada pada *file* tempat .jar *file* berada. *Command* yang diberikan berupa *command* untuk menjalankan jar *file*, yaitu java -jar <nama_file_jar> <path_data_training>. Data *training* yang diakses berupa .arff *file*. Setelah itu, program akan membaca isi dari *file* .arff dan menkonversi isi *file* menjadi data *training*. Kemudian, program membangun pohon keputusan menggunakan algoritma C4.5 dari data *training* tersebut. Setelah selesai, program

akan menampilkan teks pada *command line interface* yang menandakan pohon keputusan berhasil dibangun. Setelah itu, program akan menampilkan hasil pohon keputusan dan menampilkan teks untuk menandakan bahwa pohon keputusan telah ditulis kedalam *file* .json.

Gambar 3.3 menunjukkan *flowchart* Algoritma C4.5. Program akan mulai dengan mencari *root node* dengan memanggil fungsi createNewNode(). Setelah root node didapatkan, program melakukan pengulangan pada tiap *branch* yang dimiliki *root node* untuk menentukan *node* dari masing-masing *branch*. Terdapat tiga kemungkinan *node* dari masing-masing *branch*, yaitu *root node, internal node,* dan *leaf node*.

Program mengambil *subset* dari masing-masing *branch root* pada tiap pengulangan. *Subset* adalah *list* dari *dataset* yang telah dikelompokkan berdasarkan *value* yang sesuai dengan *value* dari masing-masing *branch*. Selanjutnya, program mengecek apakah jumlah data dari *subset* tidak sama dengan 0. Jika ya, program mengecek jumlah data dari *subset* untuk dilakukan *pre-pruning*. Jika tidak, maka program tidak dapat menentukan *node* dari *branch*, dan melanjutkan pencarian *node* dari *branch* selanjutnya. *Pre-pruning* dilakukan kepada *subset* yang memiliki jumlah data <=10.

Apabila *subset* memiliki jumlah data <= 10, maka *node* dari *branch* tersebut adalah *leaf node*, dan program mencari *majority label* untuk menjadi *label* pada *leaf node*. Jika jumlah data pada *subset* > 10, maka program akan menghitung nilai *entropy* dari *branch* tersebut.



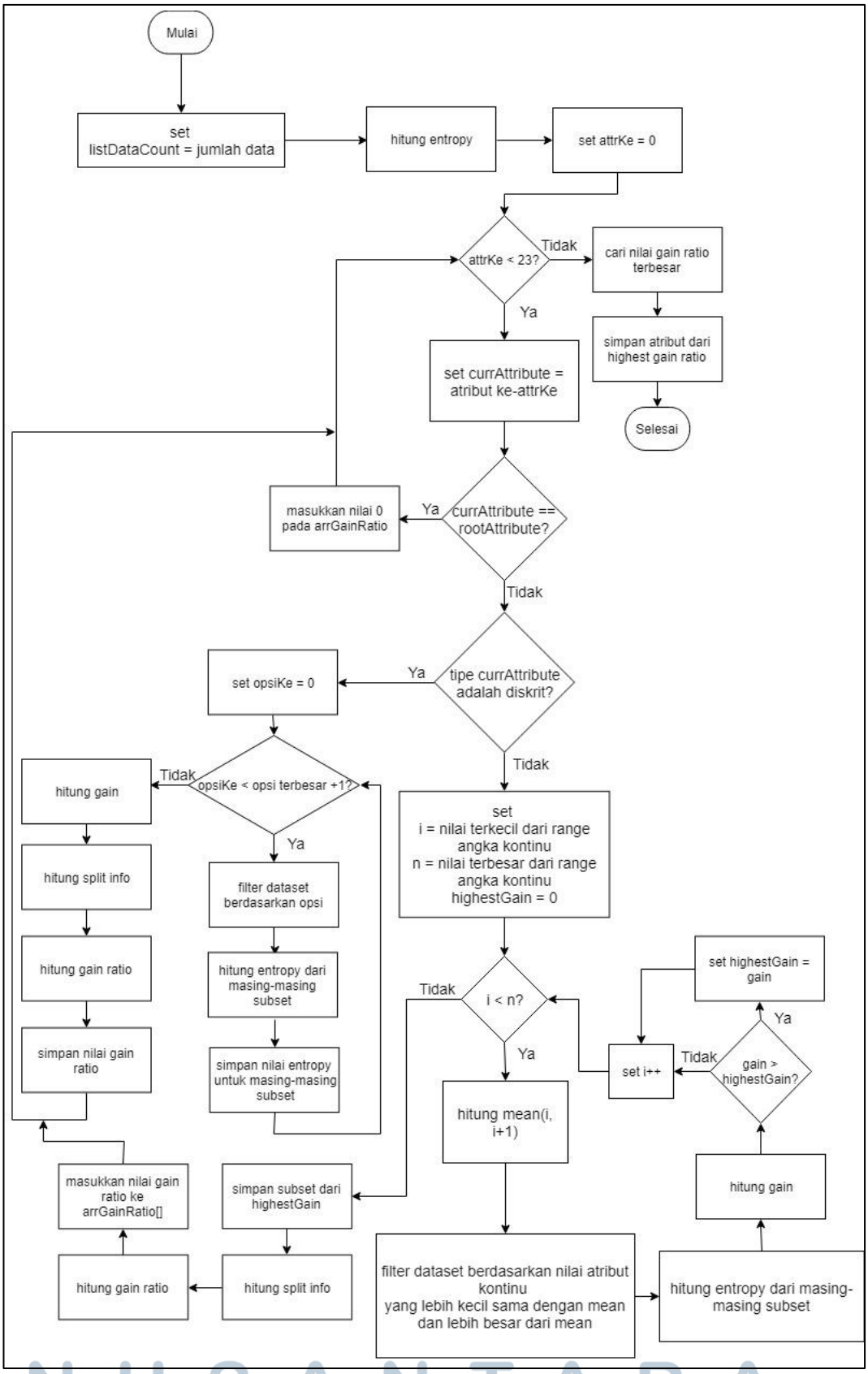
Gambar 3.3 Flowchart Algoritma C4.5

Apabila nilai *entropy* sama dengan nol, atau *depth* dari *tree* sudah mencapai jumlah atribut – 1, maka *node* dari *branch* tersebut berupa *leaf*. Program kemudian mencari *majority label* untuk menentukan *label* dari *leaf node*.

Gambar 3.4 menunjukkan *flowchart* createNewNode. createNewNode berfungsi untuk membuat *root node* baru. Awalnya, program akan melakukan

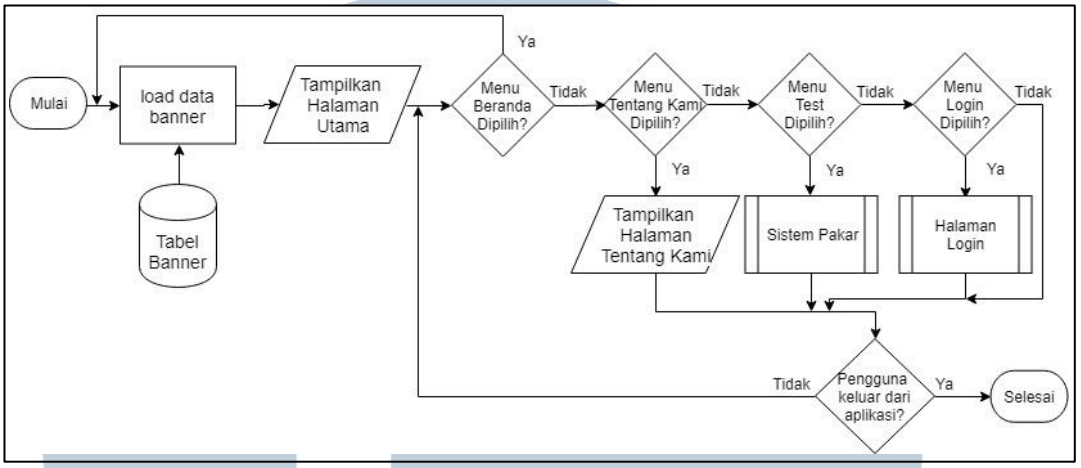
perhitungan *entropy* total keseluruhan *dataset*, dan melakukan perhitungan pada masing-masing atribut. Jika atribut saat ini sama dengan atribut *root node*, maka program akan melewati perhitungan untuk atribut tersebut. Jika tidak, program akan melanjutkan perhitungan dan melakukan pengecekan tipe atribut. Apabila tipe atribut berupa kontinu, maka program akan melakukan perhitungan *entropy*, *gain*, dan *split info* yang sesuai untuk atribut kontinu. Apabila tipe atribut berupa diskrit, maka program akan melakukan perhitungan *entropy*, *gain*, dan *split info* yang sesuai untuk atribut diskrit.

Setelah *gain* dan *split info* didapatkan, program melakukan perhitungan *gain ratio* dan menyimpan hasilnya ke dalam sebuah *array*. Lalu, setelah nilai *gain ratio* dari semua atribut telah dicari, maka program mencari nilai *gain ratio* tertinggi. *Root node* ditentukan oleh atribut dengan nilai *gain ratio* tertinggi.



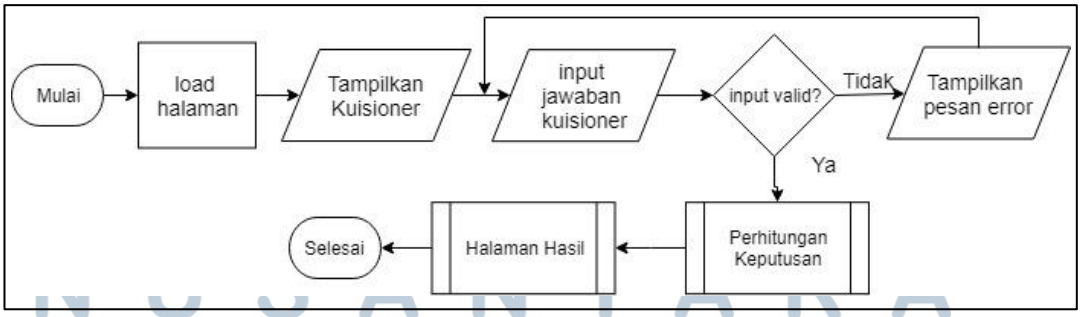
Gambar 3.4 Flowchart Create New Node

3.3.3 Flowchart Aplikasi Web



Gambar 3.5 Flowchart Halaman Utama Aplikasi Web

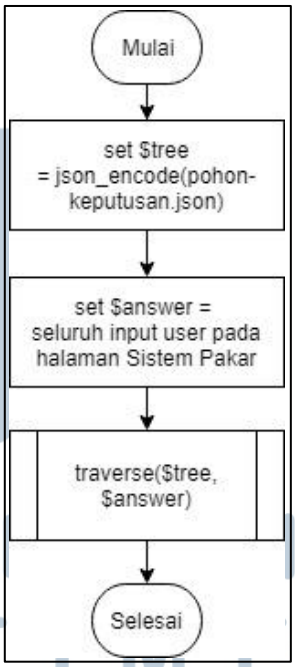
Flowchart halaman utama aplikasi *web* ditunjukkan pada Gambar 3.5. Halaman utama aplikasi *web* berisi *banner* yang menunjukkan tampilan *carousel* gambar yang berhubungan dengan masalah konsentrasi dan hiperaktivitas. Gambar diambil dari tabel *banner* pada *database*. Pengguna dapat melakukan navigasi ke halaman lain melalui *navigation bar*. Apabila pengguna memilih menu Beranda, maka program akan mengarahkan pengguna ke halaman utama aplikasi *web*. Apabila pengguna memilih menu Tentang Kami, program akan mengarahkan pengguna ke halaman Tentang Kami. Halaman ini menampilkan profil pakar dan perancang aplikasi. Apabila menu *Test* dipilih, maka pengguna akan diarahkan ke halaman Sistem Pakar, dan apabila menu *Login* dipilih, pengguna akan diarahkan ke halaman *Login Admin*.



Gambar 3.6 Flowchart Sistem Pakar

Gambar 3.6 menunjukkan alur halaman Sistem Pakar. Program akan menampilkan pertanyaan masalah konsentrasi dan hiperaktivitas mengikuti kuesioner CBCL dan CPRS yang digunakan pada saat perancangan pohon keputusan. Pengguna memberikan *input* berupa jawaban kuesioner dan menekan tombol *submit*, kemudian program akan melakukan cek terhadap *input* pengguna apabila *input* sudah benar. *Input* yang sudah benar artinya tidak ada pertanyaan kuesioner yang belum diisi. Jika ya, maka program akan melakukan perhitungan keputusan dan menampilkan keputusan yang didapat pada halaman Hasil.

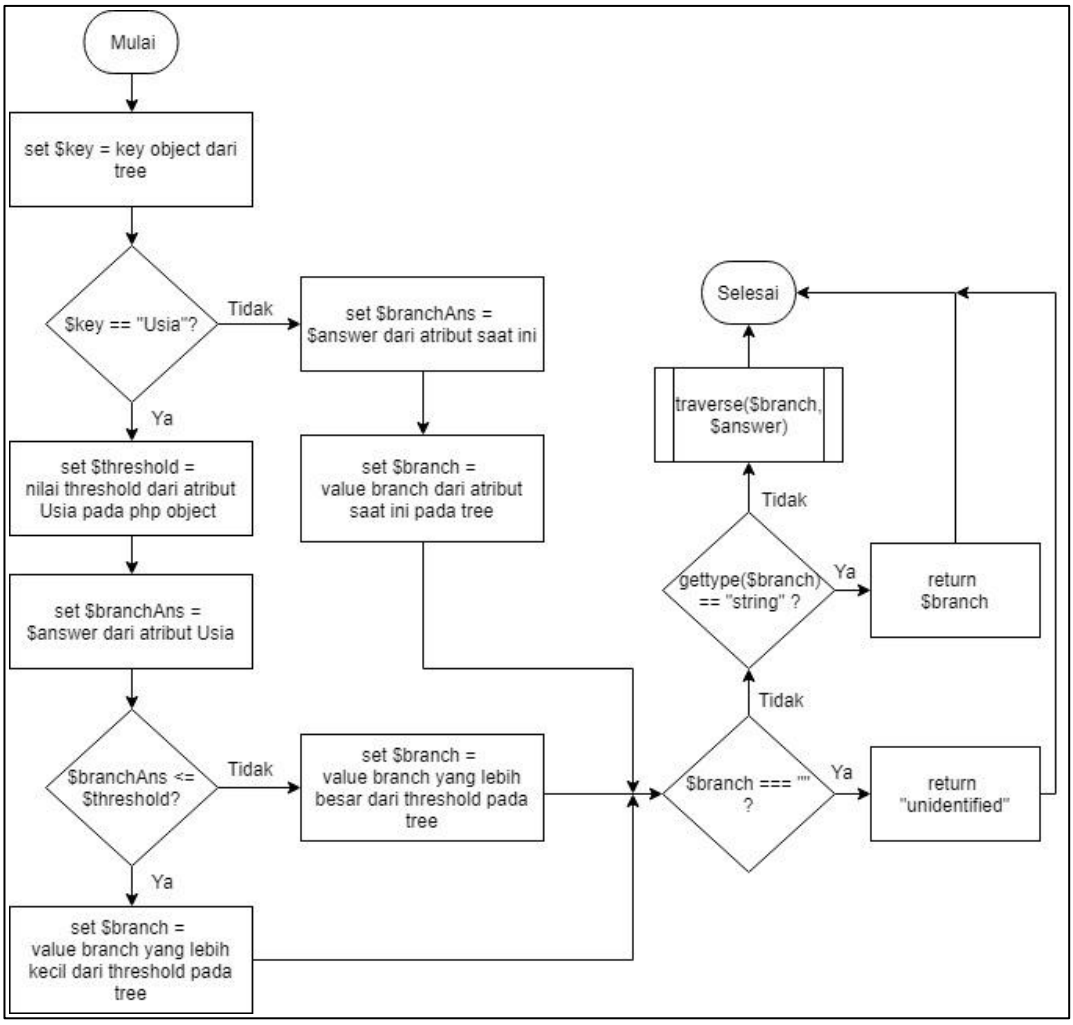
Alur perhitungan keputusan dapat dilihat pada Gambar 3.7. Program mendeklarasi variabel $tree sebagai isi dari json *file* pohon keputusan, dan juga mendeklarasi variabel $answer sebagai *array* dari *input* pengguna pada halaman Sistem Pakar. Selanjutnya, program memanggil fungsi traverse() untuk melakukan pencarian hasil berupa *leaf* dari *branch* pada *tree* sesuai dengan *input* pengguna.



Gambar 3.7 Flowchart Perhitungan Keputusan

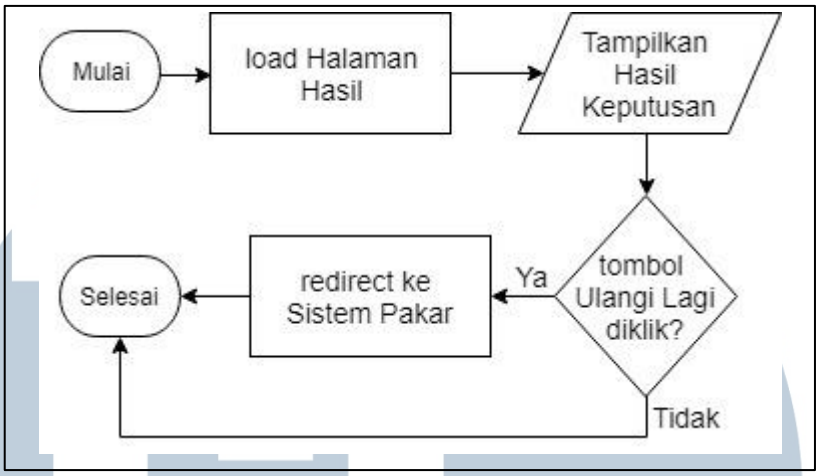
Gambar 3.8 menunjukkan alur fungsi traverse(). Variabel $key menunjukkan nama atribut dari *root*. Apabila nama atribut saat ini adalah usia, maka data input pengguna pada variabel $answer untuk atribut usia dibandingkan dengan $threshold usia. Jika nilainya lebih kecil dari $threshold, maka *branch* dari *root* yang diambil adalah *branch* pertama, yaitu *branch* yang lebih kecil dari $threshold. Jika nilainya lebih besar, maka *branch* yang diambil adalah *branch* kedua, yaitu *branch* yang lebih besar dari $threshold.

Jika atribut *root* saat ini bukan usia, maka data $answer pada atribut saat ini disimpan dalam variabel $branchAns. Variabel $branch menyimpan data *branch* yang sesuai dengan input pengguna pada atribut yang menjadi *root* dari *branch* tersebut. Kemudian, program akan melakukan pengecekan apabila isi dari variabel $branch kosong. Jika ya, maka program akan mengembalikan *string* "*unidentified"*, yang menandakan bahwa program tidak dapat memberikan hasil untuk *branch* tersebut. Jika isi $branch tidak kosong, program mengecek apakah $branch berupa *string*. Jika berupa *string*, artinya $branch merupakan *leaf node*, dan program mengembalikan isi dari *leaf node* tersebut. Jika tidak, artinya $branch merupakan *root* dan program melakukan fungsi rekursif untuk mencari *branch* dari *root* tersebut kembali.



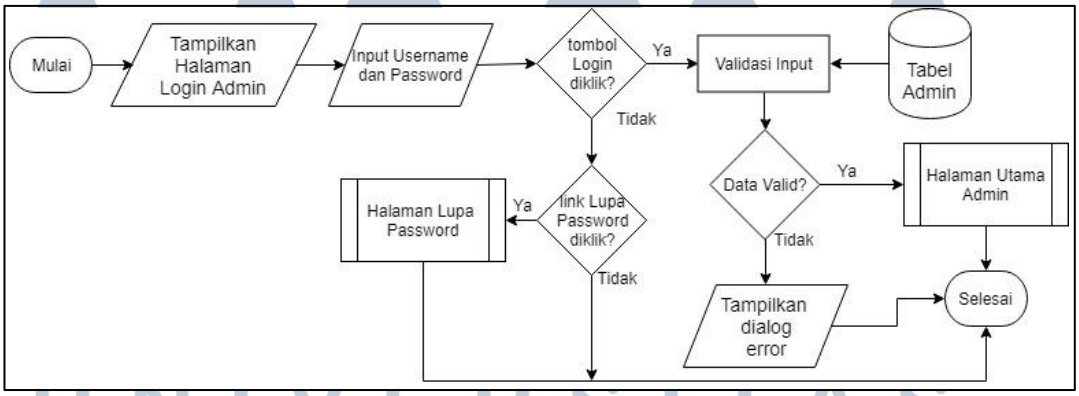
Gambar 3.8 Flowchart *Traverse*

Gambar 3.9 menunjukkan alur halaman Hasil. Program menampilkan hasil keputusan berdasarkan perhitungan keputusan sebelumnya. Selain itu, progam juga menampilkan *chart* pohon keputusan, penjelasan keputusan dan saran untuk pengguna pada halaman ini. Apabila pengguna menekan tombol Ulangi Lagi, maka program akan mengarahkan pengguna ke halaman Sistem Pakar.

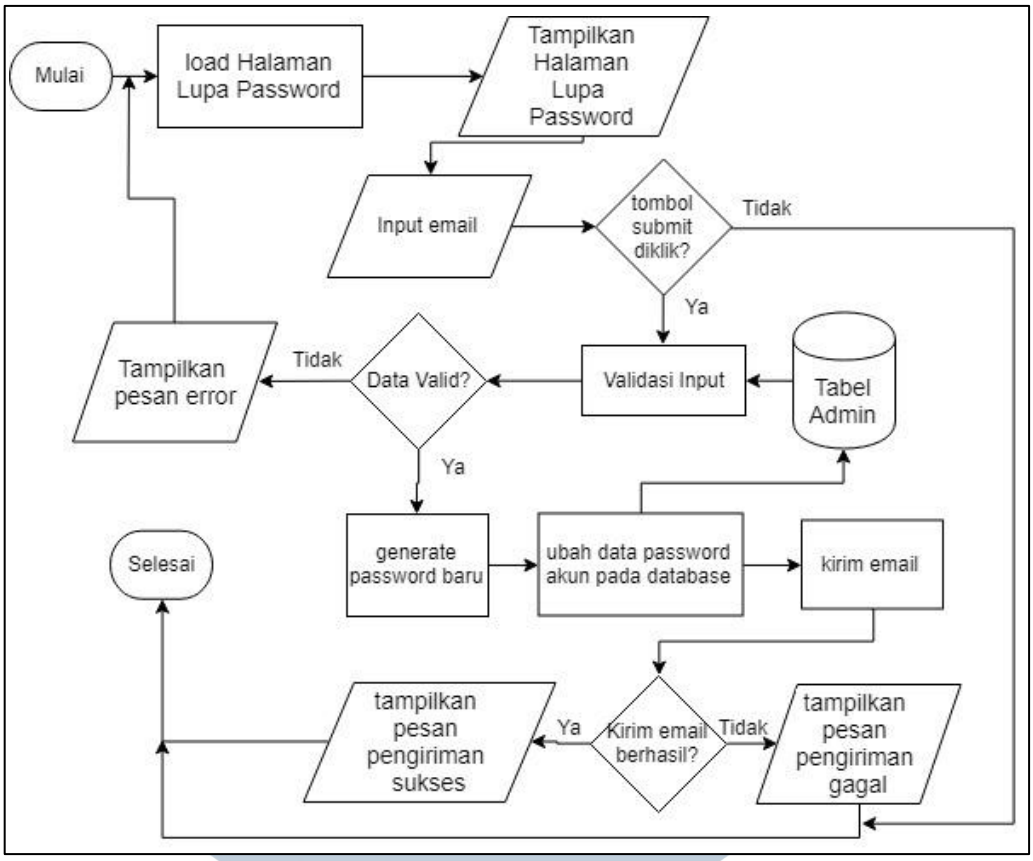


Gambar 3.9 Flowchart Halaman Hasil

Gambar 3.10 menunjukkan alur halaman *Login*. Apabila pengguna menekan *link* Lupa *Password*, pengguna akan diarahkan ke halaman Lupa *Password*. Pengguna memberikan *input* berupa *username* dan *password*, lalu program akan melakukan validasi *input* dengan melakukan cek data *username* dan *password* pada *database*. Apabila data yang diberikan *valid*, maka pengguna akan diarahkan ke Halaman Utama *Admin* oleh program. Namun, jika tidak *valid*, program akan menampilkan pesan *error* dan menunggu *input* pengguna kembali.

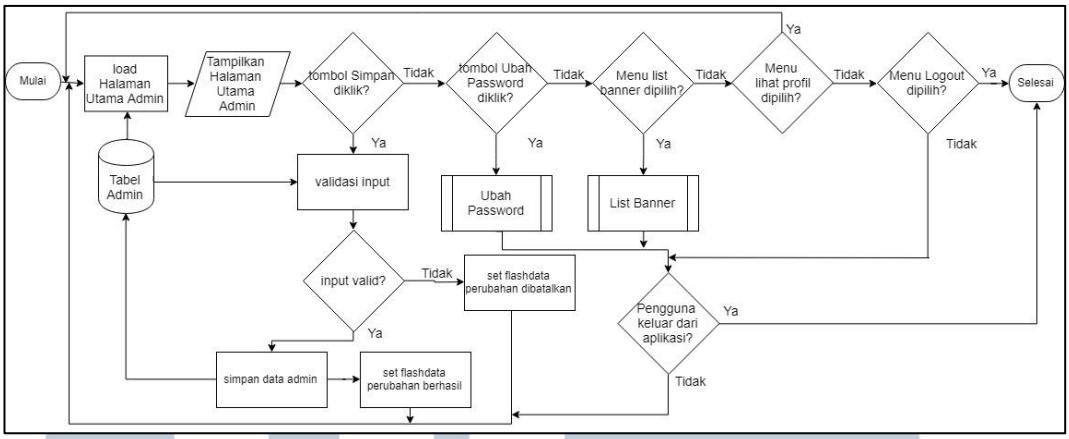


Gambar 3.10 Flowchart Halaman Login



Gambar 3.11 Flowchart Halaman Lupa Password

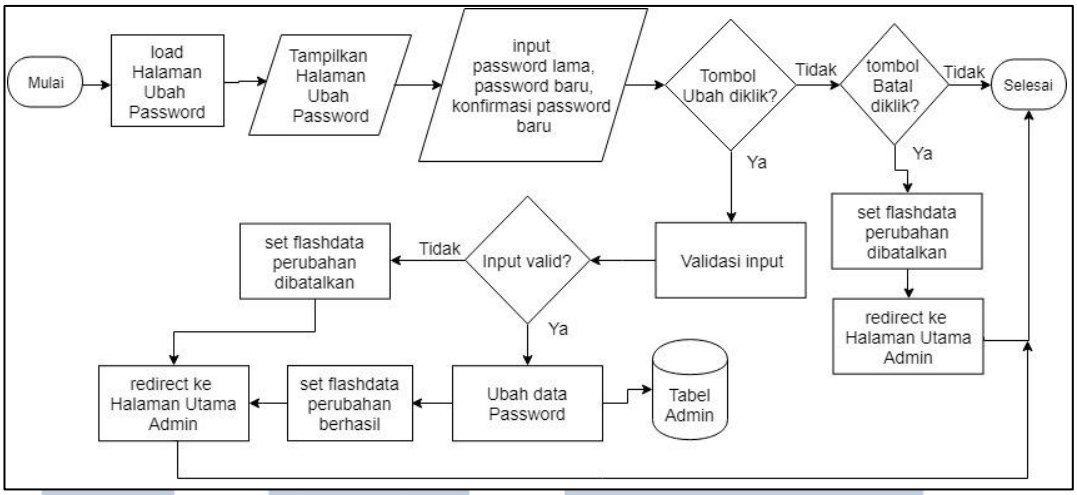
Gambar 3.11 menunjukkan alur halaman Lupa *Password*. Pengguna memasukkan *email* dari akun yang dimiliki. Apabila pengguna menekan tombol *Submit*, maka program akan melakukan validasi *input* dengan melakukan cek terhadap penulisan *email*, dan melakukan cek data *email* pada *database*. Apabila *input* sudah *valid*, program akan membuat *password* baru dengan mengambil acak nilai dari 999 sampai 99999, dan mengubah data *password* lama dari akun tersebut menjadi *password* baru pada *database*. Lalu, data *password* baru tersebut akan dikirimkan ke *email* pengguna. Apabila proses kirim *email* berhasil, program akan menampilkan pesan sukses. Jika tidak berhasil, program akan menampilkan pesan gagal.



Gambar 3.12 Flowchart Halaman Utama Admin

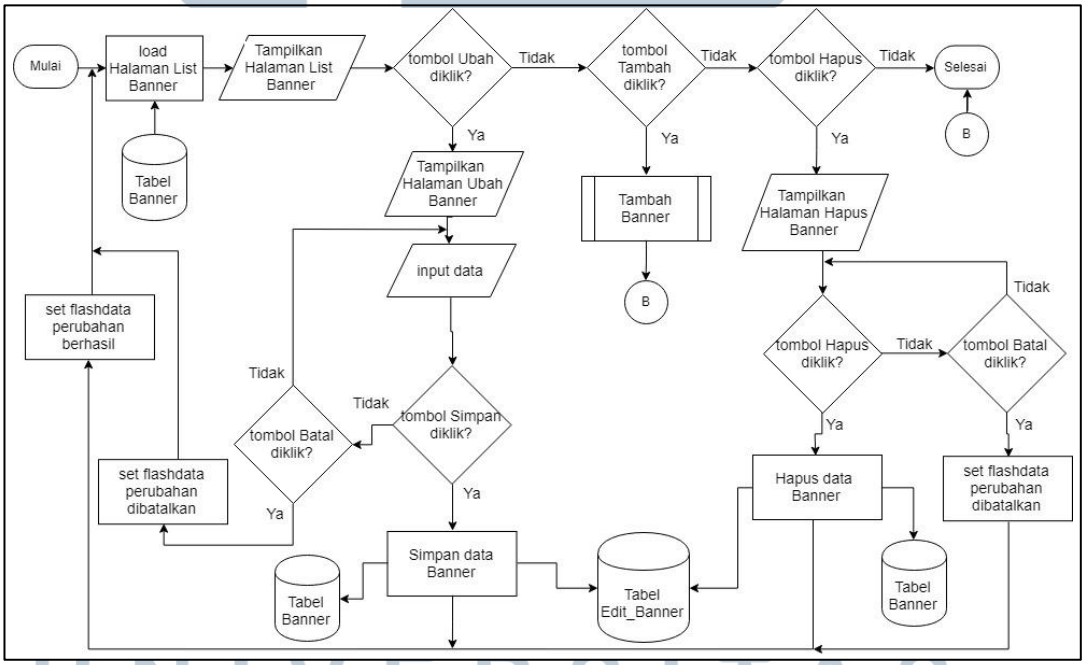
Gambar 3.12 menunjukkan alur halaman utama *admin*. Program akan menampilkan menu Lihat Profil. Pengguna dapat melakukan navigasi ke halaman *List Banner*, halaman Ubah *Password*, Halaman Utama *Admin*, ataupun melakukan *log out* aplikasi. Pengguna juga dapat melakukan modifikasi pada data profil. Apabila pengguna menekan tombol Ubah yang terletak pada sebelah kanan teks *password*, maka pengguna akan diarahkan ke halaman Ubah *Password*. Apabila pengguna menekan tombol Simpan, maka program akan melakukan validasi *input* dengan melakukan cek pada penulisan *email*, *username*, *password*, dan *fullname*. Jika *input* sudah benar, maka modifikasi pada data profil *admin* akan disimpan ke *database*.

Gambar 3.13 menunjukkan alur halaman Ubah *Password*. Pengguna memasukkan *input* berupa data *password* lama, *password* baru, dan konfirmasi *password* baru. Apabila tombol Ubah ditekan, program akan melakukan validasi *input* dengan mencocokkan ke *database* pada tabel *Admin*. Jika *input valid*, maka *password* pada *database* diubah.



Gambar 3.13 Flowchart Ubah Password

Jika tidak *valid*, maka program mengarahkan pengguna ke halaman utama *admin* dan menampilkan pesan batal. Apabila tombol Batal diklik, program mengarahkan pengguna ke halaman utama *admin* dan menampilkan pesan batal.

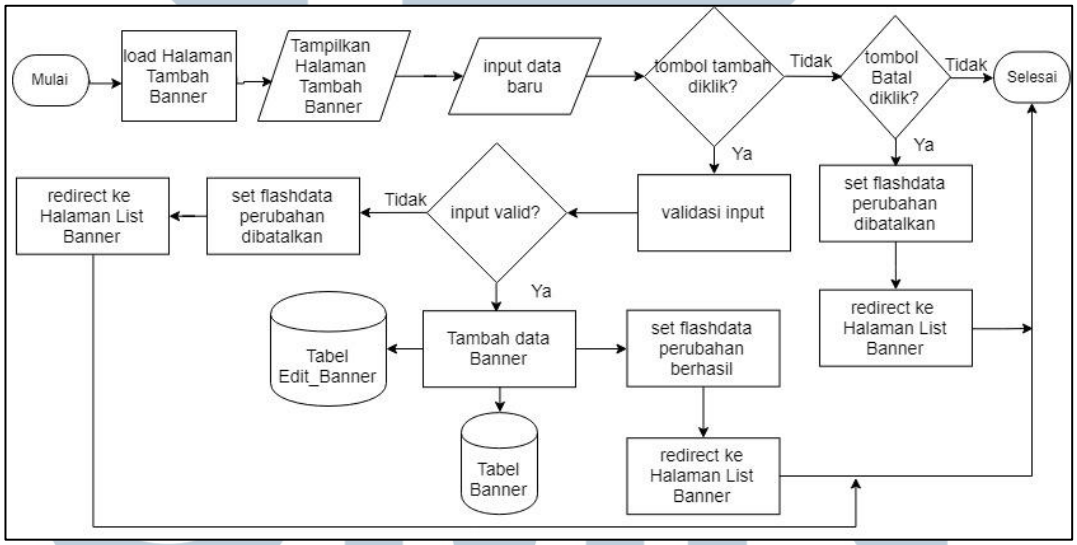


Gambar 3.14 Flowchart List Banner

Gambar 3.14 menunjukkan alur halaman *List Banner*. Apabila pengguna menekan tombol Tambah, maka pengguna akan diarahkan ke halaman Tambah *Banner*. Apabila pengguna menekan tombol Ubah, maka pengguna akan diarahkan

ke halaman Edit *Banner*. Pada halaman tersebut, pengguna dapat mengunggah gambar *banner* yang baru. Apabila pengguna menekan tombol Hapus, maka pengguna akan diarahkan ke halaman Delete *Banner*. Pada halaman tersebut, program melakukan konfirmasi penghapusan data.

Gambar 3.15 menunjukkan alur halaman Tambah *Banner*. Halaman ini berfungsi untuk menambahkan data *banner* yang baru kedalam tabel *Banner* pada *database*. Pengguna dapat menambah data *banner* baru dengan mengunggah gambar *banner*. Jika tombol Tambah ditekan, maka data *banner* yang baru akan dimasukkan ke tabel *Banner* pada *database*, dan menampilkan pesan sukses pada halaman *List Banner*. Namun, apabila tombol Batal diklik, maka program akan menampilkan pesan batal pada halaman *List Banner*.

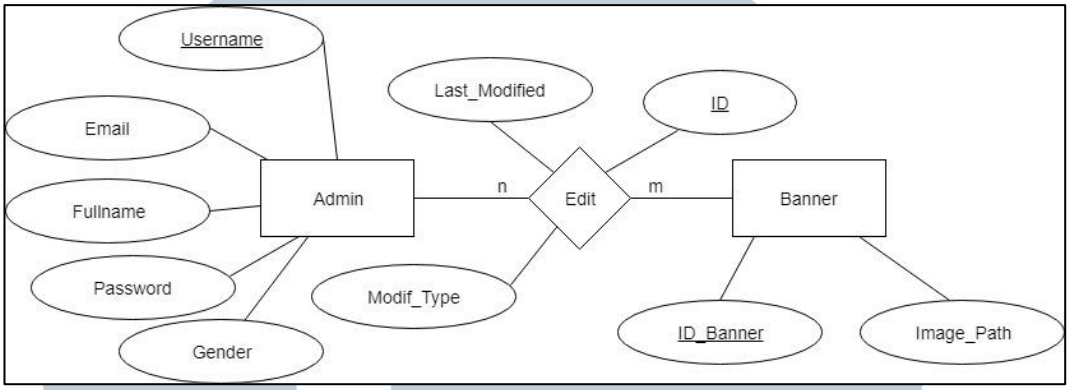


Gambar 3.15 Flowchart Tambah Banner

3.3.4 Entity Relationship Diagram

Gambar 3.16 menunjukkan Entity Relationship Diagram dari aplikasi *web* sistem pakar. Setiap *admin* dapat melakukan CRUD (*Create, Read, Update,* dan *Delete*) pada setiap data yang terdapat pada tabel *Banner*. ID untuk setiap

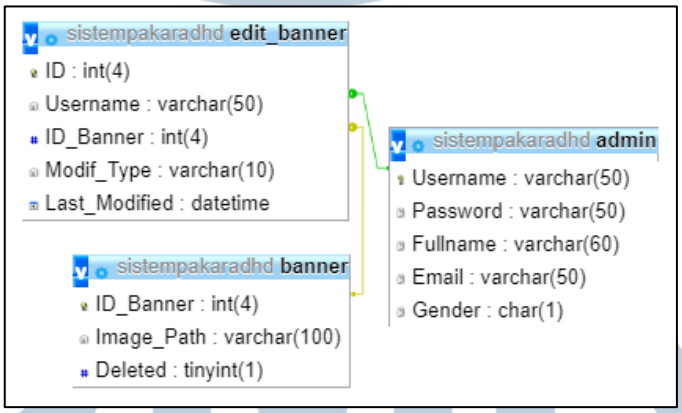
perubahan, jenis perubahan (*create*, *update*, atau *delete*) dan waktu modifikasi akan dicatat untuk setiap perubahan pada tabel *Banner* kedalam tabel Edit_Banner.



Gambar 3.16 Entity Relationship Diagram

3.3.5 Database Schema

Database Schema untuk aplikasi *web* dapat dilihat pada Gambar 3.17. Program memiliki 5 tabel, dimana tabel *Admin*, tabel Edit_Banner dan tabel *Banner* terhubung satu sama lain.



Gambar 3.17 Database Schema

3.3.6 Struktur Tabel

Program memiliki tiga tabel yang dengan fungsi masing-masing. Struktur dari masing-masing tabel dijelaskan sebagai berikut.

A. Tabel Admin

Tabel 3.3 menunjukkan Struktur Tabel *Admin* yang berfungsi menyimpan seluruh akun *admin* yang terdaftar pada sistem.

Tabel 3.3 Struktur Tabel Admin

No.	Nama Kolom	Tipe	Panjang	Keterangan
1.	Username	varchar	50	*Username* pengguna yang digunakan untuk login, dan berfungsi sebagai *primary key*
2.	Password	varchar	50	*Password* pengguna untuk validasi
3.	Fullname	varchar	60	Nama lengkap pengguna
4.	Email	varchar	50	*Email* pengguna
5.	Gender	char	1	Jenis kelamin pengguna, antara lain 'M' sebagai penanda laki-laki dan 'F' sebagai penanda perempuan

B. Tabel *Banner*

Tabel 3.4 Struktur Tabel Banner

No.	Nama Kolom	Tipe	Panjang	Keterangan
1.	ID_Banner	int	4	*Primary key*
2.	Image_Path	varchar	100	*Path* dari *image* yang ingin ditampilkan pada *banner*
3.	Deleted	boolean		Menandakan apakah suatu data sudah dihapus oleh *admin* atau belum. Jika bernilai *true*, maka data tidak akan ditampilkan pada halaman *List Banner*.

Tabel 3.4 menunjukkan Struktur Tabel *Banner*. Tabel *Banner* menyimpan data *path* dari gambar-gambar yang akan ditampilkan pada halaman utama aplikasi web.

C. Tabel Edit_Banner

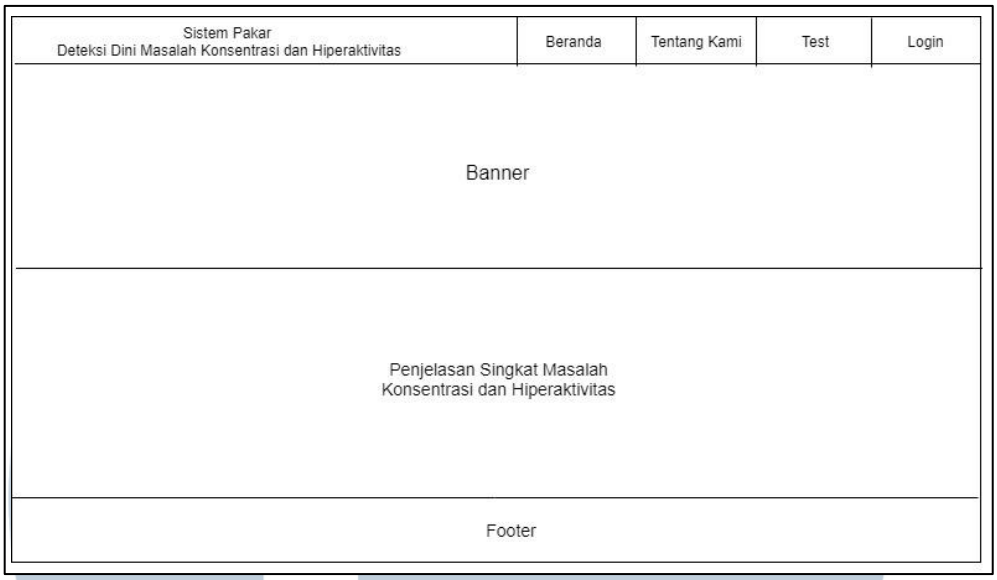
Tabel 3.5 menunjukkan Struktur Tabel Edit_Banner. Tabel ini berfungsi menyimpan data berupa tanggal dan waktu modifikasi yang dilakukan oleh admin terhadap tabel *banner*.

Tabel 3.5 Struktur Tabel Edit_Banner

No.	Nama Kolom	Tipe	Panjang	Keterangan
1.	ID	int	4	*Primary key.*
2.	Username	varchar	50	*Username* pengguna yang melakukan modifikasi pada sebuah *banner.*
3.	ID_Banner	int	4	ID dari *banner* yang dimodifikasi.
4.	Modif_Type	varchar	10	Tipe modifikasi yang dilakukan terhadap data *banner*.
5.	Last_Modified	datetime	-	Menunjukkan tanggal dan waktu modifikasi suatu *row* pada tabel *banner*.

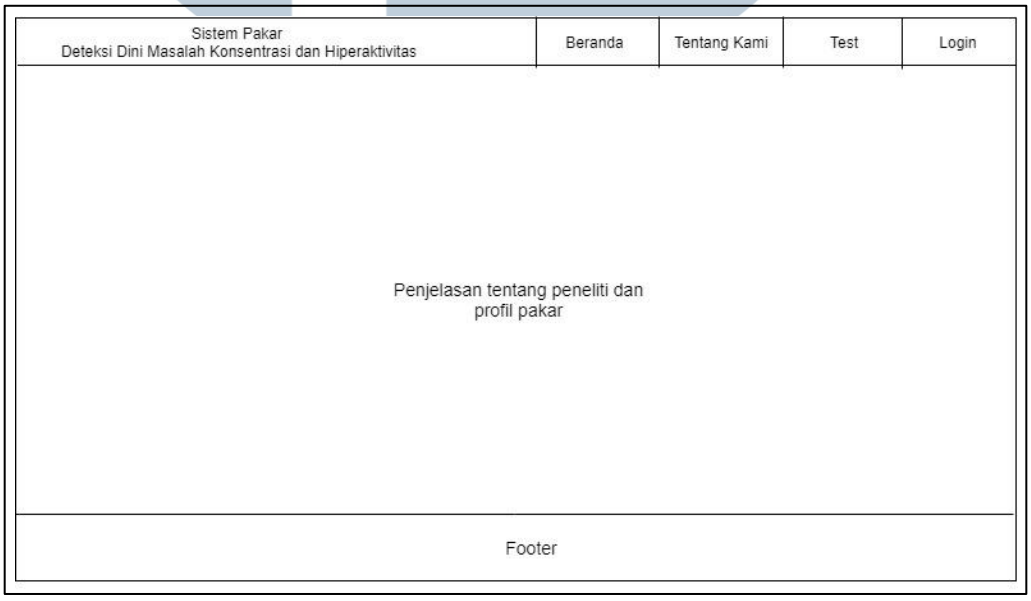
3.3.7 Desain Antarmuka

Tampilan Halaman Utama Aplikasi *Web* ditunjukkan pada Gambar 3.18. *Banner* berfungsi untuk menampilkan gambar-gambar yang berkaitan dengan masalah konsentrasi dan hiperaktivitas. Pengguna dapat melakukan navigasi ke halaman lain melalui *navigation bar*. *Footer* berfungsi untuk memberikan *copyright* pada halaman *web*.



Gambar 3.18 Rancangan Antarmuka Halaman Utama

Gambar 3.19 menunjukkan tampilan halaman Tentang Kami. Halaman ini akan memberikan informasi mengenai aplikasi, profil pakar, dan profil peneliti.

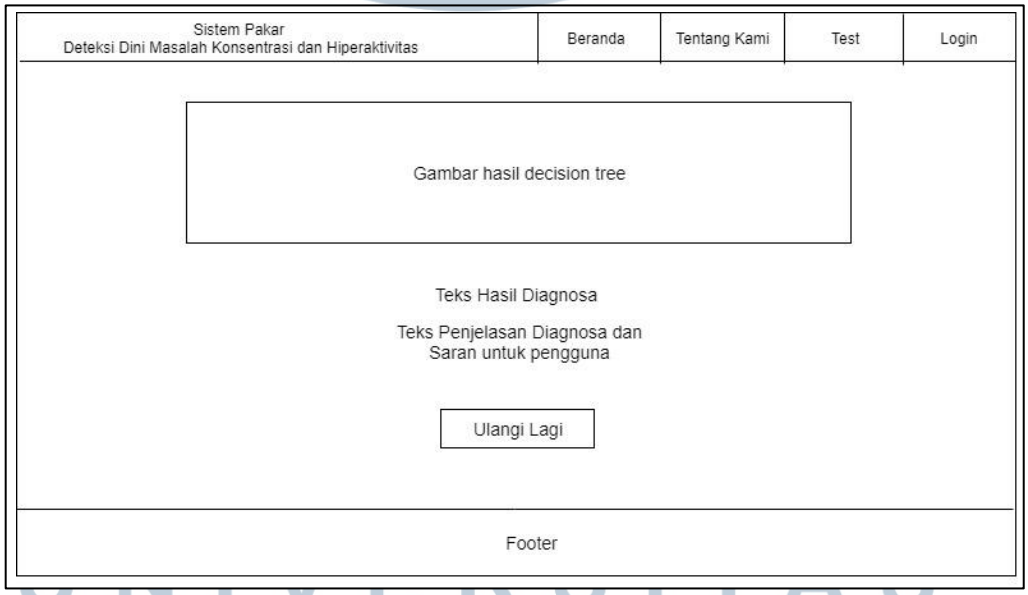


Gambar 3.19 Rancangan Antarmuka Halaman Tentang Kami

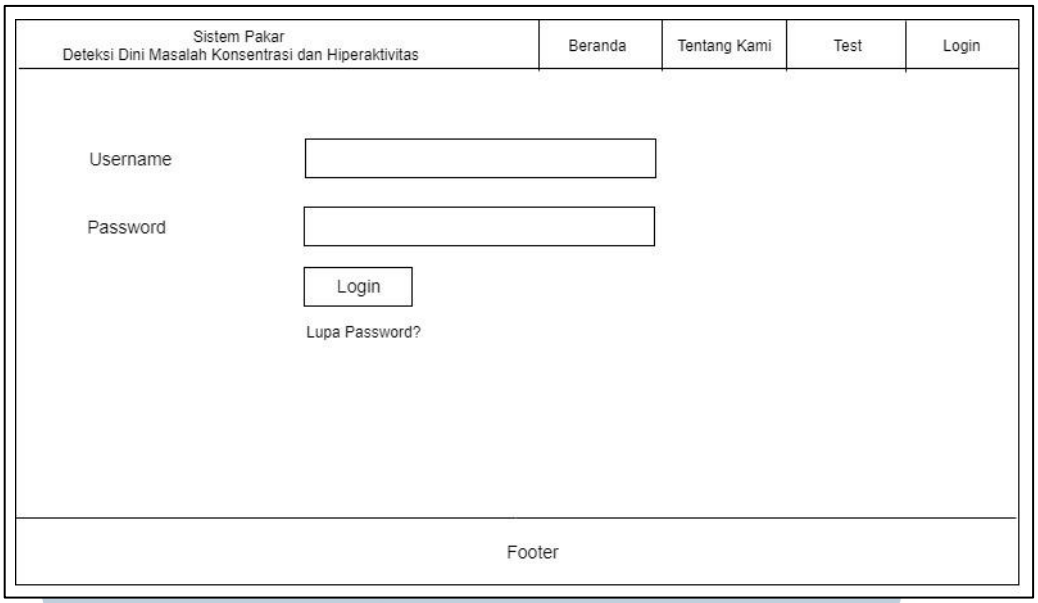
Gambar 3.20 menunjukkan rancangan tampilan halaman *Test*. Halaman ini menampilkan pertanyaan kuesioner yang digunakan untuk membentuk pohon keputusan. Pengguna dapat mengisi dan menekan tombol Submit untuk diarahkan ke halaman Hasil.

Gambar 3.20 Rancangan Antarmuka Halaman Test

Gambar 3.21 merupakan rancangan tampilan halaman Hasil. Halaman ini menampilkan pohon keputusan, hasil diagnosa sistem, penjelasan hasil diagnosa, serta saran untuk pengguna.

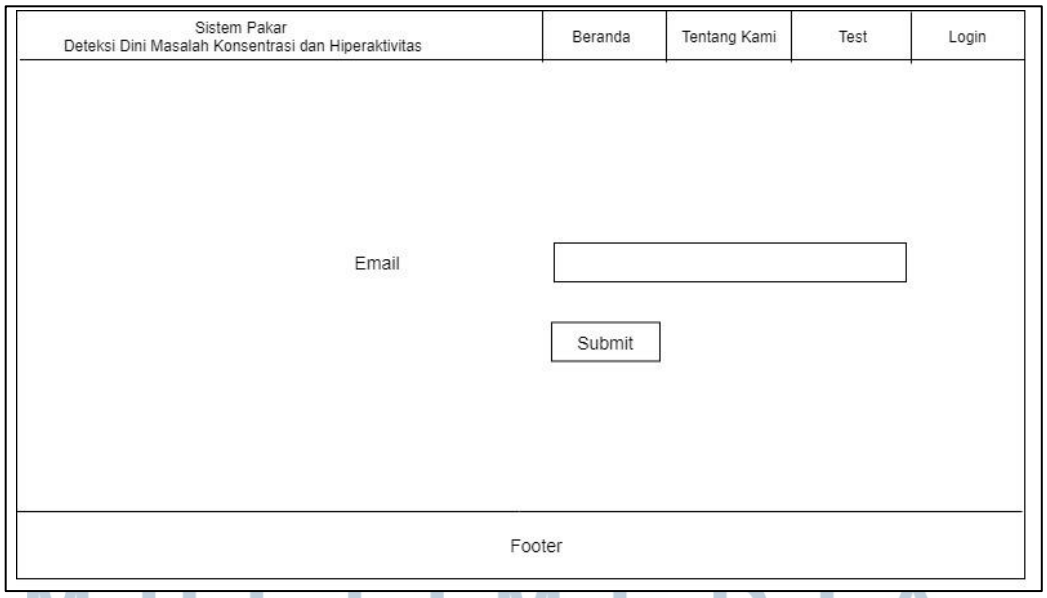


Gambar 3.21 Rancangan Antarmuka Halaman Hasil



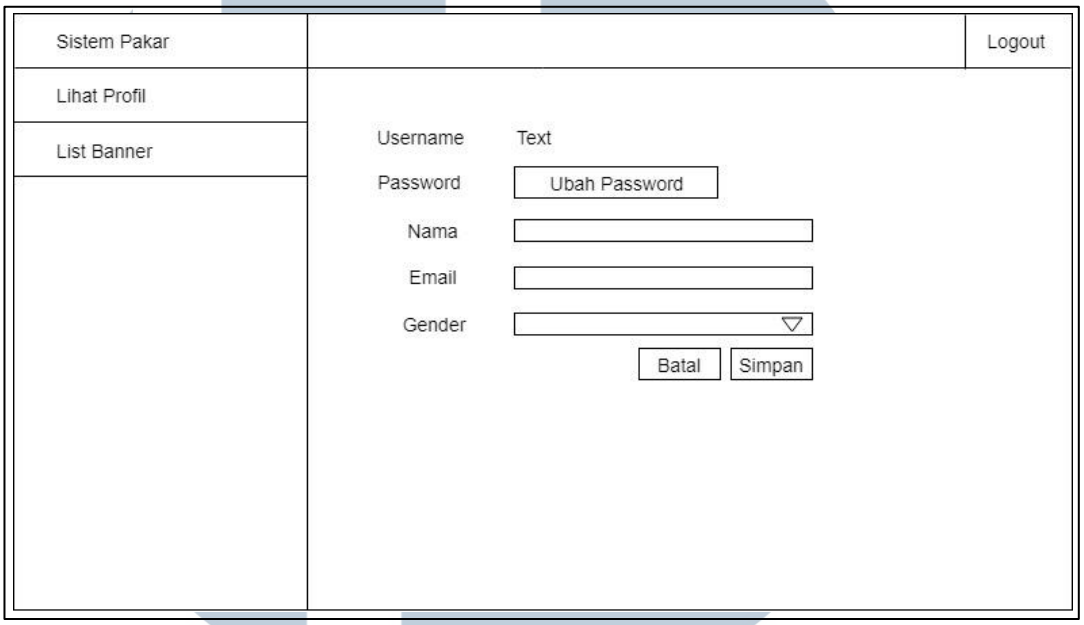
Gambar 3.22 Rancangan Antarmuka Halaman Login

Gambar 3.22 adalah tampilan halaman *Login*. Program meminta data *username* dan *password* agar pengguna dapat melakukan proses *login* dan masuk kedalam halaman utama *admin*. Pengguna dapat menekan link Lupa *Password* untuk diarahkan ke halaman Lupa *Password*.



Gambar 3.23 Rancangan Antarmuka Halaman Lupa Password

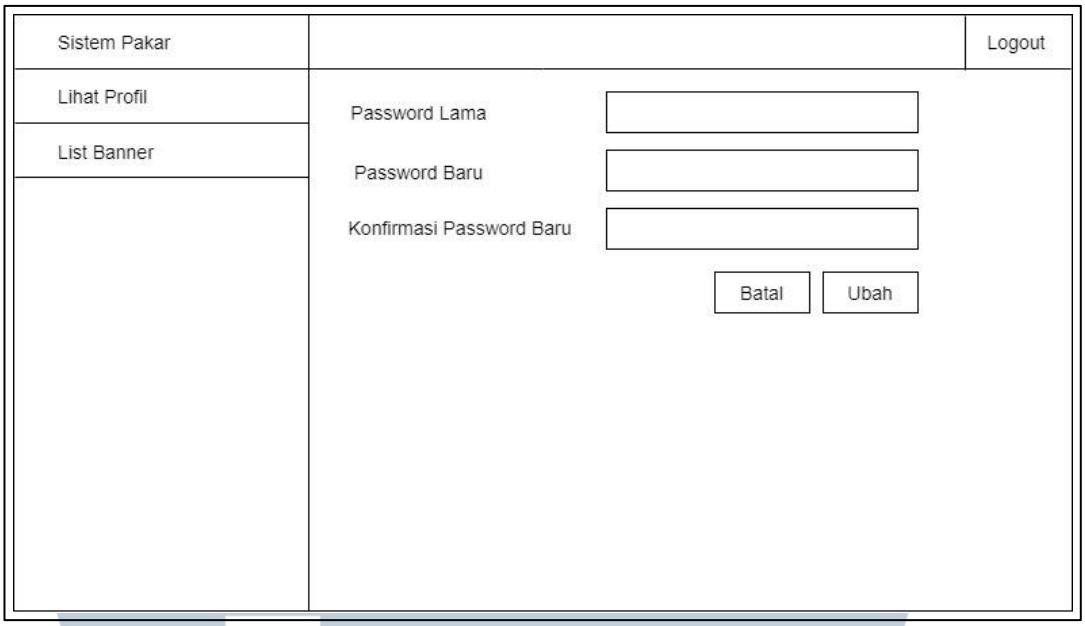
Gambar 3.23 adalah tampilan halaman Lupa *Password*. Pengguna memasukkan data *email* dari akun yang sudah teregistrasi. Apabila pengguna menekan tombol *Submit*, muncul notifikasi berupa *modal*.



Gambar 3.24 Rancangan Antarmuka Halaman Utama Admin

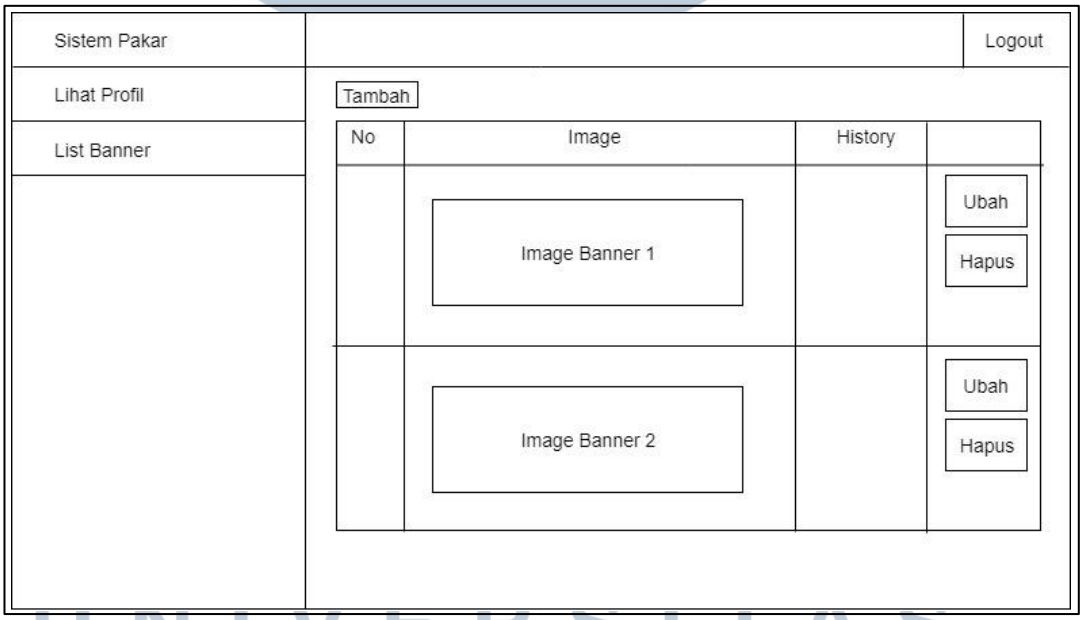
Gambar 3.24 adalah tampilan Halaman Utama *Admin*. Halaman Utama *Admin* menampilkan menu Lihat Profil. Navigasi antar halaman dapat dilakukan melalui *sidebar*. *Navigation bar* pada bagian atas halaman berfungsi menampung menu *Log Out*. Data profil pada *database* akan ditampilkan berupa teks, kecuali data *gender* yang ditampilkan berupa pilihan *drop down*. Pengguna dapat meng-*edit* data profil pada halaman ini. Pengguna dapat menekan tombol Ubah jika ingin mengubah data *password*.

Gambar 3.25 menunjukkan rancangan antarmuka halaman Ubah *Password*. Apabila tombol Batal ditekan, maka program akan mengarahkan pengguna ke halaman utama *admin* dan menampilkan pesan pembatalan perubahan *password*.

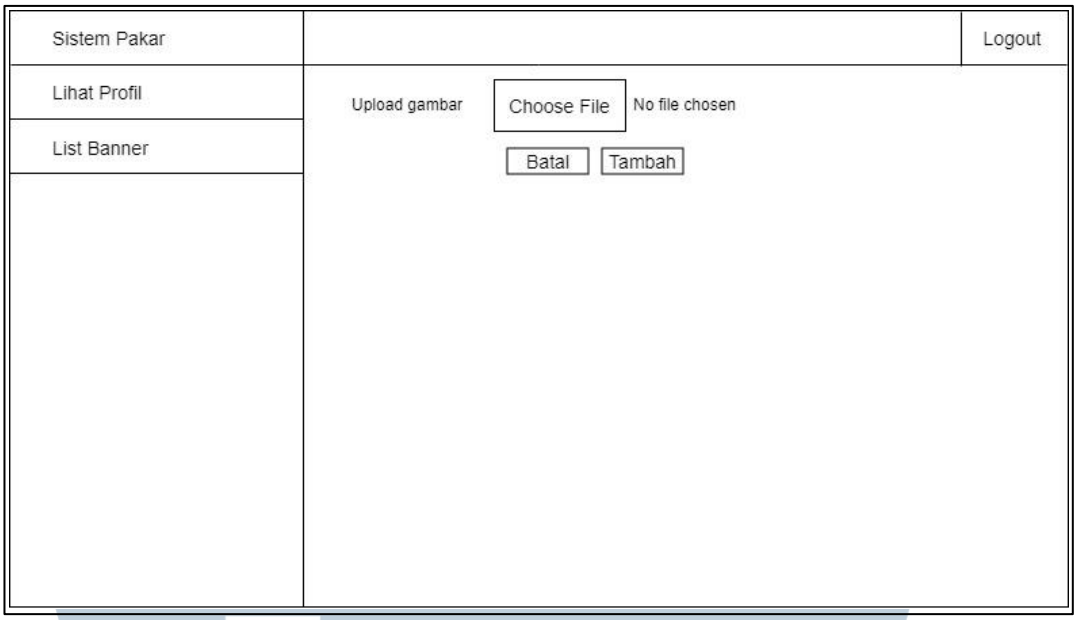


Gambar 3.25 Rancangan Antarmuka Halaman Ubah Password

Tampilan pada Gambar 3.26 merupakan desain antarmuka halaman *List Banner*. Data *banner* yang disimpan di *database* akan ditampilkan berupa tabel.

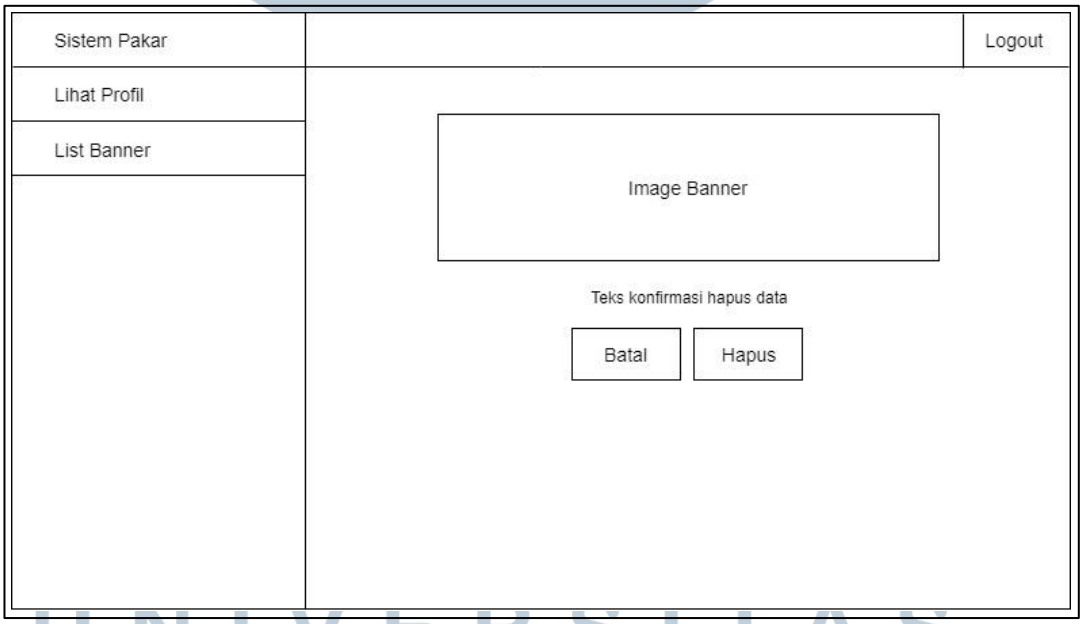


Gambar 3.26 Rancangan Antarmuka Halaman List Banner



Gambar 3.27 Rancangan Antarmuka Halaman Tambah Banner

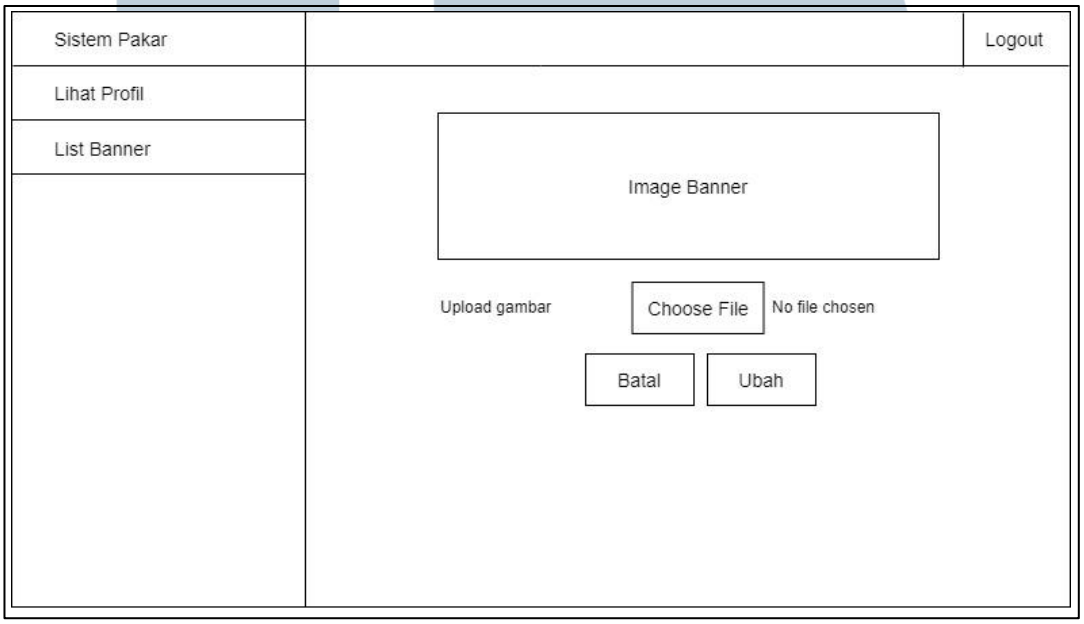
Gambar 3.27 menunjukkan rancangan antarmuka halaman Tambah *Banner*. Pengguna dapat mengunggah gambar *banner* baru.



Gambar 3.28 Rancangan Antarmuka Halaman Hapus Banner

Gambar 3.28 menunjukkan rancangan antarmuka halaman Hapus *Banner*. Pengguna dapat menghapus data *banner* dengan menekan tombol Hapus.

Gambar 3.29 menunjukkan rancangan antarmuka halaman Ubah *Banner* apabila tombol Ubah pada halaman *List Banner* diklik. Pengguna dapat mengubah data *banner* dengan menggunggah gambar *banner* yang baru pada dialog. Apabila tombol Batal diklik, program mengarahkan pengguna ke halaman *List Banner* dan memberi pesan bahwa perubahan data *banner* dibatalkan.



Gambar 3.29 Rancangan Antarmuka Halaman Ubah Banner