

BAB III

ANALISIS DAN PERANCANGAN

3.1 Analisis Kebutuhan

Kebanyakan *mobile device* atau telepon seluler yang beredar saat ini di masyarakat sudah mendukung teknologi JME dengan MIDP 2.0 serta CLDC 1.1. Selain dari sisi hiburan dan peningkatan produktivitas, aplikasi yang ditanam pada perangkat bergerak juga memungkinkan penggunanya melakukan berbagai hal secara lebih praktis.

Aplikasi *flashcard* yang akan menjadi hasil dari penelitian ini diharapkan dapat membantu pengguna telepon seluler yang memiliki fitur JME, MIDP 2.0, dan CLDC 1.1. untuk belajar secara lebih efektif dan dapat digunakan di mana saja. Keterbatasan *resources* dari sebuah perangkat bergerak juga menjadi tantangan bagi pengembangan aplikasi yang akan ditanam di dalamnya.

3.2 Spesifikasi Kebutuhan Sistem

3.2.1 Kebutuhan Fungsional

Berdasarkan analisis kebutuhan di atas, maka aplikasi *flashcard* yang akan dibangun memiliki kemampuan fungsional sebagai berikut :

1. Aplikasi ini merupakan versi sederhana dari aplikasi *flashcard* yang sudah umum terdapat dalam versi *desktop*.

Aplikasi ini memungkinkan pengguna memasukkan *card*, mengeditnya, dan menghapusnya baik sebagian maupun secara keseluruhan. Selain itu, terdapat

pula fitur pencarian untuk menemukan *card* yang diinginkan untuk selanjutnya diedit atau dihapus.

2. Aplikasi dapat berjalan pada perangkat bergerak yang memiliki fitur JME dengan MIDP 2.0.
3. Terdapat menu yang menampilkan statistik dari *cards* yang sudah diinputkan dan dipelajari.
4. Sebagai contoh kasus, akan dimasukkan beberapa *card* ke dalam emulator maupun perangkat bergerak.

3.2.2 Kebutuhan Non Fungsional

Aplikasi *flashcard* yang akan ditanam pada telepon seluler ini akan dibuat dengan menggunakan NetBeans 6.1 sebagai IDE (*Integrated Development Environment*) yang sudah dilengkapi dengan NetBeans Mobility Pack 6.1. Pengujian aplikasi pada emulator akan menggunakan JME WTK versi 2.5.2 dari Sun Microsystem. Untuk implementasi pada perangkat bergerak sebenarnya akan menggunakan telepon seluler yang sudah mendukung JME dengan MIDP 2.0 dan CLDC 1.1.

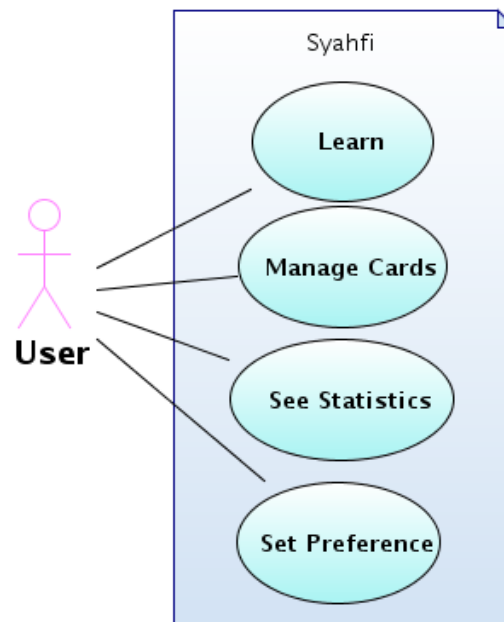
3.3 UML (*Unified Modeling Language*)

3.3.1 Use Case Diagram

Use case diagram berikut memodelkan perilaku (*behaviour*) dari sistem pada aplikasi *flashcard*. Aplikasi *flashcard* yang akan dibuat memiliki 1 (satu) aktor dan 4 (empat) *use case* yaitu

1. Manajemen data *card*, seperti menambah, mencari, menghapus, dan mengedit *card*.
2. Menampilkan menu statistik *card*.
3. Melakukan pembelajaran dari data *card* yang sudah dimasukkan sebelumnya.
4. Mengubah preferensi atau *setting* dari aplikasi.

Hubungan antara aktor dan sistem ditunjukkan oleh gambar 3.1.



Gambar 3.1. *Use case diagram* aplikasi

3.3.2. *Class Diagram*

Class diagram menggambarkan struktur dan deskripsi *class*, *package*, dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. Dalam aplikasi yang akan dibuat terdapat 7 *class*. Gambar 3.2 menunjukkan *class diagram* dari aplikasi Syahfi.

3.3.2.1 Class Syahfi

Class ini merupakan *class* yang pertama kali dijalankan. Nama *class* ini sama dengan nama aplikasi yang akan dibuat pada penelitian ini. *Class* ini merupakan turunan dari *class MIDlet*. Dalam *class* ini terdapat 5 (lima) metode penting yakni sebuah konstruktor, metode *startApp()*, metode *pauseApp()*, dan metode *destroyApp()*. Sedangkan operasi *exitMIDlet()* hanya memudahkan untuk mengeksekusi metode *destroyApp()*.

3.3.2.2 Class Learn

Class Learn dipanggil melalui metode *startApp()* dari *class Syahfi*. *Class* ini dapat dikatakan sebagai *class* penting karena setiap kali aplikasi dipanggil, *class* ini juga akan dijalankan pertama kali. *Class* ini merupakan *extends* dari *Form* dan mengimplementasikan *CommandItemListener* dan *Item CommandListener*. *Class* ini memiliki atribut *static* yang hanya dapat diakses *class Learn* (*private*) yang diberi nama *revisionQueue*. Kegunaan dari atribut *revQueue* adalah menampung *queue* atau antrian *card* yang akan dipelajari pengguna setelah sebelumnya memanggil metode *rebuildRevisionQueue()* dalam *class Store*.

3.3.2.3 Class Card

Class Card adalah *class* turunan dari *Form*. *Class* ini juga mengimplementasikan *class CommandListener*, *ItemCommandListener*, dan *class ItemStateListener*. Setelah diinisiasi, *class* ini akan menampilkan semua *card* yang

telah dimasukkan pengguna sebelumnya. Menu pada *class* ini yang akan membawa pengguna ke *class EditCard* adalah menu “Add New” atau dengan memasukkan ID *card* pada input teks yang telah disediakan dan menekan *button* “Edit Card”. *Class Card* juga akan memanggil *class Store* agar dapat menampilkan data yang berkaitan dengan *record store*.

3.3.2.4 *Class EditCard*

Class ini merupakan turunan dari *Form* serta mengimplementasikan *CommandListener*. *Class* ini hanya dapat ditampilkan dari *class Card*. Terdapat 3(tiga) parameter dalam *class* ini, yaitu *midlet*, *display*, dan *cardId*. Parameter *cardId* yang bertipe *integer* akan membedakan apakah pengguna akan menambah *card* baru (ditandai dengan nilai *cardId* yang sama dengan 0) atau mengedit *card* yang sudah ditambahkan sebelumnya (nilai *cardId* lebih besar dari 0).

Bila pengguna menghendaki mengedit *card* lama, maka nilai teks pada *TextField tfQuestion* dan *TextField tfAnswer* akan diambil dari nilai *question* dan nilai *answer* dari *cardId* yang bersangkutan. Namun bila pengguna akan menambah *card* baru, maka nilai teksnya akan bernilai kosong. Selain itu, perbedaan antara mengedit *card* lama dengan menambahkan *card* baru terletak pada diperbolehkannya pengguna mengganti *grade* dari *card*. *Card* baru dapat diinisiasi nilai *grade*-nya oleh pengguna dengan memilih rentang nilai dari 0 hingga 5, sedangkan *card* lama tidak akan berubah nilai *grade*-nya kecuali karena proses belajar.

3.3.2.5 Class Setting

Class ini adalah turunan dari *Form*. *Class* ini dapat dipanggil dari *class Learn*. Dalam *class* ini pengguna dapat mengubah nilai dari preferensi. Hanya ada dua nilai yang dapat diubah, yaitu banyaknya *card* yang memiliki *grade 0* yang akan dipelajari sekali waktu, serta pilihan apakah pengguna akan mempelajari *card* baru dengan urutan yang acak. Untuk keperluan penyimpanan preferensi pengguna, *class Store* akan dipanggil.

3.3.2.6 Class Statistic

Class ini adalah turunan dari *Form* dan dapat dipanggil melalui *class Learn* melalui menu “Statistics”. *Class* ini akan memanggil *class Store* untuk mengetahui informasi tentang *card*.

3.3.2.7 Class Store

Class Store adalah satu-satunya *class* dalam aplikasi Syahfi yang mengimpor *package javax.microedition.rms*. *Class* ini akan menangani semua hal yang berkaitan dengan *data store (record store)*. *Class* ini merupakan *class* yang paling banyak memiliki *operations*. Beberapa *operations* yang ada dalam *class* ini akan dijelaskan sebagai berikut.

1. Method open()

Metode ini selalu dipanggil setelah *class Store* dipanggil *class* lain. Gunanya adalah untuk membuka *record store*. *Record store* akan dibuat secara otomatis bila sebelumnya belum ada.

2. *Method initiateData()*

Apabila *record store* masih kosong, maka akan dimasukkan data inisiasi yang berisi preferensi standar dan nilai penting (yang sering diakses pengguna) seperti *timeOfStart* dengan memanggil metode ini.

3. *Method close()*

Metode ini berfungsi menutup koneksi *record store* yang sebelumnya dibuka.

4. *Method saveCard()*

Metode ini memiliki 12 parameter yang merupakan jumlah atribut dari tiap *card*.

5. *Method rebuildRevisionQueue()*

Metode ini berfungsi untuk mencari *card* yang akan dimasukkan ke dalam antrian revisi untuk dipelajari. Metode ini hanya memiliki satu parameter yakni *learnAhead* yang bertipe *boolean*. Bila *learnAhead* bernilai *true*, maka otomatis *card* yang dimasukkan ke dalam antrian adalah *card* yang dijadwalkan untuk hari setelah hari ini.

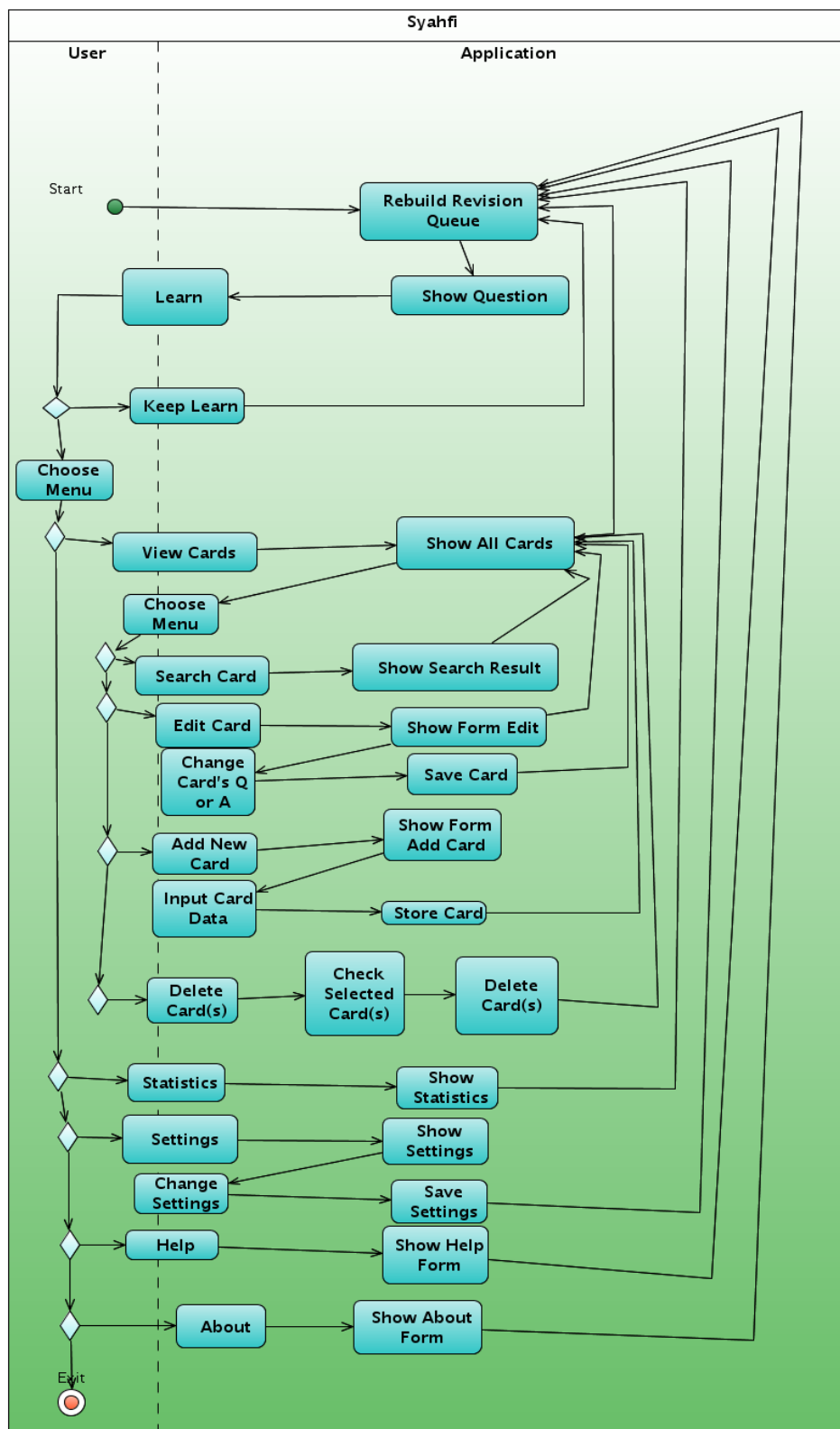
6. *Method processAnswer()*

Metode ini akan mengembalikan nilai interval dari *card* yang baru. Nilai interval menunjukkan berapa hari lagi sebuah *card* akan dipelajari atau dijadwalkan. Parameter dalam metode ini adalah *cardId* dan *newGrade*.

3.3.3 *Activity diagram*

Activity diagram menunjukkan aktivitas-aktivitas yang terdapat pada aplikasi. Aktivitas yang pertama kali dihadapkan pada pengguna ketika aplikasi

dimulai adalah aktivitas *Learn*. Selain itu pengguna dapat memilih menu yang tersedia. Bagan *activity diagram* dari aplikasi yang dibangun ditunjukkan oleh gambar 3.3.

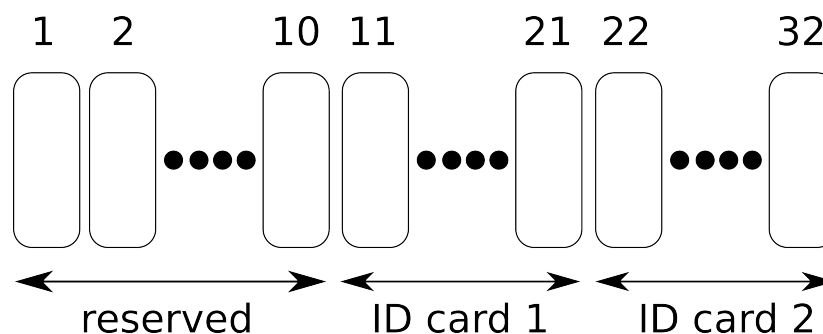


Gambar 3.3 Activity diagram aplikasi Syahfi

3.4 Rancangan Basis Data Sederhana

Pada aplikasi pada perangkat bergerak terdapat *RMS (Record Management Store)* yang merupakan basis data sederhana berorientasi *record (record-oriented database)*. Penyimpanan data pada RMS bersifat *non-volatile*, artinya data akan tetap tersimpan meskipun *MIDlet* tidak berjalan atau telepon seluler dimatikan. Pada RMS, data tersimpan dalam *record* yang masing-masingnya memiliki ID berupa integer yang sekaligus berfungsi sebagai *primary key* dan *array of bytes* untuk menyimpan data.

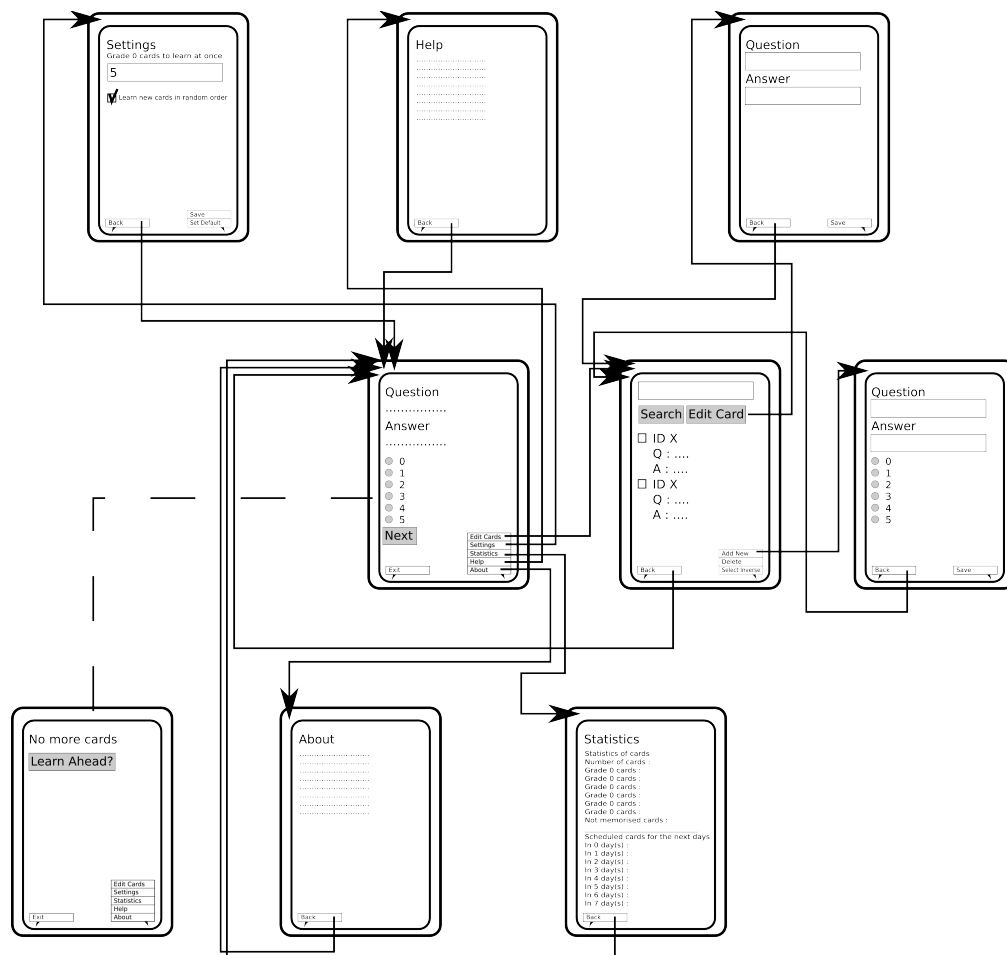
Aplikasi yang akan dibangun memiliki 1 (satu) buah *record store* yang diberi nama “SyahfiData”. Data mengenai *cards* akan dimasukkan mulai indeks ke-11, sedangkan 10 (sepuluh) indeks pertama dalam *record store* dipergunakan untuk keperluan penyimpanan preferensi aplikasi dan keperluan pengembangan aplikasi selanjutnya. Sebuah ID *record store* yang telah dihapus tidak akan dapat dipakai untuk menyimpan data. Oleh karena itu penghapusan *card* sebenarnya hanya mengubah nilai data menjadi *blank-space*. Hal ini selain untuk menjaga optimasi, juga membuat ukuran *record store* tidak terlalu membengkak. Gambaran posisi *record* ditunjukkan oleh gambar 3.4.



Gambar 3.4 Posisi *record* dan keperluannya

3.5 Rancangan User Interface

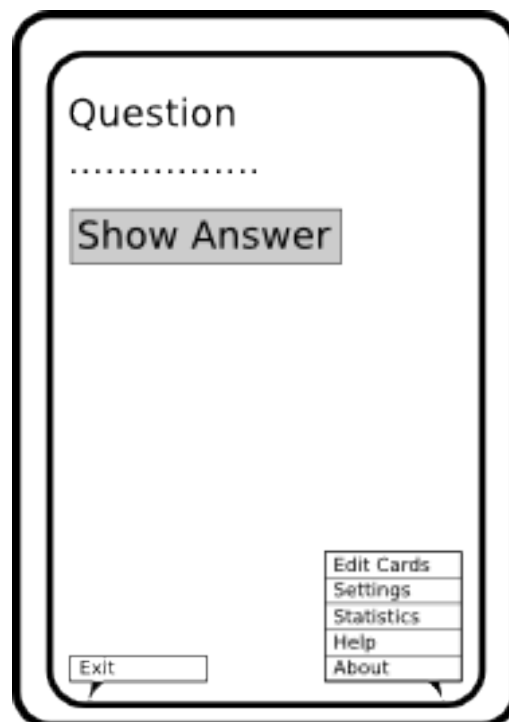
User interface atau antar muka dari aplikasi menjadi komponen penting karena merupakan jembatan yang memudahkan hubungan sistem dan pengguna. Perancangan antar muka didasari oleh perancangan UML dari aplikasi bersangkutan. Desain *interface* dari aplikasi pada perangkat bergerak berbeda dengan desain yang digunakan pada aplikasi untuk *PC* atau aplikasi *desktop*. Bagan antar muka pada aplikasi *flashcard* Syahfi ditunjukkan oleh gambar 3.5.



Gambar 3.5 Bagan user interface aplikasi Syahfi

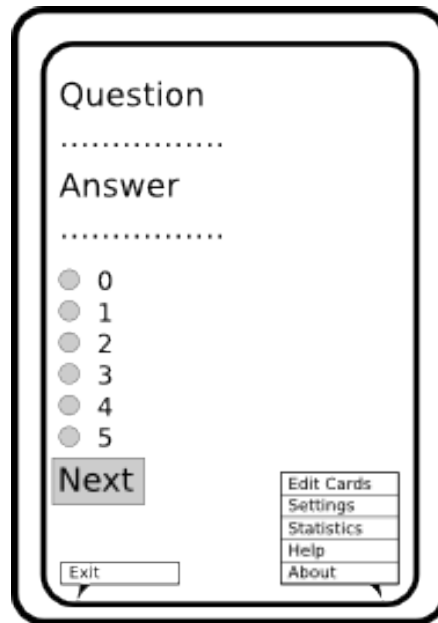
1. Antar muka halaman awal

Cards yang telah sebelumnya dimasukkan oleh pengguna akan diurut berdasarkan algoritma SM-2 dan ditampilkan hasilnya satu per satu. Desain antar muka untuk proses belajar ditunjukkan oleh gambar 3.6.



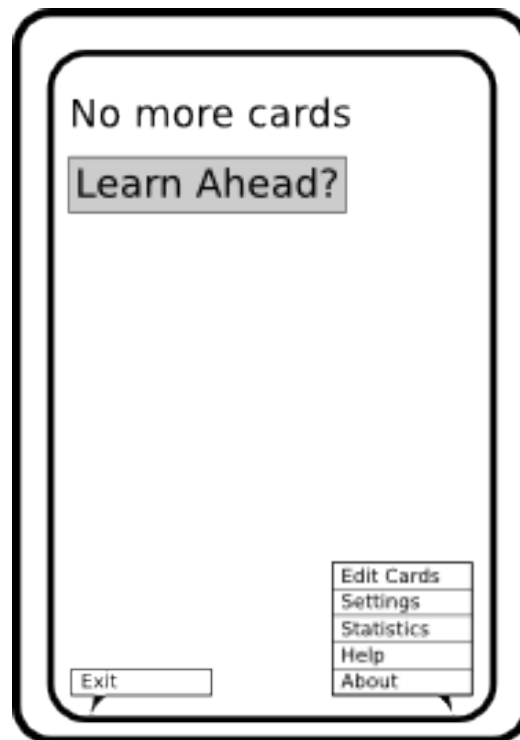
Gambar 3.6 Desain antar muka proses belajar

Selanjutnya bila pengguna menekan *button* “Show Answer”, maka jawaban dari pertanyaan bersangkutan akan ditampilkan beserta pilihan untuk menentukan *grade* dari *card*. Gambar 3.7 menunjukkan antar muka ketika *button* “Show Answer” ditekan.



Gambar 3.7 Desain antar muka proses belajar
setelah *button* “Show Answer” dipilih

Pengguna akan memberi *grade* pada *card* sesuai dengan tingkat kesulitannya. Setelah pengguna menekan *button* “Next” maka tampilan akan berubah kembali seperti gambar 3.6. Demikian seterusnya hingga tidak ada lagi *card* dalam *revisionQueue*. Hal itu menunjukkan bahwa sudah tidak ada lagi *card* yang dijadwalkan untuk hari itu. Pengguna lalu akan ditanya apakah ingin belajar *card* yang sebenarnya dijadwalkan untuk waktu mendatang. Tampilan dari *prompt* tersebut ditunjukkan seperti desain antar muka pada gambar 3.8.

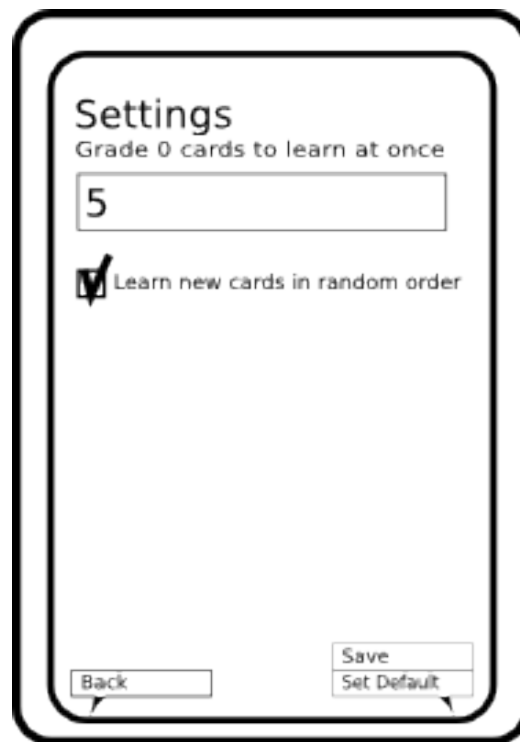


Gambar 3.8 Desain antar muka proses belajar ketika tidak ada lagi *scheduled cards*

Ketika *button* “Learn Ahead?” ditekan, maka aplikasi akan memasukkan *card* yang seharusnya dijadwalkan untuk masa mendatang dan tampilan antar muka aplikasi akan kembali tampak seperti gambar 3.6.

2. Antar muka *Setting*

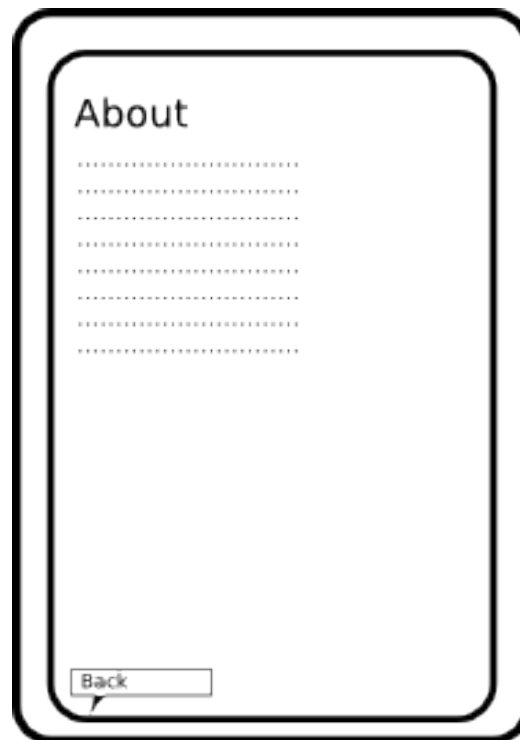
Menu *Setting* dapat dimunculkan dengan memilih menu *Setting* pada tampilan *Learn*. Halaman *Setting* memiliki satu buah input teks dan sebuah *checkbox*. Selain itu, halaman *Setting* juga memiliki menu “Set Default” yang berguna mengembalikan *setting* standar aplikasi. Desain antar muka untuk halaman *Setting* terlihat seperti pada gambar 3.9.



Gambar 3.9 Desain antar muka
halaman *Setting*

3. Antar muka *About*

Halaman *About* akan menampilkan penjelasan singkat tentang aplikasi, nama pengembang, serta kontak *e-mail* yang dapat dihubungi bila pengguna ingin memberi komentar. Desain antar muka untuk halaman *About* terlihat pada gambar 3.10.



Gambar 3.10 Desain antar muka
halaman *About*

4. Antar muka *Statistic*

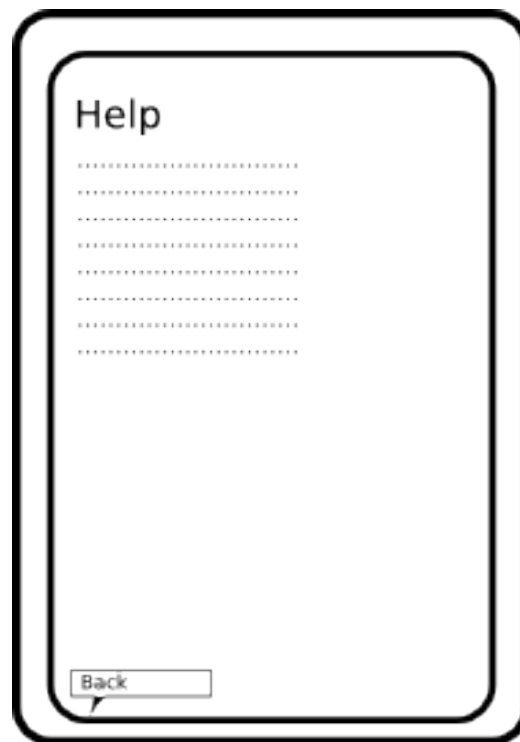
Menu *Statistic* dapat ditampilkan dari menu *Statistic* pada tampilan *Learn*. Halaman *Statistic* akan menampilkan statistik *card* yang telah dimasukkan serta jumlah *card* yang akan dipelajari mulai dari hari ini hingga satu minggu mendatang. Selain itu, informasi yang ditampilkan pada halaman *Statistic* adalah jumlah *card* yang belum diingat oleh pengguna. Gambar 3.11 adalah desain antar muka untuk halaman *Statistic*. Halaman *Statistic* dilengkapi sebuah *command button* untuk kembali ke menu *Learn* (menu utama).



Gambar 3.11 Desain antar muka
halaman *Statistic*

5. Antar muka halaman *Help*

Antar muka halaman *Help* menampilkan petunjuk singkat tentang cara penggunaan aplikasi Syahfi. Desain antar muka halaman *Help* terlihat pada gambar 3.12.



Gambar 3.12 Desain antar muka halaman *Help*

6. Antar muka halaman *Cards*

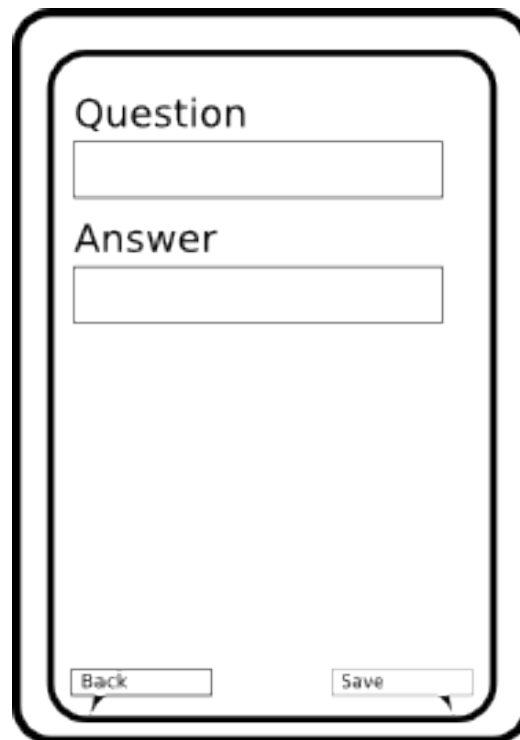
Ketika pengguna memilih menu “Edit Cards”, maka akan ditampilkan semua *card* yang telah dimasukkan. Informasi mengenai *card* yang ditampilkan hanya ID dari *card*, *question*, dan *answer*. Pada halaman *Cards* terdapat sebuah *text field* yang berguna untuk memasukkan kata kunci pada pencarian *card*. Kata kunci yang dimasukkan akan dibandingkan dengan setiap *question* dan *answer* dari *card* dan hasil pencarian akan ditampilkan.

Menu “Delete Selected” berfungsi untuk menghapus *card* dari *record store* setelah pengguna memberi centang pada *card* yang ingin dihapus. Desain antar muka untuk halaman *Cards* terlihat pada gambar 3.13.

The image shows a mobile application interface for managing cards. At the top, there is a text input field. Below it are two buttons: 'Search' and 'Edit Card'. The main area contains a list of cards. Each card entry starts with a checkbox, followed by 'ID X', and then two lines of text: 'Q :' and 'A :'. At the bottom of the screen, there is a navigation bar with four buttons: 'Back', 'Add New', 'Delete', and 'Select Inverse'.

Gambar 3.13 Desain antar muka
halaman *Cards*

Text field pada halaman *Cards* juga akan digunakan untuk mengedit *card*. Nilai dari teks yang dimasukkan pengguna harus berupa bilangan lebih dari 0. Selanjutnya pengguna akan menekan *button* “Edit Card” dan tampilan aplikasi akan berubah. Desain antar muka untuk pengeditan *card* terlihat pada gambar 3.14.



Question

Answer

Back Save

Gambar 3.14 Desain antar muka ketika pengguna akan mengedit sebuah *card*

Pada halaman *Cards* juga terdapat menu “Add New” bila pengguna ingin menambah *card* baru. Pengguna dapat menginisiasi *grade* dari *card* ketika menambahkan *card* baru. Desain antar muka untuk menambah *card* baru terlihat pada gambar 3.15.

Question

Answer

Grade

☐ 0

☐ 1

☐ 2

☐ 3

☐ 4

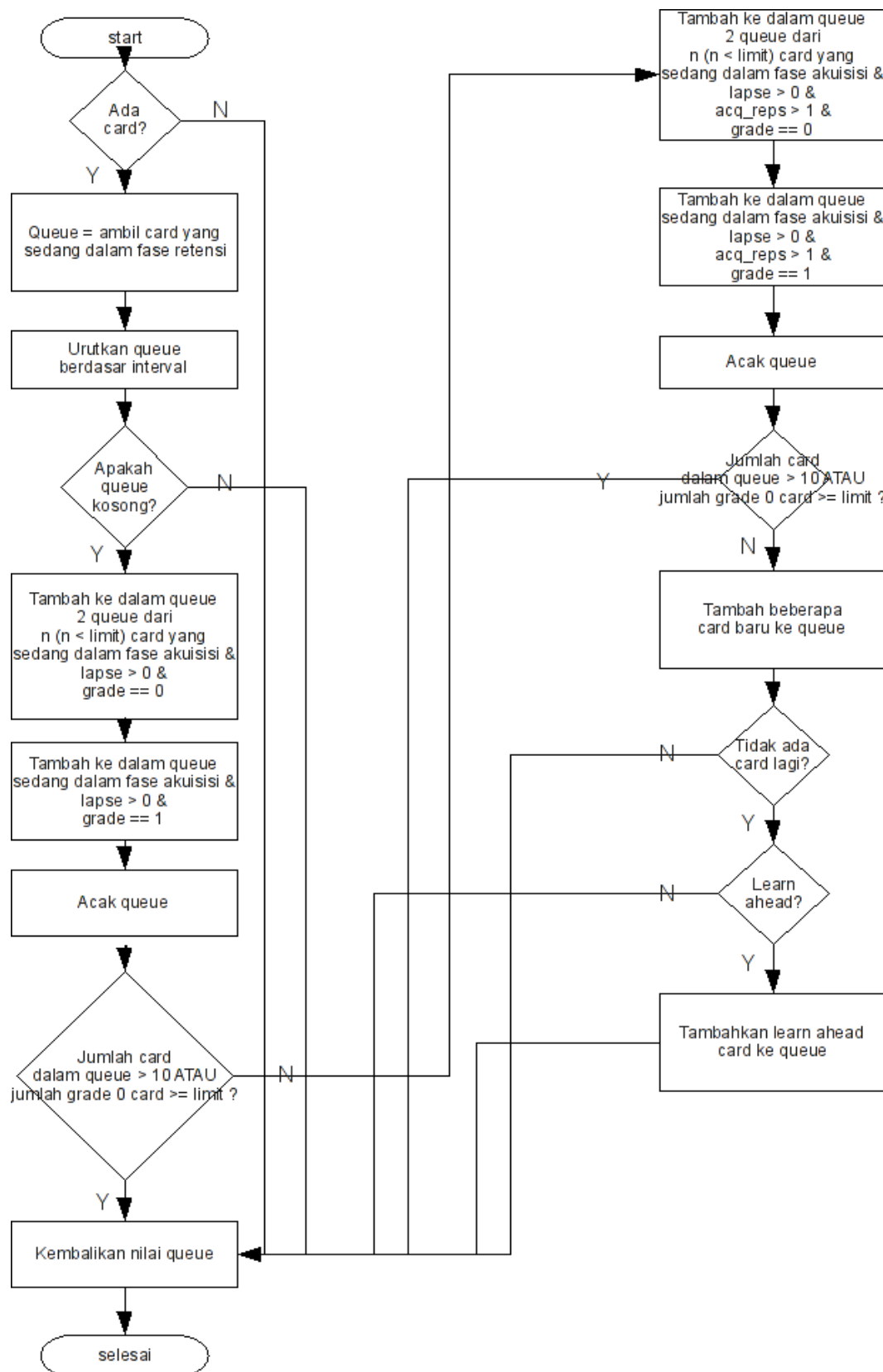
☐ 5

Back Save

Gambar 3.15 Desain antar muka menu
“Add New” *card*

3.5 Algoritma untuk *Rebuild Revision Queue* dan *Process Answer*

Algoritma untuk *Rebuild Revision Queue* adalah algoritma yang digunakan dalam aplikasi untuk menentukan antrian atau *queue* dari *card* yang akan dipelajari pengguna. Sedangkan algoritma untuk *Process Answer* adalah algoritma yang digunakan dalam menentukan interval serta beberapa atribut *card* lain (semisal *easiness factor*) dengan berpatokan pada *new grade* yang diberikan pengguna. Implementasi dari algoritma ini adalah metode `rebuildRevisionQueue()` dan `processAnswer()` pada *class Store*. *Flowchart* algoritma *Rebuild Revision Queue* ditunjukkan oleh gambar 3.16.



Gambar 3.16 Flowchart algoritma untuk Rebuild Revision Queue

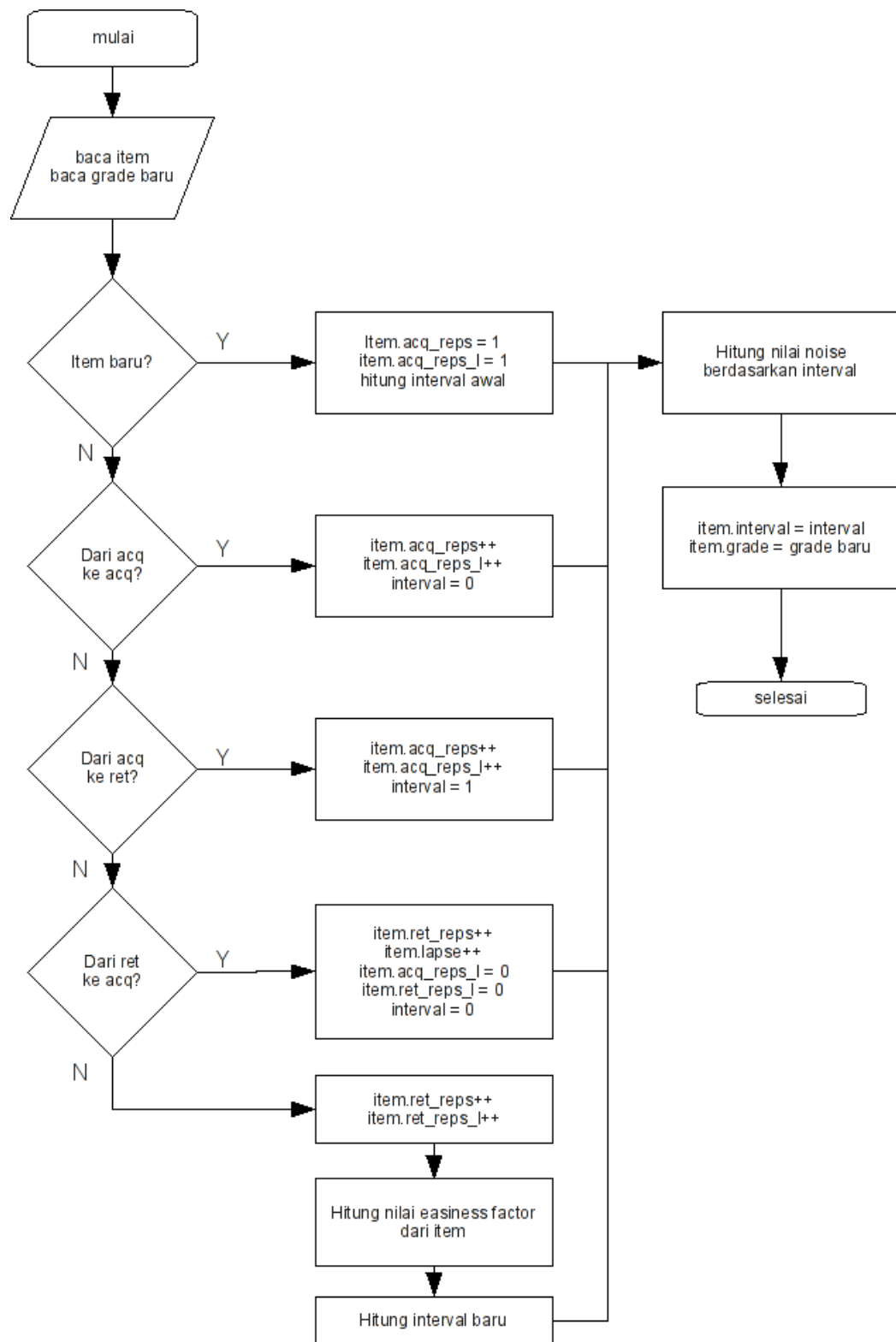
Pertama-tama *queue* akan dibuat dari *card* yang memenuhi syarat retensi atau “due for retention repetition”, yaitu *card* yang dijadwalkan untuk hari ini atau hari yang sudah lalu dan nilai *grade*-nya lebih dari sama dengan 2. Bila dalam *queue* sudah ada *card*, maka *card* dalam *queue* akan dipelajari pengguna.

Bila *queue* belum berisi *card*, maka prioritas selanjutnya adalah *card* yang memiliki *grade* 0 dan *grade* 1 atau dengan kata lain *card* sedang berada dalam fase akuisisi. Selanjutnya urutan dalam *queue* akan diacak. Bila sudah terdapat *card* dalam *queue*, maka proses belajar dapat dijalankan.

Bila dalam *queue* belum terdapat lebih dari 10 *card* atau jumlah *card* dengan *grade* 0 tidak lebih dari limit pada preferensi pengguna, maka prioritas selanjutnya adalah *card* yang memiliki nilai *acquisition repetitions* lebih dari 1. Selanjutnya *queue* akan diacak urutannya.

Bila dalam *queue* belum terdapat lebih dari 10 *card* atau jumlah *card* dengan *grade* 0 tidak lebih dari limit pada preferensi pengguna, maka *card* baru atau *new cards* akan ditambahkan ke *queue*. Bila tetap masih belum memenuhi, maka pengguna akan dihadapkan pada pilihan untuk mempelajari *card* yang seharusnya dijadwalkan di masa mendatang. Kondisi ini disebut “Learn Ahead”. Bila pengguna menghendaki untuk “learn ahead” maka semua *card* yang memenuhi syarat “learn ahead” akan dimasukkan ke dalam *queue*.

Gambar 3.17 adalah *flowchart* algoritma untuk *Process Answer*.



Gambar 3.17 Flowchart penentuan interval dari card

Langkah awal adalah mengambil nilai *grade card* yang baru dari pengguna untuk selanjutnya dijadikan parameter dalam penentuan interval *card*. Sebuah *card* akan diperiksa apakah termasuk *new card* dengan ditandai nilai *acq_reps* dan *ret_reps* yang sama dengan 0 (*card* belum dipelajari sama sekali). Bila sebuah *card* adalah *card* baru, maka nilai intervalnya ditentukan dari *grade* yang diberikan pengguna. Nilai *acq_reps* dan *acq_reps_l* dari *card* akan bernilai 1.

Bila *card* sebelumnya berada pada fase akuisisi (nilai *grade* antara 0 atau 1) dan *grade* baru berkisar 0 sampai 1 (berarti *card* tetap berada pada fase akuisisi), maka nilai interval yang baru sama dengan 0. Nilai *acq_reps* dan *acq_reps_l* dinaikkan masing-masing sebesar 1.

Bila *card* berpindah dari fase akuisisi ke fase retensi (nilai *grade* yang baru berkisar antara 2 hingga 5), maka nilai interval yang baru sama dengan 1. Nilai *acq_reps* dan *acq_reps_l* dinaikkan masing-masing sebesar 1.

Bila *card* berpindah dari fase retensi ke fase akuisisi (nilai *grade* yang baru turun menjadi 0 atau 1), maka nilai interval yang baru sama dengan 0. Kondisi ini disebut dengan istilah *lapse*. Nilai *ret_reps* dan *lapses* dari *card* akan dinaikkan masing-masing sebesar 1. Nilai *acq_reps_l* dan *ret_reps_l* diubah menjadi 0.

Bila sebuah *card* tetap berada pada fase retensi (nilai *grade* yang baru dan lama berada pada kisaran 2 hingga 5), maka nilai interval yang baru dihitung berdasarkan nilai *easiness factor* dari *card*. Nilai *ret_reps* dan *ret_reps_l* dari *card* akan dinaikkan masing-masing sebesar 1. Nilai *easiness factor* dihitung berdasarkan *grade* baru yang diberikan pengguna.

Langkah selanjutnya adalah menentukan interval lebih lanjut dengan menghitung *noise*. *Noise* adalah sebuah bilangan acak yang memiliki parameter interval. Untuk nilai interval sama dengan 0 maka nilai *noise* sama dengan 0. Selain itu, nilai *noise* tidak dapat ditentukan karena sifatnya yang acak. Kegunaan *noise* adalah untuk membuat interval antar *card* menjadi beragam.