PENGEMBANGAN APLIKASI PENDUKUNG PEMBELAJARAN BAHASA ARAB MENGGUNAKAN ORIENTASI OBJEK, PENGGUNAAN ULANG DAN POLA-POLA PERANCANGAN: KASUS PERUBAHAN KATA KERJA

SKRIPSI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh:
RAVIQUL HAIDIR FRANSCASMARA
NIM. 0910680086

KEMENTERIAN RISET TEKNOLOGI DAN PENDIDIKAN TINGGI
UNIVERSITAS BRAWIJAYA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER
MALANG
2015

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

LEMBAR PERSETUJUAN

PENGEMBANGAN APLIKASI PENDUKUNG PEMBELAJARAN BAHASA ARAB MENGGUNAKAN ORIENTASI OBJEK, PENGGUNAAN ULANG DAN POLA-POLA PERANCANGAN: KASUS PERUBAHAN KATA KERJA

SKRIPSI

LABORATORIUM REKAYASA PERANGKAT LUNAK

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh:

RAVIQUL HAIDIR FRANSCASMARA

NIM. 0910680086

Telah diperiksa dan disetujui oleh:

Pembimbing I	Pembimbing II,
Ismiarta Aknuranda, S.T., M.Sc, Ph.D.	**Budi Darma Setiawan, S.Kom., M.Cs**
NIK. 740719 06 1 1 0079	**NIP. 19841015 201404 1 002**

LEMBAR PENGESAHAN

PENGEMBANGAN APLIKASI PENDUKUNG PEMBELAJARAN BAHASA ARAB MENGGUNAKAN ORIENTASI OBJEK, PENGGUNAAN ULANG DAN POLA-POLA PERANCANGAN: KASUS PERUBAHAN KATA KERJA

SKRIPSI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun oleh :

RAVIQUL HAIDIR FRANSCASMARA
NIM. 0910680086

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 26 Juni 2015

Penguji I	Penguji II
Tri Astoto Kurniawan, ST, MT, Ph.D.	**Fajar Pradana, S.ST, M.Eng**
NIP. 197105182003121001	**NIK. 87112116110371**

Penguji III

Aryo Pinandito, S.T., M.MT.
NIP. 198305192014041001

Mengetahui
Ketua Program Studi Informatika / Ilmu Komputer

Drs. Marji, M.T.
NIP. 19670801 199203 1 001

PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 30 Juli 2015
Mahasiswa,

Raviqul Haidir Franscasmara
NIM. 0910680086

Kata Pengantar

Puji syukur kehadirat Allah SWT yang selalu memberikan rahmat, hidayah dan cahaya-Nya sehingga penulis dapat menyelesaikan Skripsi dengan judul "Pengembangan Aplikasi Pendukung Pembelajaran Bahasa Arab Menggunakan Orientasi Objek, Penggunaan Ulang dan Pola-pola Perancangan : Kasus Perubahan Kata Kerja". Skripsi ini diajukan sebagai salah satu syarat untuk mendapatkan gelar sarjana S-1 di Fakultas Ilmu Komputer Universitas Brawijaya.

Keberadaan skripsi ini tidak terlepas dari bimbingan dan bantuan dari berbagai pihak, untuk itu pada kesempatan ini penulis menyampaikan rasa terima kasih dan perhargaan sebesar- besarnya kepada:

1. Ismiarta Aknuranda, ST., M.Sc, Ph.D., selaku dosen pembimbing I yang telah memberikan ilmu dan saran untuk penelitian skripsi ini.
2. Budi Darma Setiawan, S.Kom., M.Cs., selaku dosen pembimbing II yang telah memberikan ilmu dan saran untuk penelitian skripsi ini.
3. Suprapto, ST., MT., selaku dosen pembimbing akademik yang telah memberikan pengarahan selama penulis menempuh pendidikan di Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
4. Drs. Marji, MT., selaku Ketua Program Studi Informatika/Ilmu Komputer Universitas Brawijaya.
5. Ir. Sutrisno, MT. selaku Ketua Program Teknologi Informasi dan Ilmu Komputer (PTIIK)
6. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya khususnya pada Pak Tri Astoto dan Ibu Indriati atas diskusi berkaitan dengan konsep Design Patterns.
7. Orang tua saya, Drs. Arifin, M.Pd.I dan Tatik Nuriyati, S.Pd. dan saudara kandung saya Vinora B.Anam., A. Dzulfikar Jauhari dan Luqman B.Putra. yang telah memberikan dukungan moral dan material yang tak ternilai jumlahnya.

8. Keluarga besar di Malang khususnya yang sudah saya anggap sebagai orang tua, Nur Saudah, S.Pd dan Drs. Tugija dan menjadi saudara Mufida Cahyani, Wahyu I. Fakhrudin dan N. F. Aziza.
9. Atas dukungan, motivasi serta banyak hal dari teman-teman saya yang tergabung dalam "ASTA TEAM" yaitu Bintang K. Akbar, Ichsan Wahyudi, Adien F.Habieb, Dani Prasnanto, Gery E.Angriawan, Reviangga D.Pratama, dan yang terakhir serta utama karena bersama-sama berjuang bangkit menyelesaikan tugas skripsi yang cukup berliku, Galih Julianto Purnomo.
10. Segenap teman-teman yang tergabung dalam grup "Penghuni Lab KCV" yang telah memberikan lingkungan "Kondusif" dalam mengerjakan Skripsi diantaranya Melynda, Tika Rahmadian, Anisa , Ervin Yohanes, Hendro PS., Putu Arya, Sufia A. Putri, Tian GKS dan Septa Hendra.
11. Bagi gerombolan mahasiswa yang tergabung di "TPL Legend 2009" khususnya G.'Dude' Wisudawan, G.'Cibi' Arief, Rizky 'Robot', Dw intaa Ayu, Pak Yuri Pratama, Raka nta Rifky, Luthfi Aziz, Arief 'ayiz', Bayu Jimi, Alfian 'Mbah', Bayu Nursito, Anom, Abet, Ading, Fandisya dan Anita Anta.
12. Bagi Senior dan Junior PTIIK yang telah memberikan saran dan *sharing* ilmu.
13. Ustadz Islahudin dan Ustadz Abdul Jalil yang telah berbagi, meluangkan waktu dan membantu mengenai seluk beluk Ilmu Sharaf.
14. Semua pihak yang telah membantu terselesaikannya penyusunan skripsi ini yang tidak dapat disebutkan satu per satu.

Penulis sadar bahwa penelitian skripsi ini masih banyak kekurangan, oleh karena itu kritik dan saran yang bersifat membangun sangat diharapkan untuk menyempurnakan penelitian skripsi ini. Penulis berharap skripsi ini dapat bermanfaat khususnya bagi diri sendiri dan bagi semua pihak.

Malang, Juli 2015

Penulis

repository.ub.ac.id

ABSTRAK

Raviqul Haidir F. 2015. : Pengembangan Aplikasi Pendukung Pembelajaran Bahasa Arab Menggunakan Orientasi Objek, Penggunaan Ulang dan Pola-Pola Perancangan : Kasus Perubahan Kata Kerja. Skripsi Program Studi Informatika/Ilmu Komputer, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya. Dosen Pembimbing : Ismiarta Aknuranda, S.T., M.Sc., Ph.D. dan Budi Darmawan, S.Kom., M.Cs.

Umat muslim dalam mempelajari Al-Qur'an membutuhkan keahlian bahasa Arab pada tingkatan tertentu. Ilmu Sharaf merupakan salah satu cabang ilmu bahasa Arab yang mempelajari morfologi kata dan menjadi dasar sebelum mempelajari cabang ilmu yang lebih lanjut. Proses pembelajaran ilmu bahasa Arab di Indonesia masih banyak menggunakan cara yang konvensional dan tradisional yang dianggap sulit, sementara Indonesia merupakan negara non-Arab dengan jumlah penduduk muslim yang besar di dunia. Dengan memanfaatkan kemajuan teknologi informasi, seperti penggunaan perangkat lunak, maka diharapkan bisa mengurangi permasalahan yang berhubungan dengan proses pembelajaran tersebut. Pengembangan perangkat lunak pada penelitian ini mencoba memodelkan aturan sistematik ilmu sharaf pada kasus perubahan kata kerja memanfaatkan *design patterns strategy* dan *facade* di tahap perancangan aplikasi. Tahapan implementasi menggunakan bahasa pemrograman Java dengan memanfaatkan API JQuranTree, Hibernate Search dan JavaFX dan nanti menghasilkan *desktop application*. Dari hasil pengujian fungsional, aplikasi mampu menjalankan 72% dari total kasus uji yang dibuat berdasarkan spesifikasi kebutuhan sistem. Berdasarkan pengujian *Usability* yang dilakukan oleh seorang ahli bahasa Arab, aplikasi pendukung pembelajaran ini membutuhkan perbaikan pada faktor kemudahan navigasi menu, penyediaan detail informasi kata kerja dengan arti bahasa indonesia, penggunaan istilah yang lebih tepat pada kata ganti subyek, ketersediaan kata kerja secara lengkap sesuai kamus Arab dan penggunaan bahasa instruksi serta animasi yang lebih interaktif.

Kata Kunci : bahasa Arab, *desktop application*, sharaf

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

ABSTRACT

Raviqul. Haidir F. 2015. : Development of An Arabic Language Learning Support Application using Object Orientation, Reuse and Design Patterns : Cases of Verb Conjugation. Department of Computer Science and Information Technology, University of Brawijaya. Advisors : Ismiarta Aknuranda, S.T., M.Sc., Ph.D. and Budi Darmawan, S.Kom., M.Cs.

Muslims require a certain degree of understanding and proficiency in Arabic language for learning Al-Quran. Sharaf is a branch in Arabic language studies which deals with morphology and also a foundation of other branches in Arabic language. In Indonesia many learning activities in Arabic language still use conventional and traditional method, while Indonesia is a non Arabic nation with muslim majority . The use of information technology, particularly a software application, is expected to help overcome this problem. This research sought to model the systematical rules in sharaf, particularly the verb conjugation by using strategy and facade design patterns in design activity. The software was implemented in Java programming language by reusing available APIs such as JQuranTree, Hibernate Search and JavaFX, which resulted in a desktop application. The result of functional testing of the software showed that 72% of the sytem functionality were satisfied. Based on usability testing which was done by an Arabic lecturer, this software application needs improvement in menu navigation ease of use, details of verb information with Indonesian translation, use of proper Arabic terms, availability of complete list of words similar to those in an Arabic dictionary, use of more interactive animations and more appropriate use of language instructions.

***Keywords**: Arabic languange, Desktop application, Sharaf*

UNIVERSITAS BRAWIJAYA

Daftar Isi

Daftar Gambar

Daftar Tabel

Pedoman Transliterasi Arab-Latin

Berdasarkan Keputusan Bersama Menteri Agama dan Menteri P dan K

Nomor : 158 Tahun 1987 – Nomor : 0543 b/u/1987

Konsonan

No	Arab	Latin
1	ا	tidak berlambang
2	ب	b
3	ت	t
4	ث	ṡ
5	ج	j
6	ح	ḥ
7	خ	kh
8	د	d
9	ذ	ż
10	ر	r
11	ز	z
12	س	s
13	ش	sy
14	ص	ṣ
15	ض	ḍ

No	Arab	Latin
16	ط	ṭ
17	ظ	ẓ
18	ع	‘
19	غ	g
20	ف	f
21	ق	q
22	ك	k
23	ل	l
24	م	m
25	ن	n
26	و	w
27	ه	h
28	ء	’
29	ي	y

Vokal

Vokal pendek		Vokal panjang		Diftong	
Arab	latin	Arab	latin	Arab	Latin
ـَ	a	آ	ā	اَوْ	au
ـِ	i	اِيْ	ī	اَيْ	ai
ـُ	u	اُوْ	ū		

BAB I
PENDAHULUAN

1.1 Latar belakang

Umat Islam membutuhkan proses belajar dalam memahami bahasa Arab baik secara baca maupun tulis pada tingkat tertentu untuk bisa mempelajari Al-Qur'an. Al-Qur'an merupakan kitab suci bagi umat Islam yang diyakini sebagai perantara wahyu dari Tuhan kepada Nabi Muhammad dengan menggunakan bahasa Arab sebagai media bahasa. Umat Muslim umumnya hanya belajar pada kemampuan membaca dan menulis Al-Qur'an tanpa mempelajari kaidah bahasa Arab secara utuh padahal dengan mempelajarinya umat muslim bisa lebih memperdalam pemahaman terhadap kitab Al-Qur'an.

Berdasarkan data yang dirilis oleh organisasi Pew Research Center menunjukan bahwa jumlah penganut Islam didunia sekitar 1,57 miliar jiwa pada tahun 2010 [PEW-11]. Sekitar 20 % dari total tersebut berada di negara-negara yang menggunakan bahasa Arab sebagai bahasa resmi negara. Berarti terdapat sekitar 80% populasi muslim yang hidup di negara non-Arab yang dapat diasumsikan membutuhkan proses belajar bahasa Arab pada tingkatan tertentu.

Bahasa Arab secara morfologi dapat digolongkan salah satu bahasa yang paling kompleks dan kaya [ALS-04]. Hal ini dikarenakan puluhan bahkan ratusan kata dengan makna yang berbeda dapat dihasilkan oleh satu akar kata yang sama melalui beberapa pola perubahan dan penambahan imbuhan. Berdasarkan sumber tersebut di sisi lain bahasa Arab juga memiliki tingkat ambiguitas yang cukup tinggi karena secara natural bahasa Arab tidak menggunakan tanda baca (*diacritics/vowels*) dan juga ada beberapa bagian akar kata yang mirip atau sama dengan imbuhan yang digunakan. Oleh karena itu terdapat ilmu sharaf yang menjawab permasalahan tersebut.

Ilmu sharaf merupakan salah satu cabang ilmu tata bahasa Arab yang membahas tentang perubahan suku kata menyangkut penambahan, penggantian, dan perubahannya [ANW-11]. Menurut pengamatan dan wawancara yang dilakukan pada pengajar bahasa Arab, ilmu sharaf umumnya masih diajarkan secara manual dimana para siswa diinstruksikan untuk membunyikan, menghafalkan dan menulis perubahan kata kerja pada buku atau papan tulis sehingga belum banyak mengoptimalkan penggunaan teknologi informasi. Kaidah perubahan suku kata dalam ilmu sharaf memang cukup kompleks jika dibandingkan dengan kaidah bahasa Indonesia namun kaidah tersebut tetap memiliki aturan sistematik yang jika diamati dapat dimodelkan dan diimplementasikan dalam sebuah aplikasi perangkat lunak untuk pembelajaran bahasa Arab.

Selama ini belum ada perangkat lunak yang secara luas digunakan untuk membantu proses pembelajaran dalam ilmu sharaf. Memang terdapat beberapa aplikasi yang ditemukan di Internet akan tetapi aplikasi tersebut, yaitu SARF [SAR-14] dan Quthrub [QUT-14] memiliki keterbatasan seperti sulit dikembangkan karena tidak bersifat *open-source* atau juga ditujukan pada pengguna berbahasa tertentu saja. Oleh karena itu, dalam penelitian ini dilakukan pengembangan perangkat lunak pembelajaran bahasa Arab ilmu sharaf yang disesuaikan dengan pengguna Indonesia.

Pengembangan perangkat lunak yang mampu secara lengkap mendukung pembelajaran ilmu sharaf disadari memiliki tingkat kompleksitas yang tinggi sehingga diperlukan pengembangan yang dilakukan secara iteratif dan bertahap. Oleh karena itu perangkat lunak yang akan dikembangkan adalah purwarupa awal yang mampu mendemonstrasikan latihan perubahan kata kerja secara sederhana. Pemodelan dan pemrograman berorientasi objek serta penggunaan *Design Pattern* yang tepat dianggap sesuai untuk jenis pengembangan ini sehingga nantinya ketika ada peneliti lain yang ingin membahas topik penelitian yang sama bisa dengan relatif mudah meneruskan pengembangan aplikasi ini. Hal ini sesuai dengan prinsip penggunaan ulang dimana pengembangan aplikasi saat ini menggunakan komponen yang telah tersedia untuk mempersingkat proses implementasi dan aplikasi tersebut dapat digunakan ulang untuk pengembangan selanjutnya.

1.2 Rumusan Masalah

1. Bagaimana alur untuk mendemonstrasikan pembelajaran perubahan kata kerja ilmu sharaf secara sederhana ?
2. Bagaimana memodelkan kata kerja dan prinsip-prinsip perubahan kata kerja ke dalam model desain sebuah perangkat lunak berorientasi objek yang memanfaatkan pola-pola perancangan (*Design Patterns*) ?
3. Bagaimana mengembangkan prototipe aplikasi perangkat lunak untuk pembelajaran bahasa Arab pada *platform Java* sesuai dengan model perubahan kata kerja?
4. Bagaimana menguji fungsionalitas dan *usability* purwarupa aplikasi pendukung pembelajaran bahasa Arab ?

1.3 Tujuan

Mengembangkan sebuah purwarupa aplikasi perangkat lunak pendukung pembelajaran bahasa Arab Qur'an yang mampu mendemonstrasikan perubahan bentuk dari suatu kata kerja sesuai dengan kaidah ilmu *Sharaf* .

1.4 Batasan Masalah

Dari rumusan masalah yang telah disampaikan maka penelitian ini difokuskan dengan batasan sebagai berikut:

1. Implementasi Aturan Ilmu Sharaf kata kerja dengan jumlah huruf 3 (tiga) atau dalam istilah bahasa Arab disebut *fiil Tsulasy* dan diluar kata yang mengandung huruf *Illat* (huruf lemah) atau kata dasar yang tidak teratur.
2. Aturan yang diimplementasikan Aplikasi adalah Aturan Dasar pada bentuk *Fiil Madhi, Fiil Mudhori, Fiil Amr* dan *Damir*.

3. Skenario demonstrasi latihan yang digunakan berdasarkan pengamatan cara pembelajaran ilmu sharaf yang konvensional.
4. Kata kerja yang digunakan adalah dari Bahasa Arab Qur'an (*Classical Arab*).
5. Aplikasi ini nantinya berupa *Desktop Application* yang dikembangkan dengan *platform Java*.

1.5 Manfaat

1. Bagi pengajar, aplikasi yang dikembangkan ini dapat menjadi media pembelajaran yang mendukung proses pengajaran secara manual.
2. Bagi pelajar, aplikasi ini bisa menjadi media pembelajaran yang mendukung penguasaan ilmu sharaf di institusi pendidikanya.
3. Bagi pembelajar mandiri/otodidak, aplikasi ini bisa digunakan untuk pembelajaran ilmu sharaf secara otodidak .
4. Bagi peneliti / Pengembang Perangkat Lunak, aplikasi ini bisa digunakan sebagai dasar pengembangan aplikasi perangkat lunak lebih lanjut pada ilmu yang berkaitan dengan tata bahasa Arab seperti ilmu nahwu dan sejenisnya.

1.6 Sistematika Penulisan

Sistematika penulisan dalam skripsi ini sebagai berikut:

BAB I Pendahuluan

Bab ini memuat latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, dan sistematika penulisan.

BAB II Landasan Kepustakaan

Bab ini menguraikan tentang dasar teori dan referensi yang mendasari pengembangan, perancangan, implementasi, dan pengujian aplikasi.

BAB III Metodologi Penelitian

Bab ini membahas metode yang digunakan dalam penelitian yang terdiri dari studi literatur, analisis persyaratan, perancangan, implementasi sistem perangkat lunak, pengujian dan analisis, serta penulisan laporan.

BAB IV Analisis Persyaratan dan Perancangan

Bab ini membahas analisis persyaratan dan perancangan aplikasi pendukung pembelajaran bahasa Arab.

BAB V Implementasi

Bab yang membahas implementasi berbasis *desktop* sesuai dengan perancangan sistem yang telah dibuat.

BAB VI Pengujian dan Analisis

Bab yang memuat proses dan hasil pengujian terhadap sistem yang telah direalisasikan.

BAB VII Penutup

Bab yang memuat kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunak yang dikembangkan dalam skripsi ini serta saran – saran untuk pengembangan lebih lanjut.

BAB II
LANDASAN KEPUSTAKAAN

Pada Bab ini akan dijelaskan mengenai konsep teori yang dijadikan landasan dalam pengembangan aplikasi ini . Teori yang dibahas adalah dasar-dasar Ilmu Sharaf, *Library* JQuranTree, Hibernate, JavaFX , *Design Patterns* dan *evolutionary process model : prototyping*.

2.1 Sharaf

Sharaf atau tashrif secara bahasa (etimologi) artinya adalah "perubahan" adapun secara istilah (terminologi) sharaf adalah ilmu yang membahas tentang tata cara merubah kata dasar dalam bahasa Arab menjadi kata yang berbeda-beda karena menghendaki makna yang berbeda pula [ANW-12]. Menurut Ulama Basrah asal kata itu adalah *masdar* atau kata benda sedangkan menurut Ulama Kufah asal kata itu adalah *fi'il madhi* atau kata kerja lampau.

Secara umum Kata dalam bahasa Arab dibagi menjadi 3 jenis :

a. *Fi'il* : Kata kerja
b. *Isim* : Kata benda
c. *Harf* : Kata hubung / pelengkap bagi fiil dan Isim dalam sebuah kalimat

Istilah –istilah dasar yang sering digunakan dalam ilmu sharaf :

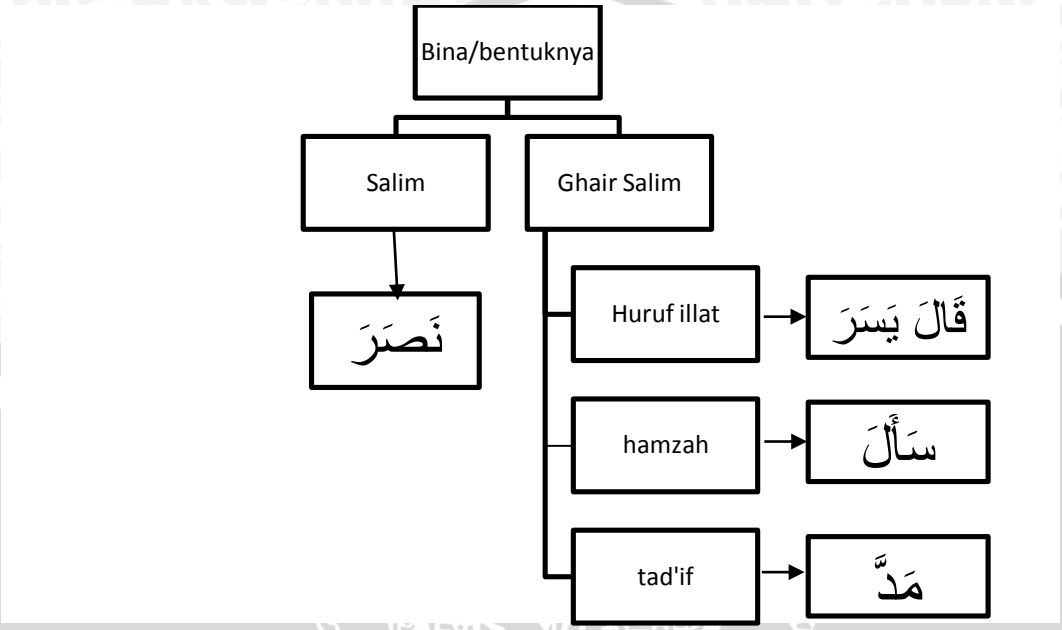
a. *Shighat* : Sebagaimana disinggung diatas bahwa sharaf adalah perubahan dari satu bentuk kata dasar menjadi beberapa bentuk yang lain. Kata dasar dalam bahasa Arab adalah Fi'il madhi yang terdiri dari 3 atau 4 huruf, dari kata dasar tersebut bisa dirubah menjadi beberapa bentuk lain yang disebut dengan shighat, adapun shighat (bentuk) tersebut adalah:
   - *Fi'il Madhi* : Kata kerja lampau
   - *Fi'il Mudhori'* : Kata kerja sekarang / masa depan
   - *Fi'il Amr* : Kata kerja perintah
b. Wazan : Rumus baku kata dalam bahasa Arab yang dijadikan acuan, dimodelkan dalam 3 huruf yaitu fa (ف), ain(ع) dan lam (ل)

c. Mauzun : Jika wazan adalah rumusnya, maka mauzun adalah kata yang dibandingkan dan disandingkan dengan wazan.

Fi'il itu ada yang asal hurufnya 3 (Tsulasy) dan 4 (ruba'i) . Dari kedua macam fi'il itu terbagi lagi menjadi yang kosong dari tambahan huruf (mujarrad) dan yang ada tambahan satu, dua atau tiga huruf (mazied) seperti yang terlihat pada gambar 2.1.

Gambar 2.1 Pembagian Fi'il berdasar jumlah huruf

Fi'il juga dibagi berdasarkan bentuk dan huruf yang menyusun kata tersebut menjadi dua yaitu mengandung huruf *Illat* (ا و ي) atau hamzah (ء) atau huruf ganda (tad'if) disebut Ghair Salim dan tidak mengandung huruf Illat atau sebanding dengan pola wazan (فَعَلَ)disebut Salim seperti pada gambar 2.2.

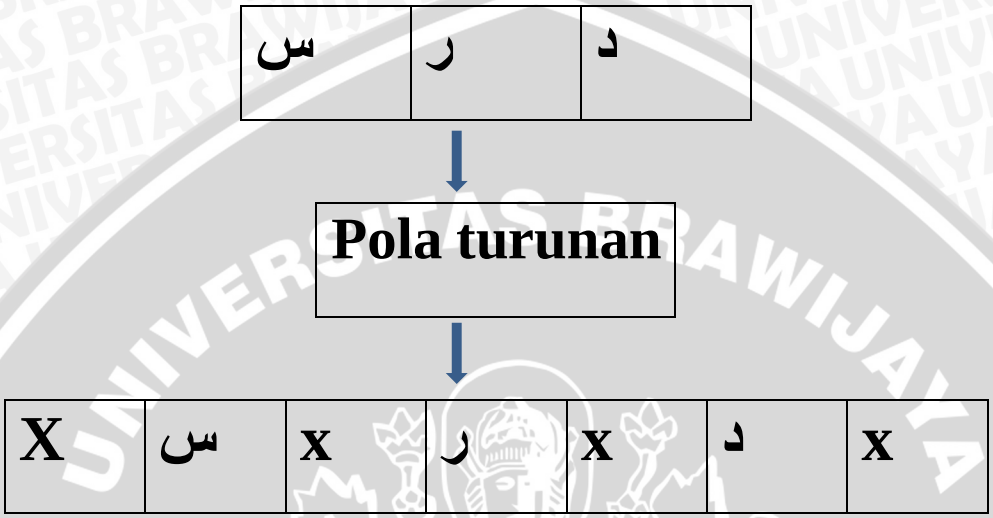


Gambar 2.2 Bagan pembagian jenis Fi'il berdasar bentuk

Aturan dasar dalam bahasa Arab Qur'an adalah hampir semua kata diturunkan dari pola aturan 3 akar huruf (*triliteral*) atau 4 akar huruf (*quadriliteral*). Huruf Arab fa ain lam (ف ع ل) digunakan sebagai rujukan bentuk dasar/model dari setiap kata kerja. Akar kata dalam bahasa Arab menyampaikan makna dasar yang kemudian dapat dikembangkan pada konsep semantik yang lebih kompleks untuk diturunkan baik itu menjadi kata kerja ataupun kata benda. Berdasarkan sistem derivasi ini kata benda dan kata kerja bisa memiliki sampai dengan 14 atau 15 bentuk berbeda walaupun umumnya hanya berjumlah 10 bentuk.

Sebagai contoh dapat diambil akar kata Dal Ra Sin (د ر س) yang mempunyai arti dasar "belajar". Dengan menambahkan huruf pada 3 huruf dasar tersebut pada sebelum, ditengah maupun diakhir huruf dasar maka bisa menghasilkan makna yang lebih kompleks seperti sekolah, guru, dan pelajaran.

Pada ilustrasi Gambar 2.3 ‘x’ adalah huruf yang bisa ditambahkan pada 3 huruf dasar. Huruf tambahan ini tidak harus semuanya harus terisi bersamaan. Di beberapa bentuk yang lain huruf dasar bisa diduplikasi atau penambahan tanda baca.



Gambar 2.3 Pola Turunan dalam kata kerja bahasa Arab

Pola turunan dipengaruhi oleh banyak faktor seperti jumlah huruf kata dasar, jenis huruf yang menyusun kata dasar tersebut, tanda baca yang melekat pada huruf, jenis fiil atau isim yang menjadi bentuk perubahan, jenis subyek yang menjadi property kata tersebut dan sebagainya. Kaidah perubahan kata kerja dalam ilmu sharaf juga memiliki urutan (*sequences*) dimana pada umumnya perubahan diawali pada kata kerja lampau (fiil madhi) kemudian mengalami perubahan pada jenis fiil atau isim yang lain lalu diikuti perubahan berdasar jenis subyek.

Kompleksitas kaidah perubahan menjadi masalah utama dalam mempelajari ilmu sharaf akan tetapi jika dicermati lebih lanjut terdapat sistematika dan ciri khas didalamnya. Secara umum dapat dikatakan bahwa ditiap jenis fiil terdapat teks kata dan juga rumus yang terikat pada fiil tersebut contohnya pada keluarga fiil madhi memiliki ciri khas menjadi patokan bagi perubahan ke fiil mudhori’ dan imbuhan biasanya diakhir huruf dasar. Setiap jenis fiil memiliki kesamaan yaitu mereka memiliki kemampuan perubahan, akan tetapi berbeda pada rumus yang terikat pada jenisnya. Salah satu poin penting dalam memahami kaidah ilmu sharaf adalah proses klasifikasi jenis fiil dan rumus perubahan sesuai dengan jenis kelompoknya.

2.2 JQuranTree Library

JQuranTree merupakan sekumpulan Java API (*Application Programming Interfaces*) untuk mengakses dan menganalisa Al-Qur'an dalam bentuk bahasa Arab yang otentik. *Library* yang dibuat oleh Kais Duke dan tim yang berasal dari Universitas Leeds, Inggris dirilis sebagai proyek *open source* yang bertujuan untuk mendorong penelitian dalam analisis komputasional Al-Qur'an. [DUK-11] JQuranTree terbagi dalam 3 bagian utama yaitu :

a. Teks Al-Qur'an
b. Sekumpulan API untuk pengaksesan (*Access APIs*)
c. Sekumpulan API untuk analisis (*Analysis APIs*)

Perbedaan dari *Access* dengan *Analysis* adalah *Access APIs* lebih membahas pada cara merepresentasikan bahasa Arab Qur'an (Surat, Ayat, Huruf dan tanda baca) sedangkan *Analysis APIs* menggunakan representasi tersebut dalam implementasinya.

Model Objek Ortografi yang dibuat pada JQuranTree disusun berdasarkan hirarki sebagai berikut :

Gambar 2.4 Model ortografi Qur'an pada JQuranTree

Definisi Ortografi adalah sistem ejaan suatu bahasa atau bunyi suatu bahasa yang berupa tulisan atau lambang. Ortografi meliputi masalah ejaan, kapitalisasi, pemenggalan kata serta tanda baca [DUK-11]. Pada JQuranTree model ortografi Al-Qur'an sifatnya *immutable* dalam artian model ini dapat dibaca dan dicari tapi tidak bisa dirubah isinya. Dari atas sampai bawah hirarki yang ada pada gambar model ortografi tersusun atas elemen berikut :

a. *Document* : representasi terstruktur keseluruhan teks Al-Quran termasuk seluruh teks ayat , nama surat, frase bismillah dan informasi mengenai dokumen lainya.

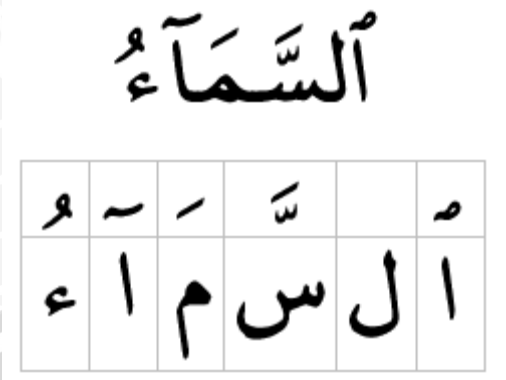
b. *Chapter* : Al-Qur'an terdiri dari 114 Surat . Setiap surat memiliki nama dan penomoran tersendiri.

c. *Verse* : Setiap Surat terdiri atas ayat-ayat yang berurutan dan didalam Al-Quran jumlahnya adalah 6236 ayat. Hampir semua surat diawali dengan frase bismillah .

d. *Token* : teks Arab/kata yang dipisahkan oleh spasi dalam sebuah ayat. Dalam bahasa Arab sebuah huruf dan banyak partikel dapat bergabung menjadi satu token ortografi.

Kelas *Document* berada pada puncak model objek. Kelas ini *singleton* dan menyediakan metode statis dalam mengakses elemen-elemen yang lain. Elemen Ortografi lainya menyediakan metode instansiasi. Kelas *Verse* dan *Token* diturunkan dari kelas *ArabicText* . Penggunaan Kelas turunan ini sesuai karena pada dasarnya *Verse* (Ayat) dan *Token* (Kata) adalah teks Arab.

Pada tingkatan yang paling bawah ortografi teks Arab dalam Al-Qur'an dimodelkan dalam susunan instansiasi kelas ArabicCharacters. Tiap karakter Arab memiliki *character type* dan *diacritics* (tanda baca). Sebagai contoh diambil kata ketiga dari ayat 70:8 (Al-Ma'arij) . Kata ini dibaca *l-samāu* . Secara ortografi kata ini direpresentasikan dengan 6 karakter , dengan *diacritic* yang menempel pada 5 hurufnya.



Gambar 2.5 Contoh pemecahan kata bahasa Arab

Teks Arab dibaca dari kanan ke kiri dan disini bisa dilihat bahwa huruf kedua tidak memiliki tanda baca, huruf ketiga memiliki 2 tanda baca yaitu *fatha* dan *shadda*. Dalam karakter *unicode* maka kata diatas terhitung sebanyak 12 karakter dimana tiap karakter berukuran 2-byte .

Dalam JQuranTree tidak menggunakan *Unicode* dalam memodelkan ortografi Qur'an, tetapi memakai ArabicCharacter yaitu merupakan sebuah kelas yang memodelkan huruf dalam bahasa Arab. Tiap ArabicCharacter direpresentasikan dengan 3 bytes. Byte pertama untuk *character type* , Byte kedua dan ketiga untuk *diacritics* (tanda baca). Jumlah maksimum kombinasi dari skema ini adalah sebanyak 256 *character type* dan kombinasi 16 *diacritic types* .

Arabic Character + Diacritics	Byte 1 Character	Byte 2 Diacritics	Byte 3 Diacritics
Alif + HamzatWasl	0	0	2^3
Lam	22	0	0
Seen + Fatha + Shadda	11	$2^0 + 2^6$	0
Meem + Fatha	23	2^0	0
Alif + Maddah	0	0	2^0
Hamza + Dammah	28	2^1	0

Gambar 2.6 Penguraian karakter Arab berdasar byte

Teks Al-Qur'an berisi 6236 ayat yang jika direpresentasikan dengan Unicode maka akan membutuhkan 1389662 bytes (1.33 megabytes) sedangkan jika menggunakan representasi diatas menjadi 1242006(1.18 megabytes) dibagi dengan 3 maka akan didapatkan sebanyak 414002 karakter untuk semua huruf dalam ayat termasuk spasi.

Instansiasi Kelas ArabicText bersifat *Immutable*. ArabicText tersusun dari instansiasi ArabicCharacters. Hal ini berbeda dengan Kelas String di Java yang tersusun dari karakter unicode 2-byte.

Character	Glyph	Description
Alif	ا	Arabic letter
Ba	ب	Arabic letter
Ta	ت	Arabic letter
Tha	ث	Arabic letter
Jeem	ج	Arabic letter
HHa	ح	Arabic letter
Kha	خ	Arabic letter
Dal	د	Arabic letter
Thal	ذ	Arabic letter
Ra	ر	Arabic letter
Zain	ز	Arabic letter
Seen	س	Arabic letter
Sheen	ش	Arabic letter
Sad	ص	Arabic letter
DDad	ض	Arabic letter
TTa	ط	Arabic letter
DTha	ظ	Arabic letter
Ain	ع	Arabic letter
Ghain	غ	Arabic letter
Fa	ف	Arabic letter
Qaf	ق	Arabic letter
Kaf	ك	Arabic letter

Gambar 2.7 Pemetaan transliterasi buckwalter huruf Arab

Pada gambar diatas menunjukkan tabel yang memetakan antara huruf-huruf Arab dengan transliterasi buckwalter mulai dari huruf alif (ا) sampai dengan huruf kaf (ك). Pemetaan ini juga menunjukkan nama konstanta yang digunakan oleh JQuranTree dalam penamaan atribut pada kelas Character Type.

Meem	م	Arabic letter
Noon	ن	Arabic letter
Ha	ه	Arabic letter
Waw	و	Arabic letter
Ya	ى	Arabic letter
Hamza	ء	Arabic letter
AlifMaksura	ى	Arabic letter
TaMarbuta	ة	Arabic letter
Tatweel	ـ	Orthographic symbol used to lengthen the previous letter. In Tanzil XML, a diacritic *hamza* may sit on a *tatwīl*.
SmallHighSeen	ۜ	Quranic symbol
SmallHighRoundedZero	۟	Quranic symbol
SmallHighUprightRectangularZero	۠	Quranic symbol
SmallHighMeemIsolatedForm	ۡ	Quranic symbol
SmallLowSeen	ۭ	Quranic symbol
SmallWaw	ۥ	Quranic symbol
SmallYa	ۦ	Quranic symbol
SmallHighNoon	ۨ	Quranic symbol
EmptyCentreLowStop	۪	Quranic symbol
EmptyCentreHighStop	۫	Quranic symbol
RoundedHighStopWithFilledCentre	۬	Quranic symbol
SmallLowMeem	۫	Quranic symbol

Gambar 2.8 Pemetaan transliterasi buckwalter pada simbol Al-Qur'an

Pada gambar diatas masih menunjukkan pemetaan antara simbol huruf Arab dengan transliterasi buckwalter tetapi yang menjadi perbedaan disini adalah adanya pemetaan simbol pada Al-Qur'an . Pada umumnya huruf Arab dikenali mulai dari huruf alif (ا) sampai dengan huruf ya' (ي) yang berjumlah 28 huruf [ALY-09] tetapi diluar itu juga terdapat huruf yang adanya pada teks Al-Qur'an seperti alif maksura (ى) dan ta marbuta (ة). Selain itu juga terdapat pemetaan pada simbol-simbol yang ditemukan pada Al-Qur'an yang digunakan sebagai tanda berhenti maupun larangan berhenti baik ditengah atau diakhir ayat.

Diacritic	Glyph	Description
Fatha	◌َ	Above
Damma	◌ُ	Above
Kasra	◌ِ	Below
Fathatan	◌ً	Double *fatha*
Dammatan	◌ٌ	Double *damma*
Kasratan	◌ٍ	Double *kasra*
Shadda	◌ّ	Above
Sukun	◌ْ	Above
Maddah	◌ٓ	Above
HamzaAbove	◌ٔ	Above
HamzaBelow	◌ٕ	Below
HamzatWasl	◌ٱ	Above *alif*
AlifKhanjareeya	◌ٰ	Superscript *alif*

Gambar 2.9 Pemetaan transliterasi pada jenis tanda baca/*diacritics type*

Pada gambar diatas menjelaskan pemetaan tanda baca harokat pada huruf Arab yang secara umum diketahui terdapat 3 tanda baca utama yaitu Fathah (◌َ), Kasra (◌ِ) dan Damma (◌ُ). Pada implementasi API JQuranTree semua tanda baca dikumpulkan pada kelas bernama DiacriticsType dengan atribut berupa konstanta.

2.3 Design Patterns

Seorang pemrogram atau pengembang perangkat lunak bertugas menuntaskan suatu permasalahan dalam membuat sebuah perangkat lunak dengan melakukan perubahan dari model, abstraksi desain ke dalam baris kode bahasa pemrograman. Tidak jarang masalah yang dihadapi seorang *programmer* bersifat rekursif atau dapat dikatakan berulang, karena dari sudut pandang pengalaman *programmer* tersebut mungkin dia sudah membuat solusi atas masalah pemrograman tersebut dimasa lalu tapi entah kenapa ketika dihadapkan pada masalah yang jika ditarik prinsip utamanya adalah sama, *programmer* seolah kembali berupaya menemukan solusi pemrograman itu.

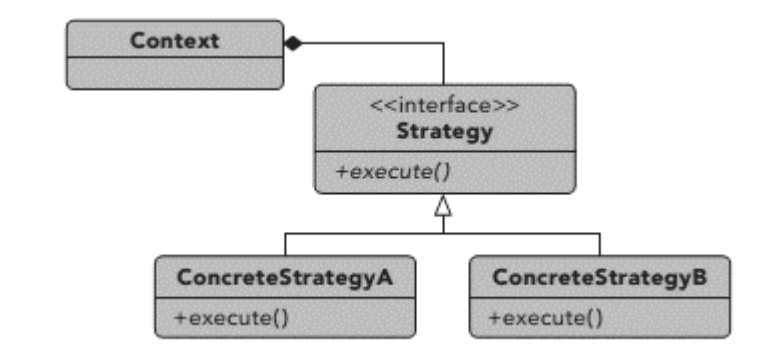
Permasalahan tersebut adalah hal yang biasa ditemui dalam dunia pengembangan perangkat lunak dan dapat dijawab dengan menggunakan *Design Patterns* atau pola-pola perancangan. *Design Patterns* merupakan sekumpulan solusi inti yang terbukti mampu diterapkan pada beberapa permasalahan pemrograman berorientasi objek yang terulang dalam situasi dan kondisi tertentu [FRE-04]. *Design Patterns* mulai populer didalam pengembangan perangkat lunak ketika Erich Gamma beserta rekan yang kemudian disebut sebagai *Gang Of Four* (GoF) membuat buku *Design Patterns*: *Elements of Reusable object-oriented Software* [GAM-95] yang didalamnya terdapat 23 jenis patterns standar walaupun hanya sebagian saja yang lebih banyak ditemui (diimplementasikan) pada dunia pengembangan perangkat lunak [HOL-06]. 23 Pattern tersebut dikategorikan menjadi tiga jenis yang berbeda berdasarkan penggunaanya :

a. *Creational Patterns*: digunakan untuk membuat konstruksi sebuah objek sehingga proses pembuatan objek (instansiasi) terpisah dari sistem yang hendak mengimplementasikanya.
b. *Structural Patterns*: digunakan untuk membuat struktur objek yang besar yang terdiri dari banyak objek-objek lain yang terpisah.
c. *Behavioural Patterns*: digunakan mengatur algoritma, relasi dan tanggung jawab antar objek dalam sebuah sistem.

Pengembangan perangkat lunak pada skripsi ini merupakan sebuah purwarupa atau *prototype* yang nantinya diharapkan dapat dikembangkan lebih lanjut. Dengan menyadari kondisi tersebut disadari pentingya penerapan sebuah *design pattern* dalam proses pengembanganya. Tujuan dalam menggunakan *design pattern* nantinya diharapkan dapat membantu meningkatkan kualitas sebuah perangkat lunak dalam hal *reusability* (pemanfaatan/penggunaan ulang), *maintainability* (penyesuaian) dan *extensibility* (penambahan fungsi). Perangkat lunak pada penelitian ini adalah aplikasi yang dapat membantu proses belajar perubahan kata kerja bahasa Arab yang didalamnya terdapat masalah seperti pada penjelasan paragraf akhir subbab 2.1. Terdapat 2 pola perancangan yang bisa jadi membantu dalam memodelkan masalah tersebut yaitu *Strategy pattern* dan *Facade Pattern*.

2.3.1 Strategy Pattern

Strategy pattern [FRE-04] adalah sebuah pola yang menyusun sekelompok atau keluarga algoritma tertentu kemudian melakukan proses enkapsulasi pada tiap anggota algoritma tersebut dan memberi kemampuan untuk saling bertukar saat *run-time*. *Strategy pattern* juga memberi keleluasan pada algoritma dimana mereka dapat berbeda dan independen dari klien (kelas lain) yang menggunakanya.



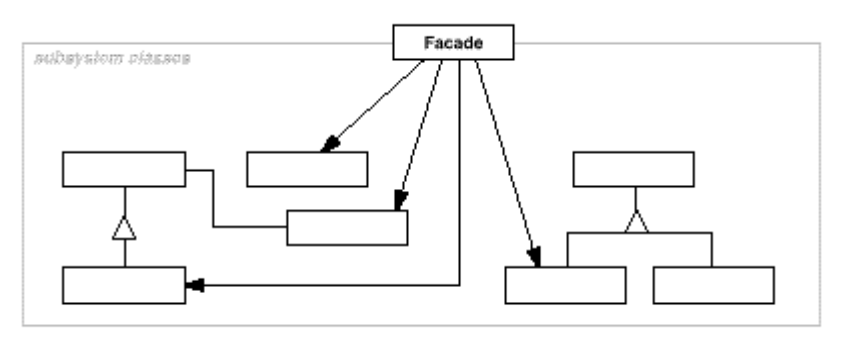
Gambar 2.10 Diagram Kelas Strategy Pattern [FRE-04]

Strategy pattern melakukan pengelompokan rumus perubahan kata kerja pada tiap jenis fiil yang kemudian direpresentasikan sebagai keluarga algoritma transformasi, Sehingga terdapat keluarga algoritma transformasi fiil madhi, mudhori dan amr.

S*trategy pattern* memisahkan bagian kode yang relatif mudah berubah yaitu algoritma transformasi, dengan bagian yang cenderung tetap yaitu struktur objek kata kerja dan hal ini memudahkan penambahan algoritma dimasa depan. Selain itu pola ini bisa memberikan kemampuan pada tiap objek kata kerja untuk merubah algoritma secara *run-time*. Hal ini bermaksud memberikan kemampuan satu jenis objek fi'il dapat menghasilkan objek fi'il lain yang berbeda tanpa perlu merubah struktur kode objek tersebut karena ketika *run-time* hanya diperlukan pergantian algoritma.

2.3.2 Facade Pattern

Facade pattern adalah pattern yang menyediakan *interface* (antarmuka) yang menyatukan set *interface* (antarmuka) dari *sub-system*. *Facade* menjadi semacam antarmuka pada lapisan yang lebih tinggi untuk memudahkan akses terhadap fungsi-fungsi subsistem [HOL-06].

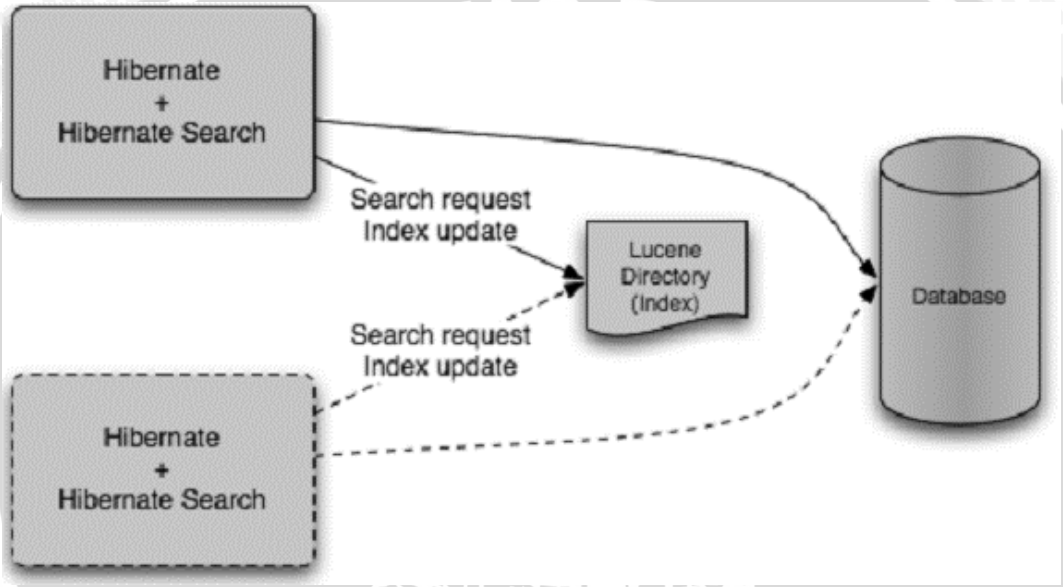


Gambar 2.11 Diagram Kelas Facade Pattern [FRE-04]

Penggunaan *Facade pattern* diharapkan mampu menjawab permasalahan pengaturan dan pengurutan proses perubahan kata kerja. Dalam desain sistem terdapat prinsip "*Least Knowledge*" yang menjelaskan bahwa interaksi antar objek sebaiknya hanya pada objek yang berasal dari kelas yang berdekatan. Di dalam mendesain sistem nantinya akan dibuat banyak kelas yang strukturnya memiliki hirarki, seperti adanya *superclass* "Fiil" (kata kerja) yang mungkin memiliki banyak *subclass* yaitu jenis-jenis Fiil. Dibutuhkan sebuah kelas pengatur antarmuka yang lebih sederhana untuk bisa mengatur interaksi antar objek yaitu kelas *facade*.

2.4 Hibernate Search

Hibernate Search adalah *search engine* API yang menggabungkan kemampuan model domain objek *Hibernate* dengan *Full Text Search Engine Apache Lucene* [BER-14]. API ini melakukan proses indeks model domain (Hibernate) menggunakan anotasi khusus yang telah didefinisikan, mengatur sinkronisasi basis data / indeks kemudian mengembalikan objek reguler yang merupakan hasil dari *free text queries* (Lucene).

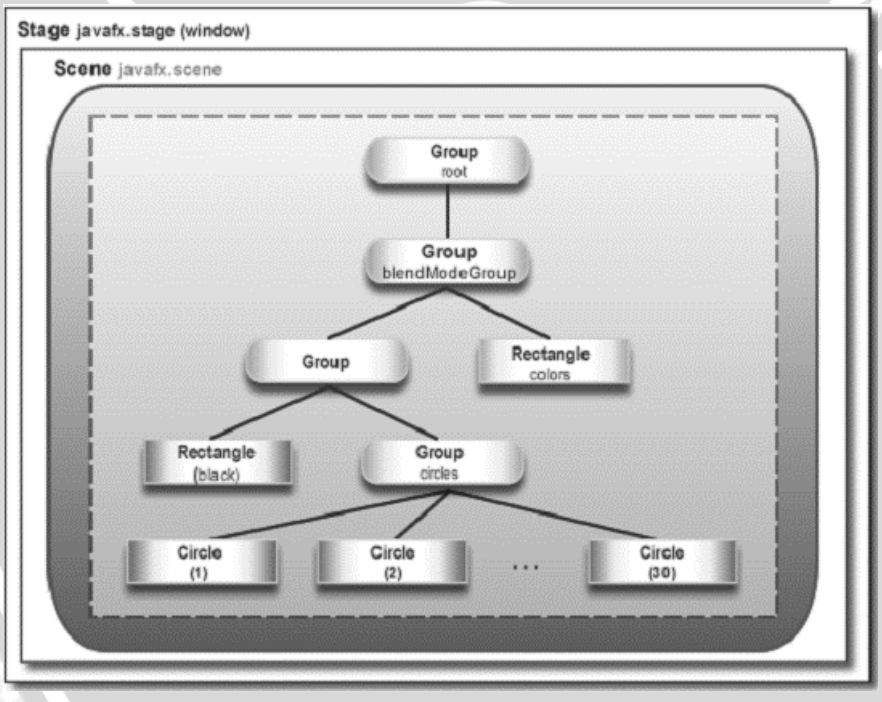


Gambar 2.12 Arsitektur Hibernate Search [BER-14]

Gambar 2.11 menunjukkan bahwa hibernate search terdiri dari komponen *indexing* dan komponen pencarian indeks yang kedua komponen tersebut didukung oleh Lucene API. *Indexing* merupakan proses pembuatan indeks dimana indeks merupakan sebuah pointer yang mereferensikan kedudukan suatu baris data tertentu dalam *database*. Indeks berisi data tertentu yang merepresentasikan informasi baris data di *database* dan sudah melalui proses pengurutan (*sorting*). Penggunaan indeks sangat diperlukan dalam mempercepat proses pencarian suatu data ketika terdapat permintaan *query* tertentu karena ketika suatu database mempunyai sebuah indeks maka pencarian dilakukan pada data indeks tersebut , tidak secara langsung mencari pada data yang asli.

2.5 JavaFX Framework

JavaFX merupakan *framework Graphic User Interface* (GUI) terbaru pada bahasa pemrograman Java yang hendak menggantikan Swing [CHA-13]. Beberapa kelebihan yang dimiliki dibandingkan dengan Swing adalah kemudahan pengaturan layout tampilan yang akan dibuat (*scene builder*), komponen *controls* yang lebih lengkap (*buttons*, *menu* dan sebagainya), adanya FXML yang digunakan sebagai pembuatan layout komponen GUI yang terpisah dari *logic application*, integrasi penggunaan *style sheets* (CSS) yang menyerupai pada pengembangan *web application*, dan penambahan *visual effect* serta animasi yang membuat aplikasi semakin atraktif.



Gambar 2.13 Scene Graph JavaFX [CHA-13]

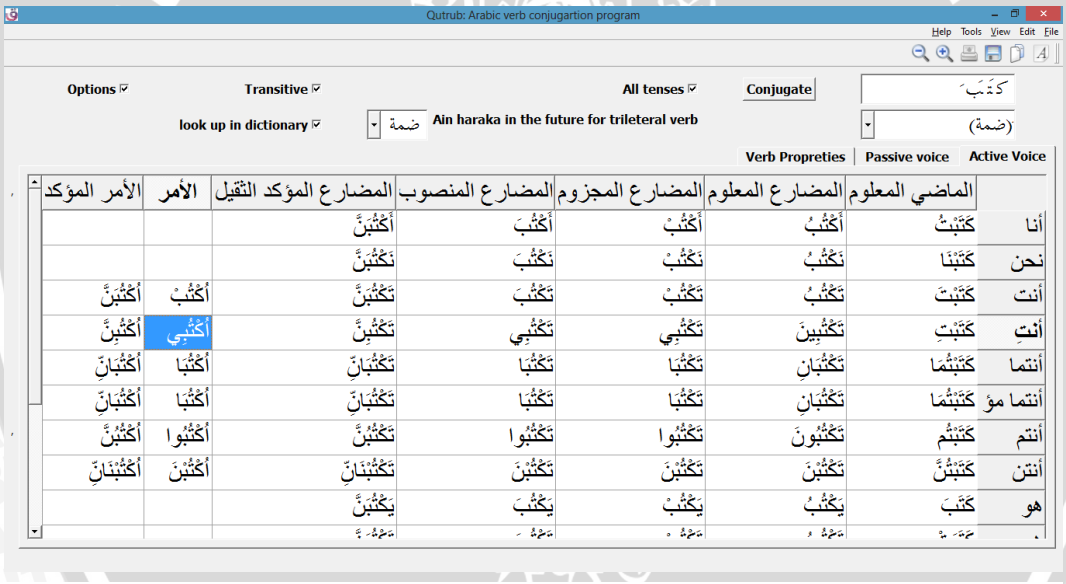
Gambar 2.12 menunjukkan layer / lapisan arsitektur dari JavaFX yang terdiri dari *Stage*, *Scene* dan *contents*. Stage merupakan layer tertinggi sebagai *container window*. *Scene* sebagai container seluruh isi komponen UI yang digunakan. *Contents* direpresentasikan sebagai hirarki atas node-node yang biasanya berupa komponen UI.

2.6 Aplikasi Pembelajaran Ilmu Sharaf : Quthrub dan SARF

Berdasarkan pencarian di Internet yang dilakukan untuk mengetahui apakah terdapat aplikasi pembelajaran ilmu sharaf yang telah beredar maka ditemukan 2 aplikasi yang dianggap memiliki kesamaan dengan aplikasi yang hendak dikembangkan oleh penelitian ini yaitu *Quthrub* dan *SARF*.

2.6.1 Quthrub : Arabic Verb Conjugation Program

Aplikasi ini merupakan program yang dibuat untuk menunjukkan hasil perubahan kata kerja bahasa Arab [QUT-14]. Program ini ditemukan pada situs sourceforge.net yang merupakan penampung aplikasi – aplikasi yang sifatnya *freeware*, *open-source* dan sebagainya.



Gambar 2.14 Tampilan dari program Quthrub

Beberapa fitur yang dikenali dari program ini adalah :

1. Mampu melakukan proses perubahan kata kerja pada bentuk aktif dan juga pasif pada jenis fiil madhi, mudhori, dan amr.
2. Menyertakan pilihan pada penggunaan kamus ketika hendak mencari bentuk perubahan kata kerja sehingga pengguna bisa memilih apakah kata kerja yang dimasukkan perlu dicari didalam kamus atau tidak.
3. Melakukan ekspor hasil perubahan kata kerja dalam format html.

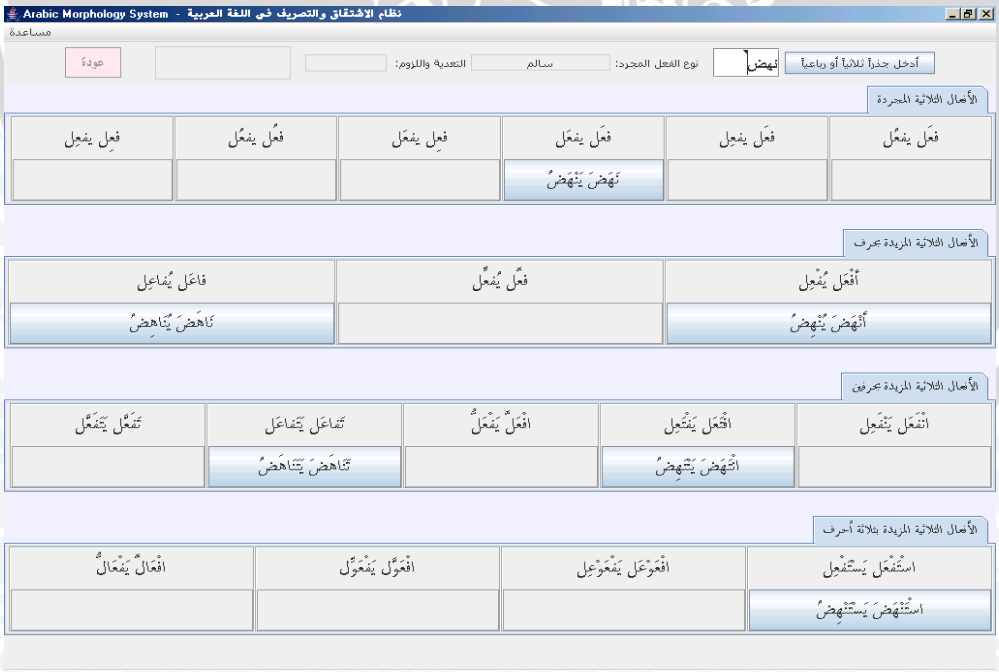
4. Melakukan pencetakan hasil perubahan kata kerja.

Secara keseluruhan aplikasi ini sudah cukup baik menunjukkan hasil perubahan kata kerja bahasa Arab, akan tetapi dalam terdapat beberapa keterbatasan antara lain :

1. Program ini tidak bersifat *open source* sehingga sulit jika ada pengembang lainya yang ingin berkontribusi dalam penambahan atau perbaikan fungsi program yang telah ada.
2. Penggunaan bahasa masih dominan dalam bahasa Arab sehingga cukup menyulitkan untuk pengguna yang masih dalam tahap awal belajar khususnya dari pengguna non-Arab.

2.6.2 SARF : Arabic Morphology System

Sarf Arabic Morphology System merupakan proyek yang dikembangkan oleh *Arab League Educational, Cultural and Scientific Organization* (ALECSO) untuk menjadi aplikasi pendukung pembelajaran ilmu kata kerja dan turunanya (Sharaf) [SAR-14]. Sama halnya dengan Quthrub, SARF juga ditemukan pada situs sourceforge.net.



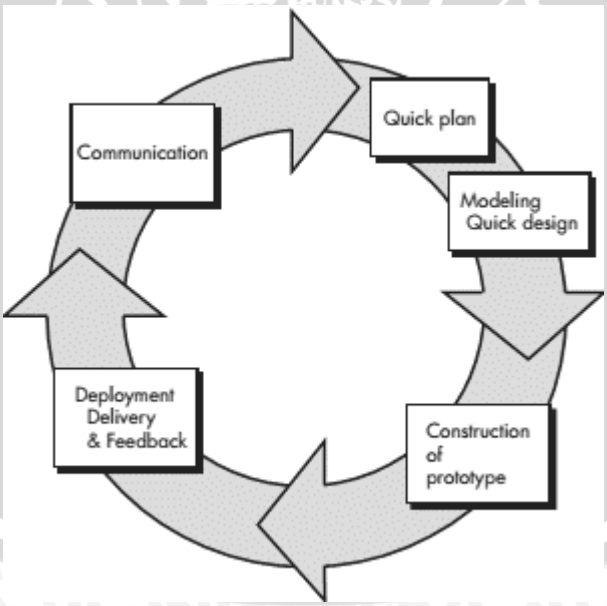
Gambar 2.15 Tampilan dari program Sarf

Beberapa kemampuan yang dikenali dari aplikasi ini adalah :

1. Hasil perubahan kata kerja meliputi rumus pada jumlah huruf dan jenis seperti 3 huruf tanpa tambahan imbuhan (*tsulatsy mujarrad*) dan menggunakan imbuhan (*tsulatsy mazied*) disertai dengan perubahan pada jenis kata benda (isim).
2. Jenis aplikasi yang sifatnya *open source*. Menyertakan kode sumber yang dibangun pada bahasa Java beserta *database* akar kata kerja dalam format xml.

Secara keseluruhan kemampuan aplikasi dalam melakukan proses perubahan dapat dikatakan cukup lengkap dan kompleks. Dalam laman yang berisi proyek pengembangan SARF juga disertakan dokumentasi yang detail dan lengkap akan tetapi semuanya dalam bahasa Arab. Sama halnya dengan Quthrub, aplikasi SARF secara user interface dikhususkan bagi pengguna yang memahami bahasa Arab.

2.7 Evolutionary Process Model : Prototyping



Gambar 2.16 Prototyping software process [PRE-10]

Prototyping adalah salah satu *software process/software development life cycle* yang masuk kategori *evolutionary process model* dengan ciri pengembangan yang iterative dan *requirement* pengguna yang tidak detail [PRE-10].

Proses tahapan yang dilalui dalam Prototyping adalah :

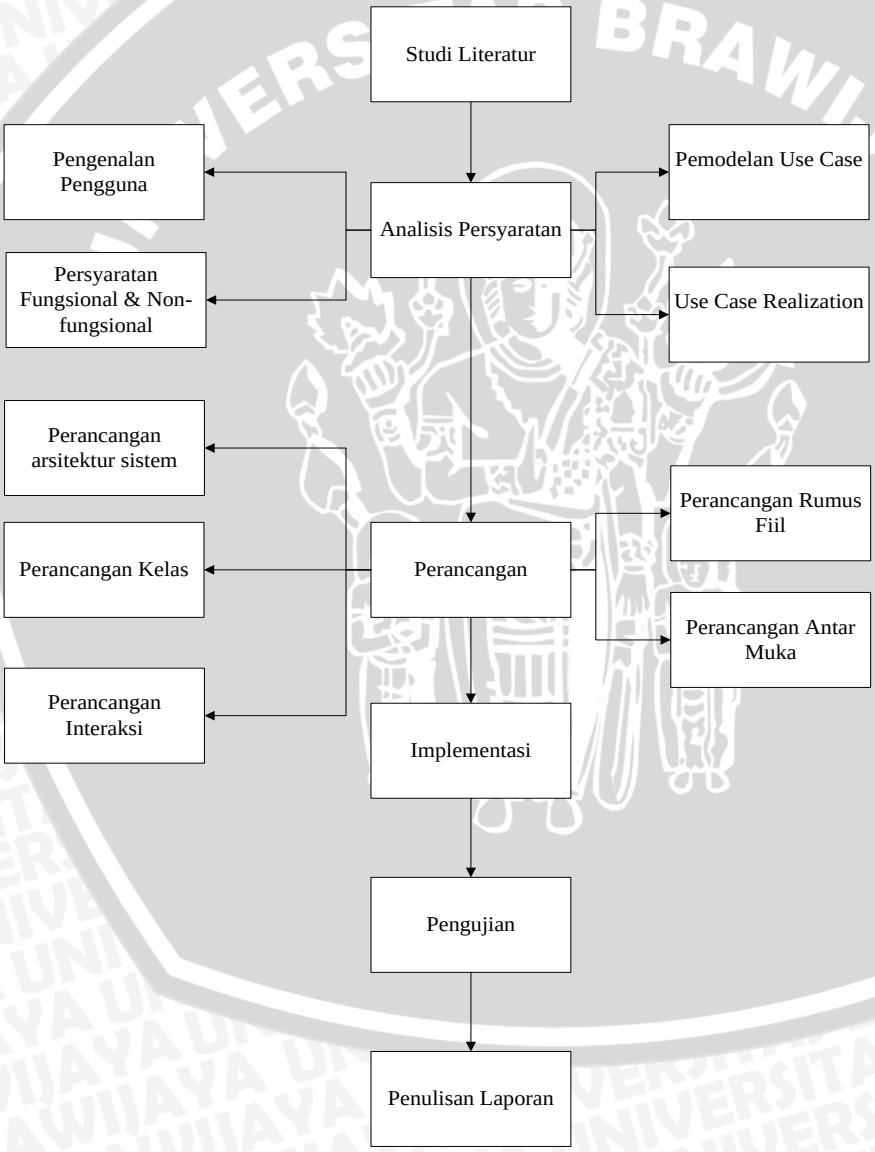
1. *Communication* : tahapan dimana pengembang bertemu dengan *stakeholders* untuk mendefinisikan kebutuhan / persyaratan yang dibutuhkan dalam membangun perangkat lunak, melakukan identifikasi kebutuhan (*requirements*) yang sudah diketahui dengan jelas dengan yang masih membutuhkan penggalian informasi lebih detail.
2. *Quick Plan* : tahapan pembuatan rencana untuk pencapaian fungsional perangkat lunak pada tiap iterasi secara umum.
3. *Modelling, Quick Design* : tahapan pembuatan model dan rancangan sistem yang berfokus pada aspek fungsionalitas sistem yang dapat ditampilkan pada pengguna / *end user*.
4. *Construction* : tahapan implementasi dari rancangan sistem yang telah dibuat.
5. *Deployment and Feedback* : tahapan dimana prototipe sistem yang telah dibuat dikirimkan pada pengguna untuk dievaluasi dan mendapatkan *feedback* yang mampu untuk memperbaiki *requirements* yang telah didefinisikan serta menetapkan pencapaian fungsionalitas sistem pada iterasi berikutnya.

Penelitian ini melakukan adaptasi dari *Prototyping software process* karena ditemukan beberapa aspek yang sesuai dalam proses pengembangan aplikasi pendukung pembelajaran bahasa Arab. Terdapat kesamaan pada aspek pengembangan yang *iterative* karena hasil dari penelitian ini merupakan iterasi awal dari proyeksi aplikasi yang nantinya memiliki fungsional yang lebih lengkap. Aspek berikutnya adalah terdapat *requirement* yang masih belum mendetail khususnya pada bagian interaksi manusia dan komputer yang tepat dalam mendemonstrasikan proses pembelajaran Sharaf dan algoritma perubahan kata kerja yang jumlahnya cukup banyak.

BAB III
METODOLOGI PENELITIAN

3.1 Metode Penelitian

Pada bab ini dijelaskan mengenai prosedur dan kegiatan yang akan dilakukan dalam pengerjaan skripsi, yaitu studi literatur, analisis persyaratan, analisis komponen, perancangan, implementasi dan pengujian dalam pengembangan perangkat lunak yang akan dibuat.



Gambar 3.1 Alur Proses Penelitian

3.1.1 Studi Literatur

Studi literatur merupakan penelusuran literatur yang bertujuan dalam menyusun dasar teori yang digunakan untuk menunjang skripsi. Penelusuran literatur dapat bersumber dari buku, media dan belajar pada ahli .

Penyusunan dasar teori dilakukan setelah mendapatkan referensi yang tepat untuk mendukung penulisan penelitian ini. Studi Literatur yang dilakukan sebagian besar mengenai Ilmu Sharaf dan yang berkaitan dengan morfologi bahasa Arab seperti buku modul pembelajaran [ISL-14] sedangkan yang berkaitan dengan implementasi dalam pengembangan aplikasi banyak berkaitan dengan *Design Patterns,* API JQuranTree, JavaFX, Hibernate dan beberapa API lainya.

3.1.2 Analisis Persyaratan

Tujuan dari tahap analisis persyaratan adalah memahami persyaratan yang dibutuhkan oleh sistem. Analisis persyaratan merupakan langkah awal untuk menentukan aplikasi seperti apa yang akan dihasilkan. Sistem yang baik dan sesuai dengan kebutuhan pengguna sangat bergantung kepada keberhasilan dalam proses ini. Identifikasi aktor digunakan untuk menjelaskan aktor pengguna yang akan menggunakan sistem sesuai dengan peran dan kebutuhanya. Setelah itu dilakukan pemodelan dengan use case.

Untuk mempermudah dalam menganalisis sistem dibutuhkan dua jenis kebutuhan. Dua jenis kebutuhan tersebut adalah kebutuhan fungsional dan kebutuhan nonfungsional. Metode analisis yang digunakan adalah dengan pendekatan berorientasi objek yang memakai bahasa pemodelan UML (*Unified Modelling Language*).

Proses pengamatan dan wawancara pada 2 pengajar bahasa Arab yang berbeda yaitu Ustadz Islahudin dan Ustadz Abdul Jalil dilakukan untuk memahami alur skenario pembelajaran sharaf. Ustadz Islahudin merupakan lulusan pasca-sarjana UIN Maulana Malik Ibrahim yang berprofesi sebagai pengajar madrasah tsanawiyah dan PPTQ (Program Pengembangan Terjemah Qur'an) Al-Ikhlas Malang. Ustadz Abdul Jalil adalah seorang pengajar yang telah lama mengajarkan sharaf, sekitar 40 tahun di pondok pesantren dan daerah pedesaan.

3.1.3 Perancangan Sistem

Hasil dari analisis persyaratan menjadi acuan dalam perancangan perangkat lunak. Perancangan perangkat lunak menggunakan pendekatan berorientasi objek yang kemudian dimodelkan ke dalam bentuk arsitektur sistem, diagram kelas, diagram interaksi, perancangan rumus, dan perancangan antar muka. Pada perancangan sistem nantinya akan menggunakan beberapa *Design Patterns* atau pola-pola perancangan khususnya pada pembuatan diagram kelas walaupun sebenarnya secara keseluruhan juga mempengaruhi proses mekanisme didalam sistem itu sendiri. Pertimbangan penggunaan ulang (*reuse*) terhadap beberapa komponen perangkat lunak yang akan digunakan juga dibahas pada bagian ini.

3.1.4 Implementasi

Implementasi Pengembangan Aplikasi Pendukung Pembelajaran Bahasa Arab dilakukan sesuai dengan hasil analisis persyaratan dan perancangan sistem. Selama tahap ini komponen-komponen yang telah ditentukan akan dilakukan penambahan atau perubahan ke dalam sistem. Implementasi meliputi :

- Pengembangan aplikasi sesuai hasil perancangan.
- Pembuatan basis data sesuai dengan kebutuhan sistem.
- Data kata kerja (*Fi'il*) dan beberapa aturan (*wazan*) bahasa Arab yang telah dimasukkan dalam basis data.
- Pembuatan antarmuka sistem dibangun sesuai dengan perancangan antarmuka.

3.1.5 Pengujian

Pengujian sistem pada penelitian ini dilakukan agar dapat menunjukan bahwa sistem telah mampu bekerja sesuai dengan spesifikasi dari kebutuhan sistem dan menemukan celah kesalahan (*error*) yang mungkin secara tidak sengaja terdapat dalam sistem aplikasi. Pengujian dilakukan dengan teknik *whitebox basis path testing* yang mengambil *sample* dari beberapa metode yang terdapat dalam aplikasi dan *blackbox testing* dimana menguji fungsional aplikasi apakah sesuai dengan spesifikasi dan juga tingkat kemudahan penggunaan yang dirasakan oleh pengguna menggunakan Pengujian *usability*.

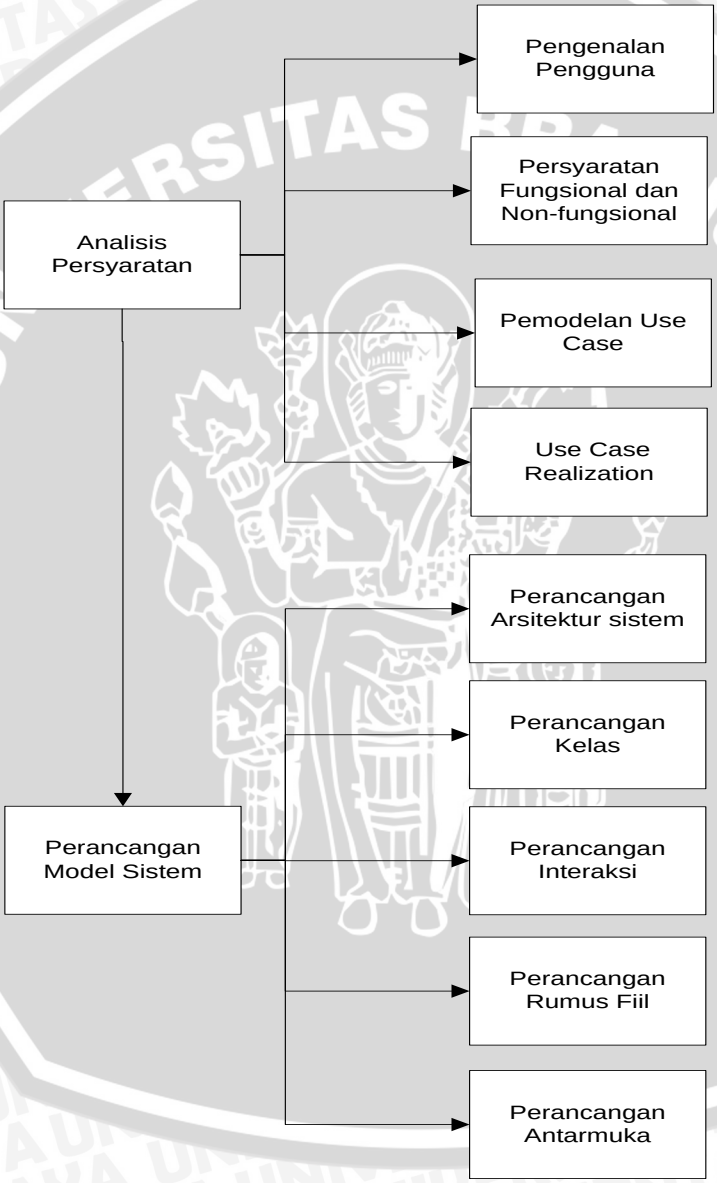
Pengujian *usability* dilakukan dengan cara pengujian aplikasi oleh pengguna yang ditentukan yaitu seorang ahli bahasa Arab dengan cara menjalankan aplikasi, mengisi kuesioner dan wawancara. Terdapat 5 kriteria yang ditentukan untuk membuat instrumen pertanyaan dan penilaian yaitu navigasi menu, informasi keluaran, informasi kesalahan, integritas konseptual antarmuka dan interaktivitas.

3.1.6 Penulisan Laporan

Laporan penelitian ditulis setelah semua proses pengerjaan skripsi dilalui. Laporan berisi dokumentasi perancangan sistem informasi beserta saran untuk memberikan pertimbangan dan pengembangan aplikasi selanjutnya.

BAB IV
ANALISIS PERSYARATAN DAN PERANCANGAN

Bab ini membahas mengenai analisis persyaratan perangkat lunak dan perancangan perangkat lunak. Perancangan dilakukan berdasarkan hasil yang telah didapatkan dari analisis persyaratan yang dibuat.



Gambar 4.1 Alur Analisis Persyaratan dan Perancangan

4.1 Analisis Persyaratan

Tahap analisis persyaratan terdiri dari tiga langkah yaitu analisa kebutuhan meliputi pengenalan pengguna, persyaratan fungsional dan non-fungsional dan pemodelan use case.

4.1.1 Pengenalan pengguna

Proses pembelajaran ilmu sharaf melibatkan 2 pihak yaitu Ustadz atau pengajar bahasa Arab dan santri atau siswa. Berdasarkan hasil pengamatan pada proses pembelajaran ilmu sharaf yang dilakukan serta hasil wawancara dengan pengajar maka dalam proses tersebut terdapat 2 hal yang dilakukan dalam setiap proses belajar sharaf ditahap awal:

a. Penjelasan perubahan kata kerja

   Pengajar akan melakukan pengenalan kata kerja (*fiil*) yang bentuk awalnya merupakan *fiil madhi* kemudian melakukan proses perubahan kata kerja (*tashrif istilahi*) ke bentuk lainya seperti *mudhori'*, *amr* dan *isim* (kata benda). Setelah itu dilakukan perubahan kata kerja berdasarkan jenis subyek/*damir* (*tashrif lughowi*) ditiap bentuk kata kerja yang telah dihasilkan. Kata kerja yang dijadikan contoh penjelasan bisa berasal dari proses pencarian di kamus Arab maupun hafalan dari pengajar itu sendiri.

b. Latihan Soal (Test) perubahan kata kerja

   Pengajar menguji pemahaman siswa dalam mengisi perubahan kata kerja yang bisa berupa perubahan berdasar bentuk / tashrif istilahi maupun perubahan berdasar subyek kata kerja / tashrif lughowi.

Dari penjelasan tersebut akhirnya ditetapkan bahwa purwarupa aplikasi yang dikembangkan akan memiliki dua fitur utama yaitu Pencarian Bentuk Tashrif Fiil dan Latihan Tashrif Fiil.

4.1.2 Persyaratan Fungsional dan Non-fungsional Sistem

Daftar persyaratan fungsional sistem akan dicantumkan pada tabel dibawah ini secara deklaratif :

Tabel 4.1 Tabel Persyaratan Fungsional Sistem

ID	Persyaratan
SRS_001_01	Sistem dapat menerima masukan kata kerja fiil madhi dengan tanda baca maupun tanpa tanda baca.
SRS_001_02	Sistem dapat memfilter masukan yang tidak sesuai dengan syarat masukan.
SRS_001_03	Sistem dapat menampilkan data hasil perubahan kata kerja dalam bentuk mudhori dan amr serta penjelasan ejaan.
SRS_001_04	Sistem dapat menampilkan data hasil perubahan kata kerja berdasarkan perubahan subyek dan disertai penjelasan ejaan.
SRS_002_01	Sistem dapat menampilkan pilihan fiil madhi yang menjadi soal latihan tashrif.
SRS_002_02	Sistem menampilkan kunci jawaban latihan tashrif secara acak.
SRS_002_03	Sistem dapat menerima masukan jawaban perubahan kata.
SRS_002_04	Sistem dapat melakukan koreksi terhadap jawaban tashrif .

Tabel 4.2 Tabel Persyaratan Non-Fungsional Sistem

Aspek	Deskripsi Kebutuhan
Usability	Perangkat lunak harus dapat digunakan dengan mudah oleh pengguna berdasarkan kriteria Navigasi menu, Informasi keluaran, Informasi kesalahan, Integritas Konseptual antarmuka dan Interaktivitas.
Maintainability	Perangkat lunak mampu beradaptasi pada perubahan algoritma rumus fiil dan penambahan jenis rumus yang baru dengan relatif mudah.

4.1.3 Pemodelan Use Case

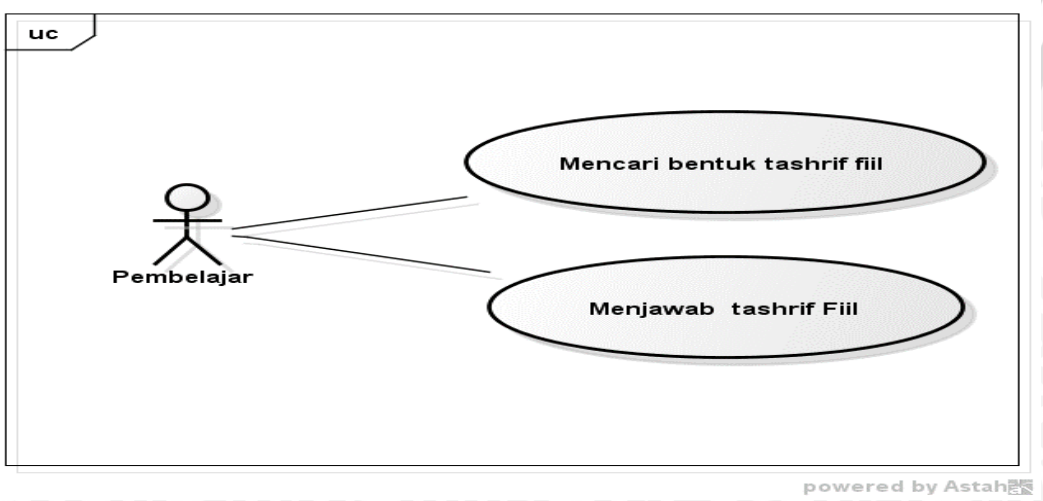
4.1.3.1 Identifikasi Aktor

Tahap ini mencoba untuk melakukan proses identifikasi terhadap pihak-pihak pengguna yang berinteraksi dengan sistem. Pada aplikasi pendukung pembelajaran bahasa Arab ini hanya terdapat 1 aktor yang menjalankan interaksi dengan sistem yaitu dinamakan sebagai "Pembelajar".

Pembelajar merupakan representasi dari pengguna yang sedang melakukan proses belajar ilmu sharaf. Pembelajar bisa jadi merupakan seorang pelajar dari sebuah lembaga pendidikan atau orang yang secara otodidak sedang menempuh pelajaran ilmu sharaf ditingkat awal. Selain itu Pembelajar juga bisa seorang pengajar ilmu sharaf yang ingin memanfaatkan teknologi dalam proses pembelajaran bahasa Arab.

4.1.3.2 Identifikasi dan Penggambaran Use Case

Dari penjelasan pengenalan pengguna ditetapkan bahwa purwarupa aplikasi yang dikembangkan akan memiliki dua tujuan utama yaitu Mencari bentuk tashrif fiil dan Menjawab tashrif fiil. Dua tujuan tersebut dapat dipetakan ke dalam dua *use cases* yang diilustrasikan pada gambar 4.2. Masing – masing *use case* menggambarkan alur proses pembelajaran ilmu sharaf secara sederhana.



Gambar 4.2 Diagram Use Case Sistem

Perancangan use case akan menggunakan RUP (*Rational Unified Process*) *style* dimana terdiri dari beberapa bagian yaitu *Brief Description, Actor, Basic Flow, Alternate Flow, Key Scenario, Pre-Condition, dan Post-Condition. Brief Description* menjelaskan secara singkat siapa yang berinteraksi dengan *Use Case* dan tujuan apa yang dicapai oleh *Use Case*.

Basic Flow menjelaskan langkah – langkah interaksi antara aktor dengan sistem yang pada umumnya akan terjadi. *Alternate flow* menjelaskan langkah-langkah pada beberapa kasus interaksi yang berbeda dari *basic flow* seperti varian operasional, kasus unik (*odd cases*) maupun *exception error*. *Key Scenarios* menyebutkan beberapa skenario yang terdiri dari *basic flow* dan *alternate flow* yang mungkin akan berjalan pada sistem. *Pre-Condition* menjelaskan kondisi awal yang mungkin perlu dipenuhi oleh *use case* sebelum menjalankan skenario. *Post-Condition* menjelaskan kondisi setelah menjalankan skenario use case.

Tabel 4.3 Tabel Istilah

No	Istilah	Arti
1	Fi'il	Kata kerja
2	Fi'il Madhi	Kata kerja masa lampau
3	Fi'il Mudhori'	Kata kerja masa sekarang
4	Fi'il Amr	Kata kerja perintah
5	Tashrif Istilahi	Perubahan dari bentuk fi'il madhi ke bentuk yang lainya.
6	Tashrif Lughowi	Perubahan bentuk Fi'il yang dipengaruhi oleh perubahan kata ganti/ subyek.
7	Wazan	Rumus perubahan fi'il
8	Mauzun	Contoh fi'il yang mengikuti wazan tertentu
9	Mujarrad	Kata kerja yang berasal dari 3 huruf atau 4 huruf tanpa tambahan.

Tabel 4.4 Spesifikasi *use case* – Mencari bentuk tashrif fiil

Brief Description	*Use case* untuk mencari hasil perubahan kata kerja dalam berbagai pola disertai dengan penjelasan rumus perubahan .
Actor	Pembelajar
Basic Flow	**1. Masuk Bagian pencarian** *Use case* dimulai ketika pembelajar memilih bagian pencarian perubahan kata kerja di bagian utama. Sistem menampilkan halaman pencarian. **2. Memasukkan kata kerja** Pembelajar memasukkan kata kerja *fiil madhi* bahasa Arab yang hendak dicari bentuk perubahanya. Sistem menampilkan hasil pencarian kata kerja *fiil tashrif istilahi* (*fiil madhi,fiil mudhori dan fiil amr*) yang mirip. **3. Melihat bentuk perubahan tashrif** Pembelajar memilih salah satu kata kerja yang ditampilkan sistem dari hasil tahapan sebelumnya. Setelah ditentukan kata kerjanya Sistem menampilkan seluruh hasil perubahan kata kerja berdasarkan jenis subyek atau *fiil tashrif lughowi*. *Use case* selesai.
Alternative Flow	1. **Kesalahan Masukan**. pada BF Memasukkan kata kerja, Pembelajar memberi masukan yang salah seperti memasukkan karakter non-Arab, jumlahnya lebih dari 3 karakter dan mengandung huruf *illat*. Sistem menampilkan pesan kesalahan/*error* pada pengguna. *Use case* kembali pada BF Memasukkan kata kerja. **2. Melihat bentuk lain** Pada BF melihat bentuk perubahan tashrif setelah pembelajar melihat hasilnya , dia juga bisa melihat bentuk yang lain jika memang terdapat lebih dari satu pilihan. Use case selesai.
Key Scenarios	*1. Basic Flow*

	2. *Basic Flow + AF Input error* 3. *Basic Flow + AF* Melihat bentuk lain
Pre-Condition	Tidak ada
Post-Condition	Notifikasi bahwa sukses menampilkan hasil perubahan kata kerja.

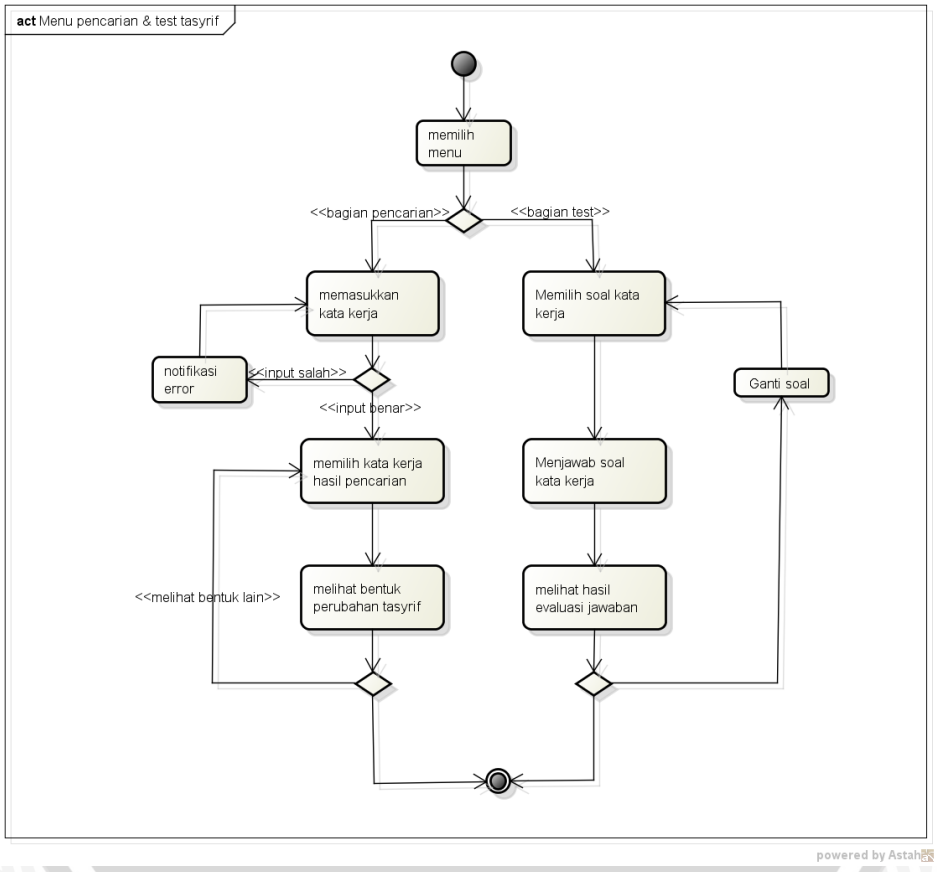
Tabel 4.5 Spesifikasi *use case* – Menjawab tashrif fiil

Brief Description	*Use case* bagi aktor pembelajar untuk menjawab hasil perubahan kata kerja dalam berbagai pola sesuai dengan kata kerja yang disediakan oleh sistem.
Actor	Pembelajar
Basic Flow	**1. Masuk bagian Test Tashrif** *Use case* dimulai ketika pembelajar memilih bagian test perubahan kata kerja tashrif dihalaman utama. Sistem menampilkan kata kerja dasar *fiil madhi* yang harus dijawab. **2. Memasukkan jawaban kata kerja** Pembelajar memasukkan jawaban kata kerja bahasa Arab yang dicari bentuk perubahanya berdasar kata yang ditampilkan sistem. **3. Melihat hasil penilaian jawaban** Pembelajar setelah selesai memasukkan jawaban maka akan mengkonfirmasi pada sistem untuk dilakukan penilaian . Sistem menampilkan hasil evaluasi jawaban perubahan kata kerja / tashrif pembelajar. Use case selesai.
Alternative Flow	1. **Ubah soal kata kerja**. Pada BF Masuk bagian test tashrif, Pembelajar mengganti soal kata kerja yang ingin dijawab. Sistem menampilkan pilihan kata kerja yang lain. *Use case* kembali pada BF Masuk Bagian Test Tashrif.
Key Scenarios	*1. Basic Flow*

	2. *Basic Flow + AF Ubah soal kata kerja*
Pre-Condition	Tidak ada
Post-Condition	Kembali pada bagiantest tashrif awal.

4.1.3.3 Penggambaran Use Case dengan Activity Diagram

Activity diagram digunakan untuk menjelaskan alur proses dari penggunaan sistem. Activity diagram pada bagian ini menjelaskan alur dari uraian use case sistem yaitu use case Mencari bentuk tashrif fiil dan Menjawab tashrif fiil.



Gambar 4.3 Activity Diagram Sistem

Pada gambar 4.3 menunjukan pada awal alur sistem pengguna (pembelajar) ditawari dalam 2 pilihan bagian yaitu : bagian pencarian bentuk tashrif dan bagian test menjawab tashrif kata kerja. Pada Bagian pencarian pembelajar akan memberikan *input* berupa kata kerja bahasa Arab pada sistem. Sistem terlebih dahulu akan melakukan pengecekan terhadap validitas *input* pengguna. *Input* dikategorikan valid jika memenuhi syarat sebagai berikut :

1. Berupa huruf Arab
2. berjumlah 3 (tiga) huruf atau disebut *Tsulatsy*.
3. Tidak mengandung huruf lemah (illat) : ا و ي
4. Tidak menggunakan tanda baca tasydid

Jika *input* bernilai benar maka masuk dalam proses selanjutnya, akan tetapi jika bernilai salah maka sistem akan memberikan notifikasi pada pengguna mengenai kesalahan *input*. Pada aktifitas selanjutnya sistem akan melakukan proses pencarian pada database atau bisa juga langsung melakukan proses analisa struktur huruf dalam kata untuk menentukan pola perubahan kata kerja. Sistem kemudian akan memberikan keluaran berupa satu atau lebih kata kerja dengan pola dasar tashrif istilahi (*Fiil Madhi, Mudhori dan Amr*) yang bisa dipilih oleh pengguna untuk mengetahui perubahan pola kata selanjutnya (tashrif lughowi). Pengguna setelah melakukan pemilihan kata kerja sistem melakukan proses analisa terhadap struktur huruf dan memberikan kembalian berupa pola kata kerja lanjutan beserta penjelasan perubahan polanya. Pengguna bisa memilih untuk melihat pola kata kerja lanjutan dari kata kerja dasar yang lainya dari proses sebelumnya jika ada atau berpindah menu.

Pada Bagian Latihan pengguna akan diuji dalam menjawab soal berupa kata kerja yang harus dicari pola perubahanya. Pada aktifitas memilih kata kerja Sistem akan secara acak menampilkan kata kerja yang dijadikan soal kepada pengguna. Pengguna akan memilih salah satu kata kerja tersebut atau mengganti soal pilihan kata kerja. Pada aktifitas menjawab soal kata kerja pengguna akan menjawab jenis-jenis hasil perubahan kata kerja yang telah dipilih sebelumnya.

Pada aktifitas melihat hasil evaluasi jawaban sistem melakukan pengecekan terhadap kebenaran jawaban dari pengguna kemudian memberikan evaluasi berupa skor dan juga pembenaran jika terdapat kesalahan pada jawaban pola perubahan kata kerja. Selanjutnya pengguna bisa mengakhiri aktifitas atau berpindah menu.

4.1.3.4 Pemetaan Persyaratan Fungsional dengan Use Case

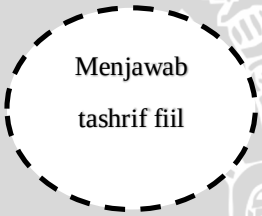
Tabel 4.6 Pemetaan *use case* – persyaratan fungsional

ID	Persyaratan	Use Case
SRS_001_01	Sistem dapat menerima masukan kata kerja fiil madhi dengan tanda baca maupun tanpa tanda baca.	Mencari bentuk tashrif fiil
SRS_001_02	Sistem dapat memfilter masukan yang tidak sesuai dengan syarat masukan.	Mencari bentuk tashrif fiil
SRS_001_03	Sistem dapat menampilkan data hasil perubahan kata kerja dalam bentuk mudhori dan amr serta penjelasan ejaan.	Mencari bentuk tashrif fiil
SRS_001_04	Sistem dapat menampilkan data hasil perubahan kata kerja berdasarkan perubahan subyek dan disertai penjelasan ejaan.	Mencari bentuk tashrif fiil
SRS_002_01	Sistem dapat menampilkan pilihan fiil madhi yang menjadi soal latihan tashrif.	Menjawab tashrif fiil
SRS_002_02	Sistem menampilkan kunci jawaban latihan tashrif secara acak.	Menjawab tashrif fiil
SRS_002_03	Sistem dapat menerima masukan jawaban perubahan kata.	Menjawab tashrif fiil
SRS_002_04	Sistem dapat melakukan koreksi terhadap jawaban tashrif .	Menjawab tashrif fiil

4.1.4 Use Case Realization

Use case realization merupakan perwujudan hasil dari proses analisis persyaratan yang kemudian akan digunakan sebagai acuan pada bagian perancangan sistem. Tabel berikut merupakan penjelasan dari kelas yang nantinya berpartisipasi dalam mewujudkan use case.

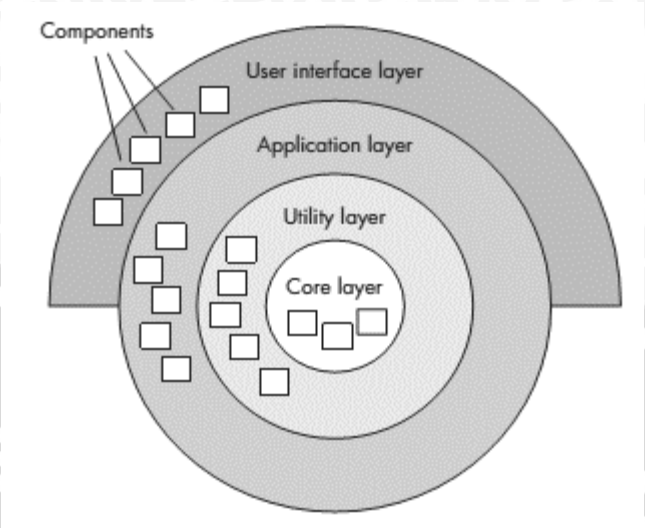
Tabel 4.7 Kolaborasi Kelas Use Case Realization

No	Use case Realization	Kelas
1		- *Screen1Controller* - *TrilateralModel* - *Fiil* - *Tasyrif* - *DBAccess*
2		- *Screen2Controller* - *Fiil* - *Tasyrif* - *DBAccess* - *TrilateralModel*

4.2 Perancangan

Perancangan perangkat lunak dengan menggunakan UML (*Unified Modelling Language*) akan dibagi dalam beberapa jenis yaitu : perancangan Arsitektur sistem, perancangan kelas (*static Class diagram*), perancangan interaksi (*Sequence diagram*) dan perancangan antarmuka. Perancangan arsitektur akan menjelaskan pola arsitektur mana yang dianggap relevan dan diadaptasi sesuai kebutuhan pengembangan aplikasi. Perancangan kelas akan disertai dengan penjelasan penggunaan design pattern yang melibatkan struktur dari kelas – kelas tertentu. Perancangan sequence diagram akan disesuaikan dengan use case yang menjelaskan interaksi objek dalam menjalankan skenario tertentu. Perancangan antarmuka menggambarkan *layout* komponen grafis pada fitur sistem

4.2.1 Perancangan Arsitektur Sistem

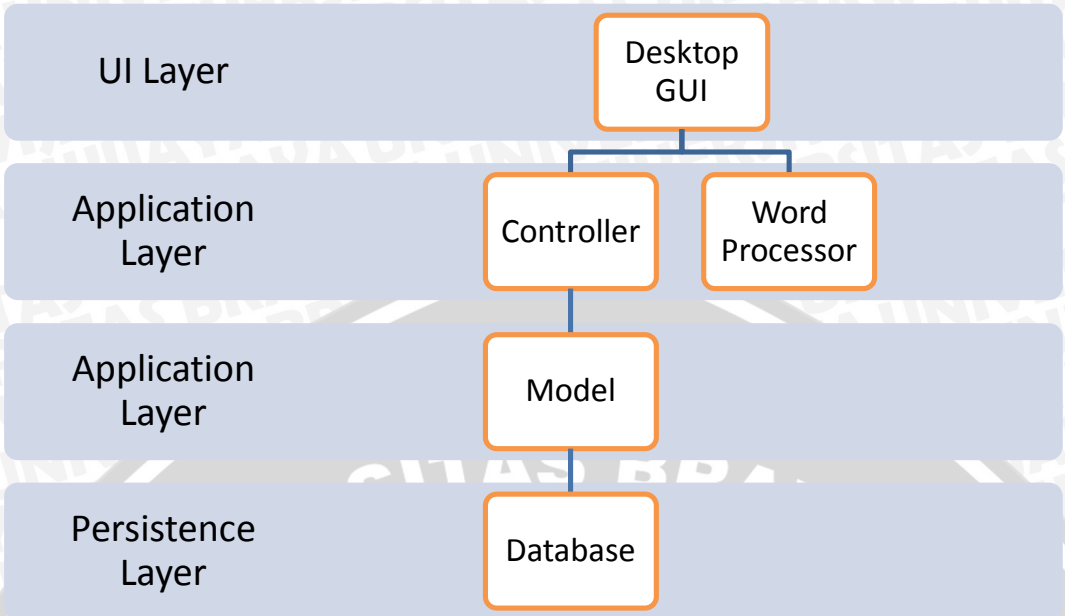


Gambar 4.4 Layered Architecture [PRE-10]

Arsitektur perangkat lunak merupakan sebuah representasi abstrak bangunan perangkat lunak yang mempunyai struktur tertentu dan disusun berdasarkan komponen-komponen yang saling berkaitan dan disesuaikan dengan kebutuhan sistem [PRE-10]. Gambar 4.6 Menunjukkan arsitektur perangkat lunak berjenis arsitektur berlapis (*layered architecture*).

Purwarupa aplikasi pendukung pembelajaran bahasa Arab menggunakan arsitektur yang diadaptasi dari jenis arsitektur berlapis dalam proses perancangan sistem. Pemilihan jenis arsitektur ini didasari pemikiran bahwa sistem yang mengadaptasinya melakukan pemisahan terhadap lapisan yang berbeda untuk presentasi (*user interface*), *application, utility* dan *core* (databases etc.). Arsitektur ini juga dinilai mendukung untuk pengembangan sistem secara bertahap karena setiap terjadi penambahan fungsionalitas dilapisan tertentu hanya akan mempengaruhi lapisan terdekat yang langsung berhubungan [SOM-11].

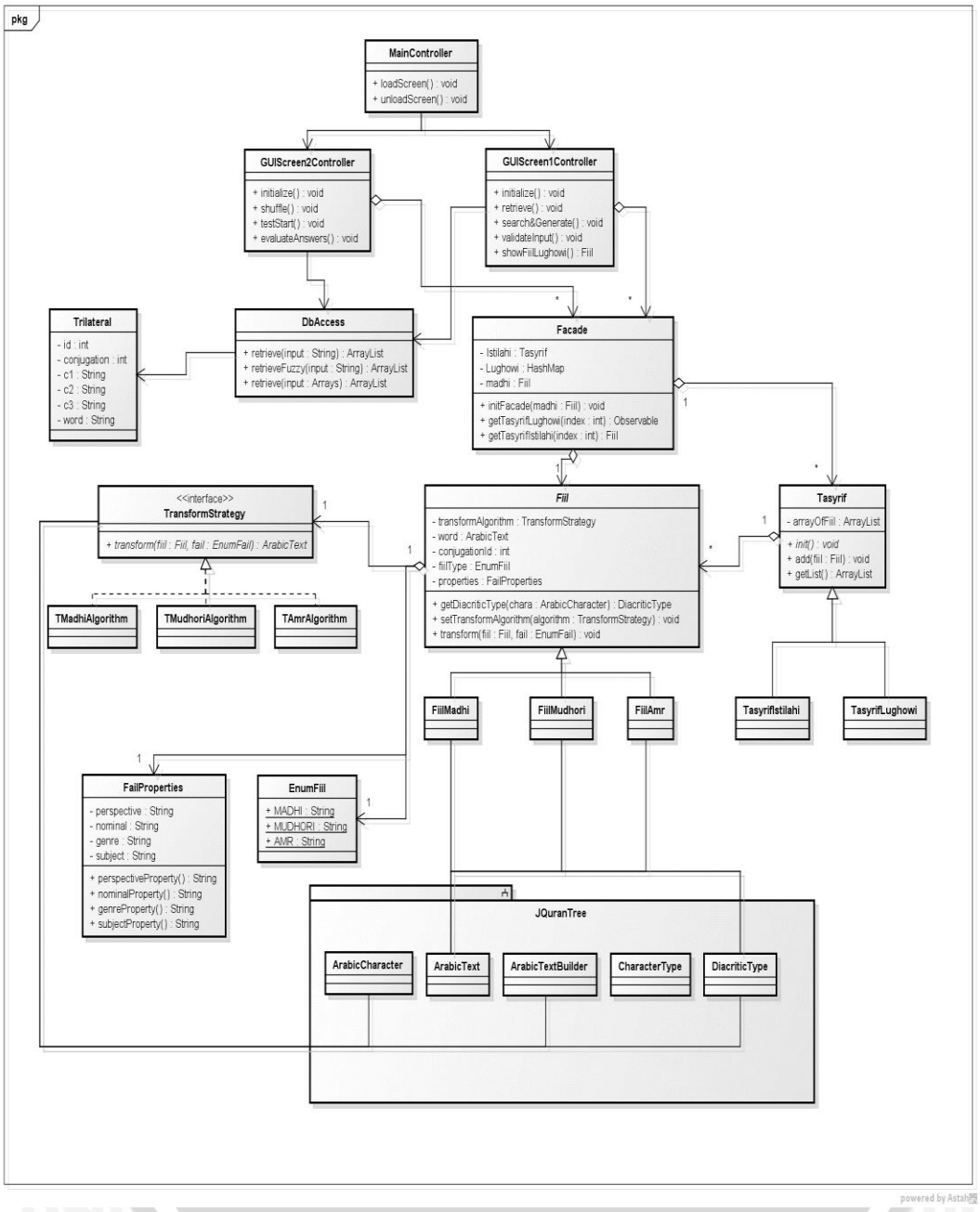
Adaptasi arsitektur ini pada lapisan *user interface* akan dibuat sebuah lapisan yang bernama *desktop graphic user interface* karena antarmuka aplikasi melalui *desktop*. Pada nantinya antarmuka ini berisi halaman yang disesuaikan dengan jumlah fungsi aplikasi pendukung pembelajaran bahasa Arab



Gambar 4.5 Implementasi Pola Layered Architecture

Pada lapisan *application* akan dibagi menjadi dua lapisan dimana lapisan yang pertama adalah *controller* dan *word processor*. Komponen *controller* akan mengatur jalan penampilan data dan informasi yang diminta oleh lapisan user interface. Komponen *word processor* adalah komponen pemroses kata dalam bahasa Arab yang mampu memisahkan antara karakter huruf Arab dengan tanda baca yang melekat pada karakter tersebut. Lapisan application berikutnya adalah model dimana menjadi komponen yang melakukan enkapsulasi data yang berasal dari database dan mengirimkan data tersebut melalui komponen *controller*. Lapisan yang terakhir diadaptasi adalah lapisan *core* dimana merupakan database yang nantinya berisi kamus kata kerja bahasa Arab beserta rumus perubahan. Komponen pada tiap lapisan pada tahap implementasi bisa menggunakan kembali komponen yang telah ada (*reuse*) pada platform tertentu dengan adaptasi sesuai dengan kebutuhan sistem yang dibuat. Komponen *Desktop GUI* yang akan digunakan pada sistem diharapkan mempunyai kemampuan untuk memisahkan antara tampilan dengan *logic* aplikasi sehingga memudahkan *maintenance code* dan juga memberi *user experience* yang baik. Mekanisme *persistence* dibutuhkan dalam hal penyimpanan data yang berkaitan dengan kebutuhan sistem dan bisa melakukan proses *mapping* dari tabel relasional ke dalam objek domain.

4.2.2 Perancangan Class Diagram



Gambar 4.6 Diagram Kelas sistem

Diagram kelas memberikan gambaran (diagram statis) tentang sistem dan relasi-relasi yang terdapat di dalamnya. Kelas-kelas yang telah teridentifikasi dapat memiliki hubungan antar kelas. Kelas-kelas juga dapat memiliki pewarisan dan ketergantungan terhadap komponen-komponen yang telah ditentukan.

Penggunaan *Design Patterns* dalam merancang sebuah sistem juga dianjurkan karena dengan menggunakanya mampu meningkatkan kualitas sebuah perangkat lunak dalam hal *reusability*, *maintainability*, *extensibility* serta bisa mengurangi waktu yang dibutuhkan dalam pengembangan perangkat lunak. Pada perancangan kelas ini menggunakan *Strategy pattern* dan *Facade pattern*.

4.2.2.1 Kelas dan Relasi

Secara umum struktur kelas pada sistem dibagi menjadi dua bagian besar yaitu kelas yang merepresentasikan kata kerja *Fi'il* dan koleksi kata kerja berdasar aturan tertentu yang disebut *Tashrif*. Hal ini didasari oleh konsep perancangan kelas berdasar domain pengetahuan menyesuaikan dengan konsep *fi'il* dalam ilmu Sharaf. Kelas abstrak Fiil mempunyai beberapa metode operasi serta atribut yang diturunkan pada sub-kelas seperti : Kelas FiilMadhi, FiilMudhori dan FiilAmr sedangkan kelas abstrak Tashrif juga mempunyai metode operasi yang diturunkan pada sub-kelasnya seperti : kelas TashrifIstilahi dan TashrifLughowi. JQuranTree merupakan sebuah API yang didalamnya terdapat kelas-kelas seperti ArabicText dan ArabicCharacter yang berguna dalam memodelkan kata kerja dalam kelas-kelas turunan Fiil.

1. Kelas Abstract Fiil

Nama Kelas	Fiil
Deskripsi	• Kelas yang berfungsi sebagai kelas induk atau abstraksi(*supertype*) untuk memodelkan sebuah kata kerja dasar (*Fi'il*) • Memiliki beberapa atribut dan juga metode yang nantinya akan diturunkan serta diimplementasikan pada sub-kelas.

2. Kelas Fiil Madhi

Nama Kelas	FiilMadhi
Deskripsi	• Kelas Turunan dari Fiil. • Memodelkan *fi'il madhi* (kata kerja lampau) yang digunakan sebagai dasar dalam mencari pola perubahan kata kerja lainya. • digunakan sebagai dasar dalam mencari pola perubahan kata kerja fi'il Madhi berdasarkan subyek

3. Kelas Fiil Mudhori

Nama Kelas	FiilMudhori
Deskripsi	• Kelas Turunan dari Fiil. • Memodelkan *fiil mudhori* (kata kerja masa sekarang) yang digunakan sebagai dasar dalam mencari pola perubahan kata kerja fi'il Amr. • digunakan sebagai dasar dalam mencari pola perubahan kata kerja fi'il Madhi berdasarkan subyek

4. Kelas Fiil Amr

Nama Kelas	FiAmr
Deskripsi	• Kelas Turunan dari Fiil. • Memodelkan *fi'il amr* (kata kerja perintah) yang digunakan sebagai dasar dalam mencari pola perubahan kata kerja fi'il Amr berdasarkan subyek.

5. Kelas Abstract Tashrif

Nama Kelas	AbstractFiilLughowi
Deskripsi	• Kelas yang berfungsi sebagai kelas induk untuk memodelkan kumpulan / koleksi banyak kata kerja sesuai dengan aturan konsep Fiil. Aturan tersebut secara umum terbagi menjadi 2 yaitu perubahan berdasarkan jenis kata kerja dan perubahan kata kerja berdasarkan subyek. • Kelas ini mempunyai atribut yang merepresentasikan kumpulan / koleksi dari kelas Fiil.

6. Kelas Tashrif Istilahi

Nama Kelas	TashrifIstilahi
Deskripsi	• Kelas Turunan dari Tashrif. • Memodelkan kumpulan kata kerja yang berdasarkan aturan perubahan berdasar jenis kata kerja atau juga perubahan “horizontal”.

7. Kelas Tashrif Lughowi

Nama Kelas	TashrifLughowi
Deskripsi	• Kelas Turunan dari Tashrif. • Memodelkan kumpulan kata kerja yang berdasarkan aturan perubahan berdasar jenis subyek atau juga perubahan “vertikal”.

8. Kelas Trilateral

Nama Kelas	Trilateral
Deskripsi	• Kelas ini merupakan model untuk table yang berada di database yang menyimpan kata kerja dengan pola fa-ain-lam dengan menggunakan tanda baca fathah pada ain fiilnya .

9. Kelas GUI Controller Layar Pencarian

Nama Kelas	GUIScreen1Controller
Deskripsi	• Kelas ini merupakan controller yang nantinya menyediakan operasi-operasi bagi main class yang banyak berhubungan dengan mengatur Graphic User Interface (GUI) pada fitur pencarian kata kerja .

10. Kelas GUI Controller Layar Test

Nama Kelas	GUIScreen2Controller
Deskripsi	• Kelas ini merupakan controller yang nantinya menyediakan operasi-operasi bagi main class yang banyak berhubungan dengan mengatur Graphic User Interface (GUI) pada fitur test menjawab kata kerja .

11. Kelas Main Controller

Nama Kelas	MainController
Deskripsi	• Kelas ini merupakan controller yang nantinya menyediakan operasi dalam mengatur layar fitur mana dari program yang akan ditampilkan kepada pengguna .

12. Kelas Facade

Nama Kelas	Facade
Deskripsi	• Kelas ini merupakan gabungan dari kelas-kelas Fiil yang berusaha menjadi fasilitator/antarmuka bagi kelas GUIController sehingga bisa menyembunyikan detail pengoperasian dan implementasi yang dibutuhkan tanpa harus mengekspos subkelas didalamnya.

13. Interface Transform Strategy

Nama Kelas	TransformStrategy
Deskripsi	• Merupakan sebuah java interface yang digunakan sebagai abstraksi untuk kumpulan Algoritma perubahan / transformasi kata kerja

14. Kelas Transform Madhi Algorithm

Nama Kelas	TMadhiAlgorithm
Deskripsi	• Merupakan kelas yang mengimplementasikan interface TransformStrategy. • digunakan sebagai detail implementasi Algoritma transformasi kata kerja fi'il madhi.

15. Kelas Transform Mudhori Algorithm

Nama Kelas	TMudhoriAlgorithm
Deskripsi	• Merupakan kelas yang mengimplementasikan interface TransformStrategy. • digunakan sebagai detail implementasi Algoritma transformasi kata kerja fi'il mudhori.

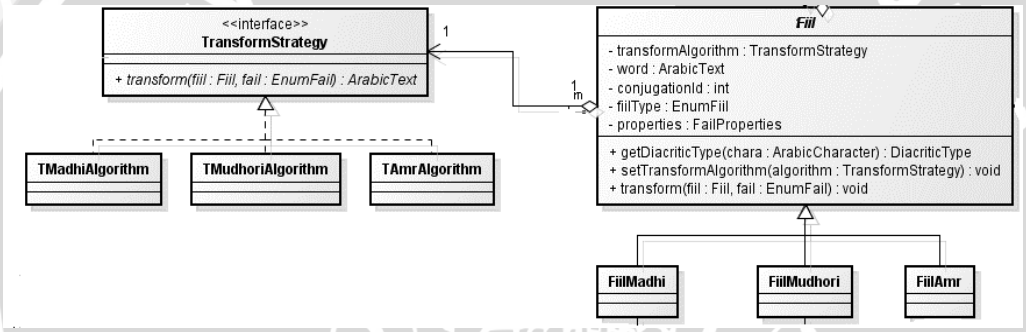
16. Kelas Transform Amr Algorithm

Nama Kelas	TAmrAlgorithm
Deskripsi	• Merupakan kelas yang mengimplementasikan interface TransformStrategy. • digunakan sebagai detail implementasi Algoritma transformasi kata kerja fi'il Amr.

17. Kelas Transform Amr Algorithm

Nama Kelas	TAmrAlgorithm
Deskripsi	• Merupakan kelas yang mengimplementasikan interface TransformStrategy. • digunakan sebagai detail implementasi Algoritma transformasi kata kerja fi'il Amr.

4.2.2.2 Pola Perancangan

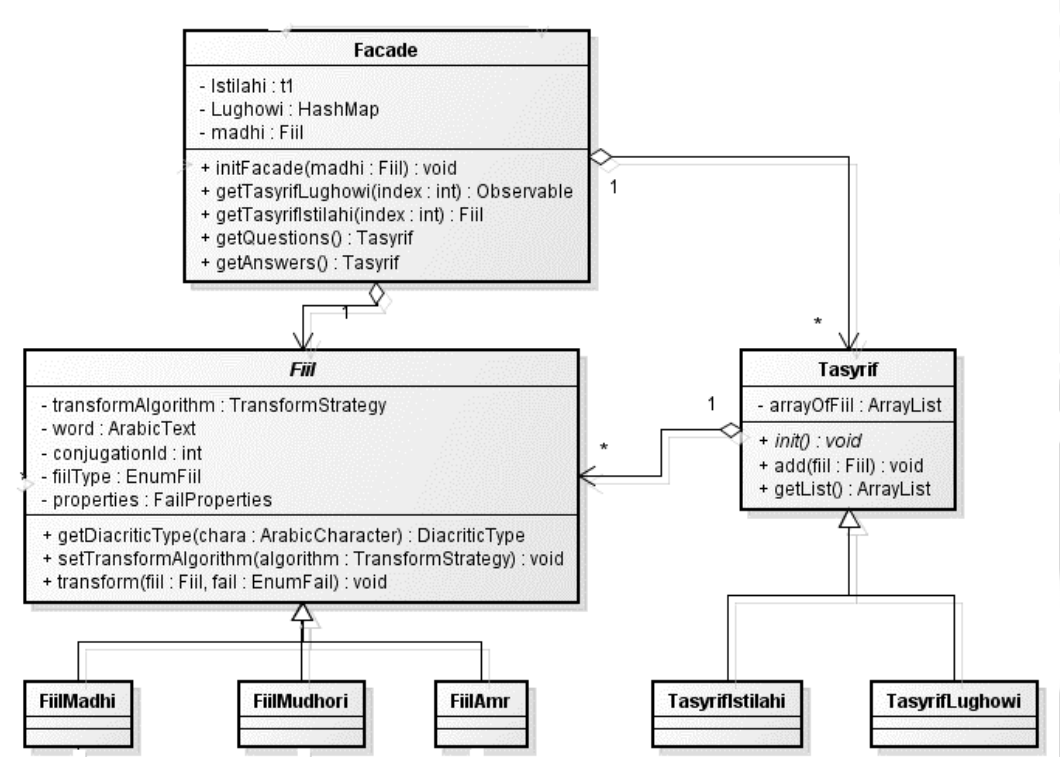


Gambar 4.7 Diagram Kelas : Pola Strategy

Perancangan kelas dalam sistem ini menggunakan *Strategy design pattern* dimana menggunakan interface TransformStrategy yang didalamnya terdapat metode algoritma perubahan kata kerja yang dikemudian diimplementasikan oleh kelas-kelas turunan yaitu TmadhiAlgorithm untuk algoritma fi'il madhi, TmudhoriAlgorithm untuk algoritma fi'il mudhori dan TamrAlgorithm untuk algoritma fi'il Amr seperti pada gambar 4.7. Penggunaan *pattern* didasari alasan sebagai berikut:

a. Menurut buku Gang of Four [GAM-95], buku yang menjadi rujukan konsep *design pattern*, *Strategy pattern* digunakan ketika mempunyai kode pemrograman yang mudah berubah (*behaviour*) dan bisa dipisahkan dari kelas utama dan pada kasus pengembangan perangkat lunak ini adalah kode algoritma transformasi fi'il.

b. Bisa melakukan pergantian algoritma saat run time untuk mendapatkan transformasi kata kerja fi'il yang berbeda.
c. Menghindari percampuran antara kode kelas utama dengan algoritma.

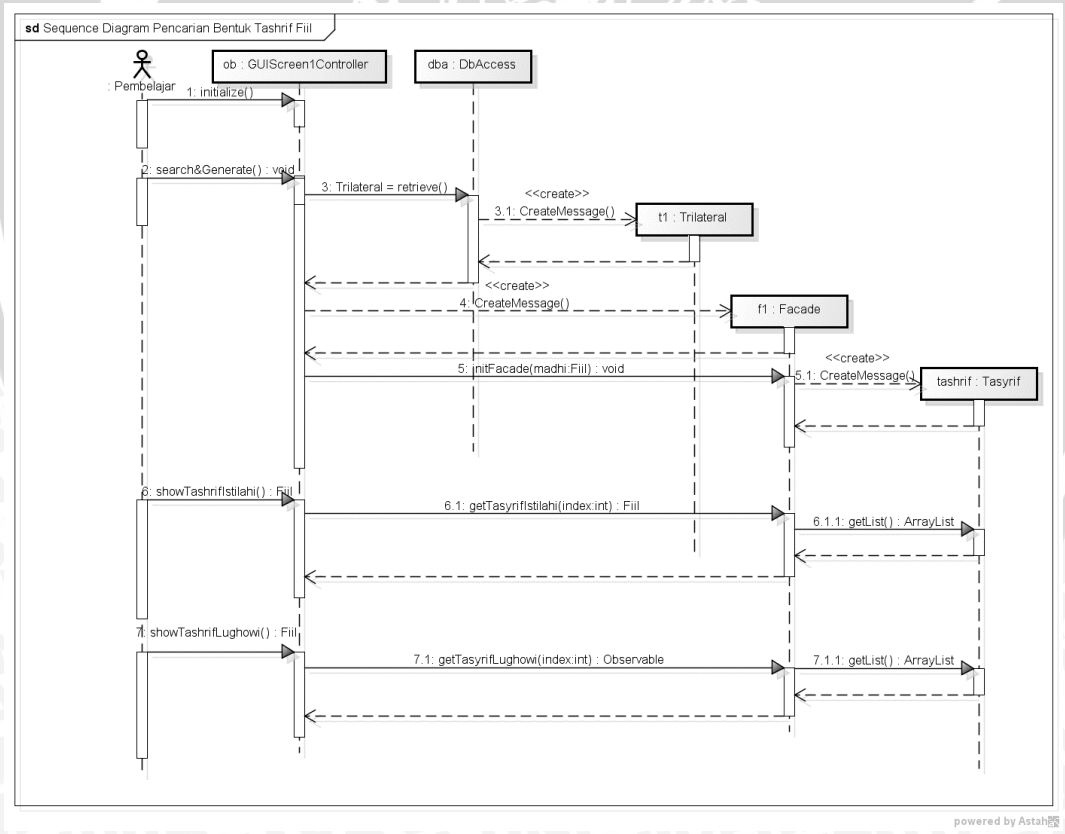


Gambar 4.8 Diagram Kelas : Pola Facade

Kelas Facade pada gambar 4.8 melakukan kolaborasi antara kelas Fiil dengan Tasyrif didalam metode initFacade(). Metode initFacade() selalu menjadi metode yang dieksekusi sebelum bisa melakukan operasi metode yang lain. Hal ini dikarenakan hasil proses metode tersebut mempengaruhi keberadaan nilai kembalian pada metode yang lain di kelas Facade. client GUI (*Graphic User Interface*) akan menggunakan kelas Facade sebagai antarmuka dalam melakukan operasi dan data-data yang dibutuhkan / diminta .

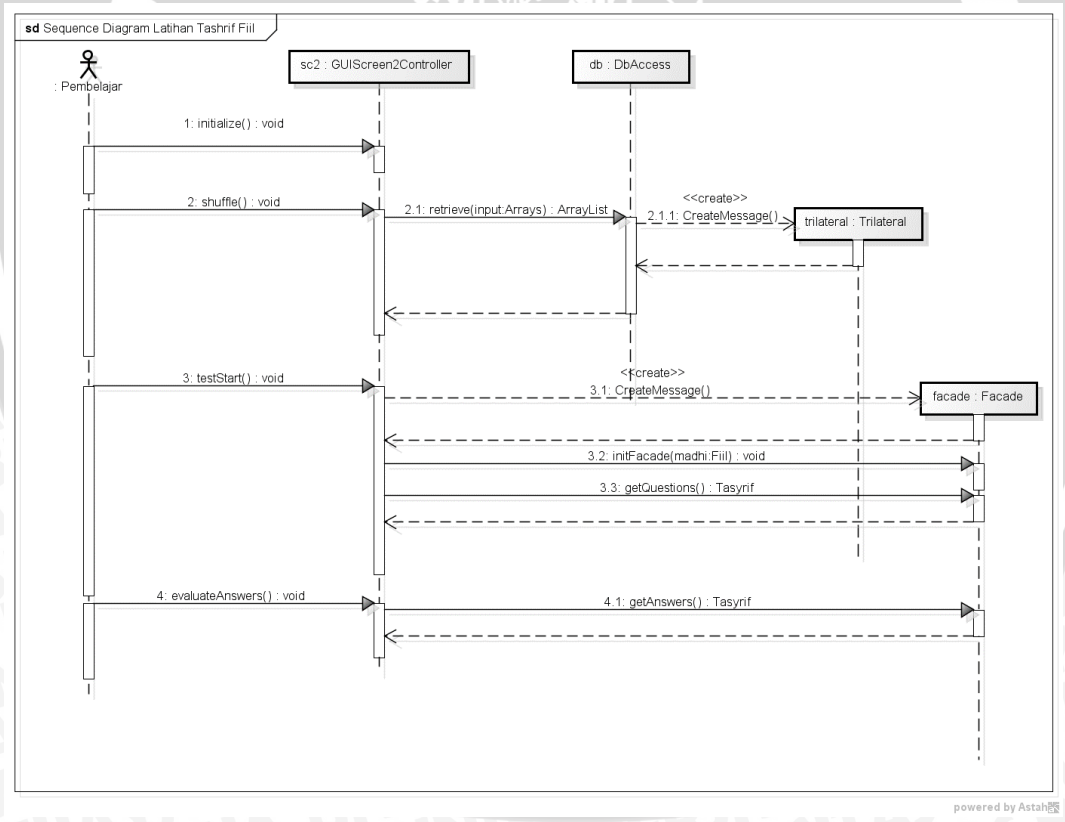
4.2.3 Perancangan Sequence Diagram

Sequence diagram digunakan untuk menggambarkan interaksi antar objek yang disusun dalam urutan waktu. *Sequence diagram* berupa skenario atau rangkaian langkah-langkah yang dilakukan dari sebuah kejadian untuk menghasilkan keluaran tertentu. Skenario pada *sequence diagram* ini menjelaskan interaksi antar kelas dalam sistem yang mana telah diuraikan berdasar *Use Case Realization.* Skenario dikelompokkan menjadi dua bagian utama yaitu Use case Mencari bentuk tashrif fiil dan Menjawab tashrif fiil. Pada bagian pencarian akan dijelaskan proses interaksi antar objek pada aktifitas Memasukkan kata kerja, memilih kata kerja dan memilih bentuk perubahan sedangkan pada bagian Test akan dijelaskan proses interaksi antar kelas pada aktifitas memilih soal kata kerja, menjawab soal kata kerja dan evaluasi jawaban pengguna.



Gambar 4.9 Diagram Interaksi Mencari bentuk tashrif fiil

Pada diagram Gambar 4.9 menjelaskan kelas yang berinteraksi pada aktifitas bagian pencarian. Kelas GUIController pada awalnya akan mengirim pesan pada objek Trilateral untuk mencari keberadaan kata kerja yang telah dimasukkan oleh pengguna. Kelas Trilateral yang merupakan kelas model dari *database* melakukan pencarian dalam database. Hasil pencarian tersebut lantas menjadi parameter acuan bagi GUIController dalam mengirim pesan dikelas Facade. Objek Kelas Tashrif pada saat objek Facade mendapat pesan dari kelas GUIScreen1Controller melalui metode searchGenerate(). Interaksi pada objek kelas Facade ketika mengirimkan pesan pada objek kelas Tashrif menjalankan metode init(). Kemudian Didalam kelas Tashrif melakukan proses instansiasi objek kelas Fiil secara berulang. Metode getList() digunakan untuk mengakses koleksi objek Fiil didalam kelas Tashrif .



Gambar 4.10 Diagram Interaksi Latihan Tashrif Fiil

Gambar 4.10 menjelaskan interaksi antar kelas pada use case Latihan Tashrif Fiil. Pada awal interaksi user otomatis mengaktifkan metode initialize() pada kelas GUIScreen2Controller yang menyiapkan komponen *graphic user interface*. Metode shuffle bertujuan mengacak soal kata kerja yang akan ditawarkan pada user untuk dikerjakan. Di dalam metode shuffle memberi pesan pada objek kelas DBAccess untuk mengakses database yang bertujuan mendapatkan objek dari kelas Trilateral dan hasil ini kemudian dilanjutkan pada kelas GUIScreen2Controller untuk diproses lebih lanjut. Setelah user memilih soal yang akan dikerjakan maka dia memberi pesan pada kelas GUIScreen2Controller agar melakukan pembuatan objek dari kelas facade. Metode evaluateAnswers() nantinya akan mengecek kebenaran hasil jawaban user apakah sudah sesuai dengan kunci jawaban.

4.2.4 Perancangan Rumus Fiil

Perancangan basis data rumus pada sistem yang dibuat terdiri kumpulan data kata kerja dan rumus bahasa Arab. Kata kerja tersebut didefinisikan dalam tabel basis data dengan memecah struktur kata tersebut per karakter. Batasan masalah pada skripsi ini adalah kata kerja dengan 3 huruf maka basis data hanya terdiri dari kata kerja 3 huruf (Trilateral).

Sesuai dengan batasan masalah yang telah disampaikan bahwa sistem ini akan mengimplementasikan kata kerja dengan properti sebagai berikut :

- 3 huruf (*Tsulasy*)
- *Mujarrad* (tanpa ada tambahan huruf)
- *Salim* (selamat dari huruf Illat)

Kemudian pola-pola aturan yang nantinya akan diimplementasikan adalah :

- Fi'il Madhi beserta perubahan kata berdasarkan subyek.
- Fi'il Mudhori beserta perubahan kata berdasarkan subyek.
- Fi'il Amr beserta perubahan kata berdasarkan subyek.

Berikut pembagian aturanya beserta kodifikasinya

1. Fi'il Madhi

   Karena hanya kata yang *mujarrad*(tanpa tambahan huruf) maka ada 3 varian :

Tabel 4.8 Tabel Wazan Fi'il Madhi

No	Wazan	Mauzun
1	فَعَلَ	نَصَرَ
2	فَعِلَ	عَلِمَ
3	فَعُلَ	حَسُنَ

2. Fi'il Madhi ke Fi'il Mudhori'

Tabel 4.9 Tabel Wazan Madhi-Mudhori'

Nomor	wazan madhi	Wazan mudhori
1	فَعَلَ	يَفْعَلُ
2	فَعَلَ	يَفْعِلُ
3	فَعَلَ	يَفْعُلُ
4	فَعِلَ	يَفْعَلُ
5	فَعِلَ	يَفْعِلُ
6	فَعُلَ	يَفْعُلُ

Penjelasan Aturan

- 1 : wazan ini dapat digunakan ketika ain fi'il atau lam fi'il terdiri dari salah satu huruf halaq (خ ح غ ع ه أ)
- 2 dan 3 : wazan ini berlaku pada bentuk umum fa'ala dan bisa bertransformasi pada salah satu atau kedua bentuk.
- 4 : wazan ini berlaku untuk bentuk fa'ila pada umumnya
- 5 : wazan ini berlaku pada beberapa fi'il yang *syadz* (diluar kaidah)
- 6 : wazan ini berlaku pada ain fi'il yang berharkat dlamma

3. Fi'il Mudhori' ke Fi'il Amr

Tabel 4.10 Tabel Wazan Mudhori' - Amr

Nomor	wazan Mudhori	Wazan Amr
1	يَفْعَلُ	اِفْعَلْ
2	يَفْعِلُ	اِفْعِلْ
3	يَفْعُلُ	اُفْعُلْ

Penjelasan Aturan

- 1 : Wazan ini berlaku dengan melihat ain fi'il pada bentuk mudhori yang kemudian dijaga pada bentuk Amr
- 2: prinsipnya sama dengan 3.1
- 3 : Wazan ini sedikit berbeda karena ketika ain fi'ilnya dlammah maka alif diawal kata mengikuti harakat dlamma.

4. Tashrif Lughowi

Pola aturan ini sebenarnya menunjukan perubahan kata kerja yang disebabkan perbedaan subjek (*Damir*). Subjek dalam bahasa Arab memiliki 3 kategori :

a. Dari Jumlah : tunggal , dual dan jamak
b. Dari Perspektif : orang pertama, kedua dan ketiga
c. Dari Kelamin : Pria dan Wanita

Fi'il Madhi dan Fi'il mudhori' dapat menurunkan semua varian subjek yang ada kecuali Fi'il Amr yang hanya berlaku pada perspektif orang kedua. Dalam proses kodifikasi pola aturan diurutkan berdasarkan Perspektif , Jumlah kemudian Kelamin.

Tabel 4.11 Tabel aturan Tashrif Lughowi

Subyek	Fi'il Madhi	Fi'il Mudhori	Fi'il Amr
Dia pria	1.x.1	2.x.1	-
Dua pria	1.x.2	2.x.2	-
Banyak pria	1.x.3	2.x.3	-
Dia wanita	1.x.4	2.x.4	-

Dua wanita	1.x.5	2.x.5	-
Banyak wanita	1.x.6	2.x.6	-
Kamu pria	1.x.7	2.x.7	3.x.1
Kamu dua pria	1.x.8	2.x.8	3.x.2
Kalian pria	1.x.9	2.x.9	3.x.3
Kamu wanita	1.x.10	2.x.10	3.x.4
Kamu dua wanita	1.x.11	2.x.11	3.x.5
Kalian wanita	1.x.12	2.x.12	3.x.6
Saya	1.x.13	2.x.13	-
Kami	1.x.14	2.x.14	-

Berikut adalah susunan detail rumus perubahan kata kerja tiap subyek :

C(x) : karakter ke-x dari kata kerja

H(x) : Tanda baca yang melekat pada karakter ke-x dari kata kerja

Tabel 4.12 Tabel aturan Tashrif Lughowi Fiil Madhi

No	Subyek	Kode	Rumus
1	Dia pria	1.x.1	C1+H1+C2+H2+C3+H3
2	Dua pria	1.x.2	C1+H1+C2+H2+C3+H3+ا
3	Banyak pria	1.x.3	C1+H1+C2+H2+C3+ْ+وْ+ا
4	Dia wanita	1.x.4	C1+H1+C2+H2+C3+H3+ث
5	Dua wanita	1.x.5	C1+H1+C2+H2+C3+H3+تَ+ا
6	Banyak wanita	1.x.6	C1+H1+C2+H2+C3+ْ+نَ
7	Kamu pria	1.x.7	C1+H1+C2+H2+C3+ْ+تَ
8	Kalian dua pria	1.x.8	C1+H1+C2+H2+C3+ْ+تُ+مَ+ا
9	Kalian pria	1.x.9	C1+H1+C2+H2+C3+ْ+تُ+مْ
10	Kamu wanita	1.x.10	C1+H1+C2+H2+C3+ْ+تِ
11	Kalian dua wanita	1.x.11	C1+H1+C2+H2+C3+ْ+تُ+مَ+ا
12	Kalian wanita	1.x.12	C1+H1+C2+H2+C3+ْ+تُ+نَّ
13	Saya	1.x.13	C1+H1+C2+H2+C3+ْ+تُ

14	Kami	1.x.14	C1+H1+C2+H2+C3+ْ+نَ+ا

Tabel 4.13 Tabel aturan Tashrif Lughowi Fiil Mudhori'

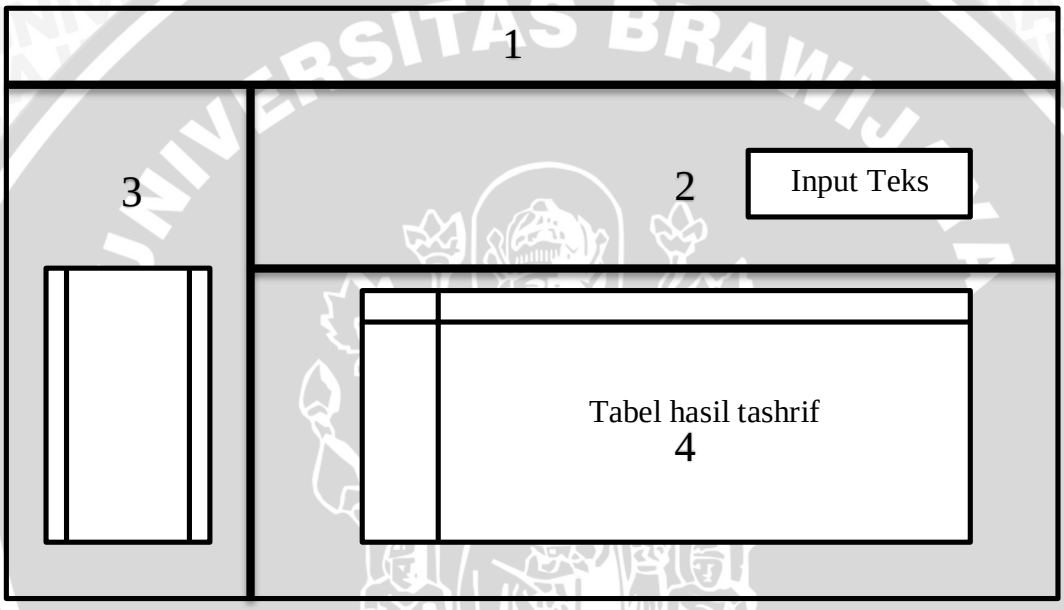
No	Subyek	Kode	Rumus
1	Dia pria	2.x.1	يَ+C1+ْ+C2+H2+C3+H3
2	Dua pria	2.x.2	يَ+C1+ْ+C2+H2+C3+َ+ا+نِ
3	Banyak pria	2.x.3	يَ+C1+ْ+C2+H2+C3+ُ+وْ+نَ
4	Dia wanita	2.x.4	تَ+C1+ْ+C2+H2+C3+H3
5	Dua wanita	2.x.5	تَ+C1+ْ+C2+H2+C3+َ+ا+نِ
6	Banyak wanita	2.x.6	يَ+H1+ْ+H2+C3+ْ+نَ
7	Kamu pria	2.x.7	تَ+C1+ْ+C2+H2+C3+H3
8	Kalian dua pria	2.x.8	تَ+C1+ْ+C2+H2+C3+َ+ا+نِ
9	Kalian pria	2.x.9	تَ+C1+ْ+C2+H2+C3+H3 +وْ+نَ
10	Kamu wanita	2.x.10	تَ+C1+ْ+C2+H2+C3+ِ+يْ+نَ
11	Kalian dua wanita	2.x.11	تَ+C1+ْ+C2+H2+C3+َ+ا+نِ
12	Kalian wanita	2.x.12	تَ+C1+ْ+C2+H2+C3+ْ+نَ
13	Saya	2.x.13	أَ+C1+ْ+C2+H2+C3+H3
14	Kami	2.x.14	نَ+C1+ْ+C2+H2+C3+H3

Tabel 4.14 Tabel aturan Tashrif Lughowi Fiil Amr

No	Subyek	Kode	Rumus
1	Kamu pria	3.x.1	ا+H0+C1+ْ+C2+H2+C3+ْ
2	Kalian dua pria	3.x.2	ا+H0+C1+ْ+C2+H2+C3+َ+ا
3	Kalian pria	3.x.3	ا+H0+C1 +ْ+C2+H2+C3+ُ+وْ+ا
4	Kamu wanita	3.x.4	ا+H0+C1+ْ+C2+H2+C3+ِ+يْ
5	Kalian dua wanita	3.x.5	ا+H0+C1+ْ+C2+H2+C3+َ+ا
6	Kalian wanita	3.x.6	ا+H0+C1+ْ+C2+H2+C3+ْ+نَ

4.2.5 Perancangan Antarmuka

Desain Antarmuka mempunyai tujuan untuk membuat sebuah medium komunikasi yang efektif antara manusia dengan komputer. Sebuah produk perangkat lunak dapat dikatakan berhasil ketika memiliki *usability* yang baik . *Usability* merupakan ukuran kualitatif atas kemudahan dan efisiensi dari seorang pengguna (user) yang mampu menggunakan fungsi dan fitur yang ditawarkan produk perangkat lunak [PRE-10] .

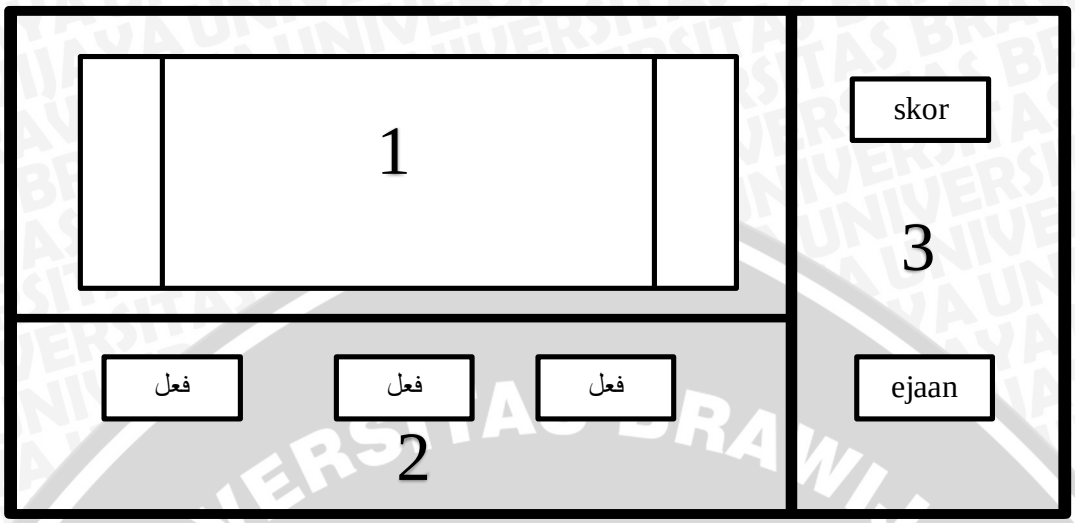


Gambar 4.11 Rancangan Antarmuka layar fitur pencarian

Gambar 4.15 merupakan rancangan antarmuka untuk fitur pencarian kata kerja bahasa Arab. Keterangan gambar sebagai berikut :

1. Area no.1 merupakan menubar yang nantinya digunakan untuk menampung fungsi jendela aplikasi seperti perpindahan antar layar fitur.
2. Area no.2 merupakan bagian dimana pengguna memberikan *input* kata kerja yang hendak dicari .
3. Area no.3 merupakan bagian yang nantinya memberikan informasi penjelasan mengenai rumus (wazan) dan hal lain yang dimiliki suatu kata kerja.

4. Area no.4 merupakan bagian yang berisi tabel hasil perubahan kata kerja.



Gambar 4.12 Rancangan Antarmuka layar fitur Test

Gambar 4.16 merupakan rancangan antarmuka untuk fitur test , dimana pengguna akan menjawab soal perubahan kata kerja (tashrif). Keterangan gambar sebagai berikut :

1. Area no.1 merupakan bagian layar dimana pengguna akan menjawab soal perubahan kata kerja dan koreksi jawaban yang salah.
2. Area no.2 merupakan kunci jawaban yang secara acak akan diberikan kepada pengguna dalam menjawab test.
3. Area no.3 merupakan bagian layar yang berisi hasil evaluasi test yang telah dilakukan oleh pengguna dan teks bantuan ejaan.

BAB V
IMPLEMENTASI

5.1 Spesifikasi Lingkungan Sistem

Hasil analisis persyaratan perangkat lunak yang telah dilakukan pada bagian bab IV dijadikan sebagai panduan dalam proses pengimplementasian perangkat lunak pendukung pembelajaran bahasa Arab. Spesifikasi lingkungan sistem yang digunakan dalam proses implementasi baik itu perangkat keras (*hardware*) maupun perangkat lunak (*software*) akan dijelaskan pada bagian tabel 5.1 dan tabel 5.2.

Tabel 5.1 Spesifikasi lingkungan perangkat keras

Nama Komponen	Spesifikasi
Prosesor	Intel Core i5-3210M 2.50 GHz
Memori (RAM)	4 GB
Harddisk Storage	750 GB
Tipe	ASUS N46VM
Kartu Grafis	NVIDIA 630M 2 GB

Tabel 5.2 Spesifikasi lingkungan perangkat lunak

Nama Komponen	Spesifikasi
Sistem Operasi	Windows 8 64-bit
Compiler	Netbeans 8.0.1
Java Development Kit (JDK) dan JRE	1.8.0

5.2 Batasan Implementasi

Beberapa batasan-batasan dalam implementasi sistem adalah :

1. Pembuatan aplikasi pendukung pembelajaran bahasa Arab dikerjakan dengan bahasa pemrograman Java dan menggunakan sistem manajemen basis data MySQL .

2. Aplikasi berupa *desktop application*.

5.3 Implementasi dengan Komponen

Aplikasi pendukung pembelajaran Bahasa Arab mempunyai dua fitur utama yaitu fitur pencarian hasil perubahan kata kerja tashrif dan fitur test menjawab soal perubahan kata kerja tashrif. Proses implementasi aplikasi juga menyertakan komponen-komponen yang dianggap dapat membantu pengembangan sehingga dapat memenuhi kebutuhan sistem dan juga meringkas waktu yang diperlukan. Komponen adalah elemen perangkat lunak yang mempunyai standar sesuai dengan model komponen, bisa secara independen digunakan, dan mempunyai *interfaces* (antarmuka) yang menyediakan akses fungsi tanpa harus mengekspos internal detail komponen tersebut [SOM-11].

Berikut ini merupakan tabel yang menjelaskan tugas dari masing-masing komponen yang digunakan dalam membangun aplikasi :

Tabel 5.3 Tabel Komponen perangkat lunak

No	Nama Komponen	Fungsi
1	JquranTree	Merubah format tipe data string ke dalam Arabic text yang kemudian dapat diolah lebih lanjut oleh sistem.
2	Hibernate	Menjadi layer pengolahan basis data pada program Java karena sistem memanfaatkan MySQL sebagai manajemen basis data.
3	Hibernate Search dan Lucene indexer	Menyediakan fungsi Konfigurasi Full Text Search.
4	JavaFX	Pembuatan Graphic User Interface

Dalam bagian implementasi akan dijelaskan potongan kode baris dari tiap kelas yang pokok beserta dengan penjelasan penggunaan design pattern dan komponen dari library atau framework tertentu.

5.3.1 Implementasi Strategy Pattern

Untuk bisa menjelaskan bagaimana implementasi strategy pattern pada sistem maka perlu dijelaskan kode yang ada pada kelas Fiil beserta kelas turunanya dan Interface TransformStrategy.

5.3.1.1 Kelas Abstrak Fiil dan Turunan

```
1   public abstract class Fiil {
2       int characters;
3       EnumFiil fiilType;
4       ArabicText word;
5       int conjugationId;
6       FailProperties properties;
7       TransformStrategy transformAlgorithm;
8
9       public void Transform(Fiil a, EnumFail fail) {
10          this.word = this.transformAlgorithm.transformMain(a, fail);
11      }
12      public StringProperty wordProperty() {
13          StringProperty kata = new
                            SimpleStringProperty(this.word.toString());
14          return kata;
15      }
16      public StringProperty perspectiveProperty() {
17          return properties.PerspektifProperty();
18      }
19      public StringProperty nominalProperty() {
20          return properties.NominalProperty();
21      }
22      public StringProperty genreProperty() {
23          return properties.GenreProperty();
24      }
25      public StringProperty dlamirProperty() {
26          return properties.padananProperty();
27      }
28    public static DiacriticType getDiacriticType(ArabicCharacter a){
29          DiacriticType d;
30          if (a.isDamma()) {
31              d = DiacriticType.Damma;
32          } else if (a.isFatha()) {
33              d = DiacriticType.Fatha;
34          } else if (a.isKasra()) {
```

```
35              d = DiacriticType.Kasra;
36          } else if (a.isSukun()) {
37              d = DiacriticType.Sukun;
38          } else {
39              d = DiacriticType.Fathatan;
40          }
41          return d;
42      }
43      public abstract ArabicText toMadhi();
44      public abstract ArabicText toMudhori();
45      public abstract ArabicText toAmr();
```

Gambar 5.1 Implementasi Kelas Fiil

Kelas Fiil merupakan abstract class (*Supertype*) yang merepresentasikan objek kata kerja bahasa Arab. Kelas Fiil nantinya akan memiliki kelas turunan yang bisa memberi detail implementasi pada metode abstrak yang berbeda di tiap kelas turunanya yaitu Kelas FiilMadhi, FiilMudhori dan FiilAmr. Keterangan berdasarkan baris kode adalah sebagai berikut :

1. Baris 2-7 merupakan deklarasi *field*/atribut kelas. Pada atribut transformAlgorithm yang bertipe TransformStrategy merupakan bagian implementasi *strategy pattern*.
2. Baris 9-11 merupakan metode transform yang nantinya memanggil metode dari atribut transformAlgorithm
3. Baris 12-27 merupakan metode akses (*getter*) untuk mendapatkan informasi mengenai objek Fiil.
4. Baris 28-42 merupakan metode statis yang berfungsi untuk mendapatkan jenis tanda baca (harakat) dari karakter Arab.
5. Baris 43-45 merupakan metode abstrak yang belum didefinisikan. Tiap kelas turunan bertugas untuk memberikan implementasi pada metode ini. Secara umum fungsi dari metode-metode ini adalah memberikan kemampuan pada objek Fiil menghasilkan kata kerja pada jenis fiil yang berbeda. Contohnya akan lebih mudah dijelaskan pada kelas turunan.

Kelas FiilMadhi

```
1   public class FiilMadhi extends Fiil {
2       public FiilMadhi(String a) {
3           this.setTransformAlgorithm(new TMadhiAlgorithm());
4           setWord(new ArabicText(a));
5           this.setConjugationId(0);
6           this.fiilType = FIIL_MADHI;
7           this.setProperties(new FailProperties(EnumFail.P3SINGLE));
8       }
9       public FiilMadhi(Trilateral a) {
10          this.setTransformAlgorithm(new TMadhiAlgorithm());
11          this.setWord(new ArabicText(a.getKata()));
12          this.setConjugationId(a.getConjugation());
13          this.fiilType = FIIL_MADHI;
14          this.setProperties(new FailProperties(EnumFail.P3SINGLE));
15      }
16      public FiilMadhi(Fiil fiil, EnumFail fail) {
17          this.setTransformAlgorithm(new TMadhiAlgorithm());
18          this.Transform(fiil, fail);
19          this.setProperties(new FailProperties(fail));
20          this.setConjugationId(fiil.getConjugationId());
21          this.fiilType = FIIL_MADHI;
22      }
23      @Override
24      public ArabicText toMadhi() {
25          this.setTransformAlgorithm(new TMadhiAlgorithm());
26          return this.word;
27      }
28      @Override
29      public ArabicText toMudhori() {
30          this.setTransformAlgorithm(new TMudhoriAlgorithm());
31          return getTransformAlgorithm()
                        .transformMain(this, EnumFail.P3SINGLE);
32      }
33      @Override
34      public ArabicText toAmr() {
35          throw new UnsupportedOperationException("Not supported");
36      }
37  }
```

Gambar 5.2 Implementasi Kelas FiilMadhi

Kelas FiilMadhi merupakan kelas yang merepresentasikan objek kata kerja lampau. Kelas ini mempunyai 3 kontruktor dikarenakan nanti sistem akan melakukan inisiasi melalui FiilMadhi dan terdapat beberapa kemungkinan parameter yang digunakan dalam instansiasi. Berikut keterangan baris kode :

1. Baris 2-8 merupakan konstruktor ke-1 dengan parameter bertipe string. Konstruktor ini digunakan ketika kata kerja yang dicari tidak terdapat pada tabel basis data trilateral.
2. Baris 9-15 merupakan konstruktor ke-2 dengan parameter bertipe Trilateral. Digunakan ketika kata kerja yang dicari terdapat dalam tabel basis data Trilateral.
3. Baris 16-22 merupakan konstruktor ke-3 dengan parameter bertipe Fiil dan EnumFail. Konstruktor ini digunakan ketika sistem hendak membuat objek kata kerja yang berubah berdasar jenis subyek (Tashrif Lughowi).
4. Pada tiap konstruktor (baris 3,10 dan17) terdapat baris kode yang menginstansiasi objek transformAlgorithm dengan jenis TmadhiAlgorithm. Hal ini merupakan bagian dari mekanisme Strategy pattern yaitu memberi pilihan pada subklas untuk menentukan algoritma yang dibutuhkan.
5. Baris 23-37 merupakan implementasi dari metode abstrak kelas Fiil.
6. Baris 24-27 merupakan metode untuk menghasilkan kata kerja masa lampau (berjenis Fiil Madhi).
7. Baris 29-32 merupakan metode untuk menghasilkan kata kerja masa sekarang (Fiil Mudhori')
8. Pada baris 34-36 perlu diperhatikan bahwa metode ini belum didefinisikan jadi hanya mengembalikan notifikasi pesan belum didukung oleh sistem.

Kelas FiilMudhori

```
1  public class FiilMudhori extends Fiil{
2      public FiilMudhori (Fiil fiil,EnumFail fail){
3          this.setTransformAlgorithm(new TMudhoriAlgorithm());
4          this.Transform(fiil,fail);
5          this.setProperties(new FailProperties(fail));
6          this.setConjugationId(fiil.getConjugationId());
7          this.fiilType=FIIL_MUDHORI;
```

```
8       }
9       @Override
10      public ArabicText toMadhi() {
11          throw new UnsupportedOperationException("Not supported");
12      }
13      @Override
14      public ArabicText toMudhori() {
15          return this.getWord();
16      }
17      @Override
18      public ArabicText toAmr() {
19          this.setTransformAlgorithm(new TAmrAlgorithm());
20          return getTransformAlgorithm()
                .transformMain(this, EnumFail.P3SINGLE);
21      }
22  }
```

Gambar 5.3 Implementasi Kelas FiilMudhori

Kelas FiilMudhori merepresentasikan objek kata kerja masa sekarang. Berikut keterangan baris kode :

1. Baris 2-8 merupakan konstruktor dengan parameter bertipe Fiil dan EnumFail. Hal ini menunjukan bahwa untuk melakukan instansiasi objek FiilMudhori membutuhkan objek Fiil lainya yang dalam konteks sistem ini adalah bisa dari FiilMadhi maupun FiilMudhori serta jenis subyeknya.
2. Baris kode 3 yang menginstansiasi objek transformAlgorithm dengan jenis TMudhoriAlgorithm. Hal ini merupakan bagian dari mekanisme *Strategy pattern* yaitu memberi pilihan pada subkelas untuk menentukan algoritma yang dibutuhkan.
3. Baris 10-24 merupakan implementasi dari metode abstrak dari kelas Fiil. Metode toMadhi (baris 11-13) belum terdapat implementasi jadi hanya mengembalikan pesan belum didukung.

Kelas FiilAmr

```
1   public class FiilAmr extends Fiil{
2       public FiilAmr (Fiil fiil,EnumFail fail){
3           this.setTransformAlgorithm(new TAmrAlgorithm());
4           this.Transform(fiil,fail);
5           this.setProperties(new FailProperties(fail));
6           this.setConjugationId(fiil.getConjugationId());
```

```
7           this.fiilType=FIIL_AMR;
8       }
9       @Override
10      public ArabicText toMadhi() {
11          throw new UnsupportedOperationException("Not supported");
12      }
13      @Override
14      public ArabicText toMudhori() {
15          throw new UnsupportedOperationException("Not supported");
16      }
17      @Override
18      public ArabicText toAmr() {
19          return getWord();
20      }
21  }
```

Gambar 5.4 Implementasi Kelas FiilAmr

Kelas FiilAmr merepresentasikan objek kata kerja perintah. Berikut keterangan baris kode :

1. Baris 2-8 merupakan konstruktor dengan parameter bertipe Fiil dan EnumFail. Hal ini menunjukan bahwa untuk melakukan instansiasi objek FiilMudhori membutuhkan objek Fiil lainya yang dalam konteks sistem ini adalah bisa dari FiilMudhori maupun FiilAmr serta jenis subyeknya.
2. Baris kode 3 yang menginstansiasi objek transformAlgorithm dengan jenis TAmrAlgorithm. Hal ini merupakan bagian dari mekanisme *Strategy pattern* yaitu memberi pilihan pada subklas untuk menentukan algoritma yang dibutuhkan.
3. Baris 10-20 merupakan implementasi dari metode abstrak dari kelas Fiil. Metode toMadhi dan toMudhori (baris 11-16) belum terdapat implementasi jadi hanya mengembalikan pesan belum didukung.

5.3.1.2 Interface TransformStrategy dan Kelas Implementasi

```
1   public interface TransformStrategy {
2     public abstract ArabicText transform(Fiil fiil);
3     public abstract ArabicText transformMain(Fiil text,EnumFail ef);
4   }
```

Gambar 5.5 Interface TransformStrategy

Penggunaan Interface disini adalah sebagai *supertype* sehingga kelas yang nantinya mengimplementasikan bisa memberi detail menurut kebutuhan algoritma dan juga memberikan metode akses yang *uniform* bagi kelas lain yang hendak menggunakanya. Keterangan kode baris sebagai berikut :

1. Baris 3 metode transform merupakan metode yang digunakan untuk melakukan perubahan kata kerja yang berbeda jenis seperti dari Fiil madhi ke Fiil Mudhori.
2. Baris 4 metode transformMain merupakan metode yang digunakan untuk melakukan perubahan kata kerja yang jenisnya sama akan tetapi berbeda subyeknya.

Tabel 5.4 Tabel Enumerasi Kata Ganti Subyek

No	Notasi	Genre	Sudut pandang	Nominal
1	P3SINGLE	Pria	Ke-3	Satu
2	P3DUAL	Pria	Ke-3	Dua
3	P3PLURAL	Pria	Ke-3	Banyak
4	P2SINGLE	Pria	Ke-2	Satu
5	P2DUAL	Pria	Ke-2	Dua
6	P2PLURAL	Pria	Ke-2	Banyak
7	W3SINGLE	Wanita	Ke-3	Satu
8	W3DUAL	Wanita	Ke-3	Dua
9	W3PLURAL	Wanita	Ke-3	Banyak
10	W2SINGLE	Wanita	Ke-2	Satu
11	W2DUAL	Wanita	Ke-2	Dua
12	W2PLURAL	Wanita	Ke-2	Plural
13	N1SINGLE	Unisex	Ke-1	Satu
14	N1PLURAL	Unisex	Ke-1	Banyak

Tabel 5.4 menunjukkan enumerasi dari jenis kata ganti subyek yang nanti akan digunakan dalam metode transformMain ditiap concrete class yang menggunakan interface TransformStrategy.

Kelas TMadhiAlgorithm

```
1  ArabicText PreTransform(Fiil fiil, DiacriticType t1,
2                          CharacterType st2, DiacriticType t2,
3                          CharacterType st3, DiacriticType t3) {
4        ArabicTextBuilder a = new ArabicTextBuilder();
5        a.add(fiil.getWord().getCharacterType(0),
6              Fiil.getDiacriticType(fiil.getWord().getCharacter(0)));
7        a.add(fiil.getWord().getCharacterType(1),
8              Fiil.getDiacriticType(fiil.getWord().getCharacter(1)));
9        a.add(fiil.getWord().getCharacterType(2), t1);
10         if (st2 != null && t2 != null) {
11             a.add(st2, t2);
12         } else if (t2 != null) {
13             a.add(t2);
14         } else if (st2 != null) {
15             a.add(st2);
16         }
17         if (st3 != null && t3 != null) {
18             a.add(st3);
19             a.add(t3);
20         } else if (t3 != null) {
21             a.add(t3);
22         } else if (st3 != null) {
23             a.add(st3);
24         }
25         return a.toText();
26     }
```

Gambar 5.6 Bagian Kelas TmadhiAlgorithm metode 1

Kelas TMadhiAlgorithm merupakan kelas yang mengimplementasi interface TransformStrategy dengan memberikan detail algoritma untuk perubahan ke dalam bentuk kata kerja masa lampau (Fiil Madhi). Gambar 5.6 menunjukkan metode-1 yang digunakan untuk merubah susunan huruf dalam kata kerja lampau dengan menggunakan asumsi 2 karakter dan 3 tanda baca tambahan.

```
1  ArabicText PreTransform(Fiil fiil, DiacriticType t1,
2            CharacterType st2, DiacriticType t2, CharacterType st3,
3            DiacriticType t3,  CharacterType st4, DiacriticType t4){
4        ArabicTextBuilder a = new ArabicTextBuilder();
5        a.add(fiil.getWord().getCharacterType(0),
6              Fiil.getDiacriticType(fiil.getWord().getCharacter(0)));
```

```
7           a.add(fiil.getWord().getCharacterType(1),
8                   Fiil.getDiacriticType(fiil.getWord().getCharacter(1)));
9           a.add(fiil.getWord().getCharacterType(2), t1);
10          a.add(st2, t2);
11          a.add(st3, t3);
12            if (st4 != null && t4 != null) {
13                a.add(st4);
14                a.add(t4);
15            } else if (t4 != null) {
16                a.add(t4);
17            } else if (st4 != null) {
18                a.add(st4);
19            }
20            return a.toText();
21      }
```

Gambar 5.7 Bagian Kelas TmadhiAlgorithm metode 2

Gambar 5.7 menunjukkan metode-2 yang digunakan untuk merubah susunan huruf dalam kata kerja lampau dengan menggunakan aturan 3 karakter dan 4 tanda baca tambahan.

```
1       @Override
2       public ArabicText transform(Fiil fiil) {
3           return fiil.getWord();
4       }
6       @Override
7       public ArabicText transformMain(Fiil text, EnumFail ef) {
8           ArabicText result;
9           switch (ef) {
10              case N1SINGLE:
11                  result = PreTransform(text, DiacriticType.Sukun,
12                          CharacterType.Ta, DiacriticType.Damma,
13                          null, null);
14                  break;
15              case N1PLURAL:
16                  result = PreTransform(text, DiacriticType.Sukun,
17                          CharacterType.Noon, DiacriticType.Fatha,
18              ..........
19              ..........
20              default:
21                  result = this.transform(text);
```

```
22                  break;
23              }
24              return result;
25          }
```

Gambar 5.8 Bagian Kelas TmadhiAlgorithm implementasi metode interface

Gambar 5.8 menunjukan bagian kelas TmadhiAlgorithm yang mengimplementasikan metode yang berasal dari interface TransformStrategy. Pada baris 6-54 (metode transformMain) terdapat penggunaan *switch-case* dimana menggunakan nilai dari variabel bertipe EnumFail dimana EnumFail merupakan enumerasi dari jenis-jenis klasifikasi subyek, kemudian memanfaatkan metode PreTransform yang sesuai dengan aturan perubahan subyek. Contoh :

1. Baris 10-13 : merubah fi'il pada subyek kata ganti berjenis orang pertama tunggal dengan menambahkan parameter harokat sukun, huruf ta dan harokat damma.
2. Baris 15-17 : merubah fi'il pada subyek kata ganti berjenis orang pertama jamak dengan menambahkan parameter harokat sukun, huruf nun dan harokat fatha.

Kelas TMudhoriAlgorithm

```
1   @Override
2       public ArabicText transform(Fiil fiil) {
3           ArabicTextBuilder mudhori = new ArabicTextBuilder();
4           mudhori.add(CharacterType.Ya, DiacriticType.Fatha);
6           mudhori.add(fiil.getWord().getCharacterType(0),
7                       DiacriticType.Sukun);
8           if (fiil.getWord().getCharacter(1).isDamma()) {
9               mudhori.add(fiil.getWord().getCharacterType(1),
10                          DiacriticType.Damma);
11          } else if (fiil.getWord().getCharacter(1).isKasra()) {
12              mudhori.add(fiil.getWord().getCharacterType(1),
13                          DiacriticType.Fatha);
14          } else if (fiil.getWord().getCharacter(1).isFatha()) {
15              if (values.contains(fiil.getWord().getCharacterType(1))
16              ||values.contains(fiil.getWord().getCharacterType(2))){
17                  mudhori.add(fiil.getWord().getCharacterType(1),
18                              DiacriticType.Fatha);
19              } else if (fiil.getConjugationId() == 1 ||
```

```
20                          fiil.getConjugationId() == 0) {
21                  mudhori.add(fiil.getWord().getCharacterType(1),
22                                  DiacriticType.Damma);
23              } else if (fiil.getConjugationId() == 2) {
24                  mudhori.add(fiil.getWord().getCharacterType(1),
25                                  DiacriticType.Kasra);
26              }
27          }
28          mudhori.add(fiil.getWord().getCharacterType(2),
29                          DiacriticType.Damma);
30          return mudhori.toText();
31      }
```

Gambar 5.9 Bagian Kelas TMudhoriAlgorithm implementasi interface

Kelas TMudhoriAlgorithm merupakan kelas yang mengimplementasi interface TransformStrategy dengan memberikan detail algoritma untuk perubahan ke dalam bentuk kata kerja masa sekarang (Fiil Mudhori'). Pada gambar 5.9 menunjukan metode transform yang bertugas merubah susunan huruf dalam kata kerja masa lampau menuju kata kerja masa sekarang atau dari fiil madhi ke fiil mudhori'.

```
1       ArabicText PreTransform(Fiil fiil, DiacriticType t1,
2                CharacterType st2,DiacriticType t2, CharacterType st3,
3                DiacriticType t3,CharacterType st4, DiacriticType t4){
4           ArabicTextBuilder a = new ArabicTextBuilder();
5           a.add(st2, t2);
6           a.add(fiil.getWord().getCharacterType(1),
7               fiil.getDiacriticType(fiil.getWord().getCharacter(1)));
8           a.add(fiil.getWord().getCharacterType(2),
9               fiil.getDiacriticType(fiil.getWord().getCharacter(2)));
10          a.add(fiil.getWord().getCharacterType(3), t1);
11          a.add(st3);
12          if (t3 != null) {
13              a.add(t3);
14          }
15          a.add(st4, t4);
16          return a.toText();
17      }
```

Gambar 5.10 Bagian Kelas TMudhoriAlgorithm metode 1

Gambar 5.10 menunjukkan metode-1 yang digunakan untuk merubah susunan huruf dalam kata kerja masa sekarang dengan menggunakan aturan 3 karakter dan 4 tanda baca tambahan.

```
1   ArabicText PreTransform(Fiil fiil, DiacriticType t1, CharacterType
2                           st2,DiacriticType t2, CharacterType st3,
3                           DiacriticType t3){
4           ArabicTextBuilder a = new ArabicTextBuilder();
5           a.add(st2, t2);
6           if (st2.equals(CharacterType.Alif)) {
7               a.add(DiacriticType.HamzaAbove);
8           }
9           a.add(fiil.getWord().getCharacterType(1),
10              fiil.getDiacriticType(fiil.getWord().getCharacter(1)));
11          a.add(fiil.getWord().getCharacterType(2),
12              fiil.getDiacriticType(fiil.getWord().getCharacter(2)));
13          a.add(fiil.getWord().getCharacterType(3), t1);
14          if (st3 != null && t3 != null) {
15              a.add(st3);
16              a.add(t3);
17          } else if (t3 != null) {
18              a.add(t3);
19          } else if (st3 != null) {
20              a.add(st3);
21          }
22          return a.toText();
23      }
```

Gambar 5.11 Bagian Kelas TMudhoriAlgorithm metode 2

Gambar 5.11 menunjukkan metode-2 yang digunakan untuk merubah susunan huruf dalam kata kerja masa sekarang dengan menggunakan aturan 2 karakter dan 3 tanda baca tambahan.

```
1   @Override
2       public ArabicText transformMain(Fiil text, EnumFail ef) {
3           ArabicText result;
4           switch (ef) {
5           ..............
6           ..............
7               case W3DUAL:
8                   result = PreTransform(text, DiacriticType.Fatha,
9                           CharacterType.Ta, DiacriticType.Fatha,
```

```
10                          CharacterType.Alif, null,
11                          CharacterType.Noon, DiacriticType.Kasra);
12                  break;
13              case W3PLURAL:
14                  result = PreTransform(text, DiacriticType.Sukun,
15                          CharacterType.Ya, DiacriticType.Fatha,
16                          CharacterType.Noon, DiacriticType.Fatha);
17                  break;
18              default:
19                  if(text.getFiilType().equals(EnumFiil.FIIL_MADHI)){
20                      result = this.transform(text);
21                  } else{
22                      result= text.getWord();
23                  }
24                  break;
25          }
26          return result;
27      }
```

Gambar 5.12 Bagian Kelas TMudhoriAlgorithm implementasi metode interface

Keterangan baris kode :

1. Baris 7-11 merubah fi'il dengan jenis subyek wanita pihak ketiga dua orang
2. Baris 13-17 merubah fi'il dengan jenis subyek wanita pihak ketiga jamak

Gambar 4.25 menunjukan bagian kelas TMudhoriAlgorithm yang mengimplementasikan metode yang berasal dari interface TransformStrategy.

Kelas TAmrAlgorithm

```
1   @Override
2       public ArabicText transform(Fiil fiil) {
3           ArabicTextBuilder Amr = new ArabicTextBuilder();
5           if (fiil.getWord().getCharacter(2).isDamma()) {
6               Amr.add(CharacterType.Alif, DiacriticType.Damma);
7           } else {
8               Amr.add(CharacterType.Alif, DiacriticType.Kasra);
9           }
11          Amr.add(fiil.getWord().getCharacterType(1),
12                  DiacriticType.Sukun);
13          Amr.add(fiil.getWord().getCharacterType(2),
14                  getDiacriticType(fiil.getWord().getCharacter(2)));
15          Amr.add(fiil.getWord().getCharacterType(3),
```

```
16                  DiacriticType.Sukun);
17          return Amr.toText();
18      }
```

Gambar 5.13 Bagian Kelas TAmrAlgorithm implementasi metode interface

Kelas TAmrAlgorithm merupakan kelas yang mengimplementasi interface TransformStrategy dengan memberikan detail algoritma untuk perubahan ke dalam bentuk kata kerja masa perintah (Fiil Amr). Pada gambar 5.13 menunjukan metode transform yang bertugas merubah susunan huruf dalam kata kerja masa sekarang menuju kata kerja perintah atau dari fiil mudhori' ke fiil amr.

```
1   ArabicText PreTransform(Fiil fiil, DiacriticType t1, CharacterType
2           st2,DiacriticType t2, CharacterType st3, DiacriticType t3){
3           ArabicTextBuilder a = new ArabicTextBuilder();
5           a.add(fiil.getWord().getCharacterType(0),
6               fiil.getDiacriticType(fiil.getWord().getCharacter(0)));
7           a.add(fiil.getWord().getCharacterType(1),
8               fiil.getDiacriticType(fiil.getWord().getCharacter(1)));
9           a.add(fiil.getWord().getCharacterType(2),
11              fiil.getDiacriticType(fiil.getWord().getCharacter(2)));
12          a.add(fiil.getWord().getCharacterType(3), t1);
13          if (t2 == null) {
14              a.add(st2);
15          } else {
16              a.add(st2, t2);
17          }
18          if (st3 != null) {
19              a.add(st3);
20          }
21          return a.toText();
22      }
```

Gambar 5.14 Bagian Kelas TAmrAlgorithm metode 1

Gambar 5.14 menunjukkan metode-1 yang digunakan untuk merubah susunan huruf dalam kata kerja perintah dengan menggunakan aturan 2 karakter dan 3 tanda baca tambahan.

```
1   @Override
2       public ArabicText transformMain(Fiil text, EnumFail ef) {
3           ArabicText result;
4           switch (ef) {
5               case P2DUAL:
```

```
6                 result = PreTransform(text, DiacriticType.Fatha,
7                                       CharacterType.Alif, null, null, null);
8                 break;
9             case P2PLURAL:
10                result = PreTransform(text, DiacriticType.Damma,
11                                    CharacterType.Waw, DiacriticType.Sukun
12                                    , CharacterType.Alif, null);
13                break;
14        ............
15        }
16        return result;
17    }
```

Gambar 5.15 Bagian Kelas TAmrAlgorithm implementasi metode interface

Gambar 5.15 menunjukan bagian kelas TAmrAlgorithm metode TransformStrategy. Jumlah jenis Subyek pada fiil amr lebih sedikit dibandingkan dengan kata kerja lainya karena hanya diterapkan pada sudut pandang orang kedua.

1. Baris 5-7 : perubahan fi'il pada jenis subyek pria, pihak kedua, dua orang
2. Baris 9-12: perubahan fi'il pada jenis subyek pria,pihak kedua, jamak

5.3.2 Implementasi Facade Pattern

Untuk bisa menjelaskan implementasi facade pattern maka perlu menjabarkan kode pada kelas abstrak Tashrif dan turunan.

```
1  public abstract class Tashrif {
2      ArrayList<Fiil> fiilGroup = new ArrayList<>();
3      public void add(Fiil fiil){
4          this.FiilGroup.add(fiil);
5      }
6      public abstract void init(Fiil a);
7      public ArrayList<Fiil> getList(){
8          return this.FiilGroup;
9      }
10     public  abstract String getExplanation();
11 }
```

Gambar 5.16 Kelas Abstract Tashrif

Kelas Abstrak tashrif merepresentasikan tentang cara perubahan fi'il atau bisa dikatakan sebagai kumpulan / koleksi dari fi'il yang berubah berdasarkan aturan tertentu. Kelas turunan dari Tashrif yaitu TashrifIsltilahi dan TashrifLughowi yang nantinya memberikan detail implementasi pada metode abstrak. Berikut keterangan dari baris kode :

1. Baris 2: atribut fiilGroup yang bertipe arraylist digunakan sebagai koleksi dari objek Fi'il.
2. Baris 3-5: metode untuk melakukan penambahan elemen ke dalam koleksi.
3. Baris 6 : metode init bertugas untuk melakukan inisiasi elemen yang nantinya akan dimasukkan ke dalam koleksi.
4. Baris 7-9: metode ini bertugas untuk mendapatkan nilai atribut fiilGroup.
5. Baris 10: metode ini bertugas untuk memberikan informasi penjelasan.

Kelas TashrifIstilahi

```
1   public class TashrifIstilahi extends Tashrif{
2       @Override
3       public void init(Fiil a) {
4           Fiil madhi = a;
5           Fiil mudhori = new FiilMudhori(a,
6                           a.getProperties().getEnumFail());
7           Fiil amr = new FiilAmr(mudhori,
8                           a.getProperties().getEnumFail());
9           this.add(madhi);
10          this.add(mudhori);
11          this.add(amr);
12      }
13      @Override
14      public String getExplanation() {
15          String explanation="Tashrif Istilahi merupakan perubahan"
16                              +" kata kerja berdasar jenis Fiil";
17          for(Fiil fiil:getList()){
18              explanation+="\n"+fiil.getExplanation();
19          }
20          return explanation;
21      }
23  }
```

Gambar 5.17 Kelas TashrifIstilahi

Kelas TashrifIstilahi merupakan kelas turunan / *concrete class* dari Tashrif yang merepresentasikan tashrif istilahi, perubahan fi'il berdasarkan jenis fi'il. Keterangan baris kode:

1. Baris 3-12 : metode init yang telah diberikan implementasi dalam menginisiasi elemen dari koleksi fiil. Tiap elemen merupakan jenis fi'il yang berbeda karena objek tersebut berasal dari tiap-tiap *concrete class* Fiil.
2. Baris 14-21: metode getExplanation yang telah diberikan implementasi dalam memberikan kembalian informasi dari tiap elemen koleksi fiil berupa tipe data String.

Kelas TashrifLughowi

```
1   public class TashrifLughowi extends Tashrif {
2       @Override
3       public void init(Fiil a) {
4           switch (a.getFiilType()) {
5               case FIIL_MADHI:
6                   for(EnumFail fail:EnumFail.values()){
7                       add(new FiilMadhi(a, fail));
8                   }
9                   break;
10              case FIIL_MUDHORI:
11                  for(EnumFail fail:EnumFail.values()){
12                      add(new FiilMudhori(a, fail));
13                  }
14                  break;
15              case FIIL_AMR:
16                  for(EnumFail fail:EnumFail.valueSecond()){
17                      add(new FiilAmr(a, fail));
18                  }
19                  break;
20          }
21      }
22      @Override
24      public String getExplanation() {
25          String explanation="Tashrif Lughowi merupakan perubahan"
26                             +" kata kerja berdasar jenis Subyek";
27          for(Fiil fiil:getList()){
28              explanation+=fiil.getExplanation();
29          }
```

```
30          return explanation;
31      }
32  }
```

Gambar 5.18 Kelas TashrifLughowi

Kelas TashrifLughowi merupakan kelas turunan / *concrete class* dari Tashrif yang merepresentasikan tashrif lughowi yaitu perubahan fi'il berdasarkan kata ganti subyek. Berikut keterangan kode baris:

1. Baris 3-21 : implementasi metode init dimana menggunakan switch-case untuk menentukan jenis fi'il yang akan diinstansiasi kemudian dijadikan bagian dari koleksi.
2. Baris 24-31: implementasi metode yang memberikan informasi penjelasan pada tiap anggota elemen koleksi fiil.

```
1   public class Facade {
2       FiilMadhi madhi;
3       Tashrif istilahi;
4       Tashrif lughowi;
5       HashMap<Fiil, Tashrif> table;
6
7       public Facade(Trilateral a) {
8           this.madhi = new FiilMadhi(a);
9           initFacade(madhi);
10      }
11
12      public Facade() {
13      }
14      ..................
```

Gambar 5.19 Kelas Facade bagian atribut dan constructor

Kelas Facade merupakan kelas yang didalamnya terdapat atribut berupa kelas lain atau bisa disebut sebagai *sub-system* yang mendelegasikan tugas mereka melalui kelas ini (Facade). Berikut keterangan dari baris kode yang merupakan bagian kelas Facade :

1. Baris 2-5:merupakan atribut dikelas Facade yaitu berupa kelas FiilMadhi yang dibutuhkan sebagai acuan untuk proses tashrif. Atribut berikutnya bernama istilahi dengan tipe Tashrif digunakan untuk menjadi bagian proses tashrif istilahi. Atribut bernama table adalah kumpulan dari TashrifLughowi yang menggunakan tipe HashMap.
2. Alasan menggunakan HashMap adalah tiap TashrifLughowi didasari oleh jenis Fiil tertentu sehingga HashMap yang merupakan jenis *Collection* yang menggunakan konsep *key* dan *value*, tepat untuk digunakan dalam implementasi.
3. Baris 7-12: merupakan konstruktor yang digunakan dalam mengintansiasi objek kelas Facade. Terdapat 2 jenis konstruktor yaitu yang berparameter objek Trilateral dan default. Untuk yang berjenis parameter Trilateral digunakan sebagai instansiasi atribut madhi kemudian digunakan untuk metode initFacade().

```
1       public void setFiil(Trilateral a) {
2           this.madhi = new FiilMadhi(a);
3       }
4       public FiilMadhi getMadhi() {
5           return madhi;
6       }
7       public void setFiil(String a) {
8           this.madhi = new FiilMadhi(a);
9       }
10      public void initFacade(Fiil madhi) {
11          this.istilahi = new TashrifIstilahi();
12          istilahi.init(madhi);
13          table = new HashMap<>(istilahi.getList().size());
14          for (Fiil fiil : istilahi.getList()) {
15              lughowi = new TashrifLughowi();
16              lughowi.init(fiil);
17              table.put(fiil, lughowi);
18          }
19      }
```

Gambar 5.20 Kelas Facade bagian metode initFacade

Keterangan kode baris gambar 5.20 :

1. Baris 1-9: merupakan metode setter dan getter untuk atribut madhi. Terdapat metode yang overload disini yaitu setFiil yang menggunakan 2 parameter yang berbeda untuk mengantisipasi jenis *input* yang didapatkan dari kelas klien yang nantinya menggunakan kelas Facade.
2. Baris 10-19: metode initFacade adalah alasan utama yang menjelaskan diperlukannya Kelas Facade karena dalam metode ini melibatkan interaksi objek dari kelas yang berbeda. Metode ini juga dipanggil di konstruktor untuk inisiasi atribut istilahi dan table. Pada awalnya dalam metode ini terdapat instansiasi objek TashrifIstilahi (atribut istilahi) menggunakan atribut madhi sebagai parameter. Kemudian tiap anggota elemen dari atribut istilahi digunakan untuk instansiasi objek TashrifLughowi yang mana merupakan bagian dari elemen atribut table.

```
1     public ObservableList getTashrifLughowi(int index) {
2         Fiil ex = istilahi.getList().get(index);
3         ObservableList<Fiil> list = FXCollections
4                         .observableArrayList(table.get(ex).getList());
5         return list;
6     }
7     public Fiil getTashrifIstilahi(int index) {
8         return istilahi.getList().get(index);
9     }
10    public String getExplanation(){
11        return istilahi.getExplanation();
12    }
13    public ArrayList<String> getQuizAnswers() {
14        ArrayList<String> temp = new ArrayList<>();
15        //menambahkan fiil berjenis madhi dan mudhori ke dalam list
16        for (int i = 0; i < 2; i++) {
17            temp.add(table.get(istilahi.getList()
18                    .get(i)).getList().get(0).getWord().toString());
19            temp.add(table.get(istilahi.getList()
20                    .get(i)).getList().get(6).getWord().toString());
21            temp.add(table.get(istilahi.getList()
22                    .get(i)).getList().get(7).getWord().toString());
23            temp.add(table.get(istilahi.getList()
```

```
24                      .get(i)).getList().get(8).getWord().toString());
25              temp.add(table.get(istilahi.getList()
26                      .get(i)).getList().get(9).getWord().toString());
27          }
28          //menambahkan fiil berjenis Amr ke dalam list
            //(berbeda karena urutan subyek tidak sama)
29          Fiil ex = istilahi.getList().get(2);
30          for(int i=0;i<4;i++){
31              temp.add(table.get(ex).getList()
32              .get(i).getWord().toString());
33          }
34          return temp;
35      }
```

Gambar 5.21 Kelas Facade bagian metode getTashrif dan getQuizAnswers

Keterangan kode baris gambar 5.21 :

1. Baris 1-5 : metode getTashrifLughowi digunakan untuk mendapatkan objek bertipe TashrifLughowi. Didalam metode ini terdapat konversi jenis *Collections* dari *ArrayList* ke *ObservableList* untuk menyesuaikan kebutuhan dari kelas Klien yang membutuhkan jenis *Collections* ini.
2. Baris 7-9: metode getTashrifIstilahi digunakan untuk mendapatkan objek bertipe Fiil yang merupakan anggota dari elemen atribut Istilahi.
3. Baris 10-12: merupakan metode yang memberikan informasi penjelasan yang dipanggil dari atribut istilahi.
4. Baris 13-35: metode getQuizAnswers digunakan untuk mendapatkan objek bertipe ArrayList yang berisi beberapa elemen dari atribut table. Metode ini digunakan untuk mendapatkan jawaban untuk fitur latihan(exercise) menjawab Tashrif pada aplikasi.

5.3.3 Implementasi Basis Data

Implementasi basis data yang menyimpan akar kata dilakukan dengan database management system MySQL. Hanya terdapat satu tabel yang menyimpan akar kata yang terdiri dari karakter huruf penyusun kata dan kode rumus perubahan (wazan). Data akar kata ini diperoleh dari sebuah proyek pengembangan perangkat lunak serupa yang bernama Sarf. *Sarf Arabic Morphology System* merupakan proyek yang dikembangkan oleh *Arab League Educational, Cultural and Scientific Organization* (ALECSO) untuk menjadi aplikasi pendukung pembelajaran ilmu kata kerja dan turunanya (Sharaf)[SAR-14]. Sarf merupakan aplikasi yang ditujukan pada pengguna bangsa Arab sehingga segala antarmuka aplikasi dan dokumentasi menggunakan bahasa Arab.

Data *root* yang digunakan awalnya berupa berformat xml sehingga bagi aplikasi yang dibuat memerlukan proses konversi format xml tersebut menjadi tabel dalam DBMS MySQL. Berikut contoh potongan file xml:

```
<roots>
	<root c1="غ" c2="ب" c3="ء" conjugation="3" transitive="م">
		<gerund symbol="H1" value="غَبْء" />
	</root>
	<root c1="غ" c2="ب" c3="ء" conjugation="4" transitive="م">
		<gerund symbol="H1" value="غَبْء" />
	</root>
	<root c1="غ" c2="ب" c3="ب" conjugation="1" transitive="ك">
		<gerund symbol="H1" value="غَبّ" />
		<gerund symbol="H2" value="غَبَب" />
		<gerund symbol="U1" value="غبيب" />
		<gerund symbol="H7" value="غبّ" />
		<gerund symbol="E3" value="غبّاب" />
	</root>
	<root c1="غ" c2="ب" c3="ب" conjugation="4" transitive="ل">
		<gerund symbol="H7" value="غِبّ" />
		<gerund symbol="H2" value="غَبَب" />
	</root>
```

Gambar 5.22 Bagian data xml Sarf

Keterangan dari kode xml gambar 5.22 :

1. Tags <root> menandakan satu kata kerja
2. C1, C2, dan C3 merupakan karakter huruf penyusun kata
3. Conjugation merupakan notasi rumus yang akan digunakan
4. Transitive menunjukkan sifat kata kerja yang transitif atau intransitif.

5. Gerund symbol dan value menunjukkan notasi rumus perubahan kata kerja menjadi kata benda.

Dari penjelasan tags diatas yang digunakan dalam implementasi sistem aplikasi hanya nomor 2 dan 3 sedangkan 4 dan 5 masih belum diimplementasikan.

Kelas Trilateral

```
1   @Entity
2   @Table(name = "trilateral", catalog = "sharaf")
3   @Indexed
4   public class Trilateral implements java.io.Serializable {
5       private Integer id;
6       private String c1;
7       private String c2;
8       private String c3;
9       private Integer conjugation;
10      private String kata;
11      public Trilateral() {
12      }
13     public Trilateral(String c1, String c2, String c3,
14                       Integer conjugation) {
15          this.c1 = c1;
16          this.c2 = c2;
17          this.c3 = c3;
18          this.conjugation = conjugation;
19      }
```

Gambar 5.23 Kelas Trilateral bagian atribut dan konstruktor

Kelas Trilateral merupakan kelas yang dihasilkan oleh proses *reverse engineering* oleh Hibernate untuk memodelkan tabel relasional sebuah basis data menjadi sebuah kelas yang berorientasi objek. Kelas ini merupakan bentuk konversi yang didapatkan dari data xml Sarf. Keterangan kode baris gambar 5.23 :

1. Baris 1-3 : anotasi yang menunjukkan hasil *reverse engineering* oleh API Hibernate yaitu berasal dari tabel “trilateral” dan basis data “sharaf”.
2. Baris 5-10 : atribut kelas trilateral yang merepresentasikan kolom pada tabel relasional trilateral. atribut berbeda yang ditambahkan adalah kata yang nantinya mempunyai nilai berupa susunan kata bertanda baca (harokat)
3. Baris 13-19: merupakan konstruktor untuk instansiasi objek Trilateral.

```
1       @Field(index = Index.YES, analyze = Analyze.YES,
2               store = Store.NO)
3       public String getKata() {
4           return kata;
5       }
6       @Column(name = "conjugation")
7       public Integer getConjugation() {
8           return this.conjugation;
9       }
10      public void setConjugation(Integer conjugation) {
11          this.conjugation = conjugation;
12          if (this.conjugation == 1 || this.conjugation == 2) {
13              ArabicText kata2 = new ArabicText(c1 + "ّ" + c2
14                                                  + "ّ" + c3 + "ّ");
15              kata = kata2.toString();
16          }
17      }
```

Gambar 5.24 Kelas Trilateral bagian metode dan anotasi HibernateSearch

Gambar 5.24 menunjukkan sebagian metode *setter* dan *getter* pada kelas trilateral tetapi pada bagian kode yang perlu diperhatikan adalah anotasi pada baris 1 yaitu anotasi HibernateSearch dimana atribut yang diberikan anotasi ini akan dilakukan proses *indexing* dalam HibernateSearch. Pada baris 11-17 menunjukkan metode setConjugation yang berfungsi untuk inisiasi nilai pada atribut Conjugation dan juga kata.

Kelas DBAccess

```
1   public class DBAccess {
2       public DBAccess(){
3       }
4       public ArrayList<Trilateral> retrieve(String input) {
5           FullTextSession fts = null;
6           ArrayList<Trilateral> resultcontain = null;
7           try {
8               Session session = NewHibernateUtil.getSessionFactory()
9                       .openSession();
10              fts = Search.getFullTextSession(session);
11              Transaction tx = fts.beginTransaction();
12              QueryBuilder qb = fts.getSearchFactory()
13                      .buildQueryBuilder()
14                      .forEntity(Trilateral.class).get();
```

```
15              org.apache.lucene.search.Query query = qb.keyword()
16                      .wildcard().onField("kata").matching(input+"*")
17                      .createQuery();
18              org.hibernate.Query hibQ = fts.createFullTextQuery
19                                              (query, Trilateral.class);
20              resultcontain = (ArrayList<Trilateral>) hibQ.list();
21          } catch (Exception e) {
22              e.printStackTrace();
23          }
24          return resultcontain;
25      }
```

Gambar 5.25 Kelas DBAccess

Kelas DBAccess merupakan kelas yang mempunyai metode dalam proses *retrieving* data yang berasal dari *database*. Terdapat beberapa metode yang didefinisikan dengan menggunakan API HibernateSearch dan Lucene. Berikut keterangan baris kode :

1. Baris 8-14 merupakan tahapan yang dilakukan untuk mendapatkan *session* dari HibernateSearch.
2. Baris 15-17 merupakan query yang dilakukan dengan cara memanfaatkan kelas QueryAPI Lucene.
3. Baris 20 merubah hasil Query yang didapatkan menjadi bertipe arraylist sesuai dengan tipe kembalian metode retrieve.

5.3.4 Implementasi Graphic User Interface

Penggunaan JavaFX dalam pembuatan GUI didasari fakta bahwa JavaFX merupakan salah satu API terbaru yang diluncurkan oleh Java untuk memenuhi kebutuhan *developer* membuat tampilan grafis aplikasi yang dinamis dan kaya akan *user experience* [CHA-13]. Dalam implementasinya *developer* diberi beberapa pilihan untuk membuat GUI salah satunya adalah menggunakan FXML. FXML merupakan sejenis bahasa XML yang menyediakan struktur *tags* untuk membangun GUI (*application design*) terpisah dari *application logic* sehingga hal ini akan mempermudah *code maintenance*.

Pada bagian implementasi ini hanya akan disertakan potongan kode dari *application logic* yang digunakan untuk mengatur interaksi antara GUI dengan application *back-end*. Terdapat 2 kelas yang menjadi *application logic* untuk fitur pencarian dan latihan yaitu SearchMenu dan ExerciseMenu.

Kelas SearchMenu

```
1   ......
2       @FXML
3       private TableColumn<Fiil, String> GenreKataFailAmr;
4       @FXML
5       private TableColumn<Fiil, String> DlamirKataFailAmr;
6       @FXML
7       private TableColumn<Fiil, String>[][] kataFail;
8       @FXML
9       ProgressIndicator progress;
10      @FXML
11      TabPane tabPane;
12      @FXML
13      TextArea info;
14  .....
15  public void initialize(URL url, ResourceBundle rb) {
16  .....
17  .....
18   table.getSelectionModel().selectedItemProperty().addListener
19     ((observable, oldValue, NewValue) -> showFiilLughowi(NewValue));
20   table.getSelectionModel().selectedItemProperty().addListener
21     ((observable, oldValue, NewValue) -> showIstilahi(NewValue));
22  .....
23      }
```

Gambar 5.26 Bagian Kelas SearchMenu atribut dan metode initialize

Keterangan baris kode gambar 5.26:

1. Baris 1-14 : menunjukkan sebagian dari atribut yang dideklarasikan dengan menggunakan anotasi @FXML. Anotasi ini menunjukan bahwa atribut ini dapat dikenali oleh komponen application design seperti tabPane, button dan sejenisnya.
2. Baris 15-23: menunjukkan sebagian metode initialize. Metode ini dipanggil awal kali ketika compiler melakukan proses pemuatan terhadap application design yang menggunakan (*application logic*) kelas ini.
3. Baris 18-21: menunjukan ketika atribut table dalam kondisi *selected* oleh *user* maka akan memanggil metode showFiilLughowi. *Statement* Baris kode ini menggunakan *lambda expression*.

```
1  @FXML
2  protected void TombolCari(ActionEvent event) {
3          if (!cont.isEmpty()) {
4              this.cont.clear();
5          }
6          if (validateInput()) {
7              search();
8          } else {
9              alert.showAndWait();
10         }
11     }
12 private boolean validateInput() {
13         boolean valid = true;
14         String messages = "";
15     if (field.getText().length() == 0 || field.getText() == null){
16             messages += "masukkan kata kerja\n";
17         }
18     if (!field.getText().matches("[ضصثقفغعهخحجدًٌٍَُِّْ]+")){
19           messages += "input yang Anda berikan harus karakter Arab"
20                     + "\nSelain  ي,ا dan و\n";
21     }
22         if (messages.length() != 0) {
23             valid = false;
24         }
25         alert.setContentText(messages);
26         return valid;
27     }
```

Gambar 5.27 Bagian Kelas SearchMenu metode tombol cari

Keterangan baris kode gambar 5.27:

1. Baris 1-15 : metode tombolCari bertugas untuk merespon ketika tombol cari pada GUI ditekan oleh user. Penggunaan anotasi @FXML menandakan bahwa metode ini dikenali oleh komponen GUI. *Input* user pertama kali dicek melalui metode validateInput jika memenuhi syarat maka akan diproses pencarian dalam database melalui metode search, jika *input* tidak memenuhi kriteria maka akan ditampilkan pesan peringatan pada user.
2. Baris 12-27: metode validateInput sesuai dengan namanya bertugas untuk mengecek apakah *input* yang diberikan oleh user valid atau tidak. Kriteria valid tidaknya *input* sudah dijelaskan pada bagian perancangan yang intinya adalah kata kerja bahasa Arab 3 huruf.

```
1  private void search() {
2       ArabicText input = new ArabicText(field.getText());
3       ArabicTextBuilder build = new ArabicTextBuilder();
4       String fix = "";
5       if (input.getLength() < 2) {
6          System.out.println("kondisi 0");
7          build.add(input.getCharacterType(0), DiacriticType.Fatha);
8          fix = build.toString();
9       } else if (getDiacriticType(input.getCharacter(0)) ==
10                  DiacriticType.Fatha
11                  && getDiacriticType(input.getCharacter(1)) ==
12                  DiacriticType.Fatha) {
13         System.out.println("kondisi 1");
14         fix = field.getText();
15      } else if (input.getLength() == 3
16                 && getDiacriticType(input.getCharacter(0)) ==
17                 DiacriticType.Fatha
18                 && (getDiacriticType(input.getCharacter(1)) ==
19                 DiacriticType.Damma
20                 || getDiacriticType(input.getCharacter(1)) ==
21                 DiacriticType.Kasra)
22                 && getDiacriticType(input.getCharacter(2)) ==
23                 DiacriticType.Fatha) {
24         System.out.println("kondisi 2");
25         Facade fa = new Facade();
26         fa.setFiil(field.getText());
27         fa.initFacade(fa.getMadhi());
28         this.cont.add(fa);
29         this.table.setItems(cont);
30      } else {
31         System.out.println("kondisi 3");
32         build.add(input.getCharacterType(0), DiacriticType.Fatha);
33         build.add(input.getCharacterType(1));
34         fix = build.toString();
35      }
36         if (fix.length() > 0) {
37             service.setInput(fix);
38             service.restart();
39         }
40     }
```

Gambar 5.28 Bagian Kelas SearchMenu metode search

Keterangan baris kode gambar 5.28 :

1. Baris 5-35 : menunjukan *statement* kondisional IF yang intinya adalah mencoba memisahkan jenis kata kerja yang telah dimasukkan oleh user. Memisahkan kata kerja yang ain fi'ilnya (huruf kedua) berharokat fatha dengan yang menggunakan harokat dhamma atau kasra.

2. Baris 36-39 : menunjukan statement kondisional apakah *input* oleh user perlu dicari ke dalam database karena hanya fi'il yang huruf keduanya. Service merupakan atribut yang mengenkapsulasi objek DBAccess.

Kelas ExerciseMenu

```
1   @FXML
2       protected void shuffle(ActionEvent event) {
3           HashSet<Integer> a;
4           a = new HashSet<>();
5           while (a.size() < 5) {
6               int value= (int) Math.round(Math.random() * 100 % 100);
7               a.add(value);
8           }
9           Integer[] cuv = a.toArray(new Integer[0]);
10          service.setInput(cuv);
11          service.restart();
12      }
13      private void initQuestion() {
14          for (int i = 0; i < con.size(); i++) {
15              buttonCont[i].setText(con.get(i).getKata());
16              buttonCont[i].setVisible(true);
17              storage.put(con.get(i).getKata(), con.get(i));
18              System.out.println(con.get(i).getId() + " ");
19          }
20      }
```

Gambar 5.29 Bagian Kelas ExerciseMenu metode shuffle

Keterangan baris kode gambar 5.29 :

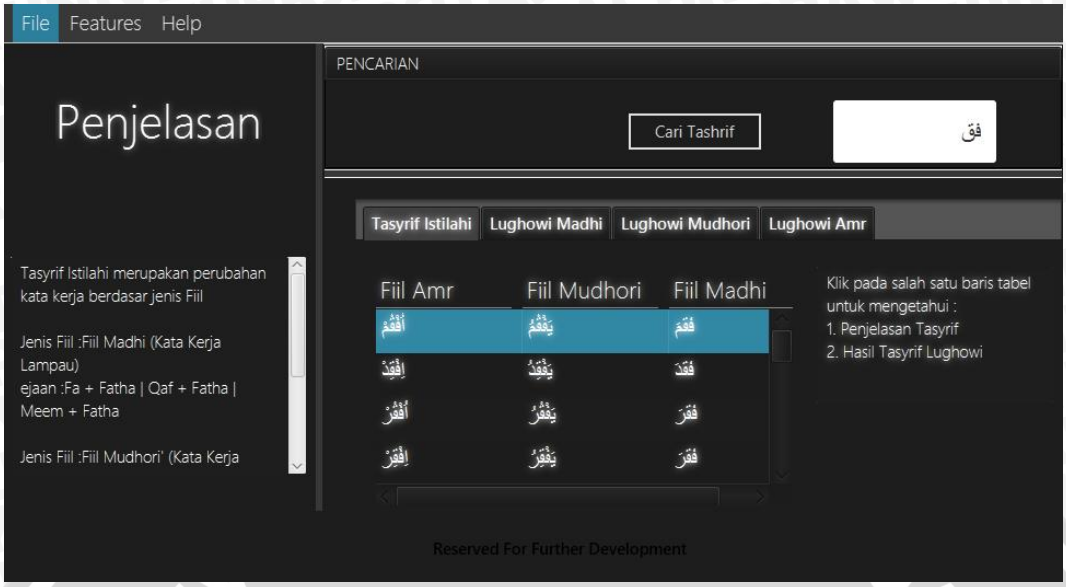
1. Baris 1-12: merupakan metode shuffle yang bertugas untuk merespon ketika tombol shuffle di tampilan aplikasi ditekan oleh user. Tombol tersebut berfungsi untuk melakukan proses pengacakan soal kata kerja yang akan dijawab oleh user. Dalam implementasinya aplikasi akan memilih 5 soal kata kerja yang dipilih secara acak dari database.
2. Baris 13-20 : metode initQuestion dipanggil ketika atribut service sukses dalam mencari 5 kata kerja secara acak. Fungsi metode ini untuk menampilkan 5 kata kerja yang terpilih dan juga menyimpanya ke dalam atribut storage.

```
1     @FXML
2     protected void testStart(ActionEvent event) {
3         Button e = (Button) event.getSource();
4         System.out.println("its working" + "" + e.getText());
5         answers = new Facade(storage.get(e.getText()));
6         second.setVisible(true);
7         first.setVisible(false);
8         acc.setExpandedPane(second);
9         for (int i = 0; i < answers.getQuizAnswers().size(); i++) {
10            labelCont[i].setText(answers.getQuizAnswers().get(i));
11        }
12        resetText();
13    }
14    @FXML
15    protected void evaluateAnswer(ActionEvent event) {
16        int right = 0;
17        int wrong = 0;
18        for (int i = 0; i < textCont.length; i++) {
19            System.out.println(i);
20            String jaw = textCont[i].getText();
21            System.out.println(jaw);
22            String kun = answers.getQuizAnswers().get(i);
23            System.out.println(answers.getQuizAnswers().get(i)
24                                    .equals(textCont[i].getText()));
25            System.out.println(kun);
26            if (!jaw.equals(kun)) {
27                textCont[i].setStyle("-fx-background-color: red;"
28                                        +"-fx-text-fill: white");
29                wrong++;
30            } else {
31                right++;
32            }
33        }
34        this.right.setText("Benar : " + right);
35        this.wrong.setText("Salah : " + wrong);
36    }
```

Gambar 5.30 Bagian Kelas ExerciseMenu metode evaluateAnswer

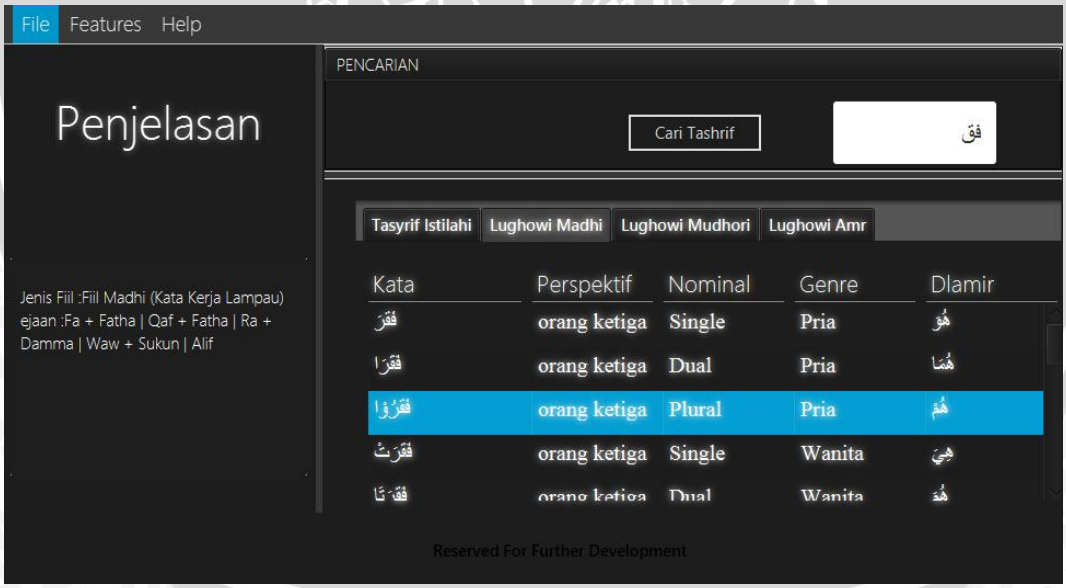
Keterangan gambar 5.30 :

1. Baris 1-13 : metode testStart bertugas untuk merespon ketika user telah memilih salah satu soal kata kerja yang akan dijawab. Awalnya dari soal kata kerja yang dipilih akan diinstansiasi menjadi objek facade kemudian memanfaatkan metode getQuizAnswers untuk menampilkan kunci jawaban pada tampilan menjawab soal.
2. Baris 14-36 : metode evaluateAnswer bertugas untuk merespon ketika tombol jawab (evaluasi) ditekan oleh user. Didalam metode ini ada statement yang mengecek jawaban user dengan kunci jawaban yang benar.



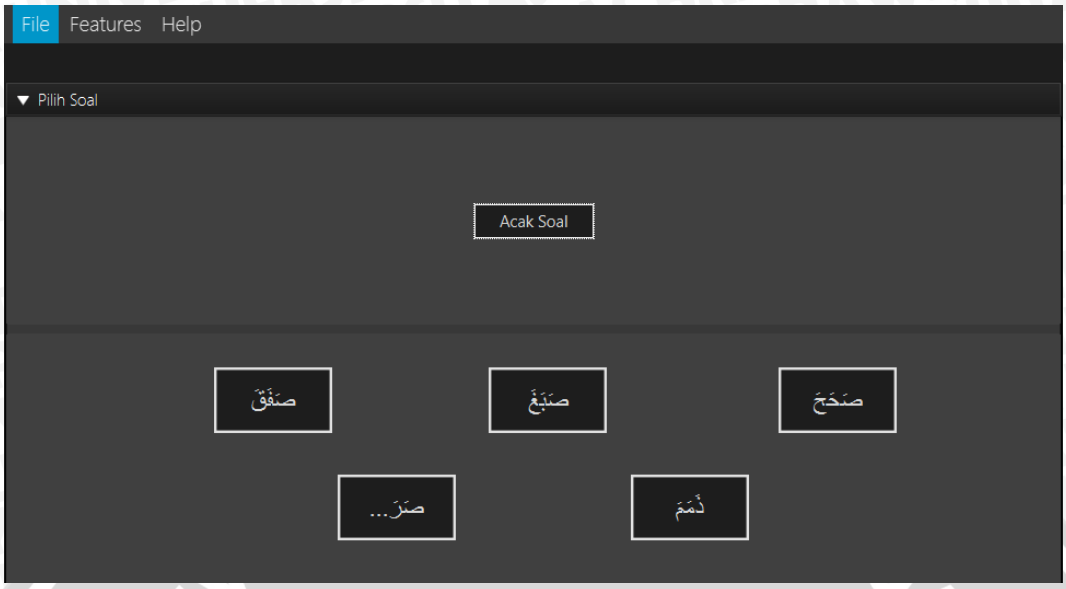
Gambar 5.31 Tampilan grafis fitur Pencarian Tashrif : Tashrif Istilahi

Gambar 5.31 menunjukkan tampilan grafis ketika pengguna menggunakan fitur Pencarian Tashrif yang menghasilkan perubahan kata kerja beserta penjelasan yang berkaitan mengenai perubahan tersebut.



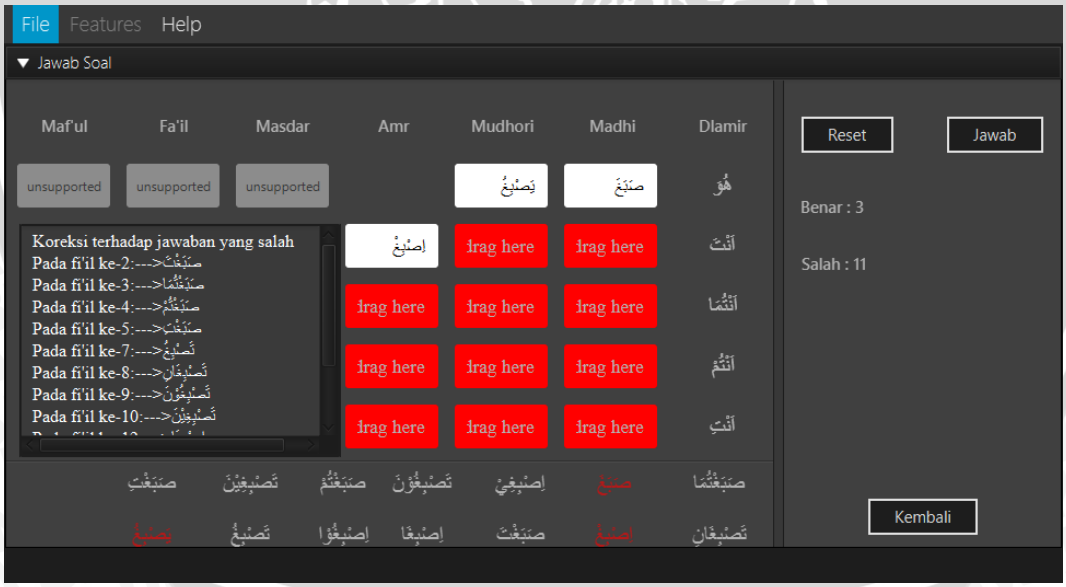
Gambar 5.32 Tampilan grafis fitur Pencarian Tashrif : Tashrif Lughowi

Gambar 5.32 menunjukan tampilan grafis ketika pengguna sudah memilih salah satu hasil pencarian (Tasyrif Istilahi) maka tabel pada Lughowi Madhi, Mudhori dan Amr akan berisi data perubahan tasyrif istilahi yang dipilih tersebut.



Gambar 5.33 Grafis fitur Latihan Tashrif : Pengacakan & Pemilihan soal

Gambar 5.33 menunjukkan bagian dari fitur Latihan Tashrif dimana pengguna awalnya diinstruksikan untuk mengacak soal dan memilih soal kata kerja yang hendak dikerjakan.



Gambar 5.34 Grafis fitur Latihan Tashrif : Menjawab soal

Gambar 5.34 menunjukkan ketika pengguna sudah memilih soal maka akan diarahkah pada halaman menjawab soal tashrif. Perlu diketahui bahwa ketika masuk pada halaman ini maka pengguna tidak bisa berpindah ke fitur Pencarian Tashrif hingga selesai menjawab atau kembali ke halaman pemilihan soal.

BAB VI
PENGUJIAN DAN ANALISIS

Dalam Bab ini akan dilakukan pengujian dan analisis pada aplikasi pendukung pembelajaran Arab perubahan kata kerja. Pengujian akan dilakukan dalam tiga tahapan yaitu pengujian *basis path*, pengujian fungsional dan pengujian *usability*. Pengujian fungsional merupakan teknik pengujian black-box. Pengujian *usability* dilakukan dengan melakukan ujicoba aplikasi pada seorang yang ahli dalam pembelajaran Arab untuk mendapatkan umpan balik penggunaan dan pengembangan dari perspektif pengguna.

6.1 Pengujian

Pengujian perangkat lunak adalah proses eksekusi suatu program atau sistem perangkat lunak dengan tujuan untuk menemukan *error* didalamnya. Menemukan dan memperbaiki *error* dalam suatu program pada akhirnya akan meningkatkan kualitas dan nilai dari program itu sendiri.[MYE-04]

Basis Path testing merupakan [MCC-96]. salah satu jenis *whitebox testing* yang digunakan untuk melihat nilai kompleksitas modul suatu program dan menguji tiap *independent path* yang terdapat dalam modul tersebut. *Blackbox testing* berfokus pada pengujian persyaratan fungsional perangkat lunak yang telah didefinisikan pada bagian analisis persyaratan perangkat lunak tanpa harus mengetahui struktur dan *logic* internal di dalam sistem. Untuk dapat menguji fungsional perangkat lunak dibutuhkan sejumlah set kondisi *input* tertentu. Penentuan set *input* tersebut dapat menggunakan metode yang telah banyak digunakan beberapa diantaranya adalah *equivalence partitioning* dan *boundary value analysis.* Aplikasi pendukung pembelajaran bahasa Arab ini merupakan *prototype* yang masih mempunyai banyak ruang dalam pengembangan fungsionalitas sehingga umpan balik yang berasal dari seorang ahli yang biasa melakukan proses pembelajaran bahasa Arab diharapkan mampu memberi visi pengembangan dimasa depan. Untuk bisa merumuskan umpan balik pengembangan maka digunakan Pengujian *usability.*

6.1.1 Pengujian Basis Path

Pengujian basis path dilakukan dengan melakukan *sampling* dari beberapa algoritma metode yang terdapat pada kelas di bab Implementasi sistem. Metode yang diambil adalah metode preTransform dari kelas TmadhiAlgorithm dan metode init pada kelas TashrifLughowi.

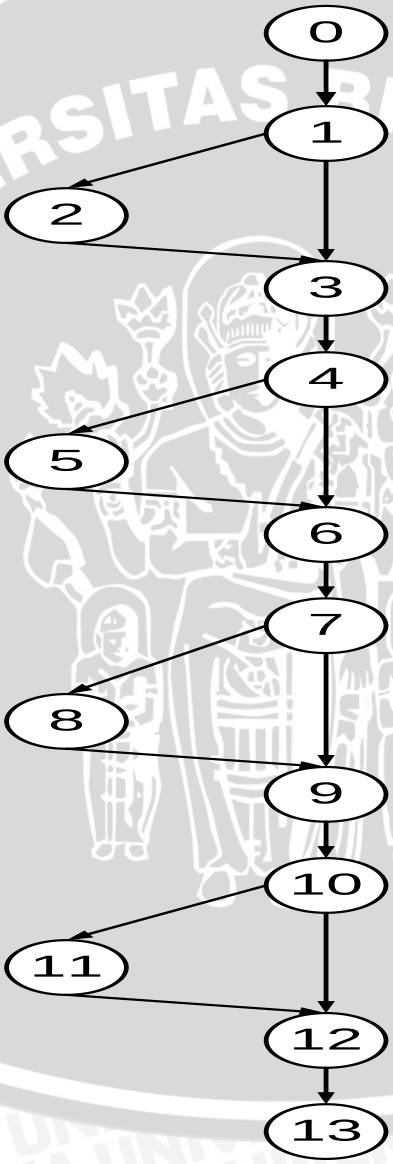
```
1   METHOD preTransform
2           Returns ArabicText
3           Accepts char1,char2 diacritic1,diacritic2
4
5       Type text IS ArabicText (0)
6
7       IF char1 NOT null (1)
8           THEN text+ = char1(2)
9       ENDIF(3)
10      IF diacritic1 NOT null (4)
11          THEN text+ = diacritic1(5)
12      ENDIF(6)
13      IF char2 NOT null (7)
14          THEN text+ = char2(8)
15      ENDIF(9)
16      IF diacritic2 NOT null(10)
17          THEN text+ = diacritic2(11)
18      ENDIF(12)
19
20  END preTransform        (13)
```

Gambar 6.1 Algoritma metode preTransform

Metode preTransform merupakan metode yang digunakan dalam merubah suatu kata kerja dengan melakukan penambahan imbuhan berupa huruf dan tanda baca sesuai dengan rumus perubahan yang ditentukan. Terdapat beberapa jenis metode preTransform yang diturunkan pada kelas-kelas yang mengimplementasikan *interface* TransformStrategy. Metode yang ditampilkan ini diambil dari kelas TMadhiAlgorithm. Nilai *Cyclomatic Complexity* dihitung menggunakan rumus perhitungan predikat sebagai berikut :

V(G) = p +1 ➔ v(G) = 4+1 ➔ v(G)=5

Metode preTransform memiliki nilai *cyclomatic complexity* berjumlah 9 sehingga terdapat 9 *independent paths* yang dapat dilalui pada metode ini. Untuk mempermudah pendefinisian path maka dibuat *control flow graph* gambar 6.1 dimana tiap statement algoritma dinotasikan dalam sebuah *node* dan *edge*. Predikat (statement kondisional) dinotasikan sebagai sebuah node yang memunculkan dua *edge* pada jalur yang berbeda. Predikat terletak pada node 1,2,4,6,9,10,12 dan 14.



Gambar 6.2 Control Flow Graph metode preTransform

Tabel 6.1 Tabel *independent paths* pada metode preTransform

No	Path	Node Sequence
1	Path 1 (baseline)	0-1-2-3-4-5-6-7-8-9-10-11-12-13
2	Path 2	0-1-3-4-5-6-7-8-9-10-11-12-13
3	Path 3	0-1-2-3-4-6-7-8-9-10-11-12-13
4	Path 4	0-1-2-3-4-5-6-7-9-10-11-12-13
5	Path 5	0-1-2-3-4-5-6-7-8-9-10-12-13

Tabel 6.1 menunjukkan varian *independent paths (basis path)* yang didefinisikan berdasarkan *control flow graph* yang telah dibuat menggunakan teknik *baseline*. Path 1 menunjukkan fungsionalitas metode preTransform dimana metode memberikan imbuhan pada kata kerja denga karakter dan tanda baca secara lengkap oleh sebab itu path 1 disebut sebagai *baseline*.

Tabel 6.2 Tabel kasus uji *independent paths* pada metode preTransform

Path	Input	Results		Validity
		Expected	Actual	
1	Sin, Kaf, fatha, kasra	فَعَلَسَكِ	فَعَلَسَكِ	Valid
2	Null, Kaf, fatha, kasra	فَعَلَ◌َكِ	فَعَلَ◌َكِ	Valid
3	Sin, Null, fatha, kasra	فَعَلَسَ◌ِ	فَعَلَسَ◌ِ	Valid
4	Sin, Kaf, Null, kasra	فَعَلَسكِ	فَعَلَسكِ	Valid
5	Sin, Kaf, fatha, Null	فَعَلَسَك	فَعَلَسَك	Valid

Tabel 6.2 menunjukkan hasil pengujian dari penetapan input yang dianggap dapat melalui *independent path* yang telah didefinisikan. Varian input ditentukan oleh kombinasi karakter arab dan tanda baca. Pengujian dilakukan dengan contoh pada penambahan imbuhan kata kerja فَعَلَ (fa'ala) dengan kombinasi karakter dan tanda baca sehingga dapat memenuhi kriteria independent path yang telah didefinisikan.

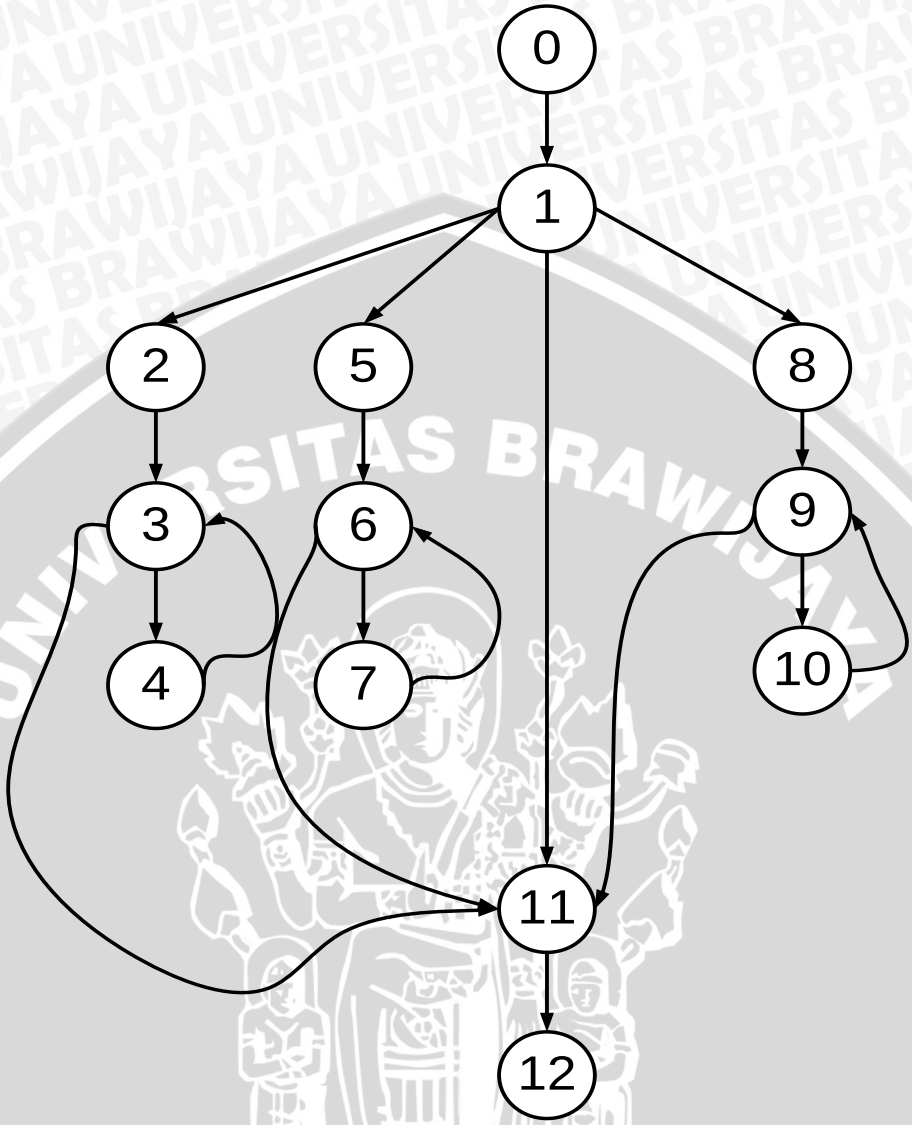
```
1   METHOD init
2       Returns Void
3       Accepts verb
4   Type verb IS Fiil
5   Type fail IS ARRAY of Fail                    (0)
6   Type collection IS ARRAY of Fiil
7
8       CHECK typeOfFiil(verb) (1)
9           IF verb=”fiil madhi” (2)
10              THEN DO FOR all elements in fail (3)
11                  ADD new instance of Fiil Madhi to collection (4)
12               ENDDOFOR
13
14          IF verb=”fiil mudhori” (5)
15              THEN DO FOR all elements in fail (6)
16                  ADD new instance of Fiil Mudhori to collection (7)
17               ENDDOFOR
18
19          IF verb=”fiil amr” (8)
20             THEN DO FOR all elements in fail (9)
21                  ADD new instance of Fiil Amr to collection (10)
22               ENDDOFOR
24
25          DEFAULT for verb = NONE
26      ENDCHECK (11)
27  END init (12)
```

Gambar 6.3 Algoritma metode init

Metode init pada gambar 6.3 merupakan salah satu metode yang terdapat pada kelas TasyrifLughowi yang bertugas untuk melakukan instansiasi objek kelas fiil sesuai dengan jenis fiil dan juga fail (jenis subyek) yang melekat pada fiil tersebut. Nilai *Cyclomatic Complexity* dihitung menggunakan rumus perhitungan predikat sebagai berikut :

V(G) = p +1 ➔ v(G) = 6+1 ➔ v(G)=7

Pada metode ini terdapat predikat yang berjenis multiway decisions yang tertulis pada baris 8-26 pada gambar 6.3 . Dalam pembuatan flow graph predikat jenis ini memiliki lebih dari 2 edges yang keluar dari node predikat tersebut.



Gambar 6.4 Control Flow Graph metode init

Pada gambar 6.4 dimulai dengan *multiway decisions* dengan 4 *edges* yang keluar dari *node* predikat. Pada 3 jalur yang utama didalamnya terdapat proses perulangan (*looping*). *Node* yang menggambarkan predikat terletak pada nomor 1,3,6 dan 9. Metode init memiliki nilai *cyclomatic complexity* berjumlah 7 sehingga terdapat 7 *independent paths* yang akan diujikan pada metode init.

Tabel 6.3 Tabel *independent paths* pada metode init

No	Path	Node Sequence
1	Path 1 (baseline)	0-1-2-3-4-.....-3-11-12
2	Path 2	0-1-2-3-11-12
3	Path 3	0-1-5-6-7-.....-6-11-12
4	Path 4	0-1-5-6-11-12
5	Path 5	0-1-8-9-10-.....-9-11-12
6	Path 6	0-1-8-9-11-12
7	Path 7	0-1-11-12

Pada tabel 6.3 terdapat tanda titik – titik (*ellipsis*) pada path 1, 3 dan 5 dimana tanda tersebut menunjukkan proses perulangan yang dilalui pada beberapa node. Tabel 6.4 menunjukkan hasil pengujian *independent paths* dimana terdapat 2 jenis input yang mempengaruhi hasil yaitu variabel jenis fiil dan jumlah elemen dalam array variabel fail. Hasil output dari metode ini dilihat dari jumlah dan jenis elemen yang tersimpan pada variable collection contohnya jika terdapat input berjenis fiil madhi dan fail berjumlah 5 subyek maka collection berisi 5 elemen fiil madhi dengan subyek yang berbeda.

Tabel 6.4 Tabel kasus uji *independent paths* pada metode init

Path	Input	Results		Validity
		Expected	Actual	
1	madhi, 5 elemen	5 elemen madhi	5 elemen madhi	Valid
2	madhi, Null	0 elemen	0 elemen	Valid
3	mudhori, 5 elemen	5 elemen mudhori	5 elemen mudhori	Valid
4	mudhori, Null	0 elemen	0 elemen	Valid
5	amr, 5 elemen	5 elemen amr	5 elemen amr	Valid
6	amr, Null	0 elemen	0 elemen	Valid
7	Null, Null	0 elemen	0 elemen	Valid

6.1.2 Pengujian Fungsional

Pengujian fungsional dilakukan pada 2 fitur aplikasi yaitu Pencarian Tashrif dan Latihan menjawab Tashrif. Penentuan set *input* yang akan menjadi kasus uji ditentukan dengan menggunakan metode *equivalence partitioning* yang akan membagi kategori *input* yang dianggap valid dan tidak valid. Pada tiap kategori ini nanti akan diambil contoh salah satu *input* dan diujikan pada fitur.

6.1.2.1 Fitur Pencarian Bentuk Tashrif Fiil

Fitur pencarian tashrif kata kerja pada aplikasi pendukung pembelajaran bahasa Arab mempunyai banyak varian *input* yang dapat diuji validitasnya. Untuk mempermudah penentuan set *input* maka diawal proses pengujian dapat digunakan pemisahan kategori kondisi *input* yang dianggap menghasilkan *output* yang valid dan yang tidak valid.

Tabel 6.5 Tabel *Equivalence Partition* pada fitur Pencarian Tashrif

Input Condition	Valid Equivalence Class	Invalid Equivalence class
Jenis karakter	1. Karakter Arabic (Saudi Arabia) 2. Karakter Arab + tanda baca	1. Non Arabic Saudi Arabia 2. *Alphanumeric* (a-z,0-9) 3. *Whitespace* 4. Hanya tanda baca
Jumlah karakter	1. 1-3 karakter	1. 0 karakter 2. >3 karakter
Bentuk Kata kerja	1. Karakter Arab non-illat	1. Karakter Arab illat (ء ا و ي) 2. Terdapat tanda baca tad'if

Metode *Boundary-Value Analysis* dapat digunakan secara komplemen dengan *equivalence partitioning* untuk menemukan kasus uji yang lebih detail dengan menguji batas nilai *input* dan *output* aplikasi. Nilai batas input pada jumlah karakter Arab adalah minimal 1 & maksimal 3. Nilai batas *output* juga diuji pada sistem, baik itu nilai minimum maupun maksimum yang dapat diberikan oleh aplikasi (sesuai dengan spesifikasi sistem).

Nilai batas *output* yang dapat diberikan oleh fitur pencarian tashrif adalah minimal 0 dan maksimal 348 kata kerja yang dapat ditampilkan.

Tabel 6.6 Tabel kasus uji *input* pada fitur Pencarian Tashrif

No.	Input Kasus uji	Keterangan Kasus uji
1	Dal ra sin (د ر س)	3 karakter Arab
2	Dal+fatha, ra+fatha, sin+fatha(دَرَسَ)	3 karakter Arab dengan harokat
3	abc123	*Alphanumeric*
4	ثقab	Karakter awal non-Arab
5	41فع	Karakter akhir non-Arab
6	ف ق	*Whitespace* ditengah karakter Arab
7	ُِ َ	Hanya tanda baca
8	ق	1 karakter Arab
9	Tidak ada *input*	*Input* kosong
10	فقلق	4 karakter Arab
11	افق	Karakter awal Huruf illat
12	فقي	Karakter akhir huruf illat
13	فَّقَلَ	Karakter awal dengan tad'if
14	فَقَلَّ	Karakter akhir dengan tad'if
15	َصَبَبَ	Diawali dengan tanda baca
16	ج	Cek output yang hasilnya nol
17	فَعُلَ	Cek output yang hasilnya 1
18	ن	Cek output yang hasilnya 348

Dari Tabel diatas didapatkan 18 kasus uji yang akan diujikan pada fitur pencarian aplikasi. Tiap kasus uji akan dieksekusi melalui prosedur skenario tertentu. Prosedur akan ditampilkan pada tabel 6.3 dan hasil pengujian akan ditampilkan pada tabel 6.4.

Tabel 6.7 Tabel Skenario uji *input* pada fitur Pencarian Tashrif

Nama Skenario	Uji Validitas pada *input* fitur Pencarian Tashrif
Tujuan pengujian	Untuk menguji validitas kinerja dari sistem dalam menyediakan pencarian tashrif kata kerja.
Persyaratan Sistem	SRS_001_01 sampai SRS_001_04
Prosedur uji	1. Masuk ke fitur pencarian Tashrif 2. Memasukkan kasus uji yang telah didefinisikan pada sejumlah 18 (delapan belas) jenis *input*. 3. Melihat respon aplikasi dari tiap jenis *input*. 4. Mengulang langkah ke-2-3 hingga kasus uji selesai.

Tabel 6.8 Tabel Hasil uji *input* pada fitur Pencarian Tashrif

Kasus uji	**Hasil yang diharapkan**	**Hasil yang didapatkan**	**Status Validitas**
1	Menampilkan kata kerja yang sama dan berdekatan / mirip.	Sesuai dengan yang diharapkan	valid
2	Menampilkan kata kerja yang sama dan berdekatan / mirip.	Sesuai dengan yang diharapkan	valid
3	Menampikan pesan *error* terdapat karakter selain Arab (alphanumeric)	Sesuai dengan yang diharapkan	valid
4	Menampikan pesan *error* terdapat karakter selain Arab (alphanumeric)	Sesuai dengan yang diharapkan	valid
5	Menampikan pesan *error* terdapat karakter selain Arab (alphanumeric)	Sesuai dengan yang diharapkan	Valid

6	Menampikan pesan *error* terdapat karakter selain Arab (whitespace).	Menampilkan pesan *error* tapi tidak spesifik penyebab kesalahan.	Tidak valid
7	Menampikan pesan *error* kata kerja diawali tanda baca/harokat	Tidak menampilkan pesan *error*. mengembalikan 0 output	Tidak valid
8	Menampilkan kata kerja yang diawali oleh karakter ini.	Sesuai dengan yang diharapkan	valid
9	Menampikan pesan *error* tidak ada *input*.	Sesuai dengan yang diharapkan	valid
10	Menampilkan kata kerja yang sama dan berdekatan / mirip dengan 3 huruf awal.	Sesuai dengan yang diharapkan	valid
11	Menampikan pesan *error* terdapat karakter huruf Arab illat	Sesuai dengan yang diharapkan	valid
12	Menampikan pesan *error* terdapat karakter huruf Arab illat	Sesuai dengan yang diharapkan	valid
13	Menampikan pesan *error* terdapat tanda baca tad'if	Menampilkan pesan *error* tapi tidak spesifik pada tanda baca tad'if.	Tidak valid
14	Menampikan pesan *error* terdapat tanda baca tad'if	Menampilkan pesan *error* tapi tidak spesifik pada tanda baca tad'if.	Tidak valid
15	Menampikan pesan *error* kata kerja diawali tanda baca/harokat	Tidak menampilkan pesan *error*. Mengembalikan 0 output.	Tidak valid

16	Menampilkan pesan tidak ada kata kerja yang ditemukan	Sesuai dengan yang diharapkan	valid
17	Menampilkan tepat 1 kembalian kata kerja	Sesuai dengan yang diharapkan	valid
18	Menampilkan tepat 348 kembalian kata kerja yang diawali karakter ini.	Sesuai dengan yang diharapkan	valid

6.1.2.2 Fitur Latihan Tashrif Fiil

Pada fitur latihan tashrif pengguna tidak bisa menentukan secara bebas *input* yang digunakan seperti pada fitur pencarian tashrif. Sehingga pada bagian pengujian ini akan dilakukan ujicoba terhadap spesifikasi dan urutan cara kerja sistem aplikasi.

Tabel 6.9 Tabel *Equivalence Partition* pada fitur Latihan Tashrif

Input Condition	Valid Equivalence Class	Invalid Equivalence class
Jumlah Soal	1. 5 soal	1. <5 soal 2. >5 soal
Kunci jawaban	1. 14 kunci jawaban	3. <14 kunci jawaban 4. >14 kunci jawaban
Pengacakan	1. Menampilkan 5 soal Fi'il yang berbeda 2. Menghasilkan set soal (5) yang berbeda ditiap pengacakan.	1. Terdapat soal Fi'il yang sama dalam 1 set (>2 soal) 2. Selalu menghasilkan set soal yang sama
Evaluasi jawaban	0-14 skor benar dan salah	1. <0 skor 2. >14 skor

Terdapat beberapa kasus uji yang akan digunakan untuk menguji beberapa *equivalence classes* sekaligus. Hal ini dikarenakan terdapat spesifikasi yang terikat satu sama lain seperti dalam hal menghasilkan jumlah soal dan pengacakan yang dilakukan dalam satu waktu yang bersamaan.

Tabel 6.10 Tabel Daftar Kasus uji pada fitur Latihan Tashrif

No.	Kondisi Kasus uji	Keterangan Kasus uji
1	5 kali menghasilkan soal	Melihat apakah jumlah soal konsisten berjumlah 5 dan terpilih secara acak
2	Memilih 5 soal berbeda	Melihat apakah jumlah kunci jawaban yang dihasilkan adalah tepat 14 jawaban.
3	Menjawab soal dengan 0 skor benar	Menguji nilai batas minimum skor benar = 0 (maks. skor salah = 14)
4	Menjawab soal dengan 14 skor benar	Menguji nilai batas maksimum skor benar = 14 (minimum skor salah = 0)

Terdapat 4 kasus uji yang didapatkan dari proses *equivalence partitioning*, berikut merupakan skenario pengujian :

Tabel 6.11 Kasus uji fitur Latihan Tashrif : Jumlah dan Pengacakan Soal

Nama Skenario	Menghasilkan dan Mengacak soal kata kerja fi'il
Tujuan pengujian	Untuk menguji validitas fungsi menghasilkan dan mengacak soal kata kerja.
Persyaratan Sistem	SRS_002_01
Prosedur uji	1. Masuk ke fitur latihan Tashrif 2. Menekan tombol acak soal 3. Melihat soal kata kerja yang dihasilkan 4. Mengulangi langkah ke-2-3 sebanyak 5 kali atau hingga terdapat *error*.
Hasil yang diharapkan	1. Fungsi selalu menghasilkan 5 soal kata kerja

	2. Fungsi selalu menghasilkan kata kerja yang berbeda dalam satu set / 5 soal. 3. Fungsi selalu menghasilkan set soal kata kerja yang acak.
Hasil yang didapatkan	Kadang mengembalikan 4 soal saja walau sangat jarang terjadi.
Validitas	Tidak Valid

Tabel 6.12 Kasus uji fitur Latihan Tashrif : Pilih Soal dan Jumlah Kunci Jawaban

Nama Skenario	Pilih Soal dan Jumlah Kunci Jawaban
Tujuan pengujian	Untuk menguji validitas fungsi memilih soal kata kerja dan jumlah kunci jawaban yang dihasilkan.
Persyaratan Sistem	SRS_002_02
Prosedur uji	1. Masuk ke fitur latihan Tashrif 2. Menekan tombol acak soal 3. Memilih satu soal yang ditampilkan 4. Melihat jumlah kunci jawaban yang dihasilkan 5. Pilih kembali ke pemilihan soal 6. Ulangi langkah 3-5 pada semua soal yang ditampilkan (5 soal)
Hasil yang diharapkan	1. Fungsi menampilkan soal tashrif yang tepat 2. Fungsi selalu menghasilkan sejumlah 14 kunci jawaban.
Hasil yang didapatkan	Sesuai dengan yang diharapkan
Validitas	valid

Tabel 6.13 Kasus uji fitur Latihan Tashrif : Menjawab Soal Skor minimum

Nama Skenario	Menjawab soal dengan skor minimum
Tujuan pengujian	Untuk menguji validitas fungsi evaluasi jawaban dengan kondisi user mendapat nilai skor minimum (0).
Persyaratan Sistem	SRS_002_03 dan SRS_002_04
Prosedur uji	1. Masuk ke fitur latihan Tashrif 2. Menekan tombol acak soal

	3. Memilih satu soal yang ditampilkan 4. Tidak mengisi jawaban , langsung tekan tombol evaluasi. 5. Pilih reset jawaban 6. Mengisi jawaban secara penuh dan sengaja diisi jawaban salah. 7. Tekan evaluasi
Hasil yang diharapkan	Skor benar: 0 dan skor salah: 14
Hasil yang didapatkan	Sesuai dengan harapan
Validitas	valid

Tabel 6.14 Kasus uji fitur Latihan Tashrif : Menjawab Soal Skor maksimum

Nama Skenario	Menjawab soal dengan skor maksimum
Tujuan pengujian	Untuk menguji validitas fungsi evaluasi jawaban dengan kondisi user mendapat nilai skor maksimum (14).
Persyaratan Sistem	SRS_002_03 dan SRS_002_04
Prosedur uji	1. Masuk ke fitur latihan Tashrif 2. Menekan tombol acak soal 3. Memilih satu soal yang ditampilkan 4. Mengisi jawaban secara penuh dan diisi jawaban benar. 5. Tekan evaluasi
Hasil yang diharapkan	Skor benar: 14 dan skor salah: 0
Hasil yang didapatkan	Sesuai dengan harapan
Validitas	valid

6.1.3 Pengujian Usability

Pengujian *Usability* digunakan untuk menemukan permasalahan kemudahan penggunaan sistem aplikasi dari sudut pandang pengguna atau faktor manusia. Dalam konteks pengembangan aplikasi pendukung pembelajaran ini maka dibutuhkan beberapa pengujian yang digunakan untuk mengukur tingkat kemudahan penggunaan aplikasi.

Secara umum tingkat kemudahan penggunaan aplikasi dirumuskan dalam 2 hal utama yaitu dalam hal navigasi dan memahami cara kerja penggunaan sistem. Berikut adalah tabel yang merupakan susunan kriteria kasus uji yang lebih detail dari 2 hal diatas berdasarkan adaptasi [MYE-04] :

Tabel 6.15 Kriteria Pengujian *usability* aplikasi pendukung pembelajaran

No.	Kriteria	Keterangan Kriteria
1	Navigasi menu	Melihat apakah pengguna dapat mengetahui dengan pasti menu yang digunakan dan cara berpindah antar menu.
2	Informasi Keluaran (*Output*)	Melihat apakah keluaran (*output*) yang diberikan oleh sistem berisi informasi yang dapat dipahami dan relevan.
3	Informasi Kesalahan *(Error*)	Melihat apakah informasi pesan kesalahan (*error*) yang diberikan oleh sistem mudah dipahami dan ditindaklanjuti oleh pengguna.
4	Integritas Konseptual Antarmuka	Melihat apakah antarmuka yang ditampilkan memiliki konsistensi dalam format grafik dan penggunaan istilah yang sesuai dengan konvensi.
5	Interaktivitas Antarmuka	Melihat apakah antarmuka grafik yang ditampilkan memiliki semacam umpan balik atau pesan interaktif pada pengguna seperti animasi tombol, proses pemuatan data (*loading progress*), sehingga pengguna mudah memahami.

Kelima (5) kriteria yang telah didefinisikan pada tabel 6.11 akan dijadikan acuan penilaian yang diberikan oleh pengguna pada aplikasi pendukung pembelajaran ini. Skenario pengujian yang dibuat nantinya akan terdapat beberapa kriteria tersebut untuk mendapatkan umpan balik dari pengguna. Pengguna yang nantinya akan menjalankan skenario pengujian adalah seorang pengajar bahasa Arab di salah satu madrasah kota Malang.

Tabel 6.16 Kasus uji Pengujian *usability*: Skenario Uji Navigasi

Nama Skenario	Uji Navigasi
Tujuan pengujian	Untuk menguji kemudahan navigasi antar fitur aplikasi berdasarkan sudut pandang pengguna sehingga bisa mengidentifikasi fitur yang digunakan dengan benar.
Prosedur uji	1. Pengguna diberikan instruksi untuk melakukan proses dari awal membuka aplikasi 2. Berpindah – pindah fitur dan mengidentifikasi kegunaan tiap fitur. 3. Keluar dari aplikasi
Kriteria yang dinilai	1. Navigasi Menu 2. Integritas Konseptual Antarmuka 3. Interaktivitas Antarmuka

Tabel 6.17 Tabel Hasil Kuesioner: Skenario Uji Navigasi

No	Kriteria	Pertanyaan	Jawaban
1	Navigasi Menu	Apakah Anda dapat menemukan fitur Pencarian Tashrif dan Latihan Tashrif?	Ya Saya menemukan
2	Integritas Konseptual Antarmuka	Apakah pendapat Anda mengenai tata letak Menu aplikasi?	Tata letaknya cukup baik namun terlihat banyak ruang kosong.

		Apakah pendapat Anda tentang jenis dan ukuran *font* dan tema warna yang digunakan?	Jenis dan ukuran fontnya kurang menarik, warnanya kurang variasi.
3	Interaktivitas Antarmuka	Apakah menurut Anda program menunjukkan perbedaan dalam proses perpindahan menu?	Ya menunjukkan perbedaan
		Apakah menurut Anda tombol navigasi mudah dikenali?	Tombol navigasi agak susah dikenali

Tabel 6.18 Kasus uji Pengujian *usability*: Skenario uji fitur Pencarian Tashrif

Nama Skenario	Mencari Macam-macam Bentuk Tashrif
Tujuan pengujian	Untuk menguji kemudahan penggunaan fitur mencari tashrif oleh pengguna.
Prosedur uji	1. Pengguna diberi instruksi untuk masuk ke fitur pencarian Tashrif dan mencari bentuk tashrif dari kata berikut : a. صَبَبَ b. سَهُلَ c. عَلِمَ d. حَسِبُ e. حَسُنَ f. فَقَلَ g. افق h. قثab 2. Mengamati keluaran yang diberikan oleh aplikasi
Kriteria yang dinilai	1. Informasi keluaran 2. Informasi kesalahan 3. Integritas Konseptual Antarmuka 4. Interaktivitas Antarmuka

Tabel 6.19 Tabel Hasil Kuesioner: Skenario Uji fitur Pencarian Tashrif

No	Kriteria	Pertanyaan	Jawaban
1	Informasi keluaran	Apakah Anda dapat memahami informasi hasil Pencarian Tashrif ?	Ya Saya memahami
		Apakah pendapat Anda mengenai kelengkapan informasi tashrif tiap kata kerja yang dihasilkan?	• Informasinya cukup lengkap • Ada kata yang tidak ditemukan • Bentuk fi'il yang salah seperti قَثَّ<--قَثَثَ • Terdapat penggunaan nama dlamir yang tidak tepat :اَنْتُنَّ
		Apakah terdapat informasi yang tidak relevan /tidak diperlukan / sulit dipahami dari hasil pencarian fi'il ?	Ya seperti Perspektif , Nominal dan Genre alangkah lebih baik langsung arti kamu, kalian, aku dst.
2	Informasi kesalahan	Apakah terdapat pesan kesalahan yang muncul ketika Anda mengoperasikan fitur pencarian?	Ya
		Apakah pesan *error* yang Anda temui memang menampilkan informasi kesalahan yang tepat / mudah dipahami?	Ya informasi kesalahanya sudah tepat namun ada beberapa yang perlu dirubah seperti tad'if menjadi tasydid + huruf illat

3	Integritas Konseptual Antarmuka	Apakah pendapat Anda mengenai tata letak tampilan informasi / tabel aplikasi?	Ya cukup baik
		Apakah tombol pencarian dan juga fi'il hasil pencarian mudah dikenali?	Mudah dikenali
		Apakah ukuran dan jenis font yang digunakan untuk menampilkan hasil pencarian mudah dibaca?	Huruf fontnya bisa diperbesar lagi
4	Interaktivitas Antarmuka	Apakah Aplikasi memberikan informasi / tanda bahwa ia sedang dalam proses pencarian dan pemuatan data fi'il (*loading*)?	Ada
		Apakah Anda merasa mudah dalam menemukan data hasil tashrif istilahi dan Lughowi setelah proses pencarian?	Kurang mudah karena hurufnya belum diurutkan sesuai huruf hijaiyyah

Tabel 6.20 Kasus uji Pengujian *usability*: Skenario Uji Fitur Latihan Tashrif

Nama Skenario	Menjawab latihan soal Tashrif
Tujuan pengujian	Untuk menguji kemudahan penggunaan fitur latihan tashrif kata kerja.
Prosedur uji	1. Pengguna diinstruksikan masuk ke dalam menu latihan tashrif

	2. Pengguna diinstruksikan untuk menjawab 2 soal kata kerja yang diinginkan 3. Menjawab salah satu dari soal dengan kesalahan yang disengaja. Jumlah jawaban yang salah diserahkan pada pengguna.
Kriteria yang dinilai	1. Informasi keluaran 2. Navigasi Menu 3. Integritas Konseptual Antarmuka 4. Interaktivitas Antarmuka

Tabel 6.21 Tabel Hasil Kuesioner: Skenario Uji fitur Latihan Tashrif

No	Kriteria	Pertanyaan	Jawaban
1	Informasi keluaran	Apakah Anda dapat Menjawab soal dalam Fitur Latihan Tashrif ?	Ya
		Apakah pendapat Anda mengenai informasi koreksi jawaban tashrif yang dihasilkan?	Informasi koreksi sudah tepat
		Apakah pendapat Anda terhadap cara evaluasi / pemberian skor difitur pencarian Tashrif?	Cara evaluasi sudah tepat
2	Navigasi Menu	Apakah perpindahan dari halaman memilih soal ke menjawab soal diperlukan ?	Ya tetap diperlukan
		Apakah informasi pada jendela konfirmasi dapat mudah dipahami?	Mudah

3	Integritas Konseptual Antarmuka	Apakah pendapat Anda mengenai tata letak tampilan informasi/ lembar menjawab tashrif (tombol,jawaban dan sebagainya)?	Ya sudah baik tata letaknya
		Apakah istilah bahasa Arab yang digunakan sesuai dengan ilmu Sharaf ?	Ya sudah sesuai
		Apakah ukuran dan jenis huruf (font) yang digunakan untuk menampilkan kunci jawaban dan koreksi jawaban mudah dibaca?	Kurang jelas , agak susah dibaca
4	Interaktivitas Antarmuka	Apakah Program menunjukkan informasi /tanda bahwa ia dalam proses pengacakan soal Fi'il?	Ya ada
		Apakah Anda merasa mudah dalam menemukan kunci jawaban dan juga cara menjawab soal ?	Belum karena tidak ada instruksi
		Apa tombol *Acak Soal*, *Reset*, *Jawab* dan *Kembali* mudah dipakai?	Ya mudah

6.2 Analisis Pengujian

Proses analisis bertujuan untuk mendapatkan kesimpulan dari hasil pengujian aplikasi pendukung pembelajaran bahasa Arab yang telah dilakukan. Proses analisis yang dilakukan meliputi hasil pengujian basis path, pengujian fungsional dan pengujian *usability*.

6.2.1 Analisis Pengujian Basis Path

Proses analisis terhadap pengujian basis path meliputi hasil dari pengujian dari metode preTransform yang berada di kelas TMadhiAlgorithm dan metode init yang berada di kelas TasyrifLughowi. Tidak terdapat permasalahan pada pengujian metode init karena semua kasus uji berjalan baik berbeda dengan metode preTransform. Metode preTransform awalnya memiliki 9 *independent paths* yang perlu untuk diuji dan dari 9 *independent paths* yang didefinisikan hanya 7 yang dapat dilalui dan 2 jalur lainya tidak dapat dilalui. Kedua jalur itu tidak dapat dilalui disebabkan oleh tidak adanya input yang mampu untuk merealisasikan / mengeksekusi jalur tersebut. Terdapat dua node yang diharuskan memiliki nilai yang berlainan pada saat eksekusi pengujian sehingga terjadi kontradiksi logika dan akhirnya tidak ada input yang mampu memenuhi eksekusi *independent path* ini. Dalam pengujian basis path fenomena ini disebut sebagai *Unnecessary Complexity*. *Unnecessary complexity* merupakan struktur logika yang berlebihan dan bisa jadi tidak memenuhi fungsionalitas pada modul yang ditempatinya [MCC-96]. Permasalahan ini jika tidak ditangani akan membuat fungsionalitas modul menjadi sulit dipahami dan pada akhirnya mempersulit proses *maintenance* kode. Perekayasaan ulang pada modul dengan prinsip *complexity reduction* harus dilakukan dan akhirnya dihasilkan metode preTransform pada Gambar 6.1 dan juga flow graph pada Gambar 6.2 . Pada metode preTransform yang sudah diperbaiki terdapat 5 *independent paths* dan dari semua kasus uji yang dilakukan berhasil memenuhi eksekusi *paths* yang ada.

6.2.2 Analisis Pengujian Fungsional

Proses analisis terhadap pengujian fungsional meliputi hasil dari pengujian pada fitur Pencarian Tashrif dan Latihan Tashrif pada Subbab 6.1.2.1 dan 6.1.2.2. Secara keseluruhan terdapat total 22 kasus uji, 18 dari Pencarian Tashrif dan 4 dari Latihan Tashrif. Sebanyak 16 kasus uji dikategorikan valid (72%) dan 6 kasus uji yang dikategorikan tidak valid berdasarkan hasil pengujian yaitu 5 berasal dari Pencarian Tashrif dan 1 berasal dari Latihan Tashrif. Kasus uji pada hasil pengujian Fitur Pencarian Tashrif yang dikategorikan tidak valid adalah sebagai berikut :

Tabel 6.22 Tabel hasil kasus uji yang tidak valid pada fitur Pencarian Tashrif

No.	Kasus uji	Keterangan
1	ف ق	Menampilkan pesan kesalahan (*error*) tapi tidak spesifik ketika terdapat *Whitespace* ditengah karakter Arab.
2	ُ ِ َ	Tidak menampilkan pesan kesalahan (*error*) ketika terdapat hanya tanda baca dan tidak ada keluaran.
3	فَّقَلَ	Menampilkan pesan kesalahan (*error*) tapi tidak spesifik ketika terdapat karakter awal dengan tad'if.
4	فَقَلَّ	Menampilkan pesan kesalahan (*error*) tapi tidak spesifik ketika terdapat karakter akhir dengan tad'if.
5	َصَبَبَ	Tidak menampilkan pesan kesalahan (*error*) ketika diawali dengan tanda baca dan tidak ada keluaran.

Kegagalan aplikasi dalam melakukan validasi jenis masukan (*input*) disebabkan oleh kesalahan dalam merumuskan jenis *input* yang valid pada awal proses rancangan aplikasi. Kesalahan tersebut mempengaruhi proses penulisan kode program yang menggunakan statement kondisi dalam proses validasi *input* kata kerja (*fi'il*).

Kasus uji yang dikategorikan tidak valid pada fitur Latihan Tashrif adalah pada fungsi pengacakan dan menghasilkan soal. Fungsi berhasil menghasilkan soal secara acak akan tetapi dalam beberapa percobaan jumlah soal yang dihasilkan hanya 4 soal, padahal seharusnya terdapat 5 soal. Penggunaan *collection classes* (*List*, *Set*, *Map etc*.) yang kurang tepat diduga merupakan penyebab kesalahan pada kasus ini.

Pada kasus ini fungsi menggunakan *collection* bertipe *Set*. *Set* adalah salah satu jenis kelas koleksi yang tiap anggota elemen didalamnya bersifat unik / tidak bisa terdapat elemen yang sama (duplikasi). Sifat inilah yang dijadikan alasan penggunaan tipe *Set* sehingga soal kata kerja yang dihasilkan bisa berbeda.

6.2.3 Analisis Pengujian Usability

Proses analisis Pengujian *usability* meliputi semua proses pengujian yang dijelaskan pada subbab 6.1.2 dengan melakukan evaluasi terhadap hasil pengujian yang didapatkan. Tiap Kriteria yang telah didefinisikan untuk mengukur tingkat kemudahan penggunaan aplikasi akan ditinjau ulang dan dievaluasi.

1. **Navigasi Menu** : Berdasarkan hasil jawaban kuesioner dan wawancara yang dilakukan pada pengguna (dosen bahasa Arab) menunjukkan bahwa pengguna mampu melakukan perpindahan antar fitur dan juga mengidentifikasi kegunaan yang ada pada tiap fitur dengan cukup mudah.
2. **Informasi Kesalahan (*error*)** : Pengguna merasa informasi yang diberikan oleh pesan *error* cukup tepat dengan catatan untuk mengganti istilah yang lebih tepat pada pesan *error* seperti *tad'if* dengan *tasydid* sehingga lebih mudah untuk dipahami dan ditindaklanjuti.
3. **Integritas Konseptual Antarmuka** : secara umum tampilan tata letak dinilai cukup baik oleh pengguna dengan catatan terlihat ruang kosong yang bisa dioptimalkan. Penggunaan jenis font sudah tepat akan tetapi ukuran yang digunakan terlihat kecil sehingga pengguna menginginkan agar ukuran font diperbesar. Warna tema pada aplikasi dinilai kurang menarik oleh pengguna karena dianggap terlalu gelap, kurang bervariasi dalam menggunakan warna.

4. **Interaktivitas Antarmuka** : Pengguna merasa aplikasi memberikan tanda / perubahan transisi dari satu fitur ke yang lainya. Pengguna juga merasa aplikasi memberikan tanda pada proses pencarian dan pemuatan data (*loading*) sehingga aplikasi tidak terkesan mengalami gangguan ketika melakukan *background process*. Terdapat catatan bahwa tombol yang digunakan pada navigasi agak sulit dikenali. Pada fitur Latihan Tashrif pengguna juga cukup kesulitan mengenali kunci jawaban yang sebenarnya bisa melakukan proses *drag and drop* untuk menjawab soal tashrif. Dari hasil pengamatan ketika proses pengujian ada dugaan bahwa masalah ini disebabkan penggunaan istilah bahasa Inggris dan komponen jawaban yang kurang interaktif.
5. **Informasi Keluaran (*output*)**: Pada fitur Pencarian Tashrif pengguna merasa mampu memahami informasi yang dihasilkan oleh aplikasi akan tetapi terdapat beberapa catatan mengenai hasil keluaran yaitu terdapat kata kerja yang tidak bisa ditemukan. Hal ini disebabkan karena keterbatasan aplikasi yang masih belum sepenuhnya memiliki data kamus yang lengkap. Pengguna juga merasa kesulitan untuk menemukan hasil pencarian kata kerja yang tidak terurut berdasarkan huruf hijaiyah. Penggunaan istilah untuk menjelaskan informasi subyek kata kerja seperti Perspektif, Nominal dan Genre dianggap pengguna kurang sesuai dan sebaiknya diganti menggunakan kata ganti subyek secara langsung seperti dia, kamu dan aku dan sebagainya. Hasil tashrif kata kerja juga diharapkan dapat dilengkapi arti bahasa Indonesia seperti telah membaca, sedang membaca, bacalah dan sebagainya. Terdapat 1 penulisan *dlamir* (kata ganti subyek) yang kurang tepat pada informasi tabel tashrif lughowi. Pada fitur Latihan Tashrif pengguna tidak memberikan catatan tertentu atau dianggap informasi keluaran yang diberikan aplikasi dapat diterima dan dipahami.

BAB VII
PENUTUP

7.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi dan pengujian yang telah dilakukan dapat diambil kesimpulan sebagai berikut :

1. Alur untuk mendemonstrasikan pembelajaran perubahan kata kerja direpresentasikan dalam dua use cases sistem yaitu Pencarian Tashrif dan Latihan Jawab Tashrif. Alur diawali dari masukan fiil madhi diikuti dengan proses perubahan fiil berdasarkan jenis waktu (tashrif istilahi) lalu berdasarkan subyek (tashrif lughowi). Alur tersebut merupakan penyederhanaan dari yang diperoleh melalui proses pengamatan dan wawancara dengan pengajar bahasa Arab.
2. Kata kerja dan prinsip perubahanya dimodelkan sesuai dengan prinsip ilmu Sharaf. Pemodelan menggunakan konsep turunan dan *interface*. Pola perancangan (*Design Pattern*) yang digunakan adalah *Strategy pattern* untuk adaptasi dan penambahan algoritma perubahan kata kerja. *Facade pattern* digunakan untuk mempermudah kolaborasi antara objek Fiil dengan objek Tasyrif.
3. Implementasi pengembangan aplikasi pendukung pembelajaran bahasa Arab menggunakan bahasa pemrograman Java dengan memanfaatkan komponen perangkat lunak yang telah tersedia (*reuse*) antara lain : JQuranTree, Hibernate, Lucene Indexer & JavaFX.
4. Berdasarkan hasil pengujian fungsional yang menggunakan teknik *blackbox testing* mendapatkan nilai prosentase 72%. 5 kasus uji yang gagal berasal dari fitur Pencarian Tashrif disebabkan oleh perancangan logika validasi input yang kurang tepat. Satu kasus uji yang gagal berasal dari fitur Latihan Tashrif diduga disebabkan oleh penggunaan kelas koleksi Java yang tidak tepat.

5. Berdasarkan hasil Pengujian *usability* yang digunakan untuk mendapatkan umpan balik dari seorang ahli bahasa Arab ditemukan banyak saran dan kritik yang bisa digunakan untuk mengembangkan aplikasi pendukung pembelajaran bahasa Arab khususnya pada kriteria informasi keluaran diperlukan pembenahan pada jumlah kata kerja yang dikenali, pengurutan huruf hijaiyah yang tepat, penggunaan istilah subyek yang lebih simpel dan ditambahkan arti bahasa Indonesia

7.2 Saran

Saran yang diberikan untuk pengembangan penelitian dan aplikasi perangkat lunak selanjutnya antara lain :

1. Aplikasi hasil penelitian ini diharapkan mampu menjadi alternatif media yang digunakan dalam proses pembelajaran ilmu Sharaf bagi pengajar maupun siswa sehingga proses pembelajaran lebih variatif.
2. Pada penelitian selanjutnya bisa difokuskan pada pengaruh penggunaan perangkat lunak ini dalam proses pembelajaran dan juga interaksi manusia komputer yang lebih ideal serta interaktif dalam mendemonstrasikan proses belajar bahasa Arab ilmu Sharaf.
3. Memberikan fitur baru yaitu Materi Sharaf yang berfungsi untuk menjelaskan dasar ilmu sharaf kepada pengguna aplikasi sehingga pengguna bisa langsung mempelajari teori sharaf tanpa menggunakan buku.
4. Model Latihan Tashrif yang menggunakan arti kata sebagai soal latihan. Hal ini berbeda dengan fitur Latihan Tashrif yang memberikan soal pada perubahan bentuk kata.
5. Penambahan algoritma perubahan kata kerja khususnya dari kata kerja menjadi kata benda (*isim*). Dalam ilmu sharaf terdapat konsep *Isim Musytaq* atau kata benda yang berasal dari kata kerja. Terdapat 5 jenis kata benda yaitu *Mashdar* (pekerjaan), *Fa'il* (subjek pelaku), *Maf'ul* (objek), *Zaman* (waktu) dan *Makan* (tempat).
6. Penggunaan kamus ilmu Sharaf tertentu yang bisa disesuaikan karena pada Madrasah/institusi pendidikan tertentu ada kemungkinan penggunaan kamus Sharaf yang berbeda.

DAFTAR PUSTAKA

[ANW-12] Anwar, M. 2011. *Ilmu Sharaf*. Bandung: Sinar Baru Algesindo

[ALS-04] Al-Sughaiyer, Imad A. dan Al-Kharasi, Ibrahim A., 2004, *Arabic Morphological Analysis Techniques : A Comprehensive Survey*

[BER-14] Bernard, Emmanuel, Ferentschik, Hardy, Fernandez, Gustavo, Grinovero, Sanne, Memon, Nabeel A., dan Morling, Gunnar. 2014. *Hibernate Search Reference Guide*.

[CHA-13] Chappell, Gail, Potts, Jasper, dan Hildebrandt, Nancy. 2013. *Getting Started with JavaFX*.

[DUK-11] Dukes, Kais. 2011. *JQuranTree Java API*. Diperoleh dari http://corpus.quran.com/java.

[FRE-04] Freeman, Eric, Freeman, Elisabeth, Sierra, Kathy dan Bert Cartens. 2004. *Head First Design Patterns*. Sebastopol: O'Reilly Media, Inc.

[GAM-95] Gamma, Erich, Helm, Richard, Johnson, Ralph, dan Vlissides, John. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software.*

[HOL-06] Holzner, Steve. 2006. *Design Pattern for Dummies*. Indiana: Wiley Publishing, Inc.

[HUS-13] Huseno, Ahmad. 2013. *60 Hari Bisa Menerjemahkan Al-Quran Sendiri Panduan Belajar Bahasa Arab Metode Al-Huda*. Jakarta: Turos Pustaka

[ISL-14] Islahudin. 2014. Modul Pembelajaran Ilmu Sharaf

[MCC-96] McCabe, Thomas J. 1996. *Structured Testing: A Testing Methodology Using Cyclomatic Complexity Metric*.

[MYE-04] Myers, Glenford J. 2004. *The Art of Software Testing.* 2nd Edition. New Jersey: John Wiley & Sons, Inc.

[PEW-11] Pew Research Center's Forum on Religion dan Public Life, *The Future of the Global Muslim Population*, 2011, www.pewforum.org

[PRE-10] Pressman, Roger S. 2010. *Software Engineering A Practitioner's Approach*. 7th Edition. Boston: McGraw-Hill.

[QUT-14] Quthrub, tersedia di: http://sourceforge.net/p/qutrub/wiki/Home/

[SAR-14] Sarf, diperoleh dari http://sourceforge.net/projects/sarf/

[SOM-11] Sommerville, Ian. 2011. *Software Engineering*. 9th Edition. New York: Addison-Wesley.

LAMPIRAN

Kuesioner Pengujian usability Kepada Pengguna

Aplikasi Pendukung Pembelajaran Bahasa Arab

Berikan jawaban Anda beserta alasan dan saran jika diperlukan !

Skenario ke-1

Prosedur uji		1. Buka program Aplikasi 2. Jelajahi segala fitur yang ada dalam aplikasi 3. Keluar dari aplikasi
No	Pertanyaan	Jawaban
1	Apakah Anda dapat menemukan fitur Pencarian Tashrif dan Latihan Tashrif?	
2	Apakah pendapat Anda mengenai tata letak Menu aplikasi?	
3	Apakah pendapat Anda dengan jenis dan ukuran huruf (*font*) dan tema warna yang digunakan?	
4	Apakah Aplikasi menunjukkan perbedaan dalam proses perpindahan menu?	
5	Apakah tombol navigasi mudah dikenali?	

Skenario ke-2

Prosedur uji	1. Cari bentuk tashrif dari kata kerja berikut melalui fitur aplikasi : a. صَبَبَ b. سَهَلَ c. عَلِمَ d. حَسِبُ e. حَسُنَ f. فَقَلَ g. افق h. فقثab

No	Pertanyaan	Jawaban
1	Apakah Anda dapat memahami informasi hasil Pencarian Tashrif ?	
2	Apakah pendapat Anda mengenai kelengkapan informasi tashrif tiap kata kerja yang dihasilkan?	
3	Apakah terdapat informasi yang tidak relevan /tidak diperlukan / sulit dipahami dari hasil pencarian fi'il ?	
4	Apakah terdapat pesan kesalahan yang muncul ketika Anda mengoperasikan fitur pencarian?	
5	Apakah pesan kesalahan yang Anda temui memang menampilkan informasi kesalahan yang tepat / mudah dipahami?	

6	Apakah pendapat Anda mengenai tata letak tampilan informasi / tabel aplikasi?	
7	Apakah tombol pencarian dan juga fi'il hasil pencarian mudah dikenali?	
8	Apakah ukuran dan jenis huruf (*font*) yang digunakan untuk menampilkan hasil pencarian mudah dibaca?	
9	Apakah Aplikasi memberikan informasi / tanda bahwa ia sedang dalam proses pencarian dan pemuatan data fi'il (*loading*)?	
10	Apakah Anda merasa mudah dalam menemukan data hasil tashrif istilahi dan Lughowi setelah proses pencarian?	

Skenario ke-3

Prosedur uji	1. Masuk pada menu Latihan Tashrif. 2. Pilih salah satu soal yang telah diacak aplikasi. 3. Jawab dan lengkapi soal tashrif. 4. Ulangi langkah 2-3 tapi dengan sengaja ketika menjawab memberikan jawaban yang salah (jumlah jawaban terserah Anda).

No	Pertanyaan	Jawaban
1	Apakah Anda dapat Menjawab soal dalam Fitur Latihan Tashrif ?	
2	Apakah pendapat Anda mengenai informasi koreksi jawaban tashrif yang dihasilkan?	
3	Apakah pendapat Anda terhadap cara evaluasi / pemberian skor difitur pencarian Tashrif?	
4	Apakah perpindahan dari halaman memilih soal ke menjawab soal diperlukan ?	
5	Apakah informasi pada jendela konfirmasi dapat mudah dipahami?	
6	Apakah pendapat Anda mengenai tata letak tampilan informasi/ lembar menjawab tashrif (tombol,jawaban dan sebagainya)?	
7	Apakah menurut Anda istilah bahasa Arab yang digunakan sesuai dengan Ilmu Sharaf ?	

8	Apakah pendapat Anda mengenai ukuran dan jenis huruf yang digunakan untuk menampilkan kunci jawaban dan koreksi jawaban mudah dibaca?	
9	Apakah Aplikasi menunjukkan informasi /tanda bahwa dalam proses pengacakan soal Fi'il?	
10	Apakah Anda merasa mudah dalam menemukan kunci jawaban dan juga cara menjawab soal ?	
11	Apakah tombol *Acak Soal*, *Reset*, *Jawab* dan *Kembali* mudah dipakai?	

Saran Pengembangan Aplikasi di Masa Depan

.....
.....
.....
.....

Terima kasih atas partisipasi Anda dalam pengisian kuesioner dan komentar untuk pengembangan lebih lanjut dari Aplikasi Pendukung Pembelajaran Bahasa Arab.