BAB III

METODOLOGI DAN PERANCANGAN APLIKASI

3.1 Metodologi Penelitian

Langkah-langkah penelitian dalam perancangan dan pembangunan sistem ini dipaparkan sebagai berikut.

a. Studi Fisibilitas

Studi Fisibilitas digunakan untuk mengukur kelayakan dari ide pembangunan aplikasi *voice chatbot*. Pengukuran dilakukan dengan cara mewawancarai pihak *marketing* UMN dan ketua Program Studi Informatika UMN. Hasil wawancara digunakan untuk mengetahui jumlah pertanyaan yang biasa diterima dan pendapat marketing dan ketua program studi terhadap ide pembuatan aplikasi ini. Hasil wawancara dapat dilihat pada lampiran. Selain itu, daftar pertanyaan serta jawaban yang didapat akan digunakan sebagai basis pengetahuan aplikasi. Pengembangan pertanyaan dan jawaban juga dapat dilihat pada lampiran.

b. Telaah Literatur

Telaah literatur dilakukan dengan pembelajaran terhadap teori-teori yang ada. Teori-teori tersebut dicari dari literatur, jurnal, paper, atau sumber bacaan lain yang berhubungan dengan aplikasi yang akan dirancang dan dibangun. Teori-teori tersebut antara lain Program Dual Degree Informatika UMN, *voice chatbot*, algoritma Jaro-Winkler, Wit.ai dan EUCS. Uraian mengenai telaah literatur dapat dilihat pada Bab II.

c. Analisis dan Perancangan Aplikasi

Pada tahap ini dilakukan analisis kebutuhan yang diperlukan untuk pengembangan aplikasi. Data-data untuk pengetahuan dasar mengenai Program Dual Degree Informatika UMN yang didapatkan dari pihak *marketing* dan program studi akan diperiksa dan dibersihkan sesuai kebutuhan. Melihat data jawaban yang dapat berubah sewaktu-waktu maka pengguna aplikasi akan dibagi menjadi pengguna biasa dan admin. Admin akan memiliki kemampuan untuk mengubah dan menambah jawaban. Untuk menjadi admin maka akan dibuat sistem autentikasi dengan pengenalan *username* dan *password*. Setelah itu akan dilakukan perancangan terhadap desain antar muka web yang menjadi *user interface* dari aplikasi ini. Interaksi *chatbot* dengan pengguna dirancang dengan menggunakan tahap salam, perkenalan, tanya jawab, dan ucapan terima kasih. Setiap interaksi harus dilakukan secara bergantian yaitu ada yang mendengarkan dan ada yang berbicara. Selanjutnya untuk analisis kebutuhan dapat dilihat pada Bab 3 Subbab 3.2.

d. Pembangunan Aplikasi

Setelah perancangan aplikasi, pembangunan aplikasi berbasis web ini dilakukan dengan membuat antarmuka terlebih dahulu dalam bentuk HTML. Aplikasi akan menggunakan Web Speech API untuk mengubah suara menjadi teks dan sebaliknya, teks menjadi suara. Teks yang didapat kemudian akan diolah berdasarkan Natural Language Processing dengan platform Wit.ai. Setelah dimengerti maksud dari kalimat tersebut maka akan dihasilkan tanggapan sesuai pengetahuan yang dimiliki oleh *chatbot*. Pembahasan mengenai pembangunan atau implementasi dapat dilihat pada Bab IV Subbab 4.2.

e. Pengujian dan Evaluasi

Pengujian aplikasi dilakukan untuk mengetahui apakah aplikasi sudah berfungsi sesuai kebutuhan yang didefinisikan. Pengujian ini dilakukan secara *black box*. Pengujian bertujuan untuk memastikan aplikasi yang telah terintegrasi memberikan hasil yang sesuai. Evaluasi dilakukan dengan menyebarkan kuesioner setelah penggunaan *voice chatbot* berdasarkan pada model *End User Computing Satisfaction*. Penilaian yang digunakkan terdiri dari 6 tingkat skala Likert yang pilihannya dapat berupa sangat baik hingga sangat buruk. Pengujian dan evaluasi yang dilakukan dapat dilihat pada Bab IV Subbab 4.3.

f. Penulisan laporan

Proses dan hasil yang dilakukan dan didapat dalam penelitian akan dituangkan ke dalam laporan sebagai bukti telah melakukan penelitian.

3.2 Analisis Kebutuhan Aplikasi

Analisis kebutuhan aplikasi yang dihasilkan dilihat menjadi dua bagian. Pertama dari sisi pengguna aplikasi, untuk mengatasi pembaruan informasi jawaban yang diberikan *voice chatbot*, yang dinamai Jacob, maka diperlukan admin untuk melakukan perubahan tersebut. Oleh karena itu, aplikasi akan mengenal 3 jenis pengguna, yaitu

- Pengguna biasa
- Admin
- Super admin

Admin dan super admin akan memiliki hak untuk menambahkan jawaban yang mana digunakan sebagai basis pengetahuan Jacob. Selain itu super admin juga memiliki kemampuan khusus yaitu menambahkan admin baru ke dalam aplikasi.

Kedua dilihat dari fitur yang akan dimiliki Jacob. Selain fitur utama yaitu halaman awal dan berinteraksi dengan Jacob, terdapat fitur lain seperti

- Lihat halaman tentang / *about*
- Lihat jawaban
- Pengubahan jawaban
- Lihat *sample*
- Pengubahan *sample*
- Lihat *entity*
- Pengubahan *entity*
- Lihat log percakapan
- Hapus log percakapan
- Login
- Logout

Pengguna biasa (bukan admin) hanya dapat mengakses halaman awal, halaman *about*, dan berinteraksi dengan Jacob.

Proses interaksi pengguna dengan Jacob menggunakan bahasa Inggris sehingga pelafalan nama yang digunakan mungkin berbeda dalam kata bahasa Inggris. Dengan begitu rentan terjadi salah dari pendengaran nama yang diberikan pengguna. Untuk mengatasi atau mengurangi dampak kesalahan pelafalaan nama tersebut, maka nama yang disebutkan oleh pengguna akan diperiksa kembali dengan algoritma Jaro-Winkler yang diimplementasi. Nama dengan nilai Jaro-Winkler terbesar akan ditetapkan menjadi nama panggilan yang digunakan.

Untuk membuat Jacob dapat mendengar dan berbicara maka digunakan Web Speech API. Web Speech API tersedia sebagai JavaScript API dan memiliki

beberapa pengaturan seperti aktivasi dan penggunaan waktu sehingga mudah untuk diterapkan pada aplikasi web. Web Speech API dapat digunakan untuk beberapa hal seperti pencarian dengan suara, pendeteksi aktivitas suara, penerjemah suara, dan sistem dialog.

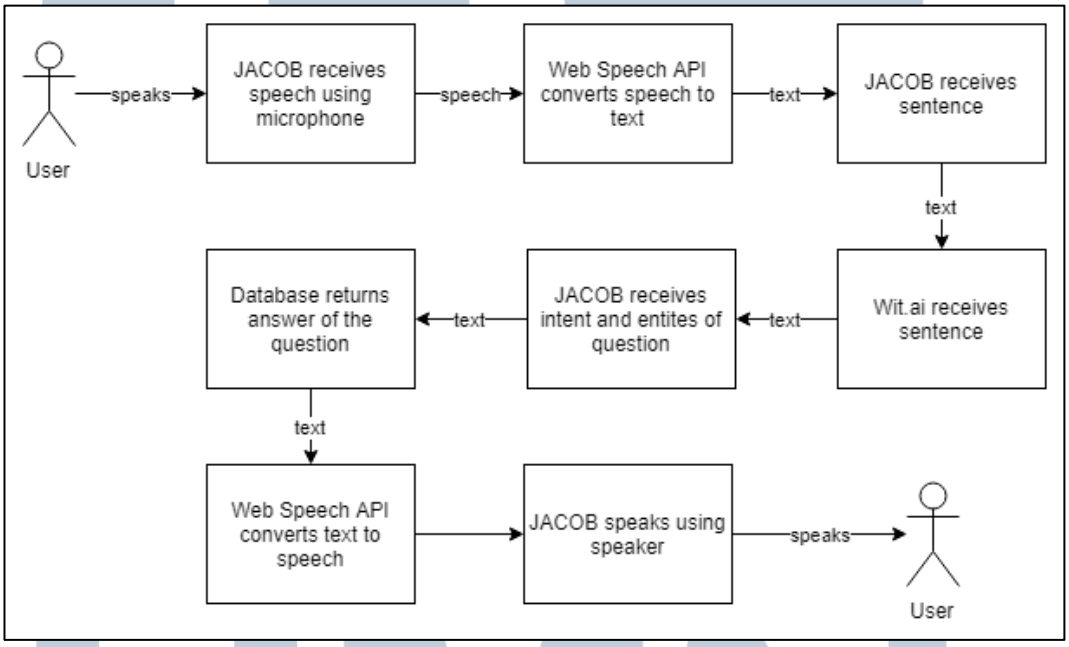
Web Speech API terbagi menjadi dua bagian yaitu SpeechRecognition dan SpeechSynthesis. SpeechRecognition diakses untuk mengenali suara masukan yang kemudian diubah menjadi teks. Bagian ini diimplementasikan menjadi objek yang memiliki beberapa e*vent handlers* dan juga antarmuka untuk menggunakan gramatika tertentu yang bertujuan mengenali kata yang diinginkan. SpeechSynthesis diakses untuk membacakan teks yang diberikan sehingga akan terdengar suaranya. Bagian ini juga diimplementasikan menjadi objek yang memiliki *event handlers* dan antarmuka untuk mengganti tipe suara dan teks yang digunakan untuk dibacakan. Kedua bagian ini memiliki perbedaan dukungan untuk penjelajah web yang digunakan. SpeechRecognition hanya sepenuhnya didukung dengan menggunakan penjelajah Google Chrome sedangkan SpeechSynthesis lebih banyak didukung pada penjelajah web yang ada.

Penerapan Wit.ai untuk memproses kalimat yang didapat dari *speech recognition* Web Speech API akan menghasilkan *intent* dan *entities*. *Intent* dan *entities* yang didapat akan digunakan Jacob untuk mencari jawaban yang sesuai. Dalam hal ini memungkinkan untuk satu *intent* memiliki jawaban yang berbeda tergantung pada *entity* yang dimilikinya. Kemudian, dari pasangan *intent* dan *entity* juga memungkinkan untuk memiliki lebih dari satu variasi jawaban. Sehingga dalam penyimpanan di dalam *database* akan terbagi menjadi 3 tabel yang

merepresentasikannya yaitu tabel knowledge_bases, tabel required_entities, dan tabel answers.

3.3 Perancangan Sistem

Perancangan sistem dari penelitian ini dibagi menjadi 6 bagian, yaitu *class diagram, use case diagram, activity diagram, sequence diagram*, struktur tabel dan perancangan antarmuka.



Gambar 3.1 Rancangan Umum Aplikasi Voice Chatbot Jacob

Gambar 3.1 menggambarkan rancangan umum untuk aplikasi Jacob. Pengguna akan berinteraksi dengan Jacob menggunakan suara. Suara yang diucapkan oleh pengguna akan diubah menjadi teks atau kalimat dengan menggunakan bantuan Web Speech API. Kalimat tersebut kemudian diterima Jacob dan diteruskan kepada *platform* Wit.ai dengan tujuan untuk mendapatkan *intent* dan *entities* dari kalimat. Wit.ai dapat menentukan *intent* dan *entities* dari kalimat-kalimat yang sudah diberikan dan dilatih. Kalimat tersebut berupa kalimat-kalimat tanya yang didapat dari hasil wawancara. *Intent* dan *entities* kemudian didapatkan

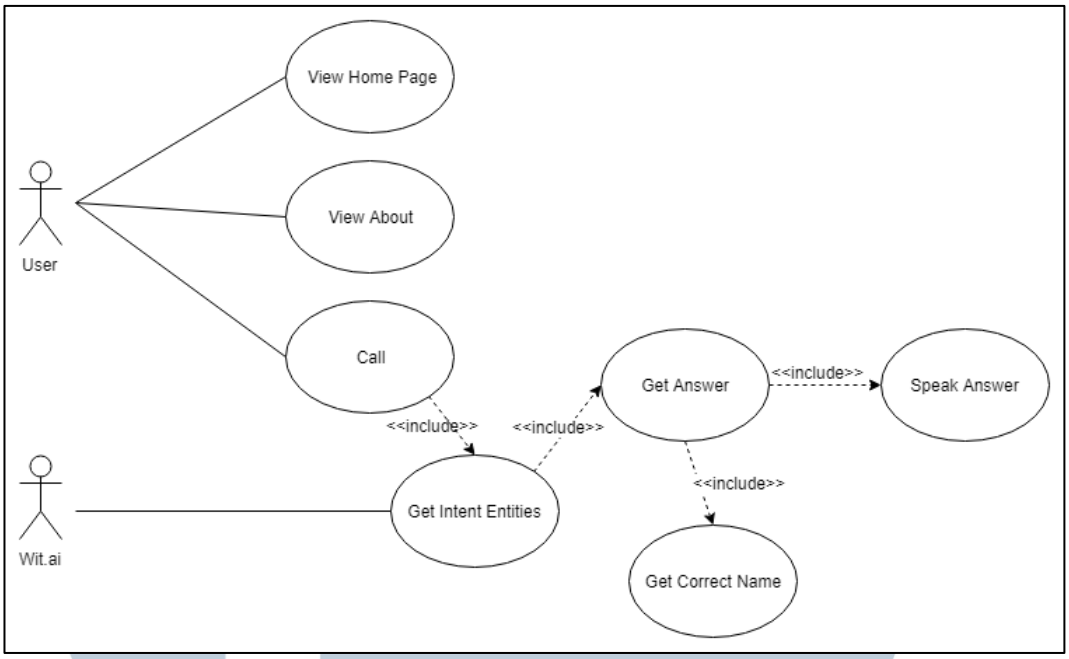
oleh Jacob untuk mencari jawaban yang ada pada *database*. *Database* dirancang dan dibangun dengan menyesuaikan hasil Wit.ai sesuai dengan analisis untuk mencari jawaban. Selain itu *database* juga berisi data-data untuk admin. Jawaban yang didapat kemudian akan dikirim ke pengguna. Pengguna akan mendengarkan jawaban dengan bantuan dari suara yang dihasilkan oleh Web Speech API.

3.3.1 Use Case Diagram

Use case diagram pada aplikasi Jacob terdiri dari 4 diagram yaitu Jacob, Admin, Register, dan Login.

A. Use Case Diagram Jacob

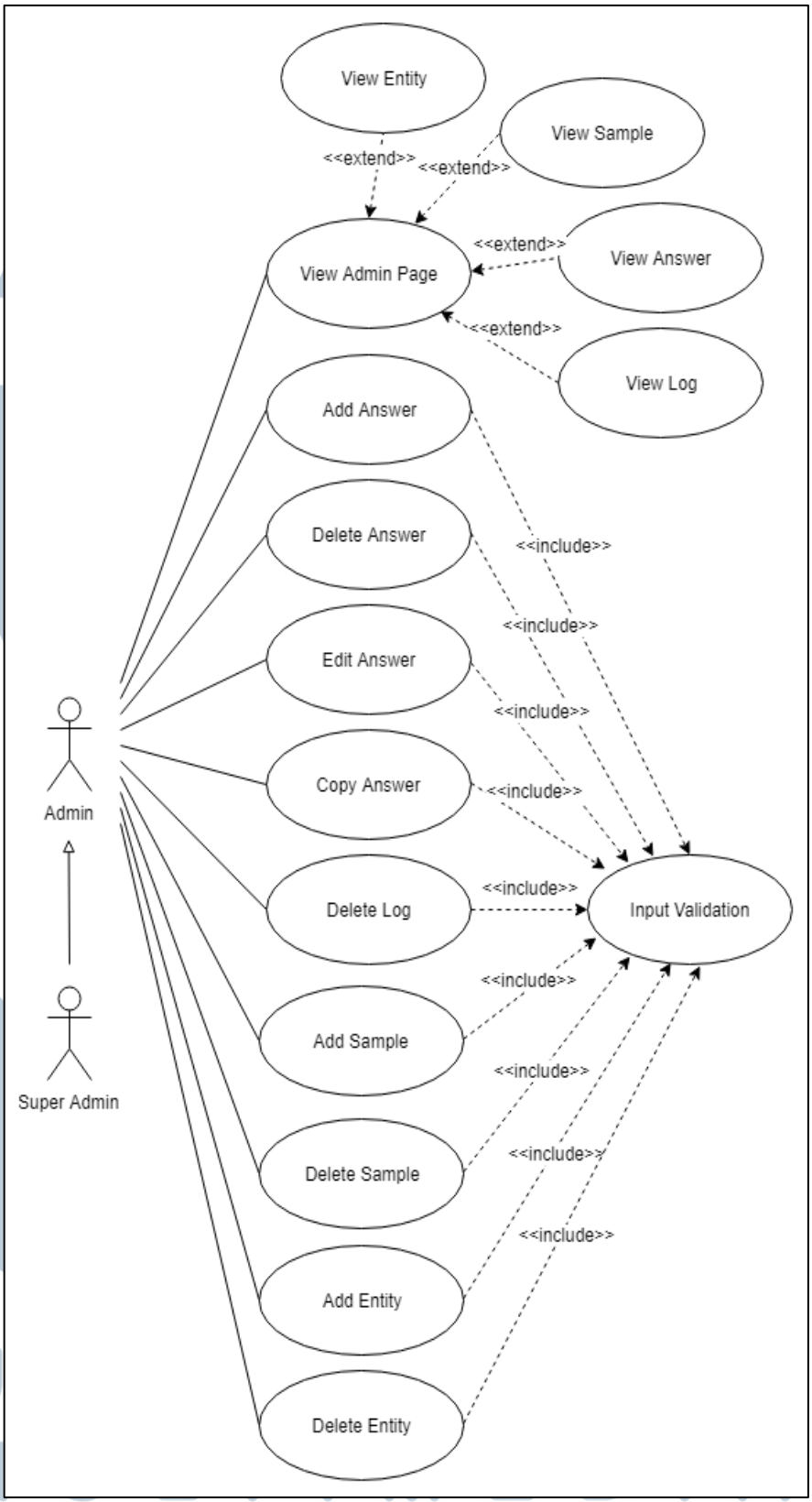
Gambar 3.2 menunjukkan *use case diagram* pada aplikasi Jacob. *Use case* terdiri dari 3 yaitu *view home page*, *view about*, dan *call*. *View home page* dan *view about* akan menampilkan sebuah halaman / *view* masing-masing yaitu HomePage dan AboutPage. Terdapat satu aktivitas yang dapat dilakukan *user* yaitu *call*. Dengan melakukan *call*, maka *user* dapat mulai berbicara dan mendapatkan jawaban. Pembicaraan akan diambil *intent* dan *entity*-nya dengan bantuan dari Wit.ai kemudian dicari jawabannya. Jawaban berupa nama *user* juga akan diolah dengan alogritma Jaro-Winkler melalui *function* getCorrectName(). Jawaban akan dimuat dalam bentuk suara kembali oleh HomePage dengan *function* speak().



Gambar 3.2 *Use Case Diagram* Jacob

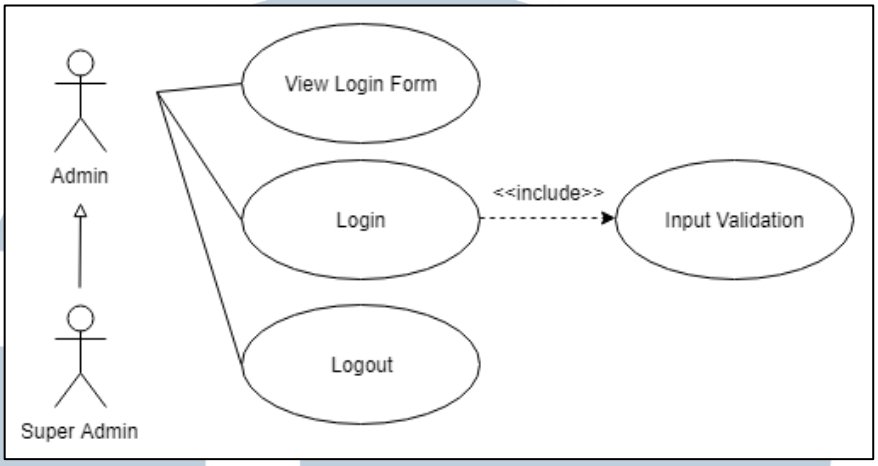
B. Use Case Diagram Admin

Gambar 3.3 menggambarkan *use case diagram* admin yang terdiri dari aktivitas melihat halaman admin yang dapat terdiri *entity* atau *sample* atau jawaban atau log dan aktivitas pengubahan untuk jawaban, *entity*, *sample*, dan jawaban. Log yang sudah ada dan tidak terpakai juga dapat dihapus pada halaman admin. Masukan yang diberikan pada saat pengubahan dilakukan akan divalidasi oleh aplikasi apakah dapat diterima sebagai perubahan.



Gambar 3.3 *Use Case Diagram* Admin

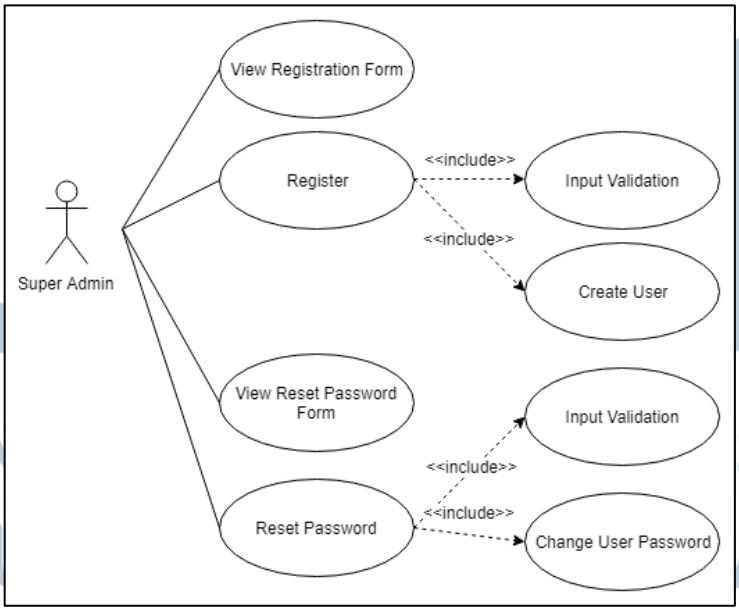
C. Use Case Diagram Login



Gambar 3.4 *Use Case Diagram* Login

Gambar 3.4 menunjukan *use case diagram* Login. Kegiatan yang dapat dilakukan lebih ditujukkan kepada admin, karena hanya admin yang dapat melakukan *login*. Masukan *login* akan divalidasi terlebih dahulu dan diperiksa apakah admin terdaftar. Selain itu admin juga dapat *logout* dari aplikasi.

D. Use Case Diagram Register



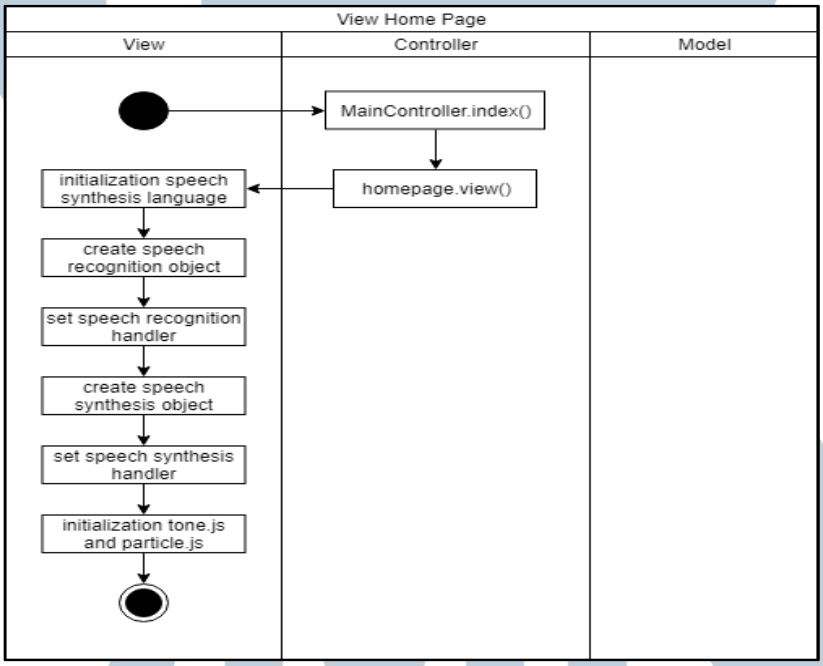
Gambar 3.5 *Use Case Diagram* Register

Gambar 3.5 menunjukkan *use case diagram* dari Register. Aktor dalam *use case* ini hanya dapat seorang admin yang memiliki tingkat super admin. Hanya super admin yang dapat mendaftarkan admin baru pada aplikasi dan juga melakukan reset kata sandi atau *password*.

3.3.2 Activity Diagram

Activity diagram pada aplikasi Jacob dibagi berdasarkan *use case* yang ada, yaitu

A. Activity Diagram View Home Page

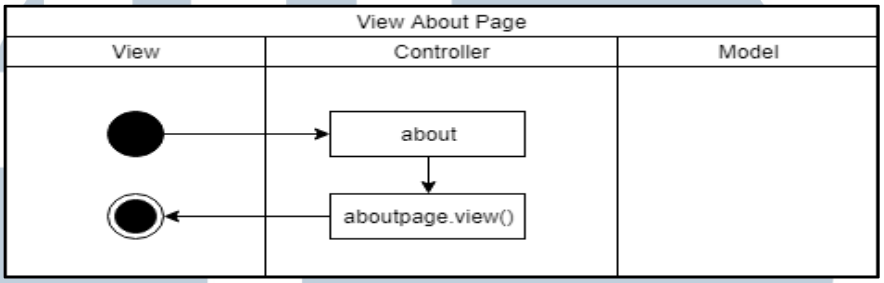


Gambar 3.6 *Activity Diagram* View Home Page

Gambar 3.6 menunjukan *diagram activity* ketika mengakses halaman utama Jacob. Fungsi Index pada MainController bertugas untuk memberikan tampilan dari homepage. Setelah homepage diterima pengguna maka akan dilakukan inisialisasi bahasa yang digunakan untuk *speech synthesis*. Kemudian membuat objek *speech recognition* dan *speech synthesis* berserta masing-masing fungsi *handler*-nya.

Terakhir dilakukan inisialisasi dari tone.js yang berguna sebagai animasi pada halaman.

B. Activity Diagram View About Page

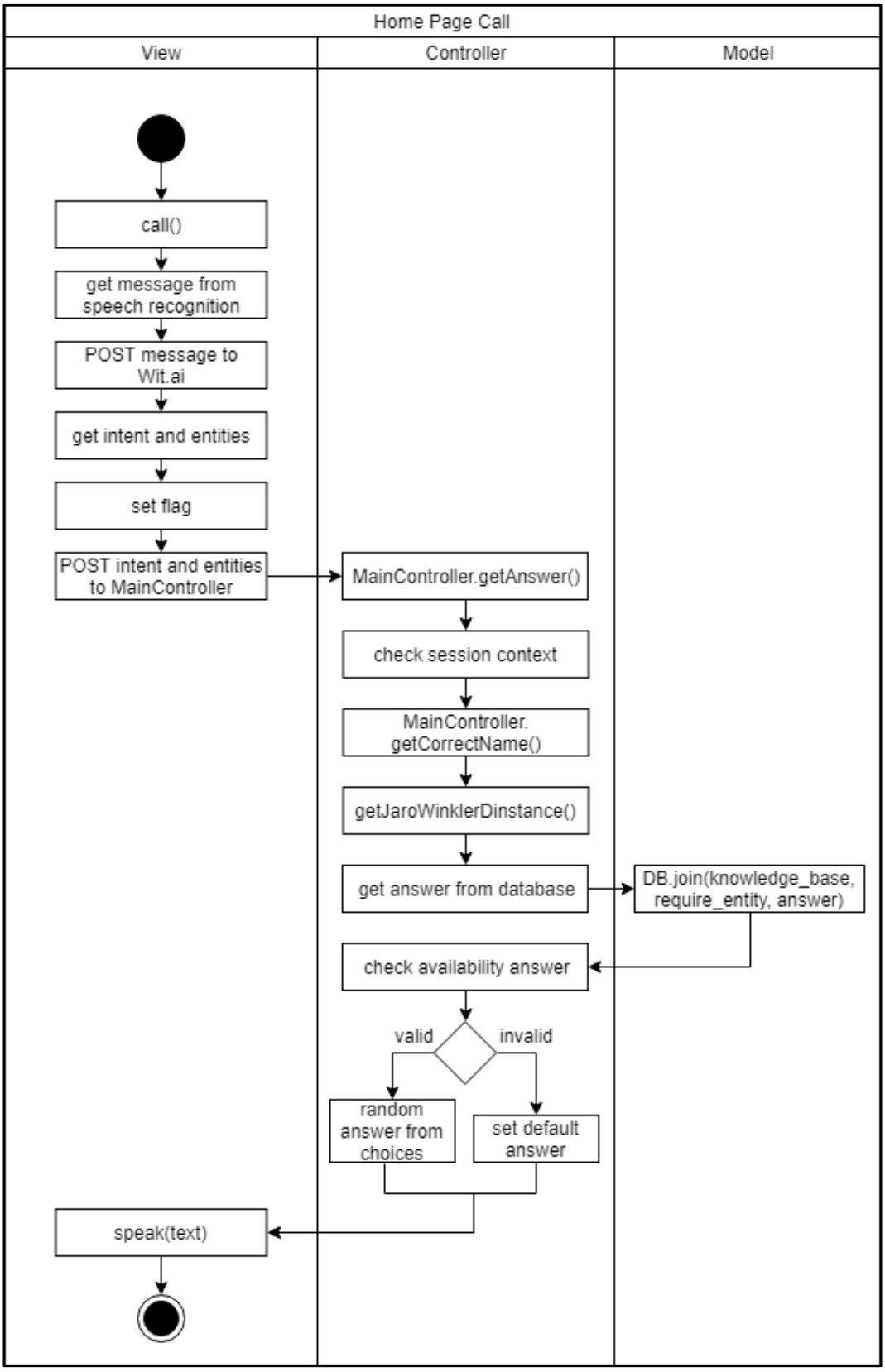


Gambar 3.7 *Activity Diagram* View About Page

Gambar 3.7 menggambarkan *activity diagram* ketika mengakses halaman *about*. MainController bertugas untuk memberikan tampilan halaman *about*. Dalam halaman tersebut terdapat penjelasan tentang *voice chatbot* dan juga tim pengembang.

C. Activity Diagram Call

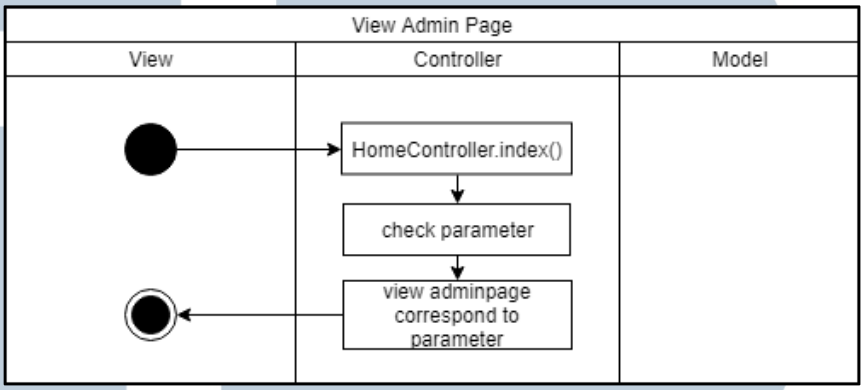
Gambar 3.8 merupakan *activity diagram* untuk fungsi call. Pengguna yang berinteraksi dengan Jacob akan diperoleh kalimatnya yang kemudian dikirim ke Wit.ai untuk didapatkan maksud (*intent*) dan entitas (*entities*) dari kalimat tersebut. Setelah didapatkan maka akan diatur beberapa penanda yang lalu *intent* dan *entities* akan dikirim ke MainController untuk dicari jawabannya. Di MainController akan ada pemeriksaan data-data *session* sebagai *context*. Jika ada perubahan untuk nama maka nama akan diperiksa dengan algoritma Jaro-Winkler untuk mendapatkan nama yang lebih benar. Selain itu akan diperiksa *intent* dan *entities* untuk mendapatkan jawaban yang sesuai. Jika jawaban tidak ada maka akan diberikan jawaban standar. Jawaban kemudian diberikan ke pengguna dengan suara.



Gambar 3.8 *Activity Diagram* Call

D. Activity Diagram View Admin Page

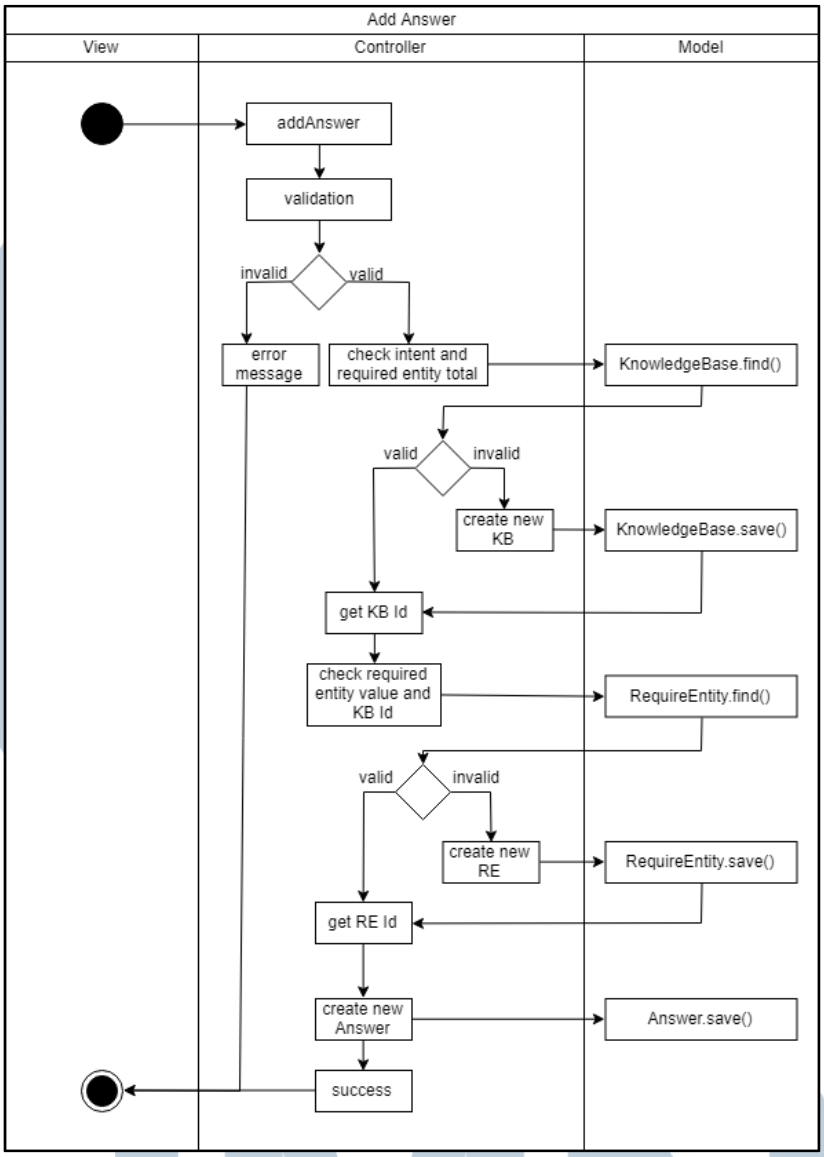
Gambar 3.9 merupakan *activity diagram* ketika mengakses halaman admin. Isi halaman admin akan disesuaikan dengan parameter yang diterima, seperti *entities*, *sample*, *list answers*, dan *logs*.



Gambar 3.9 *Activity Diagram* View Admin Page

E. Activity Diagram Add Answer

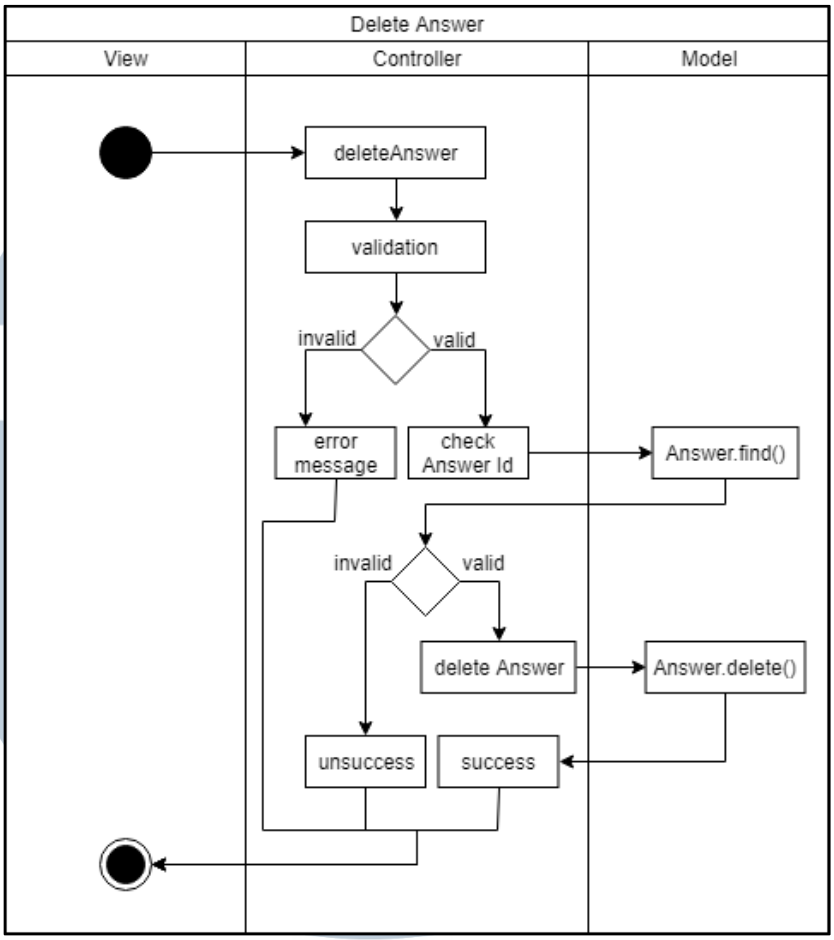
Gambar 3.10 menggambar *activity diagram* ketika admin menambahkan jawaban. Jawaban akan divalidasi terlebih dahulu sebelum disimpan ke dalam basis data atau *database*. Setelah valid maka akan dimasukkan ke *database* berurutan mulai dari tabel knowledge_bases, tabel required_entities, dan tabel *answers*. Sebelum dimasukkan ke dalam tabel knowledge_bases dan tabel required_entities akan diperiksa terlebih dahulu apakah sudah ada. Jika sudah ada maka akan akan digunakan yang lama, namun jika belum maka akan dibuatkan data yang baru. Setelah itu baru akan dibuatkan data untuk *answers* yang baru.



Gambar 3.10 *Activity Diagram* Add Answer

F. Activity Diagram Delete Answer

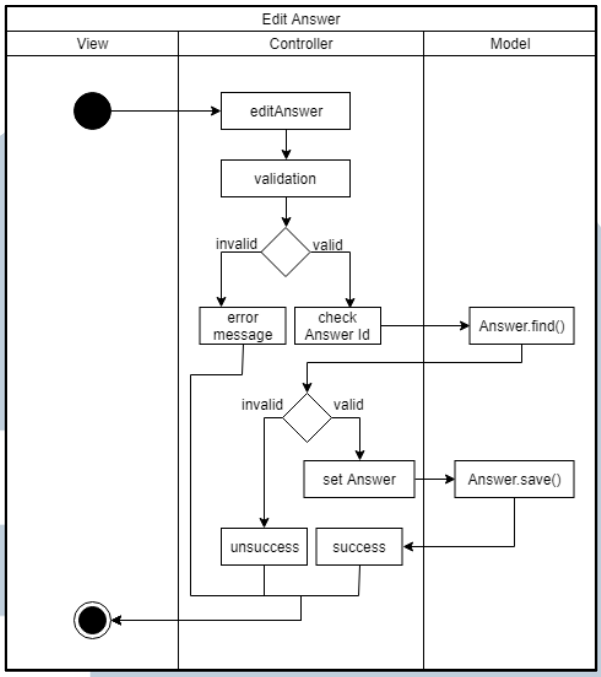
Gambar 3.11 menggambarkan *activity diagram* ketika admin menghapus jawaban. Masukan akan divalidasi dari HomeController yang kemudian jika valid akan mencari jawaban dengan model Answer dan menghapusnya.



Gambar 3.11 *Activity Diagram* Delete Answer

G. Activity Diagram Edit Answer

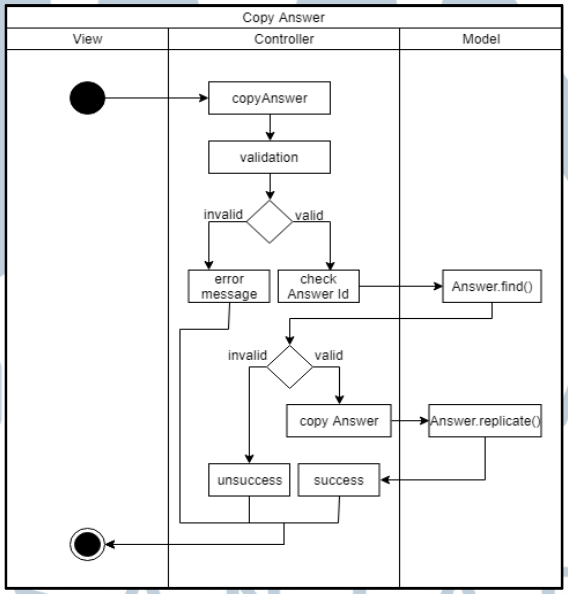
Gambar 3.12 menunjukkan *activity diagram* untuk edit jawaban. Masukan akan divalidasi oleh MainController yang kemudian jika valid akan dicari jawabannya dengan model Answer. Setelah diubah jawabannya maka akan disimpan ke *database*.



Gambar 3.12 *Activity Diagram* Edit Answer

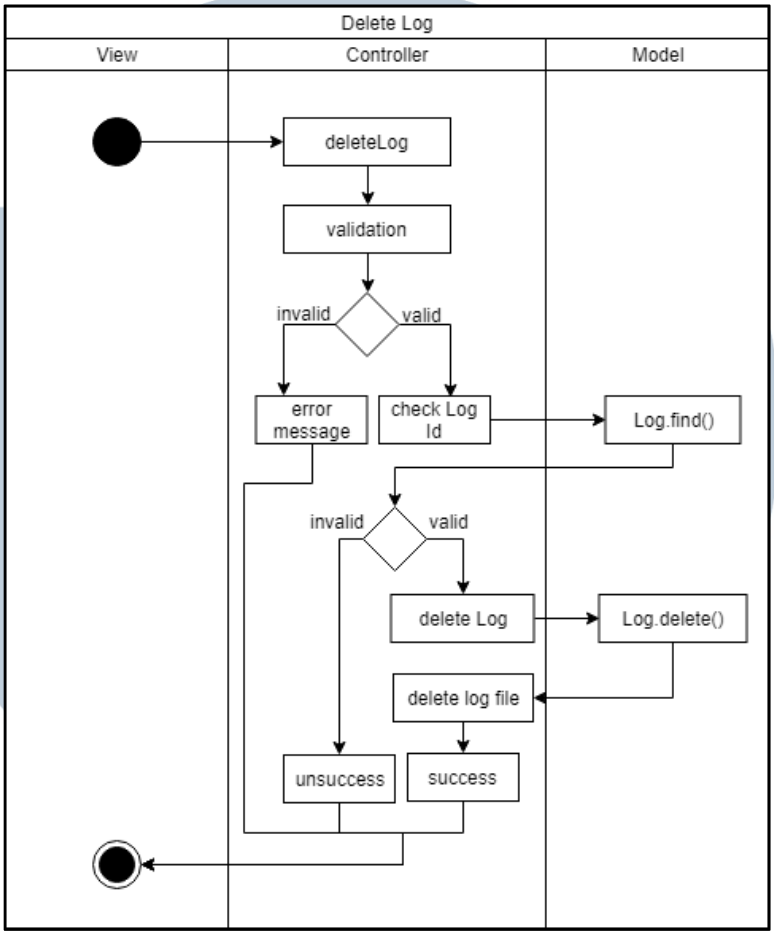
H. Activity Diagram Copy Answer

Gambar 3.13 menunjukkan *activity diagram* untuk menggandakan jawaban. Masukan akan divalidasi dan jika valid maka HomeController akan menggunakan model Answer untuk mencari jawabannya. Setelah ketemu maka akan digandakan.



Gambar 3.13 *Activity Diagram* Copy Answer

I. Activity Diagram Delete Log

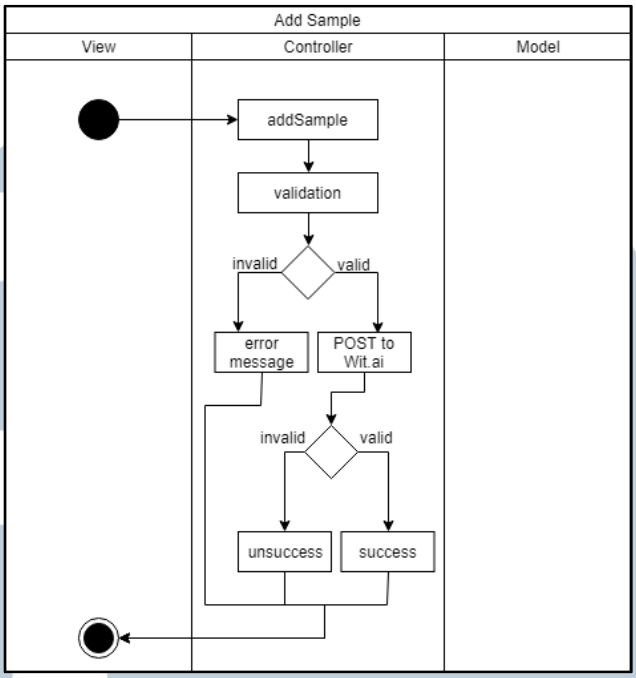


Gambar 3.14 *Activity Diagram* Delete Log

Gambar 3.14 merupakan *activity diagram* untuk menghapus log. Masukan untuk menghapus akan divalidasi dan jika sudah valid maka akan dicari log dengan model Log. Nama log akan dihapus dari *database* beserta *file* yang terdapat pada aplikasi.

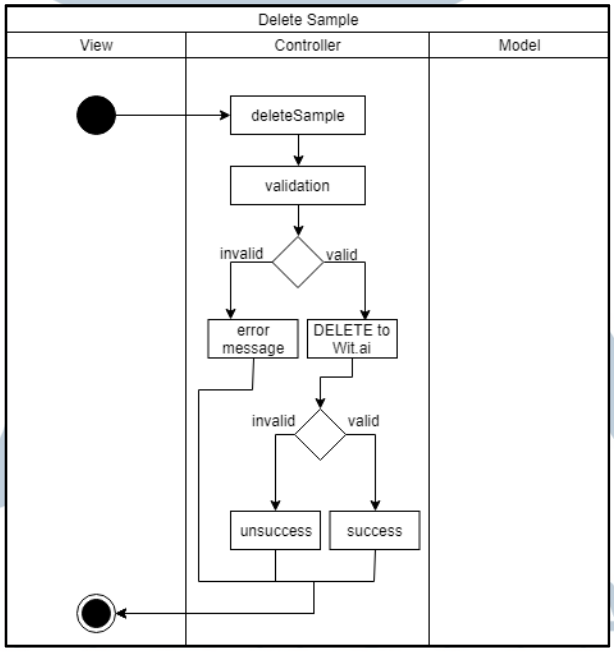
J. Activity Diagram Add Sample

Gambar 3.15 menggambarkan *activity diagram* untuk menambahkan *sample*. Masukan yang diterima akan divalidasi terlebih dahulu. Jika valid maka data-data akan diteruskan ke Wit.ai melalui HTTP *request*.



Gambar 3.15 *Activity Diagram* Add Sample

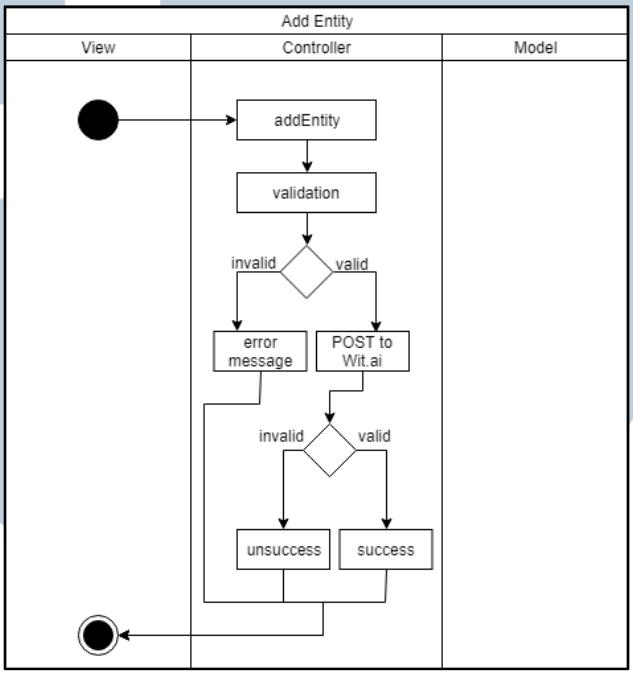
K. Activity Diagram Delete Sample



Gambar 3.16 *Activity Diagram* Delete Sample

Gambar 3.16 menunjukkan *activity diagram* dari menghapus *sample*. Ketika masukan untuk menghapus *sample* diterima maka akan dilakukan validasi. Masukan yang tervalidasi akan diarahkan ke Wit.ai melalui HTTP *request*.

L. Activity Diagram Add Entity

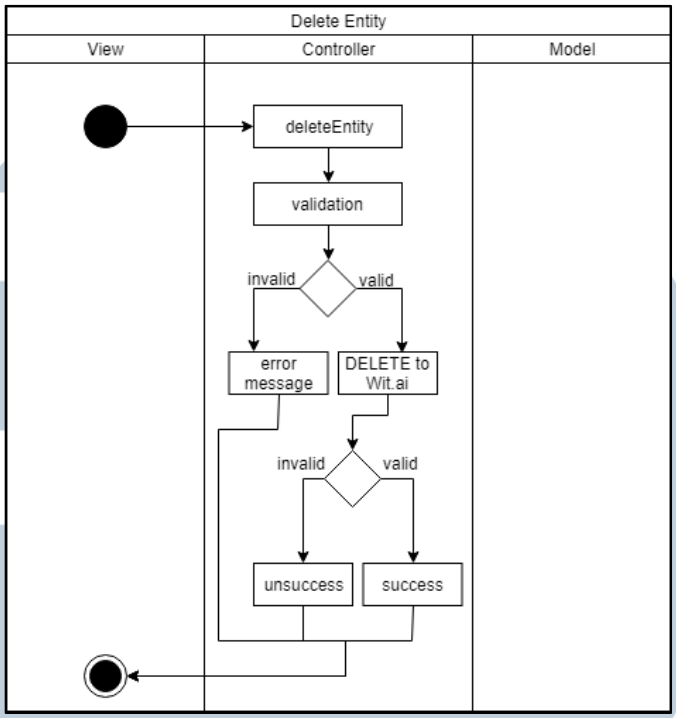


Gambar 3.17 *Activity Diagram* Add Entity

Gambar 3.17 merupakan *activity diagram* ketika menambahkan *entity*. Masukan akan divalidasi dan jika berhasil maka akan dikirim ke Wit.ai untuk *entity* baru.

M. Activity Diagram Delete Entity

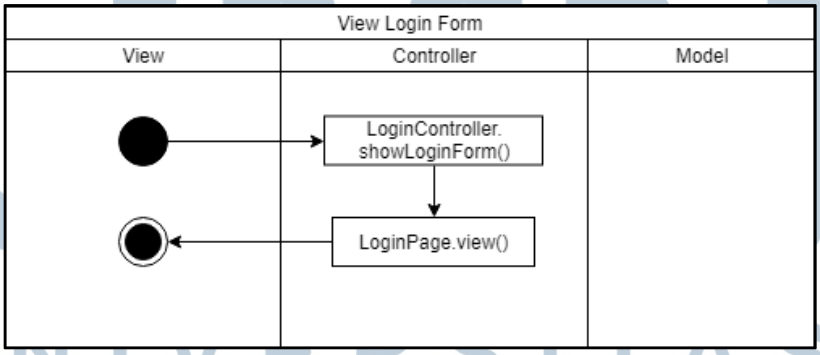
Gambar 3.18 menunjukkan gambar *activity diagram* untuk menghapus *entity*. Sebelum menghapus *entity* yang digunakan pada Wit.ai maka masukan akan divalidasi terlebih dahulu.



Gambar 3.18 *Activity Diagram* Delete Entity

N. Activity Diagram View Login Form

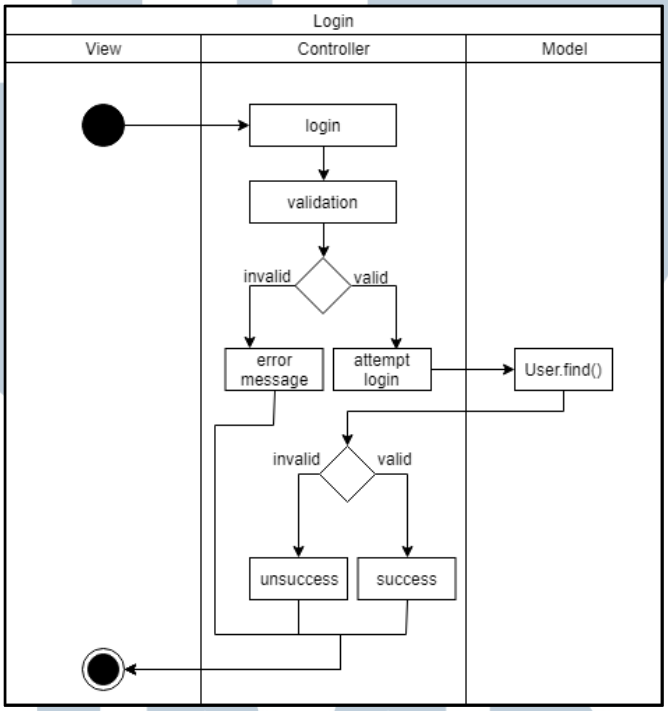
Gambar 3.19 menggambarkan *activity diagram* untuk melihat halaman *login*. Halaman *login* akan ditangani oleh LoginController dengan *method* showLoginForm.



Gambar 3.19 *Activity Diagram* View Login Form

O. Activity Diagram Login

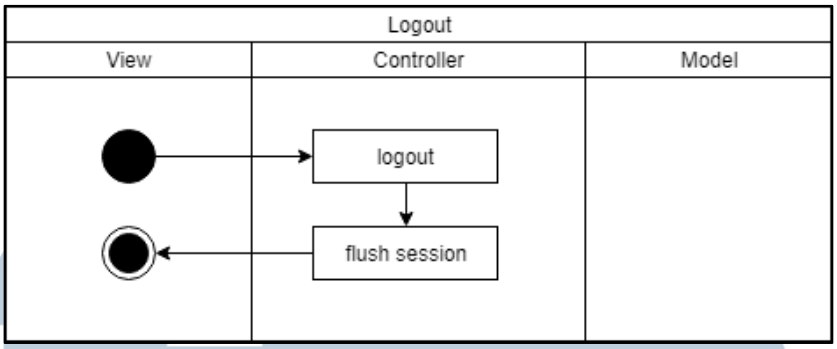
Gambar 3.20 merupakan *activity diagram* untuk *login*. Ketika pengguna melakukan *login* maka LoginController akan melakukan validasi terhadap masukan yang diterima. Jika valid maka akan dicoba untuk memeriksa *username* dan *password* yang telah diberikan. Jika berhasil ada yang cocok dengan yang ada di *database* maka pengguna akan masuk sebagai admin.



Gambar 3.20 *Activity Diagram* Login

P. Activity Diagram Logout

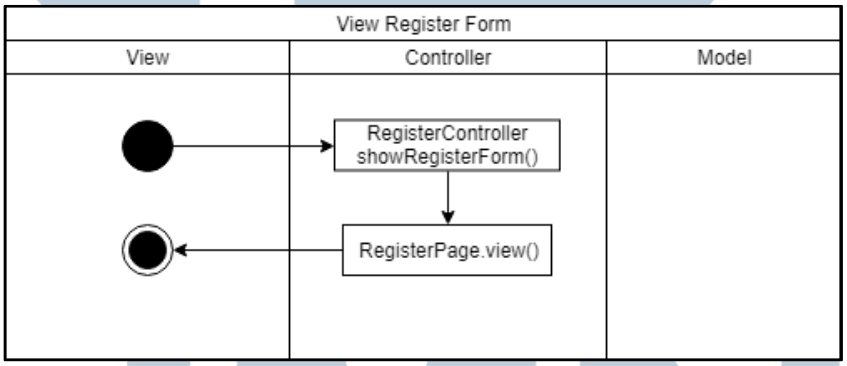
Gambar 3.21 merupakan *activity diagram* untuk *logout*. LoginController akan menangani *logout*. Ketika hal ini dilakukan maka seluruh *session* yang sedang ada akan dihapus.



Gambar 3.21 *Activity Diagram* Logout

Q. Activity Diagram View Register Form

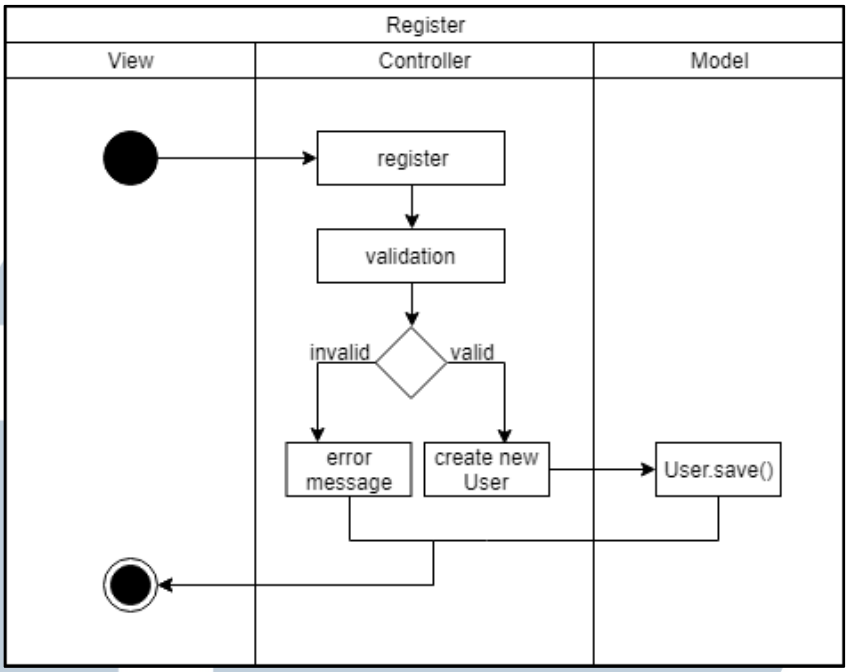
Gambar 3.22 menunjukkan *activity diagram* ketika melihat halaman *register*. RegisterController akan menangani hal ini dengan menampilkan RegisterPage kepada pengguna.



Gambar 3.22 *Activity Diagram* View Register Form

R. Activity Diagram Register

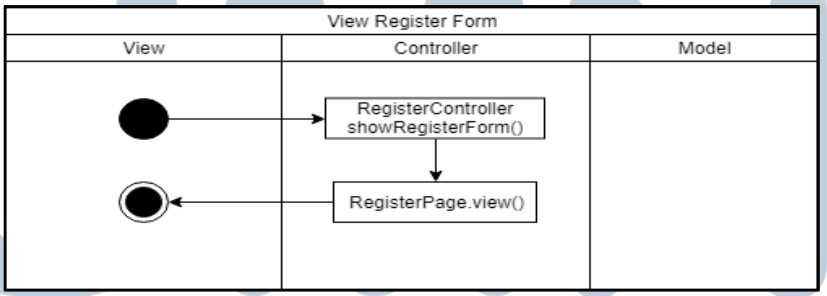
Gambar 3.23 merupakan *activity diagram* ketika *register* dilakukan. Masukan dari halaman *register* yang diterima RegisterController akan divalidasi terlebih dahulu. Jika valid maka akan dibuatkan *user* yang mana admin baru. *User* tersebut kemudian akan disimpan dalam *database*. Jika tidak valid maka akan ada pesan eror yang dikembalikan.



Gambar 3.23 *Activity Diagram* Register

S. Activity Diagram View Reset Password Form

Gambar 3.24 adalah *activity diagram* ketika melihat halaman *reset password*. Halaman ini akan ditangani oleh RegisterController pada *method* showRegisterForm.

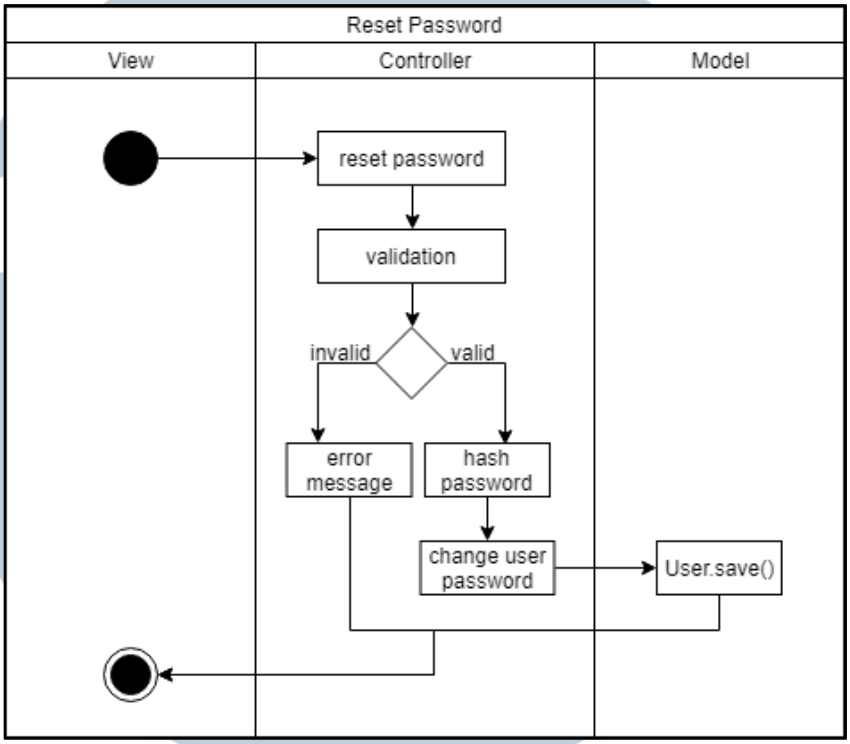


Gambar 3.24 *Activity Diagram* View Reset Password Form

T. Activity Diagram Reset Password

Gambar 3.25 menggambarkan *activity diagram* ketika melakukan *reset password*. Masukan yang diberikan dari form *reset password* akan divalidasi. Jika

tervalidasi dengan sukses maka *password* admin dengan *username* yang dipilih akan diubah.



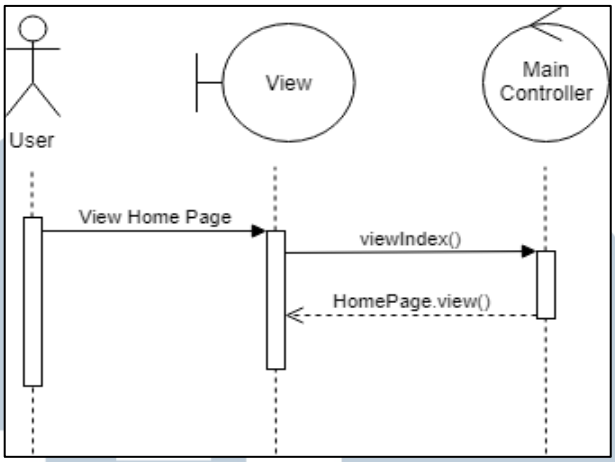
Gambar 3.25 *Activity Diagram* Reset Password

3.3.3 Sequence Diagram

Sequence diagram pada Jacob dibagi berdasarkan *use case diagram* yang ada sehingga terbagi sebagai berikut.

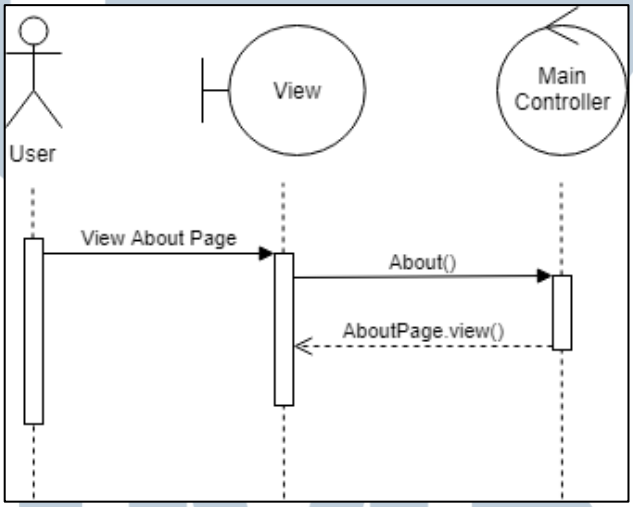
A. Sequence Diagram View Home Page

Gambar 3.26 menunjukan *sequence diagram* ketika pengguna mengakses halaman Home Page. *Method* viewIndex() pada MainController akan mengembalikan view dari Home Page.



Gambar 3.26 *Sequence Diagram* View Home Page

B. Sequence Diagram View About Page



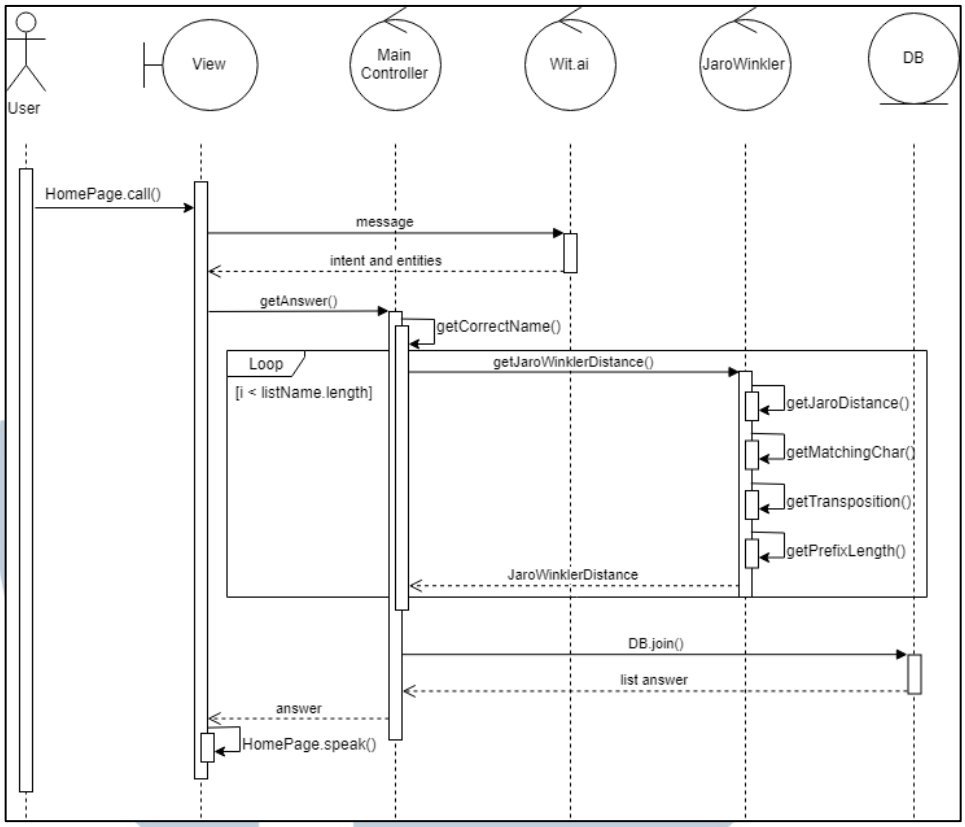
Gambar 3.27 *Sequence Diagram* View About Page

Gambar 3.27 menggambarkan *sequence diagram* ketika pengguna akan mengakses halaman *about*. *Method* about() pada MainController akan dijalankan dan mengembalikan *view* dari AboutPage.

C. Sequence Diagram Call

Gambar 3.28 menggambarkan *sequence diagram* ketika pengguna berinteraksi dengan Jacob. Teks ucapan yang dihasilkan dikirimkan ke Wit.ai untuk

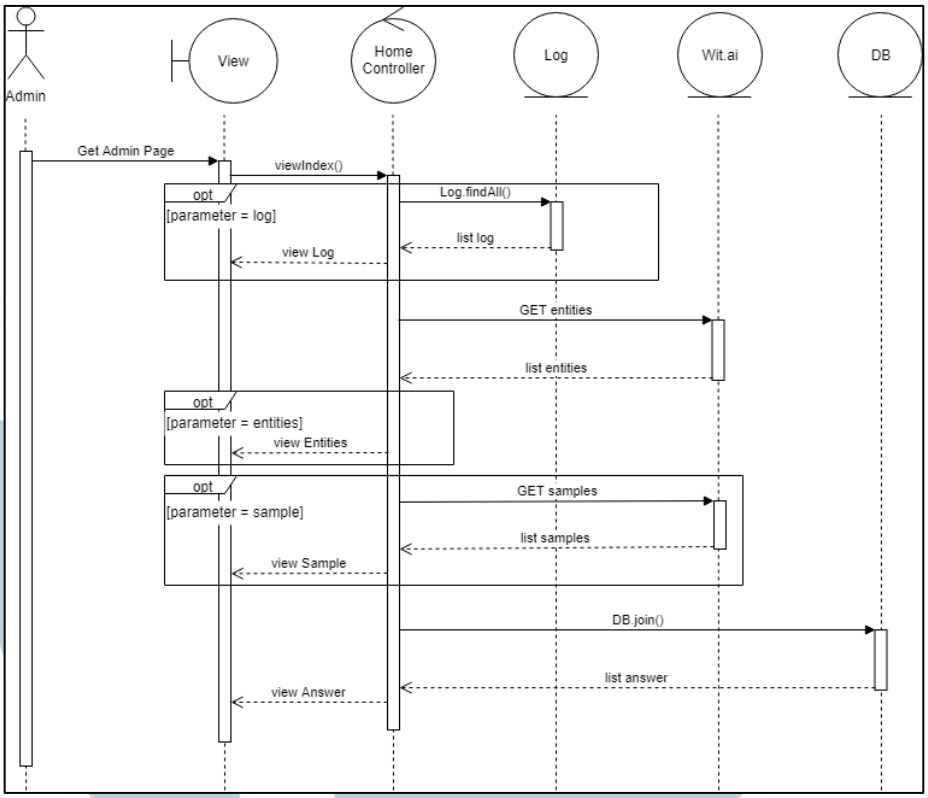
diproses dan didapatkan hasil berupa *intent* dan *entities*. Lalu *intent* dan *entities* dikirimkan ke *method* getAnswer() pada MainController. Jika ada nama yang diterima maka akan dijalankan *method* getCorrectName(). *Method* ini akan memanggil *method* getJaroWinklerDistance() dari *object* JaroWinkler. *Object* JaroWinkler akan memanggil *method*-nya sendiri untuk melakukan proses perhitungan nilai Jaro-Winkler *distance*. *Method* yang digunakan pertama adalah getJaroDistance(), yang kemudian dari *method* ini memanggil *method* getMatchingChar() untuk memperoleh karakter yang sama. Setelah itu *method* getTransposition() dipanggil untuk mendapatkan nilai transposisi dari karakter yang sama. Untuk menghitung nilai akhir dari Jaro-Winkler *distance* maka diperlukan panjang awalan nama yang sama sehingga *method* getPrefixLength() dipanggil. Dengan menggunakan rumus Jaro-Winkler maka nilai *distance*-nya dikembalikan ke MainController untuk dibandingkan dengan hasil nama yang lain. Untuk mengambil jawaban yang diperlukan, MainController melakukan *join* tabel dari *database*. Jawaban yang didapat diseleksi terlebih dahulu lalu dikembalikan ke *view* agar kalimatnya dapat digunakan untuk *speak*.



Gambar 3.28 *Sequence Diagram* Call

D. Sequence Diagram View Admin Page

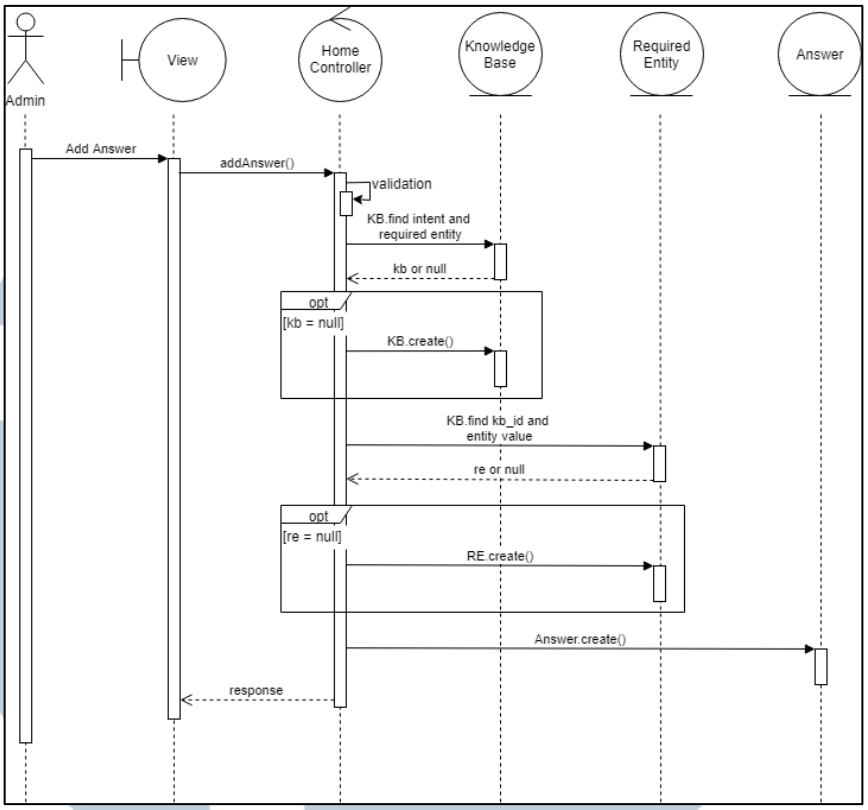
Gambar 3.29 merupakan *sequence diagram* dari *use case* View Admin Page. Tampilan yang dikembalikan oleh HomeController bergantung pada parameter yang diberikan. Pilihan dari parameter terdiri dari *entity*, *sample*, *answer*, dan log. *Entity*, *sample*, dan *answer* akan mengambil data dari Wit.ai dengan HTTP *request*. Selain itu *answer* juga akan mengambil daftar jawaban dari *database*. Parameter log akan mengambil data dari model Log.



Gambar 3.29 *Sequence Diagram* View Admin Page

E. Sequence Diagram Add Answer

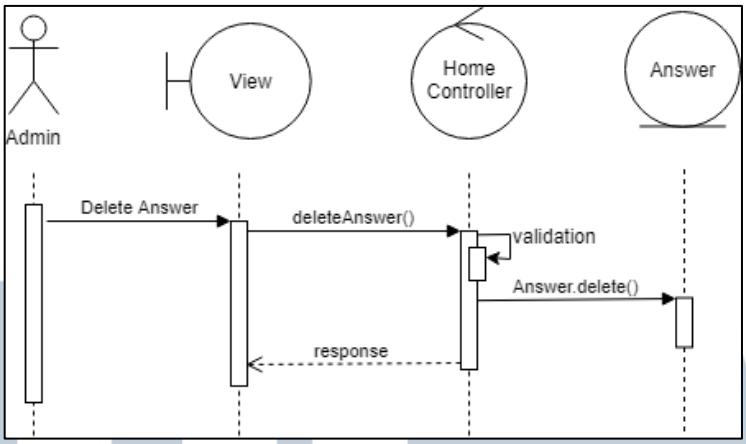
Gambar 3.30 menunjukkan *sequence diagram* dari *use case* Add Answer. Admin akan menambahkan jawaban dari *view* yang kemudian dikirim ke *methode* addAnswer() pada HomeController. Masukan akan divalidasi terlebih dahulu lalu kemudian akan dimasukkan satu per satu ke tabel knowledge_bases, required_entities, dan *answers*. *Id* dari knowledge_bases akan digunakan pada tabel *required_entities* dan *id* dari *required_entities* akan digunakan pada tabel *answers*. Kemudian jawaban baru akan disimpan ke dalam tabel *answers*.



Gambar 3.30 *Sequence Diagram* Add Answer

F. Sequence Diagram Delete Answer

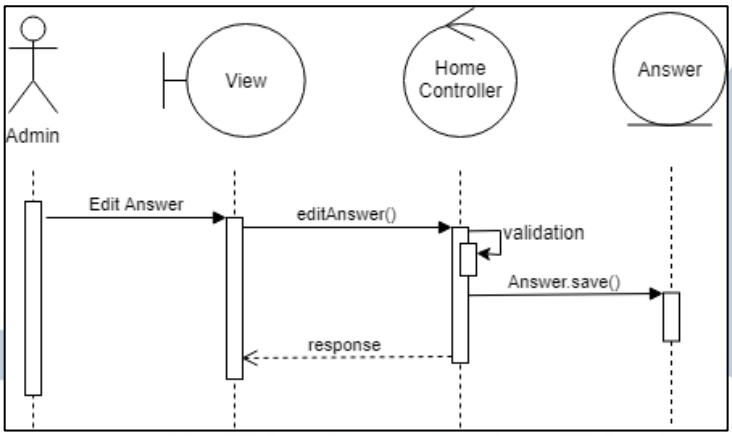
Gambar 3.31 merupakan *sequence diagram* untuk *use case* Delete Answer. Masukan dari admin akan diterima oleh *method* deleteAnswer() pada HomeController. Masukan divalidasi terlebih dahulu kemudian dicari jawabannya dengan model Answer dan dihapus. Lalu respon dikembalikan ke *view* untuk dilihat oleh admin.



Gambar 3.31 *Sequence Diagram* Delete Answer

G. Sequence Diagram Edit Answer

Gambar 3.32 merupakan *sequence diagram* dari *use case* Edit Answer. Masukan dari admin akan diterima oleh *method* editAnswer() pada HomeController. Masukan divalidasi terlebih dahulu kemudian dicari jawabannya dengan model Answer, diubah dan disimpan kembali ke *database*. Kemudian respon dikembalikan ke *view* untuk dilihat admin.

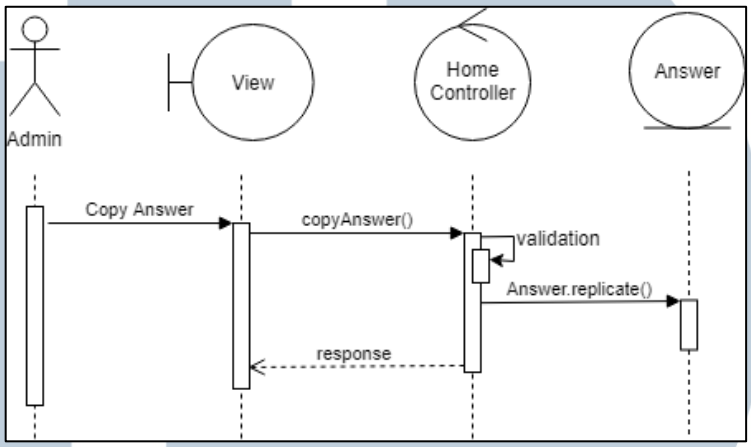


Gambar 3.32 *Sequence Diagram* Edit Answer

H. Sequence Diagram Copy Answer

Gambar 3.33 merupakan *sequence diagram* dari use case Copy Answer. Masukan dari admin akan diterima oleh *method* copyAnswer() pada

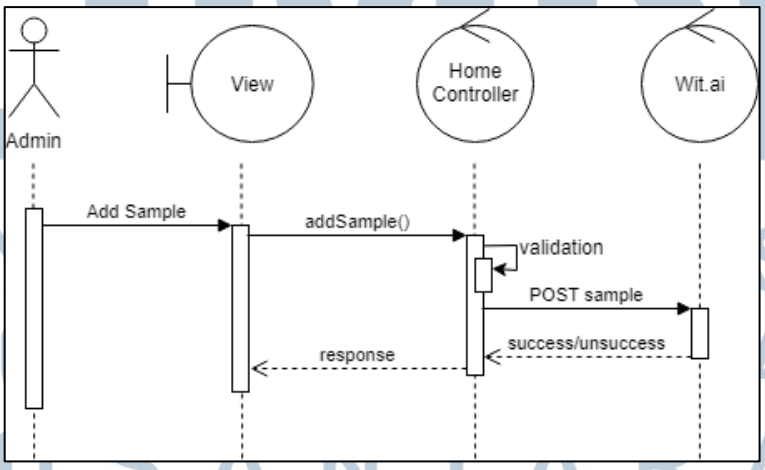
HomeController. Masukan divalidasi terlebih dahulu kemudian dicari jawabannya dengan model Answer, dan digandakan ke *database*. Kemudian respon dikembalikan ke *view* untuk dilihat admin.



Gambar 3.33 *Sequence Diagram* Copy Answer

I. Sequence Diagram Add Sample

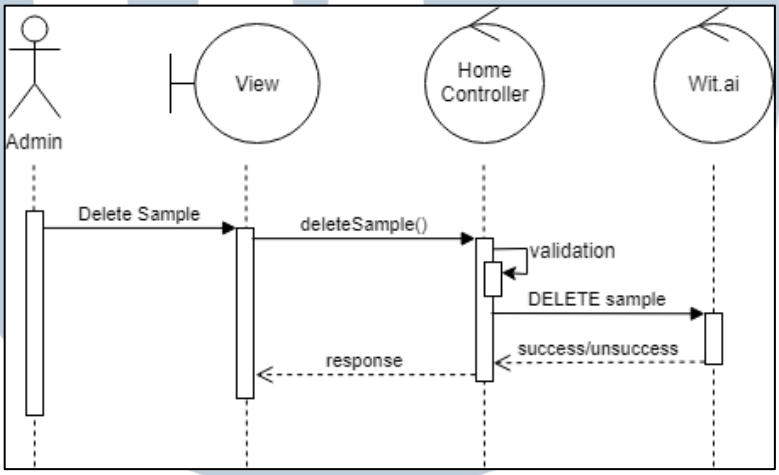
Gambar 3.34 merupakan *sequence diagram* dari use case Add Sample. Masukan dari admin akan diterima oleh *method* addSample() pada HomeController. Masukan divalidasi terlebih dahulu kemudian dilakukan HTTP *request* dengan metode POST ke Wit.ai. Kemudian respon dikembalikan ke *view* untuk dilihat admin.



Gambar 3.34 *Sequence Diagram* Add Sample

J. Sequence Diagram Delete Sample

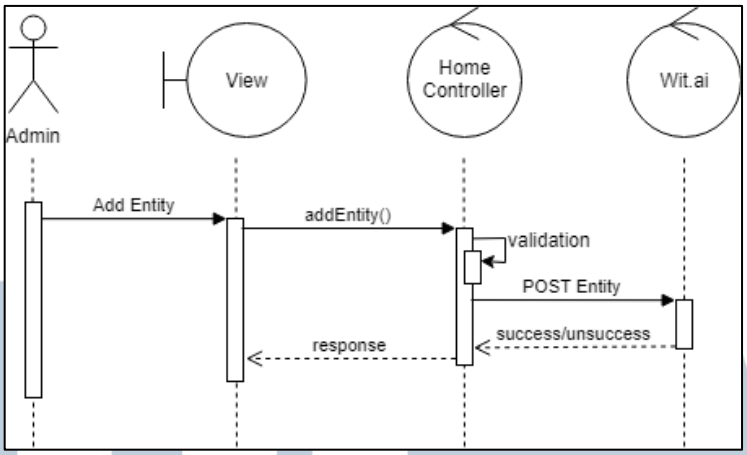
Gambar 3.35 merupakan *sequence diagram* dari use case Delete Sample. Masukan dari admin akan diterima oleh *method* deleteSample() pada HomeController. Masukan divalidasi terlebih dahulu kemudian dilakukan HTTP *request* dengan metode DELETE ke Wit.ai. Kemudian respon dikembalikan ke *view* untuk dilihat admin.



Gambar 3.35 *Sequence Diagram* Delete Sample

K. Sequence Diagram Add Entity

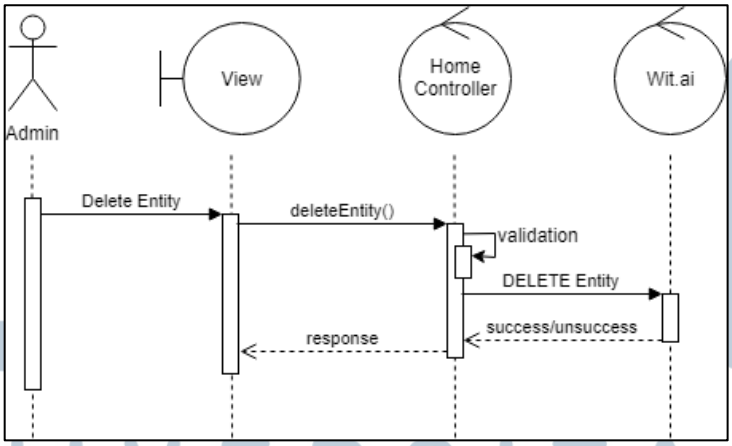
Gambar 3.36 merupakan *sequence diagram* dari use case Add Entity. Masukan dari admin akan diterima oleh *method* addEntity() pada HomeController. Masukan divalidasi terlebih dahulu kemudian dilakukan HTTP *request* dengan metode POST ke Wit.ai. Kemudian respon dikembalikan ke *view* untuk dilihat admin.



Gambar 3.36 *Sequence Diagram* Add Entity

L. Sequence Diagram Delete Entity

Gambar 3.37 merupakan *sequence diagram* dari use case Delete Entity. Masukan dari admin akan diterima oleh method deleteEntity() pada HomeController. Masukan divalidasi terlebih dahulu kemudian dilakukan HTTP request dengan metode DELETE ke Wit.ai. Kemudian respon dikembalikan ke *view* untuk dilihat admin.

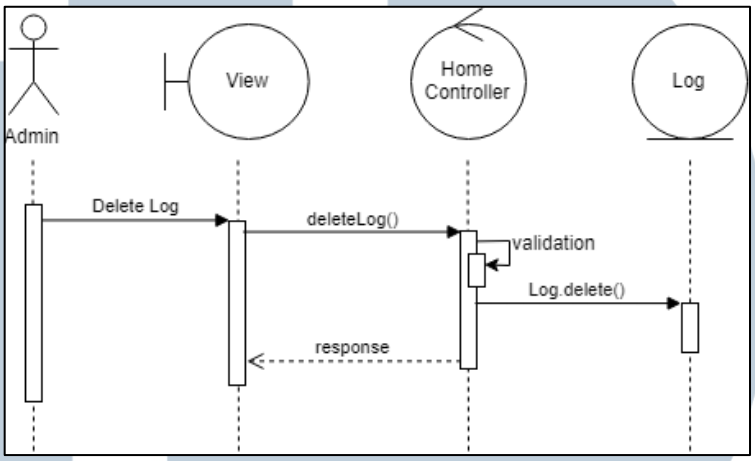


Gambar 3.37 *Sequence Diagram* Delete Entity

M. Sequence Diagram Delete Log

Gambar 3.38 merupakan *sequence diagram* dari use case Delete Log. Masukan dari admin akan diterima oleh *method* deleteLog() pada HomeController.

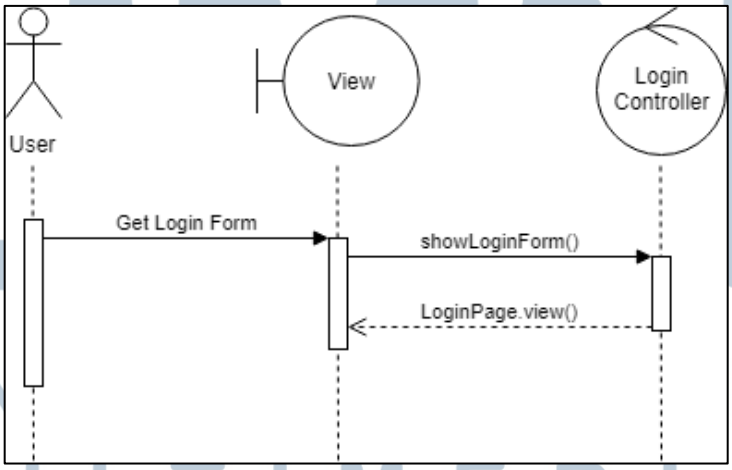
Masukan divalidasi terlebih dahulu kemudian menggunakan model Log untuk menghapus log tersebut. Penghapusan juga dilakukan terhadap *file* yang ada pada aplikasi. Kemudian respon dikembalikan ke *view* untuk dilihat admin.



Gambar 3.38 *Sequence Diagram* Delete Log

N. Sequence Diagram View Login Form

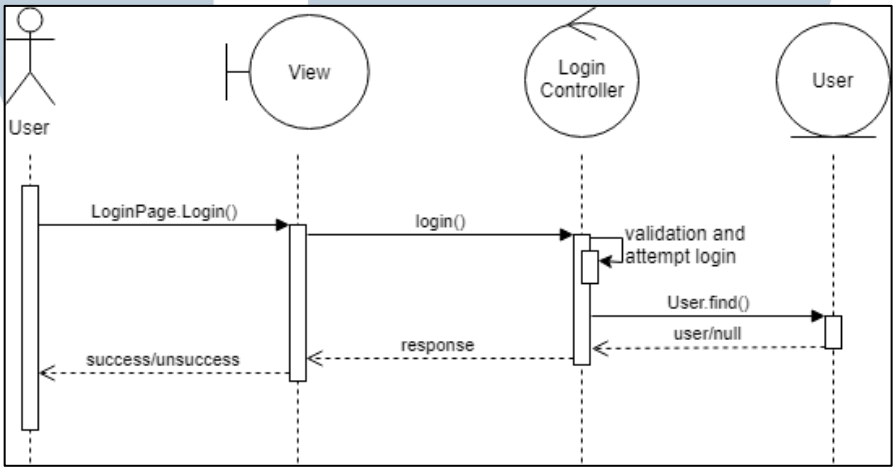
Gambar 3.39 merupakan *sequence diagram* untuk *use case* View Login Form. *Method* showLoginForm() pada LoginController akan dijalankan dan mengembalikkan *view* dari LoginPage.



Gambar 3.39 *Sequence Diagram* View Login Form

O. Sequence Diagram Login

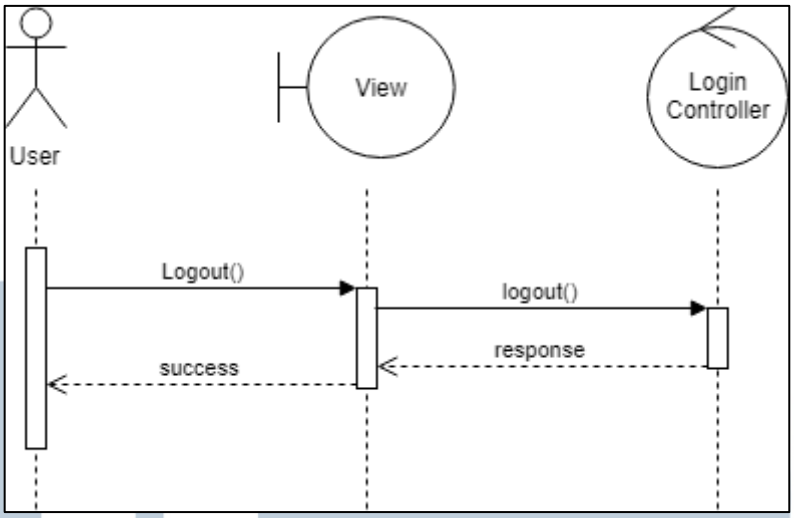
Gambar 3.40 adalah *sequence diagram* dari *use case* Login. Masukan untuk pemeriksaan *login* dikirimkan ke *method* login() pada LoginController. Setelah itu akan dilakukan validasi dan pencobaan *login*. Ketika mencoba *login* maka akan dicari *username* dan *password* yang sesuai dengan model User. Hasilnya dapat berupa *user* dan *null*. Jika *user* maka *login* sukses dan jika *null* maka *login* tidak sukses. Respon kemudian diberikan kembali ke *view* untuk ditampilkan ke pengguna.



Gambar 3.40 *Sequence Diagram* Login

P. Sequence Diagram Logout

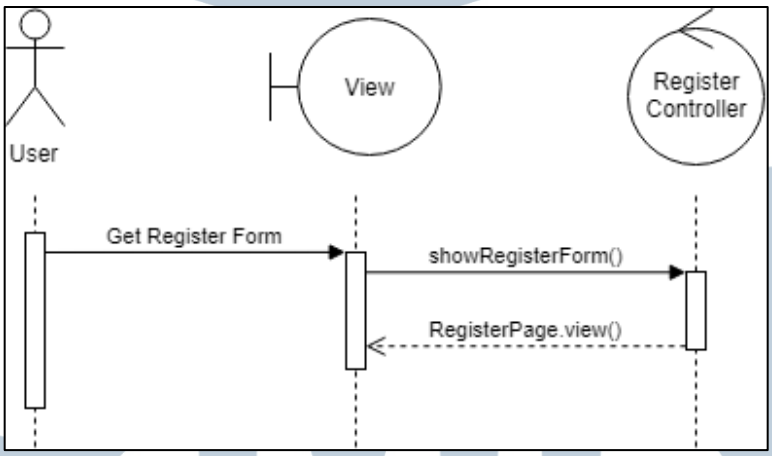
Gambar 3.41 adalah *sequence diagram* dari *use case* Logout. Permintaan akan diberikan ke *method* logout() pada LogoutController. Setelah itu respon akan dikirimkan ke *view* untuk ditampilkan kepada pengguna.



Gambar 3.41 *Sequence Diagram* Logout

Q. Sequence Diagram View Register Form

Gambar 3.42 merupakan *activity diagram* untuk *use case* View Register Form. *Method* showRegisterForm() pada LoginController akan dijalankan dan mengembalikkan *view* dari RegisterPage.

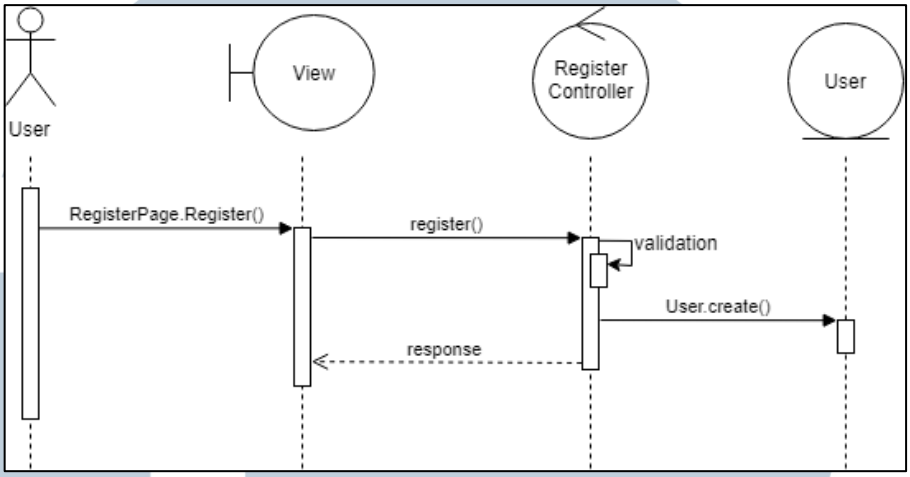


Gambar 3.42 *Sequence Diagram* View Register Form

R. Sequence Diagram Register

Gambar 3.43 adalah *sequence diagram* dari *use case* Register. Masukan untuk pemeriksaan *register* dikirimkan ke *method* register() pada RegisterController. Setelah itu akan dilakukan validasi dan setelah itu akan disimpan ke dalam *database*

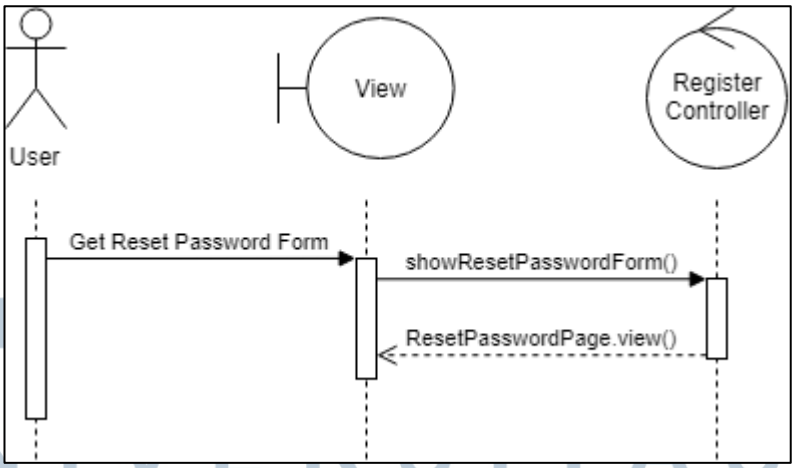
melalui model User. Respon kemudian diberikan kembali ke *view* untuk ditampilkan ke pengguna.



Gambar 3.43 *Sequence Diagram* Register

S. Sequence Diagram View Reset Form

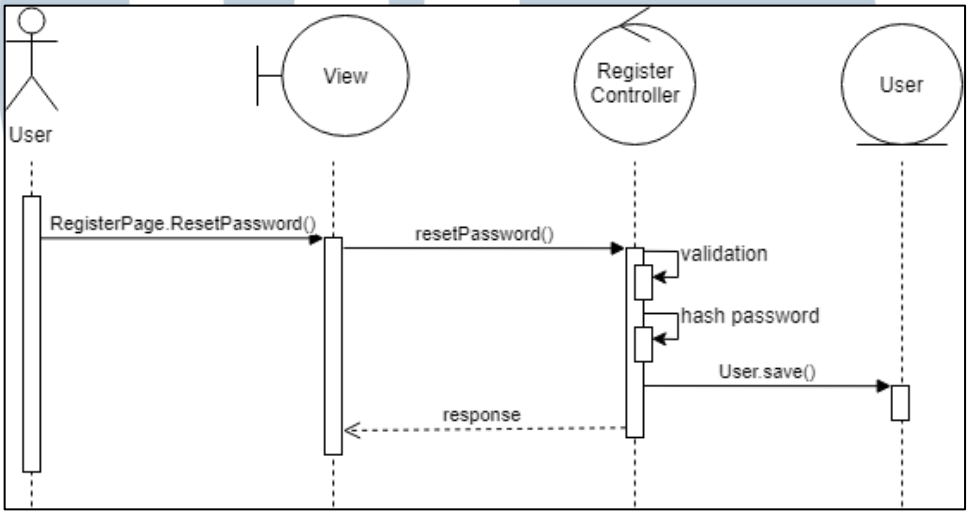
Gambar 3.44 merupakan *activity diagram* untuk *use case* View Reset Password Form. *Method* showResetPasswordForm() pada LoginController akan dijalankan dan mengembalikkan *view* dari ResetPasswordPage.



Gambar 3.44 *Sequence Diagram* View Reset Form

T. Sequence Diagram Reset Password

Gambar 3.45 adalah *sequence diagram* dari *use case* Reset Password. Masukan untuk pemeriksaan *reset password* dikirimkan ke *method* resetPassword() pada RegisterController. Setelah itu akan dilakukan validasi dan setelah itu akan disimpan ke dalam *database* melalui model User. Respon kemudian diberikan kembali ke *view* untuk ditampilkan ke pengguna.

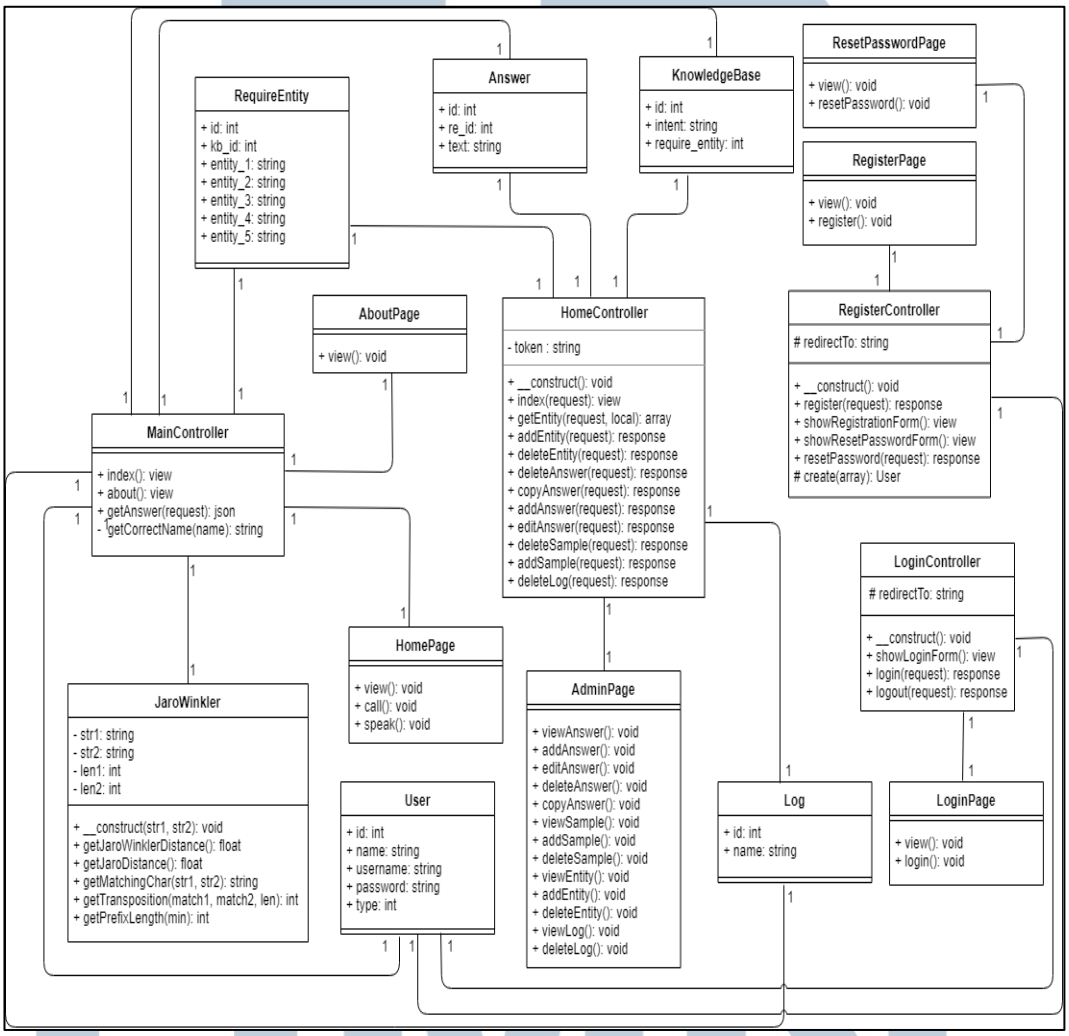


Gambar 3.45 *Sequence Diagram* Reset Password

3.3.4 Class Diagram

Gambar 3.46 menunjukan *class diagram* dari aplikasi *voice chatbot* Jacob. *Class diagram* dibentuk dari rancangan *use case diagram*, *activity diagram*, dan *sequence diagram* yang telah dirancang dan berdasarkan model pengembangan web yaitu *model*, *view*, dan *controller*. Permintaan dari pengguna atau *user* akan ditangani oleh *controller class* yang sesuai, dicari datanya dari *model class* dan kemudian dihasilkan tampilan / *view* atau respon lain seperti *Java Script Object Notation* (JSON). Main Controller digunakan untuk menangani *request-request* yang berhubungan dengan pengguna biasa seperti *home page*, *about page*, dan *call* kepada Jacob. Home Controller digunakan untuk menangani *request-request* yang

berhubungan dengan keperluan admin dan super admin seperti *add sample*, *delete sample*, *add entity*, *delete entity*, *add answer*, *delete answer*, *edit answer*, *copy answer*, dan *delete log*.



Gambar 3.46 *Class Diagram* Aplikasi Jacob

3.3.5 Struktur Tabel

Berikut adalah struktur tabel yang digunakan dalam pembangunan aplikasi *voice chatbot* Jacob. Tabel terdiri dari tabel admins, users, knowledge_bases, required_entities, answers, dan logs.

A. Tabel admins

Tabel 3.1 Struktur tabel admins

Nama Kolom	Tipe Data	Nullable	Identitas
id	tinyint(3)		PK, unsigned, auto increment
name	varchar(15)		
desc	varchar(20)		

Tabel 3.1 merupakan tabel admins yang digunakan untuk menampung tipe atau tingkatan dari admin yang dimiliki. Tabel ini terdiri dari *id*, *name*, dan *desc*. Kolom *id* merupakan *primary key*. Kolom *name* merupakan nama dari tipe admin. Kolom *desc* merupakan keterangan dari tipe admin tersebut.

B. Tabel users

Tabel 3.2 Struktur tabel users

Nama Kolom	Tipe Data	Nullable	Identitas
id	int(10)		PK, unsigned, auto increment
name	varchar(50)		
username	varchar(50)		unique
password	varchar(191)		
type	tinyint(3)		FK, unsigned
created_at	timestamp	ya	
updated_at	timestamp	ya	

Tabel 3.2 menunjukan struktur tabel *users*. Tabel ini digunakan untuk menyimpan data-data *user* admin yang dapat melakukan *login* ke dalam aplikasi. Kolom *id* merupakan *primary key* dari tabel dengan *username* yang memiliki atribut unik. Kolom *type* merupakan *foreign key* yang menunjuk kepada tabel admins.

C. Tabel knowledge_bases

Tabel 3.3 Struktur tabel knowledge_bases

Nama Kolom	Tipe Data	Nullable	Identitas
id	int(10)		PK, unsigned, auto increment
intent	varchar(50)	ya	
require_entity	tinyint(3)		unsigned
created_at	timestamp	ya	
updated_at	timestamp	ya	

Tabel 3.3 merupakan struktur tabel dari knowledge_bases. Tabel ini daftar dari *intent* yang digunakan sebagai dasar untuk mendapatkan suatu jawaban tertentu. Kolom *require_entity* merupakan jumlah dari *entity* yang diperlukan untuk membantu menentukan jawaban.

D. Tabel required_entities

Tabel 3.4 adalah struktur tabel dari required_entities. Tabel ini digunakan untuk menyimpan daftar *entity* yang diperlukan dari tabel knowledge_bases. Tabel ini memiliki kolom kb_id yang merupakan *foreign key* yang menunjuk kepada tabel knowledge_bases. Kolom entity_1 hingga entity_5 digunakan untuk menyimpan pasangan *entity* dan nilai yang diperlukan. Tidak semua kolom entity digunakan karena disesuaikan lagi ke jawaban yang akan dihasilkan.

Tabel 3.4 Struktur tabel required_entities

Nama Kolom	Tipe Data	Nullable	Identitas
id	int(10)		PK, unsigned, auto increment
kb_id	int(10)		FK, unsigned
entity_1	varchar(50)	ya	
entity_2	varchar(50)	ya	
entity_3	varchar(50)	ya	
entity_4	varchar(50)	ya	
entity_5	varchar(50)	ya	
created_at	timestamp	ya	
updated_at	timestamp	ya	

E. Tabel answers

Tabel 3.5 adalah struktur tabel answers. Tabel ini memiliki kolom re_id yang merupakan *foreign key* menunjuk ke tabel required_entities. Jawaban yang disimpan pada tabel ini merupakan kalimat jawaban yang telah diisi oleh admin.

Tabel 3.5 Struktur tabel answers

Nama Kolom	Tipe Data	Nullable	Identitas
id	int(10)		PK, unsigned, auto increment
re_id	int(10)		FK, unsigned
text	text		
created_at	timestamp	ya	
updated_at	timestamp	ya	

F. Tabel logs

Tabel 3.6 merupakan struktur tabel logs. Tabel ini digunakan untuk mencatat nama *file* log yang telah dibuat. File log dibuat setiap sesi percakapan *user* dengan *voice chatbot* Jacob.

Tabel 3.6 Struktur tabel logs

Nama Kolom	Tipe Data	Nullable	Identitas
id	int(10)		PK, unsigned, auto increment
name	varchar(30)		
created_at	timestamp	ya	
updated_at	timestamp	ya	

3.3.6 Perancangan Antarmuka Pengguna

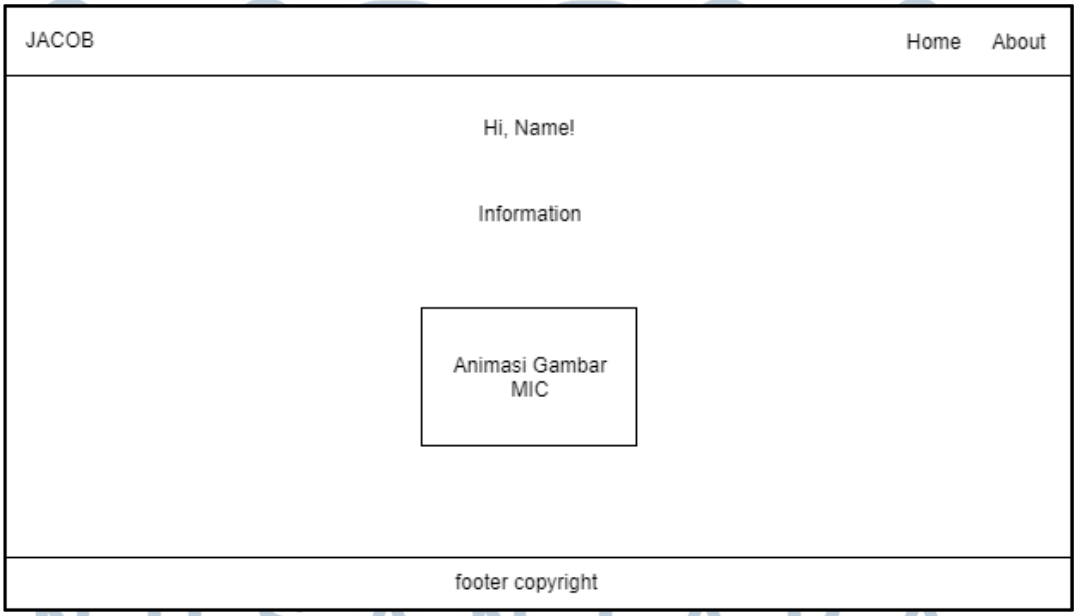
Perancangan antarmuka pengguna untuk aplikasi *voice chatbot* Jacob berbasis web terbagi menjadi antarmuka Home Page, antarmuka Voice Chatbot Jacob, antarmuka About Page, antarmuka Admin Page, antarmuka Login Form, dan antarmuka Register Form.

A. Antarmuka Home Page

Gambar 3.47 Rancangan Antarmuka Home Page

Gambar 3.47 merupakan rancangan antarmuka Home Page. Halaman inilah yang pertama akan ditampilkan ketika mengakses aplikasi Jacob. Terdapat navigasi untuk Home dan About. Home untuk mengalihkan pengguna ke halaman Home Page tersebut sedangkan About untuk mengalihkan ke halaman About Page.

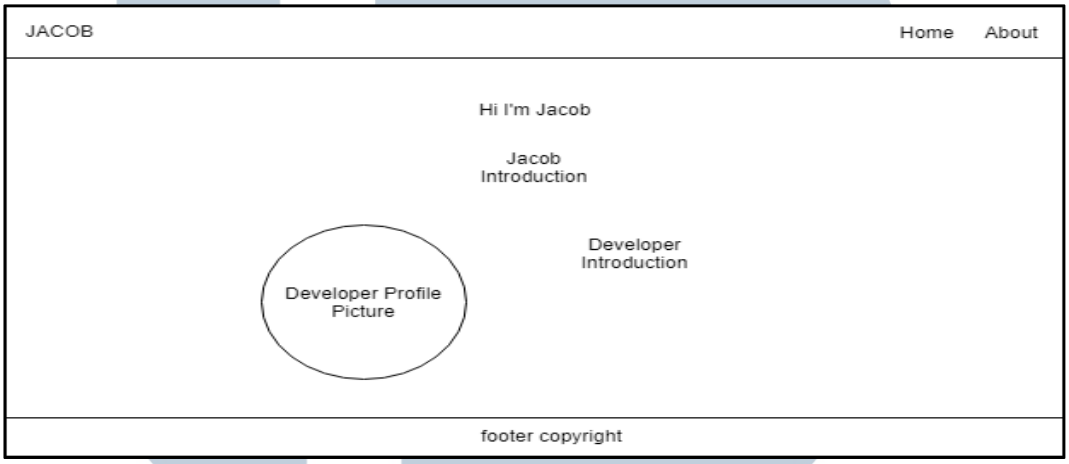
B. Antarmuka Voice Chatbot Jacob



Gambar 3.48 Rancangan Antarmuka Voice Chatbot Jacob

Gambar 3.48 merupakan rancangan antarmuka Voice Chatbot Jacob. Halaman ini muncul pada halaman Home Page ketika tombol Ask Me Now ditekan. Pada halaman ini terdapat tombol *mic* yang dapat ditekan untuk memulai berbicara.

C. Antarmuka About Page



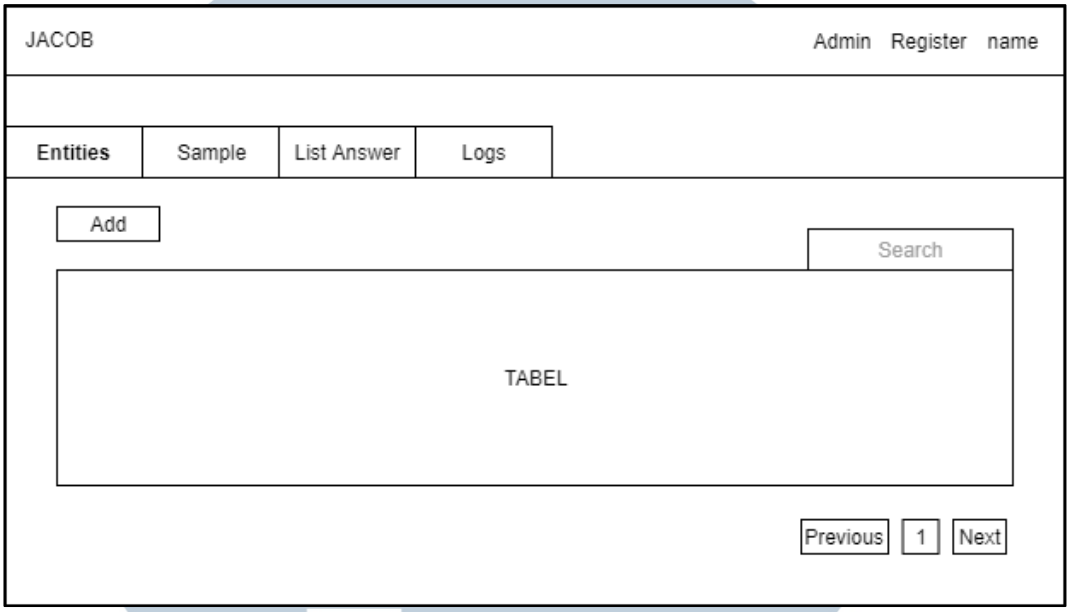
Gambar 3.49 Rancangan Antarmuka About Page

Gambar 3.49 menggambarkan rancangan antarmuka About Page. Halaman ini akan menampilkan penjelasan tentang aplikasi *voice chatbot* Jacob dengan tim pengembangnya.

D. Antarmuka Admin Page

Gambar 3.50 menggambarkan rancangan antarmuka untuk halaman admin. Halaman ini terdapat navigasi untuk admin melakukan pengubahan data yang sesuai dengan *tab* yang telah disediakan. Setiap tabel pada *tab* akan memiliki sebuah tombol *action* yang dapat digunakan untuk memudahkan melihat data atau mengubah data. Kegiatan tersebut dilakukan melalui form pada *modal* yang muncul. Perancangan untuk *tab* Sample dan *tab* List Answer sama seperti *tab* Entites, yaitu terdapat tabel data, tombol *new*, dan kolom *search*. *Tab* Log hanya memiliki tabel dan kolom *search.* Pada bagian navigasi, tombol Register akan

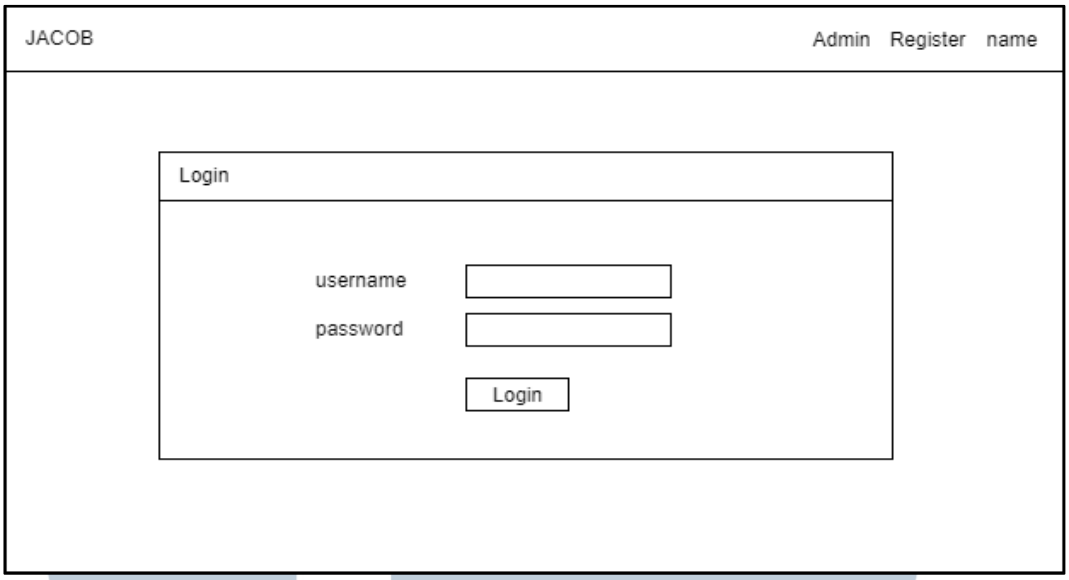
muncul jika yang *login* adalah superadmin. Tombol Jacob akan mengalihkan *admin* ke halaman Home Page.



Gambar 3.50 Rancangan Antarmuka Admin Page

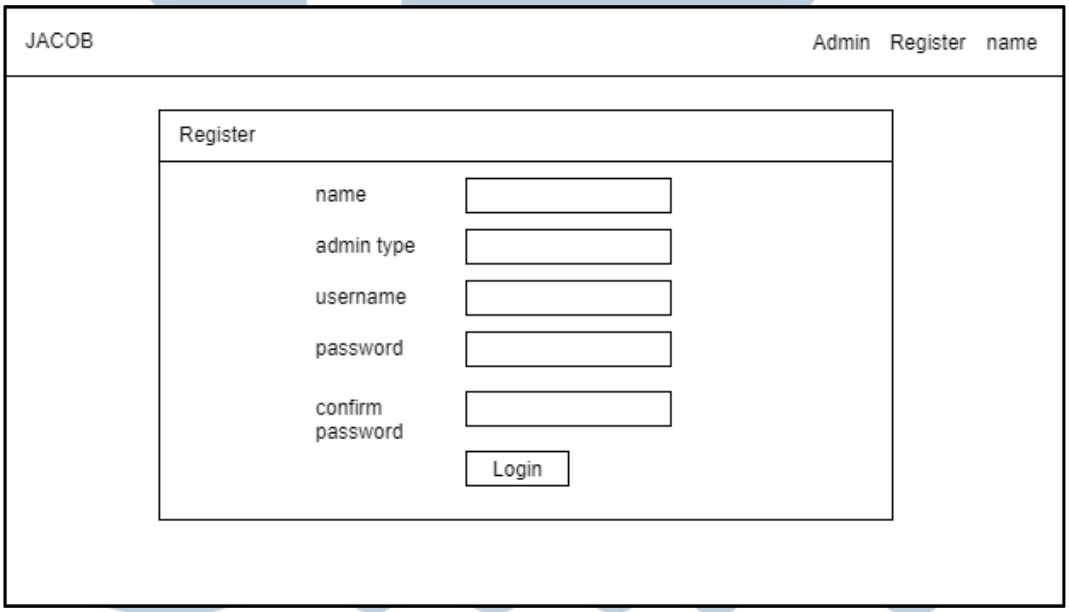
E. Antarmuka Login Form

Gambar 3.51 merupakan rancangan antarmuka untuk Login. Terdapat *field username* dan *password* yang dapat diisi admin untuk melakukan proses autentikasi. Halaman ini diarahkan dari Home Page ketika admin mengatakan bahwa ingin *login*.



Gambar 3.51 Rancangan Antarmuka Login Form

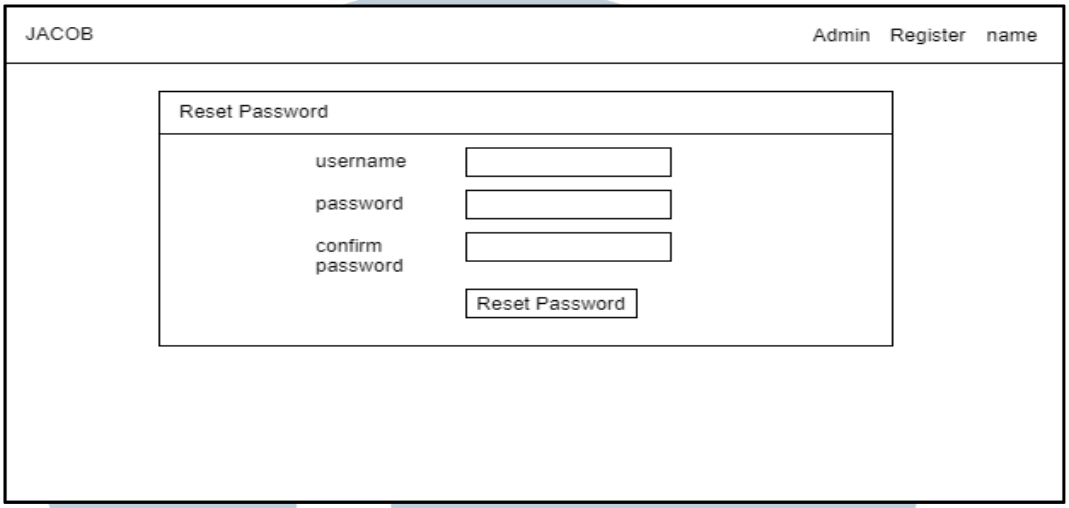
F. Antarmuka Register Form



Gambar 3.52 Rancangan Antarmuka Register Form

Gambar 3.52 merupakan rancangan antarmuka untuk Register. Pada halaman ini terdapat *field name*, *admin type*, *username*, *password*, dan *confirm password* yang dapat diisi untuk mendaftarkan admin baru. *Admin type* yang tersedia adalah admin dan superadmin.

G. Antarmuka Reset Password Page



Gambar 3.53 Rancangan Antarmuka Reset Password Form

Gambar 3.53 merupakan rancangan antarmuka untuk Reset Password. Pada halaman ini terdapat *field username*, *password*, dan *confirm password*. *Username* terdiri dari akun admin yang sudah dibuat sebelumnya. Super admin dapat mengubah *password* dari admin lain yang sudah ada.