



**KERJA PRAKTIK – IF184801**

**RANCANG ANTARMUKA APLIKASI MOBILE GARDA PANGAN BERBASIS ANDROID**

GARDA PANGAN

Wisma Mukti, Jl. Klampis Anom III F-22, Sukolilo, Surabaya

Periode: 6 Januari - 24 Februari 2020

Oleh:

Yemima Sutanto 05111740000049

Elkana Hans Widersen 05111740000127

Pembimbing Departemen

Ahmad Saikhu, S.Si., M.T.

Pembimbing Lapangan

Eva Bachtiar

DEPARTEMEN TEKNIK INFORMATIKA

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya 2020

*[Halaman ini sengaja dikosongkan]*



**KERJA PRAKTIK – IF184801**

Oleh:

Yemima Sutanto 05111740000049

Elkana Hans Widersen 05111740000127

Pembimbing Departemen

Ahmad Saikhu, S.Si., M.T.

Pembimbing Lapangan

Eva Bachtiar

DEPARTEMEN TEKNIK INFORMATIKA

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya 2020

**RANCANG ANTARMUKA APLIKASI MOBILE GARDA PANGAN BERBASIS ANDROID**

GARDA PANGAN

Wisma Mukti, Jl. Klampis Anom III F-22, Sukolilo, Surabaya

Periode: 6 Januari - 24 Februari 2020

*[Halaman ini sengaja dikosongkan]*

# LEMBAR PENGESAHAN

**KERJA PRAKTIK**

**Rancang Antarmuka Aplikasi Mobile Garda Pangan Berbasis Android**

Oleh:

Yemima Sutanto Elkana Hans Widersen

05111740000049

05111740000127

Disetujui oleh Pembimbing Kerja Praktik:

1. Ahmad Saikhu, S.Si., M.T. NIP 197107182006041001

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

(Pembimbing Departemen)

2. Eva Bachtiar .........................

(Pembimbing Lapangan)

**SURABAYA MARET, 2020**

*[Halaman ini sengaja dikosongkan]*

**Rancang Antarmuka Aplikasi Mobile Garda Pangan Menggunakan Metode User Centered Design Berbasis Android**

Nama Mahasiswa : Yemima Sutanto

### NRP 05111740000049

### Nama Mahasiswa : Elkana Hans Widersen

### NRP 05111740000127

Departemen : Teknik Informatika-ITS Pembimbing Departemen : Ahmad Saikhu, S.Si., M.T. Pembimbing Lapangan : Eva Bachtiar

# ABSTRAK

*Garda Pangan merupakan sebuah food bank yang bertujuan menjadi pusat koordinasi makanan surplus dan berpotensi terbuang, untuk disalurkan kepada masyarakat pra-sejahtera.* *Salah satu produk yang sedang dikerjakan ketika kami melakukan Kerja Praktik adalah rancang antarmuka aplikasi Garda Pangan, yaitu aplikasi mobile berbasis android yang mempermudah proses pengambilan dan pendistribusian makanan. Pengguna utama aplikasi ini adalah Food Heroes atau relawan Garda Pangan yang melakukan pengambilan dan pendistribusian makanan dari Mitra Garda Pangan kepada masyarakat yang membutuhkan.*

*Sebelumnya, aplikasi ini pernah dibuat, tetapi tidak membuahkan hasil yang baik. Oleh karena itu kami diminta untuk membuat desain antarmuka aplikasi ini lebih lanjut dengan menambahkan beberapa pengembangan fitur yang dirancang ulang.*

***Kata Kunci : Food Heroes, Garda Pangan, Android***

*[Halaman ini sengaja dikosongkan]*

# KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa, atas penyertaan dan karunia-Nya sehingga penulis dapat menyelesaikan salah satu kewajiban penulis sebagai mahasiswa Departemen Informatika ITS yaitu Kerja Praktik yang berjudul: Perancangan Antarmuka Aplikasi Mobile Garda Pangan Berbasis Android.

Penulis menyadari bahwa masih banyak kekurangan baik dalam melaksanakan kerja praktik maupun penyusunan buku laporan kerja praktik ini. Namun penulis berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi.

Melalui buku laporan ini penulis juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu menyusun laporan kerja praktik baik secara langsung maupun tidak langsung, antara lain:

* 1. Kedua orang tua penulis.
  2. Bapak Ahmad Saikhu, S.Si., M.T., selaku dosen pembimbing kerja praktik selama kerja praktik berlangsung.
  3. Bapak Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D, selaku koordinator kerja praktik.
  4. Ibu Eva Bachtiar selaku pembimbing lapangan selama kerja praktik berlangsung.

Surabaya, 1 Maret 2020

Penulis

*[Halaman ini sengaja dikosongkan]*

# DAFTAR ISI

[LEMBAR PENGESAHAN v](#_TOC_250002)

[ABSTRAK vii](#_TOC_250001)

[KATA PENGANTAR x](#_TOC_250000)

[DAFTAR ISI](#_bookmark0) [1](#_bookmark0)

[DAFTAR GAMBAR](#_bookmark1) [8](#_bookmark1)

[DAFTAR TABE](#_bookmark2)L 11

[DAFTAR KODE SUMBER](#_bookmark1) [13](#_bookmark1)

[BAB I PENDAHULUAN](#_bookmark3) [15](#_bookmark3)

[Latar Belakang](#_bookmark4) [15](#_bookmark4)

[Tujuan](#_bookmark5) [15](#_bookmark5)

[Manfaat](#_bookmark6) [16](#_bookmark6)

[Rumusan Masalah](#_bookmark7) [16](#_bookmark7)

[Lokasi dan Waktu Kerja Praktik](#_bookmark8) [16](#_bookmark8)

[Metodologi Kerja Praktik](#_bookmark9) [17](#_bookmark9)

[Perumusan Masalah](#_bookmark10) [17](#_bookmark10)

[Studi Literatur](#_bookmark11) [17](#_bookmark11)

[Analisis dan Perancangan Sistem](#_bookmark11) [17](#_bookmark11)

[Implementasi Sistem](#_bookmark11) [18](#_bookmark11)

[Pengujian dan Evaluasi](#_bookmark11) [18](#_bookmark11)

[Kesimpulan dan Saran](#_bookmark11) [18](#_bookmark11)

[Sistematika Laporan](#_bookmark12) [18](#_bookmark12)

[Bab I Pendahuluan](#_bookmark13) [18](#_bookmark13)

[**BAB II**](#_bookmark14)

### [Bab II Profil Perusahaan](#_bookmark13) [18](#_bookmark13)

### [Bab III Tinjauan Pustaka](#_bookmark13) [18](#_bookmark13)

### [Bab IV Analisis dan Perancangan Sistem](#_bookmark13) [19](#_bookmark13)

### [Bab V Implementasi Sistem](#_bookmark13) [19](#_bookmark13)

### [Bab VI Pengujian dan Evaluasi](#_bookmark13) [19](#_bookmark13)

### [Bab VII Kesimpulan dan Saran](#_bookmark13) [19](#_bookmark13)

## [PROFIL PERUSAHAAN](#_bookmark14) [21](#_bookmark14)

### [Profil CV. Aksamedia](#_bookmark15)............................................................

### [21](#_bookmark15)

### [Sejarah CV. Aksamedia](#_bookmark16) [21](#_bookmark16)

## [BAB III](#_bookmark17)

## [TINJAUAN PUSTAKA](#_bookmark17) [23](#_bookmark17)

### [Pemrograman Web](#_bookmark18) [23](#_bookmark18)

### [HTML](#_bookmark19) [23](#_bookmark19)

### [Javascript](#_bookmark19) [24](#_bookmark19)

### [VueJs](#_bookmark19) [24](#_bookmark19)

### [AxiosJs](#_bookmark19) [24](#_bookmark19)

### [Laravel](#_bookmark19) [24](#_bookmark19)

### [MySQL](#_bookmark19) [25](#_bookmark19)

## [BAB IV](#_bookmark20)

## [ANALISIS DAN PERANCANGAN SISTEM](#_bookmark20) [27](#_bookmark20)

### [Analisis Sistem](#_bookmark21) [27](#_bookmark21)

### [Definisi Umum Aplikasi](#_bookmark22) [27](#_bookmark22)

### [Analisis Kebutuhan](#_bookmark23) [27](#_bookmark23)

### [Autentikasi Agen dan Admin](#_bookmark24) [29](#_bookmark24)

### [Mengubah Data Diri](#_bookmark25) [29](#_bookmark25)

### [Menambahkan Properti](#_bookmark26) [29](#_bookmark26)

### [Melihat Daftar Properti](#_bookmark27) [29](#_bookmark27)

### [Agen Dapat Memfavoritkan Properti](#_bookmark28) [29](#_bookmark28)

### [Agen Menghapus Properti dari Daftar Favorit](#_bookmark29) [30](#_bookmark29)

### [Melakukan Pencarian Properti](#_bookmark30) [30](#_bookmark30)

### [Melihat Detail Properti](#_bookmark31) [30](#_bookmark31)

### [Admin Dapat Mengelola Data Agen](#_bookmark32) [30](#_bookmark32)

### [Admin Dapat Mengelola Data Properti](#_bookmark33) [30](#_bookmark33)

### [Sistem Mengirimkan Notifikasi Verifikasi Akun](#_bookmark34) [30](#_bookmark34)

### [Sistem Mengirimkan Notifikasi Verifikasi Properti 30](#_bookmark35) [Agen Dapat Mendaftarkan Akun](#_bookmark36) [31](#_bookmark36)

### [Perancangan Sistem](#_bookmark37) [32](#_bookmark37)

### [Desain Sistem](#_bookmark38) [32](#_bookmark38)

### [Autentikasi Agen dan Admin](#_bookmark39) [34](#_bookmark39)

### [Mengubah Data Diri](#_bookmark40) [35](#_bookmark40)

### [Menambahkan Properti](#_bookmark41) [36](#_bookmark41)

### [Melihat Daftar Properti](#_bookmark42) [37](#_bookmark42)

### [Agen Dapat Memfavoritkan Properti](#_bookmark43) [38](#_bookmark43)

### [Agen Menghapus Properti dari Daftar Favorit](#_bookmark44) [39](#_bookmark44)

### [Melakukan Pencarian Properti](#_bookmark45) [39](#_bookmark45)

### [Melihat Detail Properti](#_bookmark46) [40](#_bookmark46)

### [Admin Dapat Mengelola Data Agen](#_bookmark47) [41](#_bookmark47)

### [Admin Dapat Mengelola Data Properti](#_bookmark48) [42](#_bookmark48)

### [Sistem Mengirimkan Notifikasi Verifikasi Akun](#_bookmark49) [44](#_bookmark49)

### [Sistem Mengirimkan Notifikasi Verifikasi Properti 45](#_bookmark50) [Agen Dapat Mendaftarkan Akun](#_bookmark51) [46](#_bookmark51)

## [BAB V](#_bookmark52)

## [IMPLEMENTASI SISTEM](#_bookmark52) [48](#_bookmark52)

### [Implementasi Model](#_bookmark53) [48](#_bookmark53)

### [Lapisan Model Agen](#_bookmark54) [48](#_bookmark54)

### [Kode Sumber 5.1. Model Agen](#_bookmark55) [50](#_bookmark55)

### [Lapisan Model Users](#_bookmark56) [50](#_bookmark56)

### [Lapisan Model Properties](#_bookmark57) [50](#_bookmark57)

### [Lapisan Model Property Type](#_bookmark58) [53](#_bookmark58)

### [Lapisan Model Property Images](#_bookmark59) [53](#_bookmark59)

### [Lapisan Model Sub Districts](#_bookmark60) [54](#_bookmark60)

### [Lapisan Model Cities](#_bookmark61) [54](#_bookmark61)

### [Lapisan Model Provinces](#_bookmark62) [55](#_bookmark62)

### [Lapisan Model Favorites](#_bookmark63) [55](#_bookmark63)

### [Lapisan Model Downline](#_bookmark64) [56](#_bookmark64)

### [Implementasi Lapisan Kontrol](#_bookmark65) [56](#_bookmark65)

### [Lapisan Kontrol Agen](#_bookmark66) [56](#_bookmark66)

### [FavoriteController](#_bookmark67) [57](#_bookmark67)

### [PropertyController](#_bookmark67) [58](#_bookmark67)

### [PropertyFavoritesController](#_bookmark67) [65](#_bookmark67)

### [PropertySearchController](#_bookmark67) [66](#_bookmark67)

### [ProfileController](#_bookmark67) [68](#_bookmark67)

### [AccountController](#_bookmark67) [69](#_bookmark67)

### [SecurityController](#_bookmark67) [71](#_bookmark67)

### [DashboardController](#_bookmark67) [72](#_bookmark67)

### [LoginController](#_bookmark67) [73](#_bookmark67)

### [RegisterController](#_bookmark67) [74](#_bookmark67)

### [Lapisan Kontrol Admin](#_bookmark66) [75](#_bookmark66)

### [AgenController](#_bookmark68) [75](#_bookmark68)

### [LoginController](#_bookmark68) [79](#_bookmark68)

### [EmailNewPropertiesController](#_bookmark68) [80](#_bookmark68)

### [EmailVerifiedNotificationController](#_bookmark68) [82](#_bookmark68)

### [PropertyController](#_bookmark68) [82](#_bookmark68)

### [AdminController](#_bookmark68) [90](#_bookmark68)

### [Implementasi Antarmuka Pengguna](#_bookmark69) [91](#_bookmark69)

### [Antarmuka Admin](#_bookmark70) [91](#_bookmark70)

### [Halaman Utama](#_bookmark71) [91](#_bookmark71)

### [Halaman Pengelolaan Agen](#_bookmark72) [93](#_bookmark72)

### [Halaman Pengelolaan Properti](#_bookmark73) [94](#_bookmark73)

### [Antarmuka Agen](#_bookmark74) [95](#_bookmark74)

### [Landing Page](#_bookmark75) [95](#_bookmark75)

### [Halaman Utama](#_bookmark75) [97](#_bookmark75)

### [Halaman Profile](#_bookmark75) [99](#_bookmark75)

### [Halaman Property Management](#_bookmark75) [100](#_bookmark75)

### [Halaman Favorite](#_bookmark75) [102](#_bookmark75)

### [Antarmuka Email](#_bookmark76) [102](#_bookmark76)

## [BAB VI](#_bookmark77)

## [PENGUJIAN DAN EVALUASI](#_bookmark77) [105](#_bookmark77)

### [Tujuan Pengujian](#_bookmark78) [105](#_bookmark78)

### [Kriteria Pengujian](#_bookmark79) [105](#_bookmark79)

### [Skenario Pengujian](#_bookmark80) [106](#_bookmark80)

### [Evaluasi Pengujian](#_bookmark81) [108](#_bookmark81)

## [BAB VII](#_bookmark82)

## [KESIMPULAN DAN SARAN](#_bookmark82) [110](#_bookmark82)

### [Kesimpulan](#_bookmark83) [110](#_bookmark83)

### [Saran](#_bookmark84) [110](#_bookmark84)

## [DAFTAR PUSTAKA](#_bookmark85) [112](#_bookmark85)

## [BIODATA PENULIS I](#_bookmark86) [114](#_bookmark86)

## [BIODATA PENULIS II](#_bookmark87) [116](#_bookmark87)

*[Halaman ini sengaja dikosongkan]*

# DAFTAR GAMBAR

Gambar 4.1. Use-Case Diagram 33

Gambar 4.2. Context Diagram 34

Gambar 4.3. Activity Digram Autentikasi 35

Gambar 4.4. Activity Diagram Mengubah Data Diri 36

Gambar 4.5. Activity Diagram Menambah Properti 37

Gambar 4.6. Activity Diagram Melihat Daftar Properti 38

Gambar 4.7. Activity Diagram Menambahkan ke Favorit 38

Gambar 4.8. Activity Diagram Menghapus dari Favorit 39

Gambar 4.9. Activity Diagram Pencarian Data Properti 40

Gambar 4.10. Activity Diagram Melihat Detail Properti 41

Gambar 4.11. Activity Diagram Mengelola Data Agen 42

Gambar 4.12. Activity Diagram Mengelola Properti 43

Gambar 4.13. Activity Diagram Notifikasi Email Agen 44

Gambar 4.14. Activity Diagram Notifikasi Email Properti 45

Gambar 4.15. Activity Diagram Pendaftaran Agen 46

Gambar 5.1. Antarmuka Login Admin 92

Gambar 5.2. Antarmuka Dashboard Admin 92

Gambar 5.3. Antarmuka Admin Logout 93

Gambar 5.4. Antarmuka Admin Daftar Agen 93

Gambar 5.5. Antarmuka Admin Detail dan Edit Agen 94

Gambar 5.6. Antarmuka Admin Daftar Properti 94

Gambar 5.7. Antarmuka Admin Detail dan Edit Properti 95

Gambar 5.8. Antarmuka Admin Tambah Properti 95

Gambar 5.9. Antarmuka Registrasi Agen 96

Gambar 5.10. Antarmuka Login Agen 96

Gambar 5.11. Antarmuka Tunggu Verifikasi 97

Gambar 5.12. Antarmuka Dashboard Agen 97

Gambar 5.13. a) Antarmuka Agen Detail Properti 98

Gambar 5.13. b) Antarmuka Agen Detail Properti Lanjt 98

Gambar 5.14. Antarmuka Agen Navigasi dan Logout 99

Gambar 5.15. Antarmuka Profil Agen 99

Gambar 5.16. Antarmuka Edit Profil Agen 100

### Gambar 5.17. Antarmuka Mitra Agen 100

### Gambar 5.18. Antarmuka Daftar Properti Agen 101

### Gambar 5.19. Antarmuka Tombol Detail Properti 101

### Gambar 5.20. Antarmuka Favorit 102

### Gambar 5.21. Antarmuka Hapus Favorit 102

### Gambar 5.22. Antarmuka Email Akun Terverifikasi 103

### Gambar 5.23. Antarmuka Email Properti Baru 103

*[Halaman ini sengaja dikosongkan]*

# DAFTAR TABEL

Tabel 4.1 Kebutuhan Fungsional 28

Tabel 4.2 Kebutuhan Non-fungsional 31

Tabel 6.1 Hasil Evaluasi Pengujian 108

*[Halaman ini sengaja dikosongkan]*

# DAFTAR KODE SUMBER

### Kode Sumber 5.1. Model Agen 48

### Kode Sumber 5.2. Model Users 50

### Kode Sumber 5.3. Model Properties 50

### Kode Sumber 5.4. Model Property Types 53

### Kode Sumber 5.5. Model Property Images 53

### Kode Sumber 5.6. Model Sub Districts 54

### Kode Sumber 5.7. Model City 54

### Kode Sumber 5.8. Model Provinces 55

### Kode Sumber 5.9. Model Favorites 55

### Kode Sumber 5.10. Model Downline 56

### Kode Sumber 5.11. Favorite Controller 57

### Kode Sumber 5.12. Property Controller 58

### Kode Sumber 5.13. Property Favorite Controller 65

### Kode Sumber 5.14. Property Search Contoller 66

### Kode Sumber 5.15. Profile Controller 68

### Kode Sumber 5.16. Account Controller 69

### Kode Sumber 5.17. Security Controller 71

### Kode Sumber 5.18. Dashboard Controller 72

### Kode Sumber 5.19. Login Controller 73

### Kode Sumber 5.20. Register Controller 74

### Kode Sumber 5.21. Agen Controller 75

### Kode Sumber 5.22. Admin Login Controller 79

### Kode Sumber 5.23. Email New Properties Controller 81

### Kode Sumber 5.24. Email Verified Notification 82

### Kode Sumber 5.25. Admin Property Controller 83

### Kode Sumber 5.26. Admin Controller 91

*[Halaman ini sengaja dikosongkan]*

# BAB I PENDAHULUAN

* 1. **Latar Belakang**

Saat ini era digital makin berkembang. Semua layanan dapat disajikan secara mobile dengan adanya teknologi yang dapat dimanfaatkan untuk mempermudah manusia dalam melakukan layanan tersebut. Sebelumnya, garda pangan masih menggunakan *messenger* untuk berkoordinasi dalam setiap pelaksanaan kegiatannya*.* Sarana ini belum cukup praktis sehingga tidak dapat menutupi semua permasalahan yang dihadapi, seperti relawan yang menghilang, masalah pendataan dan lain sebagainya. Dengan aplikasi *mobile*, *business logic* garda pangan dapat diterapkan secara konsisten dalam setiap penggunaannya. Selain itu, aplikasi *mobile* dapat melibatkan database untuk setiap keperluan *data record* seperti data relawan dan datakegiatannya.

Garda Pangan merupakan sebuah *food bank* yang bertujuan menjadi pusat koordinasi makanan surplus dan berpotensi terbuang, untuk disalurkan kepada masyarakat pra-sejahtera. Semua permasalahan dapat diubah menjadi lebih *mobile* atau aktif walaupun berpindah tempat. Salah satu permasalahan yang dihadapi ialah belum adanya sarana yang efektif untuk koordinasi antar relawan. Pada saat KP, kami diberi kesempatan untuk menyelesaikan permasalahan tersebut.

* 1. **Tujuan**

` Tujuan kerja praktik ini adalah menyelesaikan kewajiban nilai kerja praktik

dan membantu perusahaan untuk mempermudah pendistribusian makanan berlebih dalam wujud aplikasi mobile berbasis *Android*.

* 1. **Manfaat**

Manfaat yang diperoleh dengan adanya aplikasi mobile ini antara lain adalah mempermudah *Food Heroes* (relawan Garda Pangan) dalam pendistribusian makanan dari mitra supaya lebih teratur, karena sebelumnya masih manual, yaitu melalui aplikasi chat *Whatsapp.*

* 1. **Rumusan Masalah**

Rumusan masalah dari kerja praktik ini adalah sebagai berikut:

* + 1. Bagaimana membuat antarmuka aplikasi berbasis mobile mengenai pendistribusian makanan?
    2. Bagaimana cara membedakan *Food Heroes* Biasa dan *Food Heroes Gold*?
  1. **Lokasi dan Waktu Kerja Praktik**

Kerja praktik dimulai pada tanggal 6 Januari 2020 hingga 24 Februari 2020 dan dilakukan secara remote working. Untuk monitoring terhadap Penanggung jawab lapangan, dilakukan rapat setiap 2 minggu di Kantor Garda Pangan yang berada di Wisma Mukti, Jl. Klampis Anom III F-22, Sukolilo, Surabaya.

* 1. **Metodologi Kerja Praktik**

Metodologi dalam pembuatan buku kerja praktik meliputi :

* + 1. **Perumusan Masalah**

Untuk mengetahui antarmuka seperti apa yang diinginkan untuk sebuah aplikasi mobile. Sebelum memulai Kerja Praktik, kami bertemu dengan *Founder* Garda Pangan untuk membahas rincian aplikasi *mobile* yang akan dibangun. Pada saat rapat, kami dijelaskan tentang kegiatan apa saja yang dilakukan oleh Garda Pangan. Setelah dijelaskan, kami merumuskan fitur-fitur apa saja yang akan diterapkan pada aplikasi *mobile* yang akan dibuat.

* + 1. **Studi Literatur**

Setelah mendapat gambaran bagaimana sistem tersebut berjalan, kami diberi waktu untuk memikirkan rancangan antarmuka berdasarkan *user journey* yang telah ditentukan.

* + 1. **Analisis dan Perancangan Antarmuka**

Setelah mempertimbangkan beberapa hal, kami memutuskan untuk menggunakan Adobe XD untuk desain antarmuka, sedangkan untuk *front-end* kami menggunakan *Android Studio* untuk implementasi desain yang telah kami buat.

* + 1. **Implementasi Antarmuka**

Implementasi merupakan realisasi dari tahap perancangan. Pada tahap ini kami membuat antarmuka aplikasi yang dirancang dari awal.

* + 1. **Pengujian dan Evaluasi**

Setelah antarmuka aplikasi yang telah direncanakan telah jadi, perlu adanya evaluasi untuk menguji apakah sudah sesuai dengan harapan *Founder*. Jika masih belum sesuai atau perlu menambah fitur, rapat akan dilakukan lagi untuk memfloorkan fitur - fitur apa saja yang perlu diperbaiki atau ditambah.

* + 1. **Kesimpulan dan Saran**

Pengujian yang dilakukan ini telah memenuhi syarat yang diinginkan, dan berjalan dengan baik dan lancar.

* 1. **Sistematika Laporan**
     1. **Bab I Pendahuluan**

Bab ini berisi latar belakang, tujuan, manfaat, rumusan masalah, lokasi dan waktu kerja praktik, metodologi, dan sistematika laporan.

* + 1. **Bab II Profil Perusahaan**

Bab ini berisi gambaran umum Garda Pangan, mulai dari sejarah, tujuan, dan visi misi perusahaan.

* + 1. **Bab III Tinjauan Pustaka**

Bab ini berisi dasar teori dari teknologi yang digunakan menyelesaikan proyek kerja praktik.

* + 1. **Bab IV Analisis dan Perancangan Antarmuka**

Bab ini berisi mengenai tahap analisis antarmuka aplikasi dalam menyelesaikan proyek kerja praktik.

* + 1. **Bab V Implementasi Antarmuka**

Bab ini berisi uraian tahap - tahap yang dilakukan untuk proses implementasi antarmuka.

* + 1. **Bab VI Pengujian dan Evaluasi**

Bab ini berisi hasil uji coba dan evaluasi dari aplikasi yang telah dikembangkan selama pelaksanaan kerja praktik.

* + 1. **Bab VII Kesimpulan dan Saran**

Bab ini berisi kesimpulan dan saran yang didapat dari proses pelaksanaan kerja praktik.

*[Halaman ini sengaja dikosongkan]*

**BAB II PROFIL PERUSAHAAN**

* 1. **Profil Garda Pangan**

Garda Pangan merupakan sebuah *food bank* yang bertujuan menjadi pusat koordinasi makanan surplus dan berpotensi terbuang, untuk disalurkan kepada masyarakat pra-sejahtera.

* 1. **Sejarah Garda Pangan**

Garda Pangan bermula dari pengalaman salah seorang founder, Dedhy Trunoyudho yang berlatar belakang pengusaha katering pernikahan, yang seringkali menghadapi masalah pembuangan makanan tiap pekannya. Dari sudut pandang bisnis, membuang makanan menjadi pilihan ideal karena cepat, murah, dan praktis untuk dilakukan.

Kebiasaan tersebut dicermati oleh Indah Audivtia, istri Dedhy yang melihat pembuangan makanan ini sebagai hal yang menyesakkan dan mengganggu. Kegelisahan itulah yang akhirnya menggerakkan mereka berdua untuk melakukan sesuatu, yaitu mendonasikan makanan berlebih. Bersama Eva Bachtiar, seseorang yang juga mempunyai semangat yang sama untuk menyelesaikan isu pembuangan makanan, mereka bertiga menginisiasi gerakan food bank di Surabaya dengan nama Garda Pangan.

*[Halaman ini sengaja dikosongkan]*

**BAB III TINJAUAN PUSTAKA**

* 1. **Pemrograman Mobile**

Pemrograman *Mobile* adalah pemrograman yang ditujukan untuk pembuatan aplikasi di perangkat *mobile*. Pemrograman *mobile* yang kami gunakan berbasis *Android*.

*Android* adalah sistem operasi seluler yang didasarkan pada versi modifikasi dari kernel Linux dan perangkat lunak open source lainnya, yang dirancang terutama untuk perangkat seluler layar sentuh seperti smartphone dan tablet. Android dikembangkan oleh konsorsium pengembang yang dikenal sebagai Open Handset Alliance, dengan kontributor utama dan pemasar komersial adalah Google.

* 1. **Java**

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik (*general purpose*), dan secara khusus dirancang untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa platform sistem operasi yang berbeda, java dikenal pula dengan slogannya, "Tulis sekali, jalankan di mana pun". Saat ini java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi.

# XML

### XML (*Extensible Markup Language*) adalah [bahasa *markup*](https://id.wikipedia.org/wiki/Bahasa_markup) untuk keperluan [umum](https://id.wikipedia.org/wiki/Umum) yang disarankan oleh [W3C](https://id.wikipedia.org/wiki/W3C) untuk membuat dokumen markup keperluan pertukaran data antar [sistem](https://id.wikipedia.org/wiki/Sistem) yang beraneka ragam. XML merupakan kelanjutan dari [HTML](https://id.wikipedia.org/wiki/HTML) ([*HyperText Markup Language*](https://id.wikipedia.org/wiki/HyperText_Markup_Language)) yang merupakan bahasa standar untuk melacak [Internet](https://id.wikipedia.org/wiki/Internet).

# Adobe XD

# Gradle

*[Halaman ini sengaja dikosongkan]*

**BAB IV**

# ANALISIS DAN PERANCANGAN ANTARMUKA

* 1. **Analisis Antarmuka**

Pada bab ini akan dijelaskan mengenai tahapan dalam membangun aplikasi Garda Pangan Mobile yaitu analisis dari sistem yang akan dibangun. Hal tersebut dijelaskan ke dalam dua bagian, definisi umum aplikasi dan analisis kebutuhan.

* + 1. **Definisi Umum Aplikasi**

Secara umum, aplikasi Garda Pangan Mobile merupakan sistem pengelolaan dan distribusi informasi penjualan properti bangunan berbasis website yang digunakan untuk saling membagikan informasi properti bangunan seperti rumah, kostan, tanah, ruko, dan lain sebagainya kepada sesama agen. Sistem ini memiliki dua *view* dari dua sisi *user* yaitu agen dan admin. Pada sisi agen, terdapat *dashboard* yang menampilkan semua properti bangunan dan fitur menambah properti. Sedangkan pada sisi admin, terdapat fitur pengelolaan dan verifikasi properti bangunan dan data agen.

* + 1. **Analisis Kebutuhan**

Dalam aplikasi ini, terdapat fungsi-fungsi yang harus dipenuhi oleh sistem. Fungsi-fungsi yang harus dipenuhi terebut adalah hasil diskusi antara *project manager* dengan klien. Kebutuhan ini terbagi ke dalam dua jenis, yakti kebutuhan fungsional dan kebutuhan non-fungsional

* + - 1. **Kebutuhan Fungsional**

Kebutuhan fungsional pada aplikasi ini menjelaskan bagaimana sistem ini bekerja yang akan dijelaskan pada Tabel 4.1

*Tabel 4.1 Kebutuhan Fungsional*

|  |  |
| --- | --- |
| Kode | Deskripsi Kebutuhan |
| F-001 | Autentikasi Agen dan Admin |
| F-002 | Mengubah data diri |
| F-003 | Menambahkan properti |
| F-004 | Melihat daftar properti |
| F-005 | Agen dapat memfavoritkan properti |
| F-006 | Agen menghapus properti dari daftar favorit |
| F-007 | Melakukan pencarian properti |
| F-008 | Melihat detail properti |
| F-009 | Admin dapat mengelola data Agen |
| F-010 | Admin dapat mengelola data properti |

|  |  |
| --- | --- |
| F-011 | Sistem mengirimkan notifikasi verifikasi Akun |
| F-012 | Sistem mengirimkan notifikasi verifikasi Properti |
| F-013 | Agen dapat mendaftarkan akun |

Untuk penjelasan dari masing-masing kebutuhan fungsional dapat dilihat pada penjelasan berikut :

**F-001. Autentikasi Agen dan Admin**

Agen dan Admin yang melakukan *login* kedalam aplikasi melalui 2 (dua) laman website yang berbeda untuk memastikan keaslian hak akses.

**F-002. Mengubah Data Diri**

Agen yang telah *login* dan terverifikasi dapat melakukan perubahan data diri.

**F-003. Menambahkan Properti**

Agen yang telah *login* dapat menambahkan properti kedalam sistem.

**F-004. Melihat Daftar Properti**

Agen yang telah *login* dapat melihat daftar properti yang ada dalam sistem.

**F-005. Agen Dapat Memfavoritkan Properti** Agen yang telah *login* dapat menambahkan satu atau lebih properti ke daftar favoritnya.

**F-006. Agen Menghapus Properti dari Daftar Favorit**

### Agen yang telah *login* dapat menghapus properti dari daftar favoritnya.

**F-007. Melakukan Pencarian Properti**

Agen yang telah *login* dapat mencari properti menurut harga, tempat, dan jenis properti.

**F-008. Melihat Detail Properti**

Agen yang telah *login* dapat melihat detail suatu properti.

**F-009. Admin Dapat Mengelola Data Agen** Admin yang telah *login* dapat mengaktifkan atau menonaktifkan Agen dan mengelola data Agen seperti mengubah dan menghapus.

## F-010. Admin Dapat Mengelola Data Properti

### Admin yang telah *login* dapat mengubah status suatu properti seperti *unverified, verified,* dan terjual dan mengelola data properti seperti menghapus, menambah, mencari, dan melihat detail properti.

**F-011. Sistem Mengirimkan Notifikasi Verifikasi Akun**

Sistem dapat mengirimkan notifikasi berupa email kepada agen yang akunnya telah diverifikasi oleh Admin

**F-012. Sistem Mengirimkan Notifikasi Verifikasi Properti**

Sistem dapat mengirimkan pesan notifikasi berupa email kepada seluruh

agen yang telah terverifikasi bahwa ada properti baru yang telah valid atau terverifikasi.

**F-013. Agen Dapat Mendaftarkan Akun** Agen dapat membuat akun dengan mengirimkan data diri di halaman yang tersedia di dalam sistem.

## Kebutuhan Non Fungsional

### Kebutuhan non-fungsional adalah kebutuhan pengguna untuk mendefinisikan bagaimana batasan dan karakteristik dari sebuah sistem yang dibangun. Kebutuhan non-fungsional dari aplikasi UPI ini terdapat pada Tabel 4.2.

*Tabel 4.2 Kebutuhan Non-fungsional*

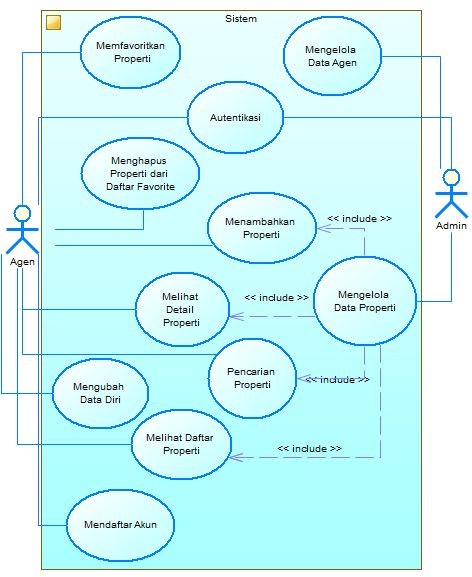
|  |  |
| --- | --- |
| Kode | Deskripsi Kebutuhan |
| NF-001 | Dapat diakses oleh semua  *web-browser* |
| NF-002 | Sistem dapat diakses 24 jam sehari |
| NF-003 | Sistem dapat diakses oleh semua perangkat yang memiliki *web-browser* yang mendukung HTML 5.0 |
| NF-004 | Sistem memiliki *feedback* yang baik |

|  |  |
| --- | --- |
| NF-005 | Sistem harus dapat memberikan hak akses data kepada pengguna yang berhak. |

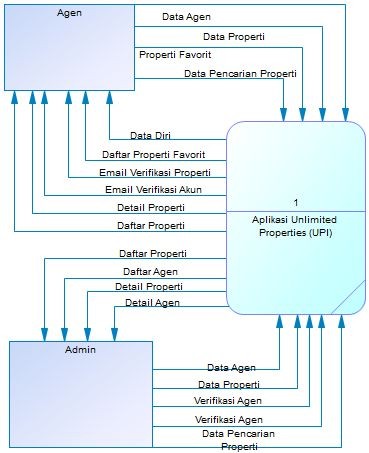
* 1. **Perancangan Sistem**
     1. **Desain Sistem**

Desain sistem digunakan untuk mengetahui jalannya proses bisnis pada suatu aplikasi sehingga pengembangan dan pemeliharaan aplikasi dapat dengan mudah dilakukan. Desain sistem yang digunakan adalah *Use Case Diagram , Context Diagram,* dan *Activity Diagram.*

Gambar 4.1. merupakan *Use Case Diagram* yang menunjukkan proses bisnis apa saja dan siapa saja yang terlibat dalam proses tersebut. Gambar 4.2 menunjukkan *Context Diagram* yang menjelaskan alur data (*data flow*) antara sistem dengan *stakeholder*. Terakhir, *Activity Diagram* menunjukkan bagaimana sistem berinteraksi dengan *stakeholder*.



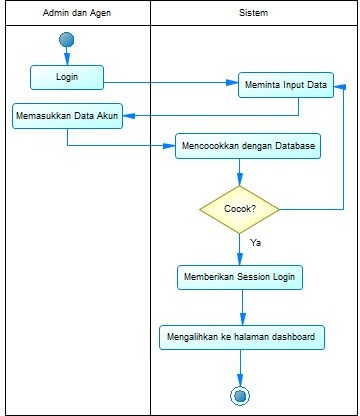
Gambar 4.1. *Use-Case Diagram*



Gambar 4.2. *Context Diagram*

## F-001. Autentikasi Agen dan Admin

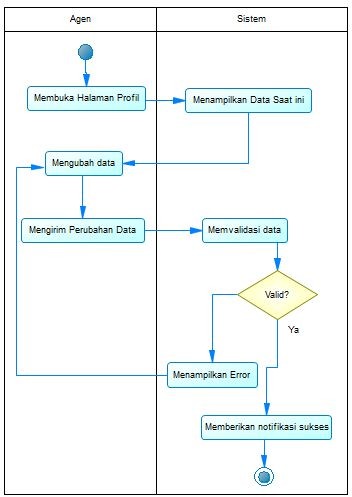
### Admin dan Agen dapat melakukan *login* kedalam aplikasi sesuai dengan masing-masing laman autentikasi. Gambar 4.3. di bawah ini merupakan diagram aktivitas yang menunjukkan alur autentikasi



Gambar 4.3. *Activity Diagram Autentikasi*

## F-002. Mengubah Data Diri

### Agen yang telah *login* dapat melihat dan mengubah data dirinya pada *form* yang telah disediakan oleh sistem. Gambar 4.4 di bawah ini merupakan diagram aktivitas yang menunjukkan alur pengubahan data diri Agen.

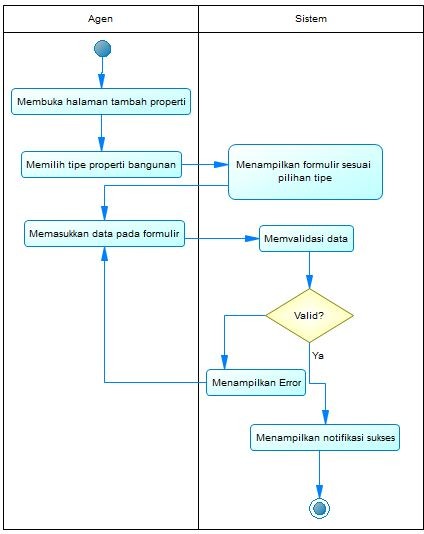


Gambar 4.4. *Activity Diagram Mengubah Data Diri*

## F-003. Menambahkan Properti

### Agen dapat menambahkan data properti baru dengan mengisi *form* yang telah disediakan oleh

sistem. Gambar 4.5 di bawah ini merupakan diagram aktivitas yang menunjukkan alur penambahan data properti ole Agen.

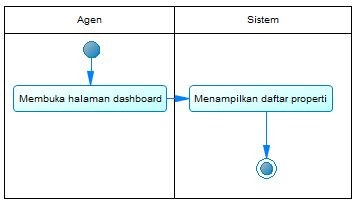


Gambar 4.5. *Activity Diagram* Menambah Properti

## F-004. Melihat Daftar Properti

### Agen dapat melihat daftar properti yang terdaftar di sistem. Gambar 4.6 di bawah ini merupakan

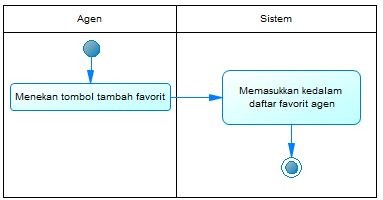
diagram aktivitas yang menunjukkan alur melihat daftar properti.



Gambar 4.6. *Activity Diagram* Melihat Daftar Properti

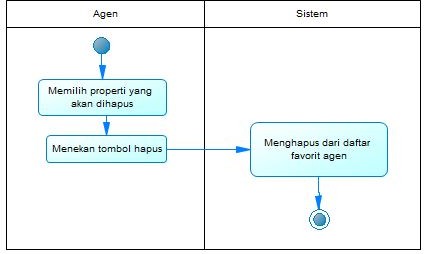
## F-005. Agen Dapat Memfavoritkan Properti

### Admin dapat menambahkan suatu properti kedalam daftar properti favoritnya dengan menekan tombol yang disediakan oleh sistem. Gambar 4.7 di bawah ini merupakan diagram aktivitas yang menunjukkan alur penambahan properti kedalam daftar favorit.



Gambar 4.7. *Activity Diagram* Menambahkan ke Favorit

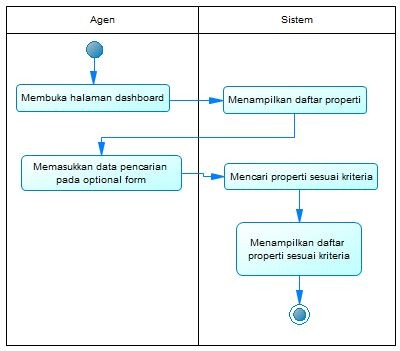
### **F-006. Agen Menghapus Properti dari Daftar Favorit** Admin dapat menghapus suatu properti dari daftar properti favoritnya dengan menekan tombol yang disediakan oleh sistem. Gambar 4.8 di bawah ini merupakan diagram aktivitas yang menunjukkan alur penghapusan suatu properti dari daftar favorit



Gambar 4.8. *Activity Diagram* Menghapus dari Favorit

## F-007. Melakukan Pencarian Properti

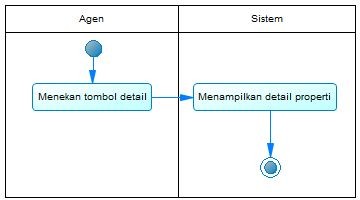
### Agen dapat melakukan pencarian properti pada halaman yang disediakan oleh sistem. Gambar 4.9 di bawah ini merupakan diagram aktivitas yang menunjukkan alur pencarian properti.



Gambar 4.9. *Activity Diagram* Pencarian Data Properti

## F-008. Melihat Detail Properti

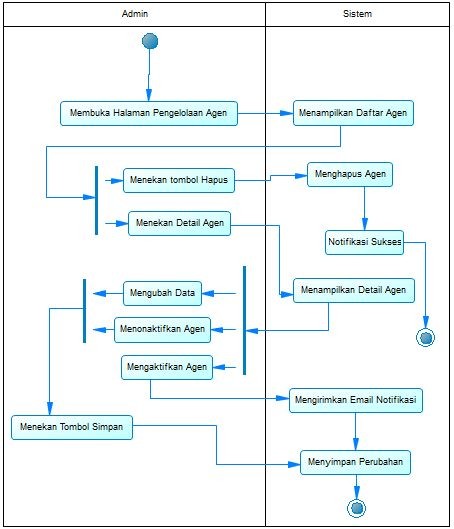
### Agen dapat melihat detail suatu properti dengan menekan tombol yang telah disediakan oleh sistem Gambar 4.10 di bawah ini merupakan diagram aktivitas yang menunjukkan alur melihat detail properti.



Gambar 4.10. *Activity Diagram* Melihat Detail Properti

## F-009. Admin Dapat Mengelola Data Agen

### Admin dapat melihat, mencari, mengubah, mengaktifkan, menonaktifkan, dan menghapus agen pada halaman yang disediakan oleh sistem. Gambar 4.11 di bawah ini merupakan diagram aktivitas yang menunjukkan alur admin mengelola agen.

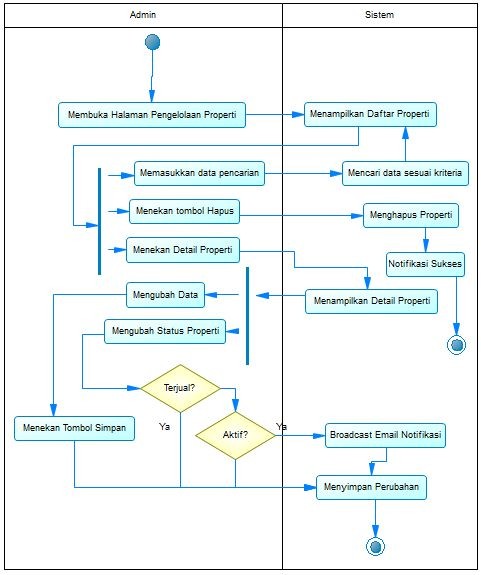


Gambar 4.11. *Activity Diagram* Mengelola Data Agen

## F-010. Admin Dapat Mengelola Data Properti

### Admin dapat melihat, mencari, menambah, mengubah, mengaktifkan, menonaktifkan, dan menghapus data properti pada halaman yang disediakan oleh sistem. Gambar 4.12 di bawah ini

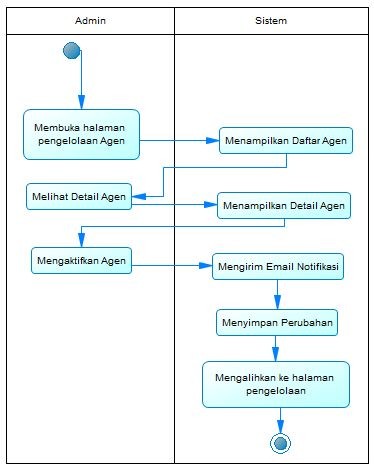
merupakan diagram aktivitas yang menunjukkan alur admin mengelola data properti.



Gambar 4.12. *Activity Diagram* Mengelola Properti

## F-011. Sistem Mengirimkan Notifikasi Verifikasi Akun

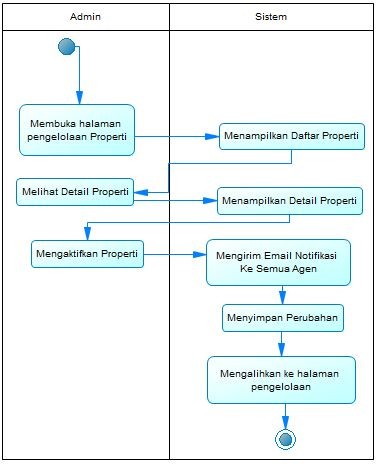
### Sistem dapat mengirimkan notifikasi email kepada Agen apabila akunnya telah diverifikasi oleh Admin. Gambar 4.13 di bawah ini merupakan diagram aktivitas yang menunjukkan alur sistem mengirimkan notifikasi.



Gambar 4.13. *Activity Diagram* Notifikasi Email Agen

## F-012. Sistem Mengirimkan Notifikasi Verifikasi Properti

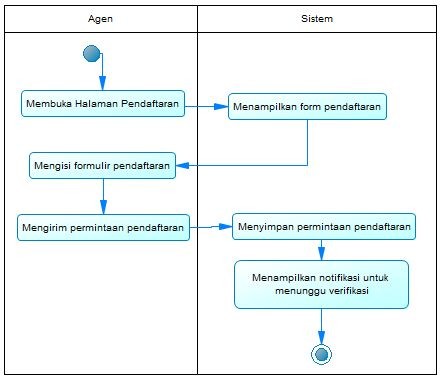
### Sistem dapat mengirimkan notifikasi email kepada seluruh Agen apabila ada properti yang telah diverifikasi oleh Admin. Gambar 4.14 di bawah ini merupakan diagram aktivitas yang menunjukkan alur sistem mengirimkan notifikasi.



Gambar 4.14. *Activity Diagram* Notifikasi Email Properti

## F-013. Agen Dapat Mendaftarkan Akun

### Agen dapat mendaftar akun baru pada halaman yang telah disediakan oleh sistem. Akan, tetapi untuk dapat menggunakan fitur lebih lanjut, akun Agen yang baru terdaftar harus diverifikasi oleh Admin terlebih dahulu. Gambar 4.15. di bawah merupakan diagram aktivitas yang menunjukkan alur pendaftaran Agen.



Gambar 4.15. *Activity Diagram* Pendaftaran Agen

*[Halaman ini sengaja dikosongkan]*

**BAB V**

**IMPLEMENTASI ANTARMUKA**

Bab ini membahas tentang implementasi dari antarmuka yang kami buat. Implementasi ini akan dibagi ke dalam beberapa bagian, yaitu bagian implementasi lapisan model, implementasi lapisan kontrol, dan implementasi antarmuka pengguna.

* 1. **Implementasi Model**

Implementasi lapisan model ini merupakan lapisan yang digunakan *framework* Laravel untuk menghubungkan aplikasi dengan *database*. Model di dalam Laravel terkadang memiliki *function* yang akan digunakan sebagai di lapisan kontroller yang biasa disebut *eloquent.* Adapun implementasi lapisan model sebagai berikut :

* + 1. **Lapisan Model Agen**

Bagian dari implementasi lapisan model Agen yang menyimpan data profil Agen ditampilkan pada Kode Sumber 5.1 berikut:

**<?php**

**namespace App**\**Models**;

**use Illuminate**\**Notifications**\**Notifiable**;

**use Illuminate**\**Foundation**\**Auth**\**User as Authenticatable**; **use Illuminate**\**Database**\**Eloquent**\**Model**;

**use App**\**Notifications**\**AgentForgotPasswordNotification**; **class Agent extends Authenticatable**

{

**use Notifiable**;

**protected** $guard = 'agents';

**protected** $fillable = [

'name', 'email','upline','phone\_number', 'password', 'id', 'plain\_text\_password', 'approved', 'address',

'nik', 'bank\_account', 'bank\_name', 'bank\_customer', 'photo','username','whatsapp'

];

**protected** $hidden = ['plain\_text\_password', 'password'];

**public function isNeededToCompleteData**(){

**return** !(@**$this**->nik && @**$this**->phone\_number &&

!is\_null(@**$this**->upline) && @**$this**->username);

}

**public function isApprovedAgent**()

{

**return** !(@**$this**->approved==1);

}

**public function Uplinenya**(){

**return $this**->belongsTo(Agent::class, 'upline','id');

}

**public function Properties**(){

**return $this**->hasMany(Property::class,'agent\_id','id');

}

**public function getPhotoAttribute**($value)

{

**return** $value ?: "/assets/img/user-placeholder.png";

}

**public function downline**(){

**return $this**->hasMany(Downline::class,'upline','id');

}

**public function sendPasswordResetNotification**($token){

**$this**->notify(**new**

AgentForgotPasswordNotification($token));

}

}

php**?>**

Kode Sumber 5.1. Model Agen

* + 1. **Lapisan Model Users**

Bagian dari implementasi lapisan model Users yang menyimpan data pengguna admin ditampilkan pada Kode Sumber 5.2 berikut:

**<?php namespace App**;

**use Illuminate**\**Notifications**\**Notifiable**;

**use Illuminate**\**Contracts**\**Auth**\**MustVerifyEmail**;

**use Illuminate**\**Foundation**\**Auth**\**User as Authenticatable**;

**class User extends Authenticatable**{ **use Notifiable**;

**protected** $fillable = [

'name', 'email', 'password',];

**protected** $hidden = [

'password', 'remember\_token',];

}

php**?>**

Kode Sumber 5.2. Model Users

* + 1. **Lapisan Model Properties**

Bagian dari implementasi lapisan model Properties yang menyimpan data properti ditampilkan pada Kode Sumber 5.3 berikut:

**<?php**

**namespace App**\**Models**;

**use Illuminate**\**Database**\**Eloquent**\**Model**; **class Property extends Model**

{

**protected** $fillable = [

'name', 'price', 'address', 'agent\_id', 'id', 'agent\_type', 'listing\_type', 'property\_type\_id', 'land\_size',

'built\_up', 'description', 'area', 'bedrooms', 'bathrooms', 'beds', 'garages', 'city\_id','subdistrict\_id'

,'approved','sold', 'maid\_bedrooms', 'maid\_bathrooms', 'certificate', 'year\_built', 'electrical\_power', 'number\_of\_floors', 'amount\_of\_down\_payment', 'estimated\_installments', 'complete\_address', 'floor\_number',

'parking\_amount', 'owner\_name', 'owner\_phone','colisting'

];

**protected** $appends = ['city\_name','agent\_name','image','created\_at\_for\_human','detai l\_url','property\_type\_name'];

**public function PropertyImages**()

{

**return $this**->hasMany(PropertyImage::class);

}

**public function PropertyType**()

{

**return**

**$this**->belongsTo(PropertyType::class,'property\_type\_id');

}

**public function getCityNameAttribute**(){ **return** (City::find(**$this**->city\_id)->name);

}

**public function Agent**()

{

**return $this**->agent\_type == "admin" ?

**$this**->belongsTo(\App\User::class, 'agent\_id') :

**$this**->belongsTo(Agent::class);

}

**public function getAgentNameAttribute**()

{

**return $this**->agent\_type == "agent" ? Agent::find(**$this**->agent\_id)->name : "Admin";

}

**public function getImageAttribute**()

{

$images = PropertyImage::where('property\_id',

**$this**->id)->first(); **return** $images ?

((!is\_null($images->tumbnail))?asset($images->tumbnail):asset($ images->name)) :

url('assets/frontend/media-demo/properties/554x360/02.jpg');

}

**public function getCreatedAtForHumanAttribute**()

{

**return $this**->created\_at->diffForHumans();

}

**public function getDetailUrlAttribute**()

{

**return** route('listing.property.show',[**$this**->id]);

}

**public function City**()

{

**return $this**->belongsTo(City::class);

}

**public function Subdistrict**(){

**return $this**->belongsTo(Subdistrict::class);

}

**public function Favorite**()

{

**return $this**->hasMany(Favorite::class);

}

**public function getPropertyTypeNameAttribute**()

{

**return** PropertyType::find(**$this**->property\_type\_id)->name;

}

}

php**?>**

Kode Sumber 5.3. Model Properties

* + 1. **Lapisan Model Property Type**

Bagian dari implementasi lapisan model Property\_typees yang menyimpan data tipe properti ditampilkan pada Kode Sumber 5.4 berikut:

**<?php**

**namespace App**\**Models**;

**use Illuminate**\**Database**\**Eloquent**\**Model**; **class PropertyType extends Model**

{

**public** $table = 'property\_typees';

}

php**?>**

Kode Sumber 5.4. Model Property Type

* + 1. **Lapisan Model Property Images**

Bagian dari implementasi lapisan model Property\_images yang menyimpan data letak foto pada *file system* ditampilkan pada Kode Sumber 5.5 berikut:

**<?php**

**namespace App**\**Models**;

**use Illuminate**\**Database**\**Eloquent**\**Model**; **class PropertyImage extends Model**

{

**protected** $fillable = ['id', 'name', 'property\_id','tumbnail'];

**public function property**()

{

**return $this**->belongTo(Property::class);

}

}

php**?>**

Kode Sumber 5.5. Model Property Images

* + 1. **Lapisan Model Sub Districts**

Bagian dari implementasi lapisan model Sub\_districts yang menyimpan data kecamatan ditampilkan pada Kode Sumber 5.6 berikut:

**<?php**

**namespace App**\**Models**;

**use Illuminate**\**Database**\**Eloquent**\**Model**; **class Subdistrict extends Model**

{

*//*

**public** $table="sub\_districts";

}

php**?>**

Kode Sumber 5.6. Model Sub Districts

* + 1. **Lapisan Model Cities**

Bagian dari implementasi lapisan model Cities yang menyimpan data kota/kabupaten ditampilkan pada Kode Sumber 5.7 berikut:

**<?php**

**namespace App**\**Models**;

**use Illuminate**\**Database**\**Eloquent**\**Model**;

**class City extends Model**{ **public function Province**(){

**return**

**$this**->belongsTo(Province::class,'province\_id','id');

}

}

php**?>**

Kode Sumber 5.7. Model City

* + 1. **Lapisan Model Provinces**

Bagian dari implementasi lapisan model Provinces yang menyimpan data provinsi ditampilkan pada Kode Sumber 5.8 berikut:

**<?php**

**namespace App**\**Models**;

**use Illuminate**\**Database**\**Eloquent**\**Model**;

**class Province extends Model**{ **public** $table= "provinces";

**public function City**(){

**return $this**->hasMany(City::class);

}

}

php**?>**

Kode Sumber 5.8. Model Provinces

* + 1. **Lapisan Model Favorites**

Bagian dari implementasi lapisan model Favorites yang menyimpan daftar properti favorit masing-masing Agen ditampilkan pada Kode Sumber

5.9 berikut:

**<?php**

**namespace App**\**Models**;

**use Illuminate**\**Database**\**Eloquent**\**Model**;

**class Favorite extends Model**{

**protected** $fillable = ['id','property\_id','agent\_id'];

**public function Property**(){ **return**

**$this**->belongsTo(Property::class,'property\_id','id');

}

}

php**?>**

Kode Sumber 5.9. Model Favorites

* + 1. **Lapisan Model Downline**

Bagian dari implementasi lapisan model Downline yang menyimpan daftar Downline suatu Agen ditampilkan pada Kode Sumber 5.10 berikut:

**<?php**

**namespace App**\**Models**;

**use Illuminate**\**Database**\**Eloquent**\**Model**; **class Downline extends Model**

{

**public** $table = 'downline';

**protected** $fillable = ['id','upline','downline'];

**public function agent**(){

**return $this**->belongsTo(Agent::class,'id','upline');

}

}

php**?>**

Kode Sumber 5.10. Model Downline

* 1. **Implementasi Lapisan Kontrol**

Implementasi lapisan kontrol ini berisi logika yang digunakan aplikasi seperti kontrol untuk memasukkan data ke *database*, menampilkan data, dan logika lainnya. Didalam pengembangannya, kami memilih untuk memisahkan *Controller* Agen dengan Admin untuk memudahkan *maintenance* dengan mempertahankan *readability*.

* + 1. **Lapisan Kontrol Agen**

Lapisan kontrol Agen merupakan kumpulan

*Controller* yang berinteraksi dengan Agen. Berikut

### adalah beberapa *Controller* yang mengatur logika dari sisi Agen :

* + - 1. **FavoriteController**

Lapisan ini bertugas untuk mengatur logika yang berurusan dengan *View* Favorit, yaitu menampilkan dan menghapus sebuah properti dari daftar favorit. Kode Sumber 5.11 berikut merupakan implementasi dari lapisan kontrol *Favorite*:

**<?php**

**namespace App**\**Http**\**Controllers**\**Frontend**\**Dashboard**\**Favorites**;

**use Illuminate**\**Http**\**Request**;

**use App**\**Http**\**Controllers**\**Controller**; **use App**\**Models**\**Property**;

**use App**\**Models**\**Favorite**;

**class FavoriteController extends Controller**

{

**public function index**(Request $request)

{

**if**($request->ajax() **and** $request->ajax==1){

$favs =

Favorite::with('Property')->where(['favorites.agent\_id'=>auth()

->user()->id])->paginate(12);

**return** response()->json($favs);

}

**return** view('frontend.dashboard.favorites.index');

}

**public function delete**(Request $request){

**if**($request->ajax()){ Favorite::destroy($request->deletefav); **return** response()->json(['success']);

}

}

}

php**?>**

Kode Sumber 5.11 *Favorite Controller*

## PropertyController

### Lapisan ini bertugas untuk mengatur logika pengelolaan data yang berurusan dengan Properti mulai dari nenampilkan, nenambahkan data dan foto, mengubah, dan menghapus. Kode Sumber

5.12 berikut merupakan implementasi dari lapisan kontrol *Property*:

#### <?php

**namespace App**\**Http**\**Controllers**\**Frontend**\**Dashboard**\**Listings**; **use Illuminate**\**Http**\**Request**;

**use App**\**Http**\**Controllers**\**Controller**;

**use App**\**Http**\**Requests**\**Frontend**\**Property**\**StoreNewProperty**; **use Illuminate**\**Support**\**Facades**\**Storage**;

**use Illuminate**\**Support**\**Facades**\**Cache**; **use Illuminate**\**Support**\**Facades**\**File**; **use App**\**Models**\**Agent**;

**use App**\**Models**\**Property**;

**use App**\**Models**\**PropertyImage**; **use App**\**Models**\**PropertyType**; **use App**\**Models**\**City**;

**use App**\**Models**\**Favorite**; **use App**\**Models**\**Subdistrict**; **use App**\**Models**\**Province**;

**use Intervention**\**Image**\**Facades**\**Image**;

**class PropertyController extends Controller**{ **public function index**(Request $request){

**if**($request->ajax() **and** $request->ajax==1){

**if**($request->type=='all'){

$properties = Property::with('PropertyImages')->paginate(9);

}**else if**($request->type=="mine"){

$properties =

Property::with('PropertyImages')->where(['agent\_id'=>auth()->us er()->id,'agent\_type'=>'agent'])->paginate(9);

}**else if**($request->type=="sold"){

$properties =

Property::with('PropertyImages')->where(['agent\_id'=>auth()->us er()->id,'agent\_type'=>'agent','sold'=>1])->paginate(9);

}

**return** response()->json($properties);

}

**return** view('frontend.dashboard.listings.index');

}

**public function create**(Request $request){

**if**($request->ajax() **and** $request->ajax==1 **and**

!is\_null($request->city\_id)){

$subdistricts = Subdistrict::select('id','name as text')->where('city\_id','=',$request->city\_id)->get();

**return** response()->json($subdistricts->toArray());

}

**else if**($request->ajax() **and**

!is\_null($request->province\_id) **and** $request->ajax==1){

$city= City::select('id','name as

text')->where('province\_id','=',$request->province\_id)->get();

**return** response()->json($city->toArray());

}

**else if**($request->ajax() **and** $request->ajax==1){

$province= Province::select('id','name as text')->get();

**return** response()->json($province->toArray());

}

$propertyTypees = PropertyType::get();

$agents=Agent::where('id','!=',auth()->user()->id)->get();

**return** view('frontend.dashboard.listings.add', compact('propertyTypees','agents'));

}

**public function store**(StoreNewProperty $request){

$data = [

'name' => $request->name, 'price' => $request->price, 'address' => $request->address, 'agent\_id' => auth()->user()->id, 'agent\_type' => 'agent',

'listing\_type' => $request->listing\_type, 'property\_type\_id' => $request->property\_type, 'land\_size' => $request->land\_size, 'built\_up' => $request->built\_up, 'description' => $request->description, 'area' => $request->area,

'bedrooms' => $request->bedrooms, 'bathrooms' => $request->bathrooms,

'beds' => $request->bed, 'garages' => $request->garages, 'city\_id' => $request->cities,

'subdistrict\_id' => $request->subdistrict, 'approved' => 0,

'sold' => 0,

'maid\_bedrooms' => $request->maid\_bedrooms, 'maid\_bathrooms' => $request->maid\_bathrooms, 'certificate' => $request->certificate, 'year\_built' => $request->year\_built, 'electrical\_power' => $request->electrical\_power, 'amount\_of\_down\_payment' =>

str\_replace('.','',$request->amount\_of\_down\_payment), 'estimated\_installments' =>

$request->estimated\_installments,

'complete\_address' => $request->complete\_address, 'owner\_name' => $request->owner\_name, 'owner\_phone' => $request->owner\_phone, 'floor\_number' => $request->floor\_number, 'parking\_amount' => $request->parking\_amount, 'number\_of\_floors' => $request->number\_of\_floors, 'colisting' => $request->colisting

];

$property = Property::create($data);

$request->session()->put('inserted\_property',

$property);

**return** redirect()->back()->with('success','Data property berhasil disimpan');

}

**public function show**(Property $property){

$isFavorite = Favorite::where(['property\_id' =>

$property->id, 'agent\_id' => auth()->user()->id])->first();

$colisting = (!is\_null($property->colisting))?

Agent::find($property->colisting):**null**;

**return** view('frontend.dashboard.listings.show', compact('property','isFavorite','colisting'));

}

**public function edit**(Request $request, Property $property){

**if**($request->ajax() **and** $request->ajax==1 **and**

!is\_null($request->city\_id)){

$subdistricts = Subdistrict::select('id','name as text')->where('city\_id','=',$request->city\_id)->get();

**return** response()->json($subdistricts->toArray());

}

**else if**($request->ajax() **and**

!is\_null($request->province\_id) **and** $request->ajax==1){

$city= City::select('id','name as

text')->where('province\_id','=',$request->province\_id)->get();

**return** response()->json($city->toArray());

}

**else if**($request->ajax() **and** $request->ajax==1){

$province= Province::select('id','name as text')->get();

**return** response()->json($province->toArray());

}

$request->session()->put('inserted\_property',

$property);

$propertyTypees = PropertyType::get();

$agents=Agent::where('id','!=',auth()->user()->id)->get();

**return** view('frontend.dashboard.listings.edit', compact('property', 'propertyTypees','agents'));

}

**public function update**(StoreNewProperty $request, Property

$property){

$data = [

'name' => $request->name,

'price' => str\_replace('.','',$request->price), 'address' => $request->address,

'listing\_type' => $request->listing\_type, 'property\_type\_id' => $request->property\_type, 'land\_size' => $request->land\_size, 'built\_up' => $request->built\_up, 'description' => $request->description, 'area' => $request->area,

'bedrooms' => $request->bedrooms, 'bathrooms' => $request->bathrooms, 'beds' => $request->bed,

'garages' => $request->garages, 'city\_id' => $request->cities,

'subdistrict\_id' => $request->subdistrict, 'approved' => 0,

'maid\_bedrooms' => $request->maid\_bedrooms, 'maid\_bathrooms' => $request->maid\_bathrooms, 'certificate' => $request->certificate, 'year\_built' => $request->year\_built,

'electrical\_power' => $request->electrical\_power, 'amount\_of\_down\_payment' =>

str\_replace('.','',$request->amount\_of\_down\_payment), 'estimated\_installments' =>

$request->estimated\_installments,

'complete\_address' => $request->complete\_address, 'owner\_name' => $request->owner\_name, 'owner\_phone' => $request->owner\_phone, 'floor\_number' => $request->floor\_number, 'parking\_amount' => $request->parking\_amount, 'number\_of\_floors' => $request->number\_of\_floors, 'colisting' => $request->colisting

];

$property->update($data);

$request->session()->put('inserted\_property',

$property);

**return** redirect()->back()->with('success','Data property berhasil disimpan');

}

**public function removeImageFromDisk**($image){

$path\_raw=(public\_path().($image->name));

$path\_tumbnail=(public\_path().($image->tumbnail));

**if**(file\_exists($path\_raw)){ unlink($path\_raw);

}

**if**(**isset**($image->tumbnail) **and**

file\_exists($path\_tumbnail)){ unlink($path\_tumbnail);

}

}

**public function delete**(Request $request){

**if**($request->ajax()){

$images = Property::find($request->deleteProp);

$images = $images->PropertyImages;

**if**(count($images)){

**foreach** ($images **as** $image) {

**$this**->removeImageFromDisk($image); PropertyImage::destroy($image->id);

}

}

Property::destroy($request->deleteProp);

**return** response()->json(['success']);

}

}

**public function deleteImages**(Request $request){

**if**($request->ajax()){

$images = PropertyImage::find($request->deleteImg);

**echo** $images;

**$this**->removeImageFromDisk($images); PropertyImage::destroy($images->id); **return** response()->json(['success']);

}

}

**public function storeImages**(Request $request)

{

**if** (@$request->hasFile('file')) {

$image = $request->file('file');

$imageSize = getimagesize($image);

$ext=

strtolower($image->getClientOriginalExtension());

**if**($ext=='jpg' **or** $ext=='jpeg'){

$imagesource = imagecreatefromjpeg($image);

}

**else if** ($ext=='png'){

$imagesource =imagecreatefrompng($image);

}

$height = $imageSize[1];

$width =$imageSize[0];

$ratio = $imageSize[0]/$imageSize[1];

$targetWidth = 1740;

$targetHeight = 960;

$tumbnailbackground = Image::canvas($targetWidth,$targetHeight);

$realImage = file\_get\_contents($image->getRealPath());

**if**($width>=$height){

$height=$targetHeight;

$width=floor($height\*$imageSize[0]/$imageSize[1]);

}

**else**{

$width=$targetWidth;

$height=floor(($width\*$imageSize[1])/$imageSize[0]);

}

$cutx = ($width>$targetWidth)?

-floor(($width-$targetWidth)/2) : floor(($targetWidth-$width)/2);

$cuty = ($height>$targetHeight)?

-floor(($height-$targetHeight)/2) : floor(($targetHeight-$height)/2);

$tobeTumbnail=imagecreatetruecolor($width,

$height);

imagecopyresampled($tobeTumbnail, $imagesource, 0,

0, 0, 0, $width, $height, $imageSize[0], $imageSize[1]);

$tumbnailbackground->insert($tobeTumbnail,'top-left',$cutx,$cut y);

$name = 'property';

$avatarName =

$name.'-'.uniqid().'.'.$image->getClientOriginalExtension();

$path = "/storage/properties/{$avatarName}"; File::put(public\_path().$path, $realImage);

$avatarName = 'tumbnail-'.$avatarName;

$path2 = "/storage/properties/{$avatarName}";

$tumbnailbackground->save(public\_path().$path2);

$data = [

'name' => $path, 'property\_id' =>

session()->get('inserted\_property')->id,

'tumbnail' => $path2

];

$property\_image =

\App\Models\PropertyImage::create($data);

}

**return** response()->json(['success']);

}

}

php**?>**

Kode Sumber 5.12 *Property Controller*

## PropertyFavoritesController

### Lapisan ini bertugas hanya untuk logika menambahkan sebuah properti kedalam daftar favorit Agen. Kode Sumber 5.13 berikut merupakan implementasi dari lapisan kontrol *PropertyFavorites*:

#### <?php

**namespace App**\**Http**\**Controllers**\**Frontend**\**Dashboard**\**Listings**; **use Illuminate**\**Http**\**Request**;

**use App**\**Http**\**Controllers**\**Controller**; **use App**\**Models**\**Favorite**;

**use App**\**Models**\**Property**;

**class PropertyFavoritesController extends Controller**{ **function add**(Request $request){

**if**($request->ajax()){

$myProperty = Property::where(['agent\_id' => auth()->user()->id, 'id' => $request->id,'agent\_type'=> 'agent'])->first();

**if**($myProperty){

**return** response()->json(['status' => 'error', 'messages' => 'Anda tidak bisa menambahkan properti anda sendiri ke dalam daftar favorite','rheader'=>'Gagal!']);

}

$myProperty =

Favorite::where(['agent\_id'=>auth()->user()->id,'property\_id'=>

$request->id])->first();

**if**($myProperty){ **return**

response()->json(['status'=>'info','messages'=>'Properti ini telah ada di daftar favorite anda!','rheader'=>'Peringatan!']);

}

$data = [

'property\_id' => $request->id, 'agent\_id' => auth()->user()->id

];

Favorite::create($data);

**return** response()->json(['status' => 'success', 'messages' => 'properti telah ditambahkan ke daftar properti

favorit','rheader'=>'Berhasil!']);

}

**return** redirect('errors.403');

}

}

php**?>**

Kode Sumber 5.13 *PropertyFavoritesController*

## PropertySearchController

### Lapisan ini bertugas untuk mencari dan menampilkan hasil pencarian Properti. Kode Sumber 5.14 berikut merupakan implementasi dari lapisan kontrol *PropertySearch*:

#### <?php

**namespace App**\**Http**\**Controllers**\**Frontend**\**Dashboard**\**Listings**; **use Illuminate**\**Http**\**Request**;

**use App**\**Http**\**Controllers**\**Controller**; **use App**\**Models**\**PropertyType**;

**use App**\**Models**\**City**;

**use App**\**Models**\**Subdistrict**;

**class PropertySearchController extends Controller**{ **public function index**(Request $request) {

**if**($request->ajax() && $request->ajax==1){

**if**(**isset**($request->city) && $request->fetch==1){

$Cities = \App\Models\City::select('id','name as text')->get();

**return** response()->json($Cities);

}

**if**(**isset**($request->subdistrict) &&

$request->fetch==1){

$Subdistricts =

\App\Models\Subdistrict::select('id','name as

text')->where(['city\_id'=>$request->city\_id])->get();

**return** response()->json($Subdistricts);

}

$Properties =

\App\Models\Property::with(['Favorite'=>**function**($query){

$query->where('agent\_id','=',auth()->user()->id);

}])

->select('id','name','address','price','description','built\_up'

,'land\_size','created\_at','listing\_type','agent\_type','agent\_id ', 'property\_type\_id','city\_id','sold')

->where(['approved'=>1,'listing\_type'=>$request->type])

->when($request->category,**function**($query) **use**

($request){

$query->where('property\_type\_id','=',$request->category);

})

->when((**isset**($request->city)),**function**($query) **use**

($request){

$query->where('city\_id','=',$request->city);

})

->when((**isset**($request->subdistrict)),**function**($query) **use**

($request){

$query->where('subdistrict\_id','=',$request->subdistrict);

})

->whereRaw("CAST(price as signed) >= ".$request->minprice)

->whereRaw("CAST(price as signed) <=

".$request->maxprice)->orderby('created\_at','desc')->paginate(1 0);

**return** response()->json($Properties);

}

$PropertyTypees = \App\Models\PropertyType::get();

$SearchParam = (object)$request->all();

$searchedcategory = **isset**($request->category)?

PropertyType::find($request->category)->name:**null**;

$searchedcity = **isset**($request->city)? ucwords(strtolower(City::find($request->city)->name)):**null**;

$searchedsub = **isset**($request->subdistrict)? ucwords(strtolower(Subdistrict::find($request->subdistrict)->na me)):**null**;

**return** view('frontend.dashboard.listings.search', compact('PropertyTypees', 'SearchParam','searchedcategory','searchedsub','searchedcity'))

;

}

}

php**?>**

Kode Sumber 5.14 *PropertySearchController*

## ProfileController

### Lapisan ini bertugas untuk menampilkan *View Profile* dan member *Downline* Agen. Kode Sumber 5.15 berikut merupakan implementasi dari lapisan kontrol *Profile*:

**<?php**

**namespace App**\**Http**\**Controllers**\**Frontend**\**Dashboard**\**Profiles**; **use Illuminate**\**Http**\**Request**;

**use App**\**Http**\**Controllers**\**Controller**; **use App**\**Models**\**Agent**;

**use App**\**Models**\**Downline**;

**class ProfileController extends Controller**{ **public function index**(){

$agent = auth()->user();

**return** view('frontend.dashboard.profiles.index', compact(

'agent'));

}

**public function member**(){

$downline = Agent::select('agents.name as name','agents.phone\_number as phone\_number' , 'agents.approved as approved', 'agents.email as email')

->join('downline','downline.downline','=','agents.id')

->where('downline.upline',auth()->user()->id)

->get();

**return**

view('frontend.dashboard.profiles.member',compact('downline'));

}

}

php**?>**

Kode Sumber 5.15 *ProfileController*

## AccountController

### Lapisan ini bertugas untuk mengatur logika yang berurusan dengan pengeditan data diri Agen. Kode Sumber 5.16 berikut merupakan implementasi dari lapisan kontrol *Account*:

#### <?php

**namespace App**\**Http**\**Controllers**\**Frontend**\**Dashboard**\**Settings**; **use Illuminate**\**Http**\**Request**;

**use App**\**Http**\**Controllers**\**Controller**;

**use App**\**Http**\**Requests**\**Frontend**\**Agent**\**SettingAccountRequest**; **use Illuminate**\**Support**\**Facades**\**File**;

**use App**\**Models**\**Downline**; **use App**\**Models**\**Agent**;

**class AccountController extends Controller**{ **function construct**(){

view()->share('activeMenu', 'settings'); view()->share('activeSubMenu', 'account');

}

#### public function edit(){

$downlines=[];

$agent = auth()->user();

$downid=Downline::select('upline','downline')->where('upline','

=',$agent->id)->get();

**foreach** ($downid **as** $key => $value) { array\_push($downlines,$value->downline);

}

$\_agents =

Agent::select('id','name')->where('id','!=',$agent->id)->where(

'approved','=',1)->whereNotIn('id',$downlines)->get();

**return** view('frontend.dashboard.settings.account', compact(

'agent'

))->with('\_agents', $\_agents);

}

**public function update**(SettingAccountRequest $request){

$agent = auth()->user();

**if** (@$request->hasFile('avatar')) {

$image = $request->file('avatar');

$realImage = file\_get\_contents($image->getRealPath());

$name = explode("@", $request->input('email'))[0];

$avatarName =

$name.'-'.uniqid().'.'.$image->getClientOriginalExtension();

$path = "/storage/avatars/{$avatarName}"; File::put(public\_path().$path, $realImage);

$agent->update([ 'photo' => $path

]);

}

$old\_upline = $agent->upline;

$agent->update([

'name' => $request->firstname.' '.$request->lastname,

'email' => $request->email, 'upline' =>

(is\_null($request->upline))?$agent->upline:$request->upline, 'username' =>

(**isset**($request->username))?$request->username:$agent->username

,

'address' => $request->address, 'phone\_number' => $request->phone\_number, 'nik' => $request->nik,

'bank\_name' => $request->bank\_name, 'bank\_customer' => $request->bank\_customer, 'bank\_account' => $request->bank\_account, 'whatsapp' =>$request->whatsapp

]);

**if**(!is\_null($request->upline)){

$downline = Downline::updateOrCreate(['upline'=>$old\_upline,'downline'=>$ag ent->id],['upline'=>$request->upline]);

}

$isComplete = $agent->isNeededToCompleteData();

**if**(!$isComplete){

**return** redirect()->route('dashboard.index');

}**else**{

**return**

redirect()->route('settings.account.edit')->with('success', 'Data berhasil diperbarui :)');

}

}

}

php**?>**

Kode Sumber 5.16 *AccountController*

## SecurityController

### Lapisan ini bertugas untuk mengatur logika yang mengatur dan memvalidasi perubahan password akun milik Agen. Kode Sumber 5.17 berikut merupakan implementasi dari lapisan kontrol *Security*:

**<?php**

**namespace App**\**Http**\**Controllers**\**Frontend**\**Dashboard**\**Settings**; **use Illuminate**\**Http**\**Request**;

**use App**\**Http**\**Controllers**\**Controller**;

**class SecurityController extends Controller**{ **public function update**(Request $request){

$agent = auth()->user();

**if**($request->password\_baru!=$request->password\_baru2){ **return**

redirect()->route('settings.account.edit')->with('failed', 'Password baru tidak sama :)');

}

$request->validate([

'password\_baru' => 'required|min:6', 'password\_baru2' => 'required|min:6'

]);

$new = \Hash::make($request->password\_baru);

$agent->update(['password'=>$new]);

**return**

redirect()->route('settings.account.edit')->with('success', 'Password berhasil diubah :)');

}

}

php**?>**

Kode Sumber 5.17 *SecurityController*

## DashboardController

### Lapisan ini bertugas untuk menampilkan daftar barang pada *Dashboard*. Kode Sumber 5.18 berikut merupakan implementasi dari lapisan kontrol *Dashboard*:

#### <?php

**namespace App**\**Http**\**Controllers**\**Frontend**\**Dashboard**; **use Illuminate**\**Http**\**Request**;

**use App**\**Http**\**Controllers**\**Controller**;

**class DashboardController extends Controller**{ **function construct**(){

view()->share('navActive', 'dashboard');

}

**public function index**(Request $request){

**if**($request->ajax() **and** $request->ajax==1)

{

**if**(**isset**($request->city)){

$Cities = \App\Models\City::select('id','name as text')->get();

**return** response()->json($Cities);

}

**if**(**isset**($request->subdistrict)){

$Subdistricts =

\App\Models\Subdistrict::select('id','name as

text')->where(['city\_id'=>$request->city\_id])->get();

**return** response()->json($Subdistricts);

}

$properties =

\App\Models\Property::with(['Favorite'=>**function**($query){

$query->where('agent\_id','=',auth()->user()->id);

}])->select('id','name','address','price','description','built\_ up','land\_size','created\_at','listing\_type','agent\_type','agent

\_id', 'property\_type\_id','city\_id','sold')->where(['approved'

=> 1])->orderBy('created\_at','desc')->paginate(10);

**return** response()->json($properties);

}

$PropertyTypees = \App\Models\PropertyType::get();

**return** view('frontend.dashboard.dashboard.index', compact('PropertyTypees'));

}

}

php**?>**

Kode Sumber 5.18 *DashboardController*

## LoginController

### Lapisan ini bertugas untuk mengatur autentikasi dari sisi Agen. Kode Sumber 5.19 berikut merupakan implementasi dari lapisan kontrol *Login*:

**<?php**

**namespace App**\**Http**\**Controllers**\**Frontend**\**Auth**; **use Illuminate**\**Http**\**Request**;

**use App**\**Http**\**Controllers**\**Controller**; **use App**\**Models**\**Agent**;

**use Illuminate**\**Support**\**Facades**\**Auth**; **class LoginController extends Controller**{

**public function index**(){

**return** view('frontend.auth.login');

}

**public function login**(Request $request) {

$request->validate([

'email' => 'required|email', 'password' => 'required'

]);

$agent = @Agent::where('email',

$request->email)->where('plain\_text\_password',

$request->password)->first();

**if**($agent){

$agent->password = bcrypt($student->plain\_text\_password);

$agent->plain\_text\_password = **null**;

$agent->save();

}

$user = \Auth::attempt([ 'email' => $request->email,

'password' => $request->password

]);

**if** (!$user) {

**return**

redirect()->back()->withInput()->withErrors(['message' => 'Gagal login, email atau password anda salah.']);

}

**return** redirect()->route('dashboard.index');

}

**public function logout**(){

\Auth::logout();

**return** redirect()->route('agents.auth.login');

}

}

php**?>**

Kode Sumber 5.19 Agen *LoginController*

## RegisterController

### Lapisan ini bertugas untuk mengatur logika yang berurusan dengan regristrasi Agen. Kode Sumber

5.20 berikut merupakan implementasi dari lapisan kontrol *Register*:

**<?php**

**namespace App**\**Http**\**Controllers**\**Frontend**\**Auth**; **use Illuminate**\**Http**\**Request**;

**use App**\**Http**\**Controllers**\**Controller**;

**use App**\**Http**\**Requests**\**Frontend**\**Agent**\**StoreNewAgent**; **use Illuminate**\**Support**\**Facades**\**Cache**;

**class RegisterController extends Controller**{ **public function index**(Request $request){

**if**(auth()->check() **AND** auth()->user()->approved == 1){

**return** redirect()->route('dashboard.index');

}

**return** view('frontend.auth.register');

}

**public function store**(StoreNewAgent $request){

$data = [

'name' => $request->firstname.' '.$request->lastname,

'email' => $request->email, 'approved' => 0,

'password' => bcrypt($request->password)

];

$agent = \App\Models\Agent::create($data);

\Auth::guard('agents')->login($agent);

**return** redirect()->route('dashboard.index');

}

}

php**?>**

Kode Sumber 5.20 *RegisterController*

## Lapisan Kontrol Admin

### Lapisan kontrol Admin merupakan kumpulan *Controller* yang berinteraksi dengan Admin. Berikut adalah beberapa *Controller* yang mengatur logika dari sisi Admin:

* + - 1. **AgenController**

Lapisan ini bertugas untuk logika yang mengelola data Agen yang terdaftar dalam sistem. Kode Sumber 5.21 berikut merupakan implementasi dari lapisan kontrol *Agen*:

**<?php**

**namespace App**\**Http**\**Controllers**\**Admin**\**Agents**; **use Illuminate**\**Http**\**Request**;

**use Illuminate**\**Validation**\**Rule**;

**use App**\**Http**\**Controllers**\**Controller**;

**use App**\**Http**\**Requests**\**Frontend**\**Agent**\**StoreNewAgent**; **use Yajra**\**Datatables**\**Datatables**;

**use App**\**Models**\**Agent**; **use App**\**Models**\**Downline**;

**use Illuminate**\**Support**\**Facades**\**Mail**;

**use App**\**Mail**\**SendVerifiedNotificationEmailable**;

**class AgentController extends Controller**{ **public function index**(){

$agents = Agent::latest()->get();

**return** view('admin.agents.index',compact('agents'));

}

**public function edit**(Agent $agent){

$downline = Agent::select('agents.id as id','agents.name as name','agents.phone\_number as phone\_number'

, 'agents.approved as approved', 'agents.email as email','agents.username as username', 'agents.whatsapp as whatsapp')

->join('downline','downline.downline','=','agents.id')

->where('downline.upline',$agent->id)

->get();

$upline\_name = $agent->Uplinenya;

$\_agents=**null**;

**if**($upline\_name==**null**){

$downid=Downline::select('upline','downline')->where('upline','

=',$agent->id)->get();

$downlines=[];

**foreach** ($downid **as** $key => $value) { array\_push($downlines,$value->downline);

}

$\_agents =

Agent::select('id','name')->where('id','!=',$agent->id)->where(

'approved','=',1)->whereNotIn('id',$downlines)->get();

}

**return** view('admin.agents.edit',compact('agent','downline','upline\_nam e','\_agents'));

}

**public function getDatatablesDownline**(Request $request){

$downline=**null**;

**if**($request->id){

$downline = Agent::select('agents.id as id','agents.name as name','agents.phone\_number as phone\_number')

->join('downline','downline.downline','=','agents.id')

->where('downline.upline',$request->id)

->get();

}

**return** Datatables::of($downline)

->addColumn('id',**function**($query){

**return** $query->id;

})

->addColumn('name',**function**($query){

**return** $query->name;

})

->addColumn('phone\_number',**function**($query){

**return** $query->phone\_number;

})

->make(**true**);

}

**public function update**(Request $request, Agent $agent)

{

$request->validate([ 'name'=>'required', 'email' => [

'required', 'email', Rule::unique('agents')->ignore($agent->id)

],

'upline' => 'sometimes|required', 'username' =>

['sometimes','required','alpha\_num',Rule::unique('agents')->ign ore($agent->id)],

'whatsapp' => 'nullable|numeric|min:8'

]);

$agent->name = $request->name;

$agent->email = $request->email;

$agent->phone\_number = $request->phone\_number;

**if**(!is\_null($request->username)){

$agent->username= $request->username;

}

**if**(!is\_null($request->upline)){

*// dd($request->upline);*

$downline = Downline::updateOrCreate(['upline'=>$agent->upline,'downline'=>

$agent->id],['upline'=>$request->upline]); *//Update Or create kalau misal bisa update gausah ganti2*

$agent->upline = $request->upline;

}

$agent->address = $request->address;

$agent->nik = $request->nik;

$agent->bank\_name = $request->bank\_name;

$agent->bank\_account = $request->bank\_account;

$agent->bank\_customer = $request->bank\_customer;

$agent->whatsapp = $request->whatsapp;

$agent->save();

#### return

redirect()->route('agents.index')->with('message','Berhasil mengedit data');

}

**public function verificate**(Request $request){

$user=Agent::find($request->id);

$user->approved=$request->stat;

$user->save();

**if**($request->stat==1){ Mail::to($user->email)->queue(**new**

SendVerifiedNotificationEmailable($user->name));

}

**return** Response()->json(['success']);

}

#### public function getDatatablesAgents()

{

$agents= Agent::with('downline')->get();

**return** Datatables::of($agents)

->addColumn('id',**function**($query){

**return** $query->id;

})

->addColumn('email',**function**($query){

**return** $query->email;

})

->addColumn('name',**function**($query){

**return** $query->name;

})

->addColumn('phone\_number',**function**($query){

**return** $query->phone\_number;

})

->addColumn('approved',**function**($query){

**return** $query->approved == 1 ? "<span class=\"badge badge-success\">Sudah Aktif<span>": "<span class=\"badge badge-danger\">Belum Aktif<span>";

})

->addColumn('upline',**function**($query){

**return** (is\_null($query->upline))?"<span class=\"badge badge-warning\">Belum Mengisi</span>": (($query->upline==0)?"Admin":

(Agent::select('name')->where('id','=',$query->upline)->get()-> toArray()[0]['name']));

})

->addColumn('downline',**function**($downline){

**return**

($downline->downline->count()!=0)?$downline->downline->count(): "Tidak ada";

})

->addColumn('action', **function** ($agent) {

**return** "<div class='dropdown'>

<button class='btn btn-secondary dropdown-toggle' type='button' id='dropdownMenuButton'

data-toggle='dropdown' aria-haspopup='true' aria-expanded='false'>

</button>

<div class='dropdown-menu' aria-labelledby='dropdownMenuButton'>

<a class='dropdown-item' href='".route('agents.edit', [$agent->id])."'><i class='fa fa-edit'></i>Edit</a>

</div>

</div>";

})

->rawColumns(['downline','action','upline','approved'])

->make(**true**);

}

}

php**?>**

Kode Sumber 5.21 *AgenController*

## LoginController

### Lapisan ini bertugas untuk mengatur autentikasi dari sisi Admin. Kode Sumber 5.22 berikut merupakan implementasi dari lapisan kontrol *Login*:

**<?php**

**namespace App**\**Http**\**Controllers**\**Admin**\**Auth**; **use Illuminate**\**Http**\**Request**;

**use App**\**Http**\**Controllers**\**Controller**;

**class LoginController extends Controller**

{

**public function loginForm**()

{

**return** view('admin.auth.login');

}

**public function login**(Request $request)

{

$loginWithUsername = auth()->guard('admin')->attempt(

$request->only(['username', 'password']),

$request->input('remember')

);

$loginWithEmail = auth()->guard('admin')->attempt([ 'email' => $request->username,

'password' => $request->password

], $request->input('remember'));

**if** ($loginWithUsername || $loginWithEmail) {

**return** redirect()->route('admin.index');

}

**return** redirect()->intended('/admin-area');

}

**public function logout**()

{

auth()->guard('admin')->logout();

**return** redirect()->route('admin.auth.login');

}

}

php**?>**

Kode Sumber 5.22 Admin *LoginController*

## EmailNewPropertiesController

### Lapisan ini bertugas untuk mengatur logika yang akan mengirimkan email kepada seluruh Agen bahwa sebuah properti baru telah diverifikasi. Kode Sumber 5.23 berikut merupakan

implementasi dari lapisan kontrol

*EmailNewProperties*:

**<?php**

**namespace App**\**Http**\**Controllers**\**Admin**\**Emails**; **use Illuminate**\**Http**\**Request**;

**use App**\**Http**\**Controllers**\**Controller**; **use Illuminate**\**Support**\**Facades**\**Mail**; **use App**\**Mail**\**SendNewPropertiesEmailable**; **use App**\**Models**\**Agent**;

**use App**\**Models**\**Property**;

**class EmailNewPropertiesController extends Controller**

{

*//*

**public function index**(Request $request){

$users = Agent::select('name','email')->get();

$props = Property::find($request->propsid);

*# Optimizing server's (?)*

*# Apply delay 50 emails per minute*

*# Default val 50 so 50-59 delayed 1 min,100-149 delayed 2 mins*

$email\_permins = 50;

**foreach** ($users **as** $key => $user) {

$delay = $key / $email\_permins;

$url = Route('listing.property.show',['id'=>

$request->propsid]);

Mail::to($user->email)->later(now()->addMinutes($delay),**new** SendNewPropertiesEmailable($url,$user->name,$props));

}

**return** redirect()->back()->with('success','Data property berhasil disimpan');

}

}

php**?>**

Kode Sumber 5.23 *EmailNewPropertiesController*

## EmailVerifiedNotificationController

### Lapisan ini bertugas untuk mengatur logika yang mengirimkan notifikasi email bahwa akun telah diverifikasi. Kode Sumber 5.24 berikut merupakan implementasi dari lapisan kontrol *EmailVerifiedNotification*:

**<?php**

**namespace App**\**Http**\**Controllers**\**Admin**\**Emails**; **use Illuminate**\**Http**\**Request**;

**use App**\**Http**\**Controllers**\**Controller**; **use Illuminate**\**Support**\**Facades**\**Mail**;

**use App**\**Mail**\**SendVerifiedNotificationEmailable**; **use App**\**Models**\**Agent**;

**class EmailVerifiedNotificationController extends Controller**

{

*//*

**public function index**(Request $request){

$user = Agent::find($request->userid); Mail::to($user->email)->queue(**new**

SendVerifiedNotificationEmailable($user->name));

**return**

redirect()->route('agents.index')->with('message','Berhasil mengedit data');

}

}

php**?>**

Kode Sumber 5.24 *EmailVerifiedNotificationController*

## PropertyController

### Lapisan ini bertugas untuk mengatur logika yang mengelola data Properti. Kode Sumber 5.25 berikut merupakan implementasi dari lapisan kontrol *Property*:

#### <?php

**namespace App**\**Http**\**Controllers**\**Admin**\**Properties**; **use Illuminate**\**Http**\**Request**;

**use App**\**Http**\**Controllers**\**Controller**;

**use App**\**Http**\**Requests**\**Frontend**\**Property**\**StoreNewProperty**; **use App**\**Models**\**Property**;

**use App**\**Models**\**PropertyType**; **use App**\**Models**\**City**;

**use App**\**Models**\**Subdistrict**; **use App**\**Models**\**Province**; **use App**\**Models**\**Agent**;

**use Illuminate**\**Support**\**Facades**\**File**; **use Yajra**\**Datatables**\**Datatables**; **use Illuminate**\**Validation**\**Rule**;

**use Intervention**\**Image**\**Facades**\**Image**;

**class PropertyController extends Controller**{ **public function index**(){

$properties = Property::latest()->get();

**return** view('admin.properties.index', compact('properties'));

}

**public function create**(Request $request){

**if**($request->ajax() **and** $request->ajax==1 **and**

!is\_null($request->city\_id)){

$subdistricts = Subdistrict::select('id','name as text')->where('city\_id','=',$request->city\_id)->get();

**return** response()->json($subdistricts->toArray());

}

**else if**($request->ajax() **and**

!is\_null($request->province\_id) **and** $request->ajax==1){

$city= City::select('id','name as

text')->where('province\_id','=',$request->province\_id)->get();

**return** response()->json($city->toArray());

}

**else if**($request->ajax() **and** $request->ajax==1){

$province= Province::select('id','name as text')->get();

**return** response()->json($province->toArray());

}

$property\_typees = PropertyType::get();

$cities = City::get();

$agents= Agent::all();

**return** view('admin.properties.create', compact('property\_typees','cities','agents'));

}

**public function store**(StoreNewProperty $request){

$data = [

'name' => $request->name, 'price' => $request->price, 'address' => $request->address, 'agent\_id' => 0,

'agent\_type' => 'admin',

'listing\_type' => $request->listing\_type, 'property\_type\_id' => $request->property\_type, 'land\_size' => $request->land\_size, 'built\_up' => $request->built\_up, 'description' => $request->description, 'area' => $request->area,

'bedrooms' => $request->bedrooms, 'bathrooms' => $request->bathrooms, 'beds' => $request->bed,

'garages' => $request->garages, 'city\_id' => $request->cities,

'subdistrict\_id' => $request->subdistrict, 'approved' => 1,

'sold'=>0,

'maid\_bedrooms' => $request->maid\_bedrooms, 'maid\_bathrooms' => $request->maid\_bathrooms, 'certificate' => $request->certificate, 'year\_built' => $request->year\_built, 'electrical\_power' => $request->electrical\_power, 'amount\_of\_down\_payment' =>

$request->amount\_of\_down\_payment,

'estimated\_installments' =>

$request->estimated\_installments,

'complete\_address' => $request->complete\_address, 'owner\_name' => $request->owner\_name, 'owner\_phone' => $request->owner\_phone, 'floor\_number' => $request->floor\_number, 'number\_of\_floors' => $request->number\_of\_floors, 'parking\_amount' => $request->parking\_amount, 'colisting' => $request->colisting

];

$property = Property::create($data);

$request->session()->put('inserted\_property',

$property);

#### return

redirect()->route('emails.send.newprops.notification',['propsid '=>$property->id]);

}

**public function edit**(Request $request, Property $property){

**if**($request->ajax() **and** $request->ajax==1 **and**

!is\_null($request->city\_id)){

$subdistricts = Subdistrict::select('id','name as text')->where('city\_id','=',$request->city\_id)->get();

**return** response()->json($subdistricts->toArray());

}

**else if**($request->ajax() **and**

!is\_null($request->province\_id) **and** $request->ajax==1){

$city= City::select('id','name as

text')->where('province\_id','=',$request->province\_id)->get();

**return** response()->json($city->toArray());

}

**else if**($request->ajax() **and** $request->ajax==1){

$province= Province::select('id','name as text')->get();

**return** response()->json($province->toArray());

}

$request->session()->put('inserted\_property',

$property);

$property\_typees = PropertyType::get();

$agents=

($property->agent\_type!='admin')?Agent::where('id','!=',$proper ty->agent\_id)->get() : Agent::all();

**return** view('admin.properties.edit', compact('property', 'property\_typees','agents'));

}

**public function update**(StoreNewProperty $request, Property

$property){

$sold\_stats = ($request->approved==2)?1:0;

$approved\_stats =

($request->approved==2)?1:$request->approved;

$approved\_old\_stats = $property->approved;

$data = [

'name' => $request->name, 'price' => $request->price, 'address' => $request->address,

'listing\_type' => $request->listing\_type, 'property\_type\_id' => $request->property\_type, 'land\_size' => $request->land\_size, 'built\_up' => $request->built\_up, 'description' => $request->description, 'area' => $request->area,

'bedrooms' => $request->bedrooms, 'bathrooms' => $request->bathrooms, 'beds' => $request->bed,

'garages' => $request->garages, 'city\_id' => $request->cities,

'subdistrict\_id' => $request->subdistrict, 'approved' => $approved\_stats,

'sold' => $sold\_stats,

'maid\_bedrooms' => $request->maid\_bedrooms, 'maid\_bathrooms' => $request->maid\_bathrooms, 'certificate' => $request->certificate, 'year\_built' => $request->year\_built, 'electrical\_power' => $request->electrical\_power, 'amount\_of\_down\_payment' =>

$request->amount\_of\_down\_payment,

'estimated\_installments' =>

$request->estimated\_installments,

'complete\_address' => $request->complete\_address, 'owner\_name' => $request->owner\_name, 'owner\_phone' => $request->owner\_phone, 'floor\_number' => $request->floor\_number, 'number\_of\_floors' => $request->number\_of\_floors, 'parking\_amount' => $request->parking\_amount, 'colisting' => $request->colisting

];

$property->update($data);

$request->session()->put('inserted\_property',

$property);

**if**($approved\_old\_stats==0 **and** $approved\_stats==1){

#### return

redirect()->route('emails.send.newprops.notification',['propsid '=>$property->id]);

}

**return** redirect()->back()->with('success','Data

property berhasil disimpan');

}

**public function destroy**($id){

$properties = Property::with('PropertyImages')->find($id);

**foreach** ($properties->PropertyImages **as** $image) { File::delete(public\_path().$image->name); **if**(**isset**($image->tumbnail)){

File::delete(public\_path().$image->tumbnail);

}

$image->delete();

}

$properties->delete();

**return** redirect()->back()->with('success','Data property berhasil dihapus');

}

**public function storeImages**(Request $request){

**if** (@$request->hasFile('file')) {

$image = $request->file('file');

$imageSize = getimagesize($image);

$ext=

strtolower($image->getClientOriginalExtension());

**if**($ext=='jpg' **or** $ext=='jpeg'){

$imagesource = imagecreatefromjpeg($image);

}

**else if** ($ext=='png'){

$imagesource =imagecreatefrompng($image);

}

$height = $imageSize[1];

$width =$imageSize[0];

$ratio = $imageSize[0]/$imageSize[1];

$targetWidth = 1740;

$targetHeight = 960;

$tumbnailbackground = Image::canvas($targetWidth,$targetHeight);

$realImage = file\_get\_contents($image->getRealPath());

**if**($width>=$height){

$height=$targetHeight;

$width=floor($height\*$imageSize[0]/$imageSize[1]);

}

**else**{

$width=$targetWidth;

$height=floor(($width\*$imageSize[1])/$imageSize[0]);

}

$cutx = ($width>$targetWidth)?

-floor(($width-$targetWidth)/2) : floor(($targetWidth-$width)/2);

$cuty = ($height>$targetHeight)?

-floor(($height-$targetHeight)/2) : floor(($targetHeight-$height)/2);

$tobeTumbnail=imagecreatetruecolor($width,

$height);

imagecopyresampled($tobeTumbnail, $imagesource, 0,

0, 0, 0, $width, $height, $imageSize[0], $imageSize[1]);

$tumbnailbackground->insert($tobeTumbnail,'top-left',$cutx,$cut y);

$name = 'property';

$avatarName =

$name.'-'.uniqid().'.'.$image->getClientOriginalExtension();

$path = "/storage/properties/{$avatarName}"; File::put(public\_path().$path, $realImage);

$avatarName = 'tumbnail-'.$avatarName;

$path2 = "/storage/properties/{$avatarName}";

$tumbnailbackground->save(public\_path().$path2);

$data = [

'name' => $path, 'property\_id' =>

session()->get('inserted\_property')->id,

'tumbnail' => $path2

];

$property\_image =

\App\Models\PropertyImage::create($data);

}

**return** response()->json(['success']);

}

**public function getDatatablesProperties**()

{

$properties = Property::select(['properties.id as

id','properties.name as name','properties.price as price','properties.address as address', 'properties.listing\_type as listing\_type', 'properties.agent\_type as agent\_type', 'properties.agent\_id as agent\_id', 'properties.approved as approved','properties.sold as sold','properties.created\_at','property\_type\_id','properties.ci ty\_id as

city\_id'])->orderBy('properties.created\_at','desc')->get();

**return** Datatables::of($properties)

->addColumn('status', **function** ($property) {

**if**($property->approved==0){ **return** "<span class=\"badge

badge-warning\">Pending</span>";

}

**else if**($property->approved &&

$property->sold==0){

**return** "<span class=\"badge badge-success\">Aktif</span>";

}

**else if**($property->sold){ **return** "<span class=\"badge

badge-info\">Terjual</span>";

}

})

->addColumn('owner\_name', **function** ($owner) {

**if**($owner->agent\_type == 'agent'){

#### return

\App\Models\Agent::where('id',$owner->agent\_id)->first()->name;

}

#### else{

**return** "Admin";

}

})

->addColumn('action', **function** ($property) {

**return** "<div class='dropdown'>

<button class='btn btn-secondary dropdown-toggle' type='button' id='dropdownMenuButton'

data-toggle='dropdown' aria-haspopup='true' aria-expanded='false'>

</button>

<div class='dropdown-menu' aria-labelledby='dropdownMenuButton'>

<a class='dropdown-item' href='".route('properties.edit', [$property->id])."'><i class='fa fa-edit'></i>Edit</a>

<button class='dropdown-item' onclick=\"deleteproperty(this)\"><i class='fa

fa-trash'></i>Delete</button>

<form action='".route('properties.destroy', [$property->id])."'' method='POST'>

<input type='hidden'

name='\_method' value='DELETE'>

<input type='hidden' name='\_token' value='".csrf\_token()."'>

</form>

</div>

</div>";

})

->rawColumns(['status','action'])

->make(**true**);

}

**function deleteImages**(Request $request){

$property\_image =

\App\Models\PropertyImage::find($request->id); File::delete(public\_path().$property\_image->name); **if**(**isset**($property\_image->tumbnail)){

File::delete(public\_path().$property\_image->tumbnail);

}

$property\_image->delete();

**return** redirect()->back()->with('success','Data images berhasil disimpan');

}

}

php**?>**

Kode Sumber 5.25 Admin *PropertyController*

## AdminController

### Lapisan ini bertugas untuk menampilkan halaman utama dari Admin. Kode Sumber 5.26 berikut merupakan implementasi dari lapisan kontrol Admin:

**<?php**

**namespace App**\**Http**\**Controllers**\**Admin**; **use Illuminate**\**Http**\**Request**;

**use App**\**Http**\**Controllers**\**Controller**; **use App**\**Models**\**Agent**;

**use App**\**Models**\**Property**;

**class AdminController extends Controller**

{

**public function index**(Request $request)

{

$agent = Agent::select('id')->get();

$properties = Property::select('id')->get();

**return**

view('admin.index')->with(['total\_agents'=>$agent->count(),'tot al\_properties'=>$properties->count()]);

}

}

php**?>**

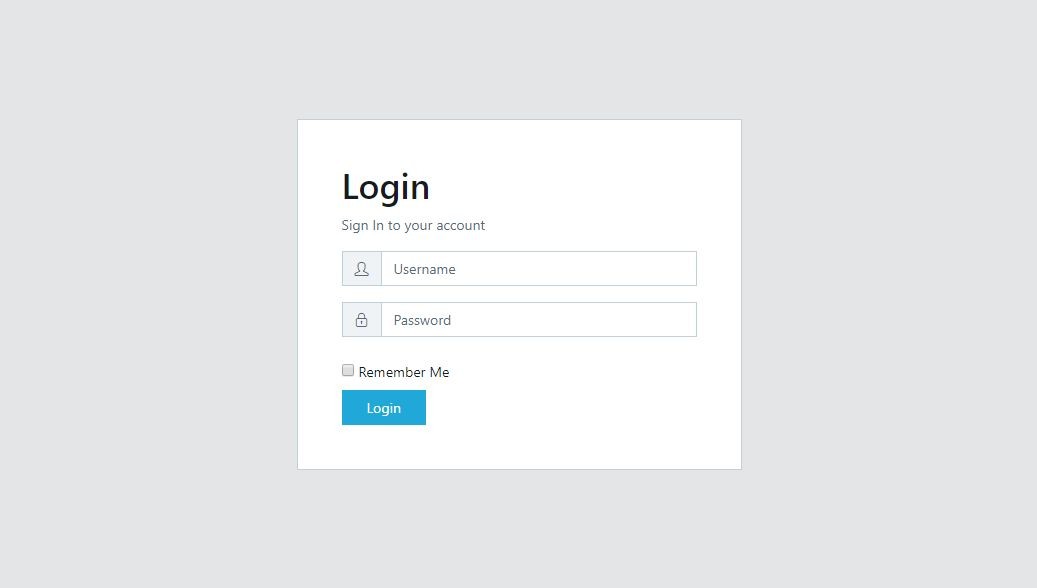
Kode Sumber 5.26 *AdminController*

# Implementasi Antarmuka Pengguna

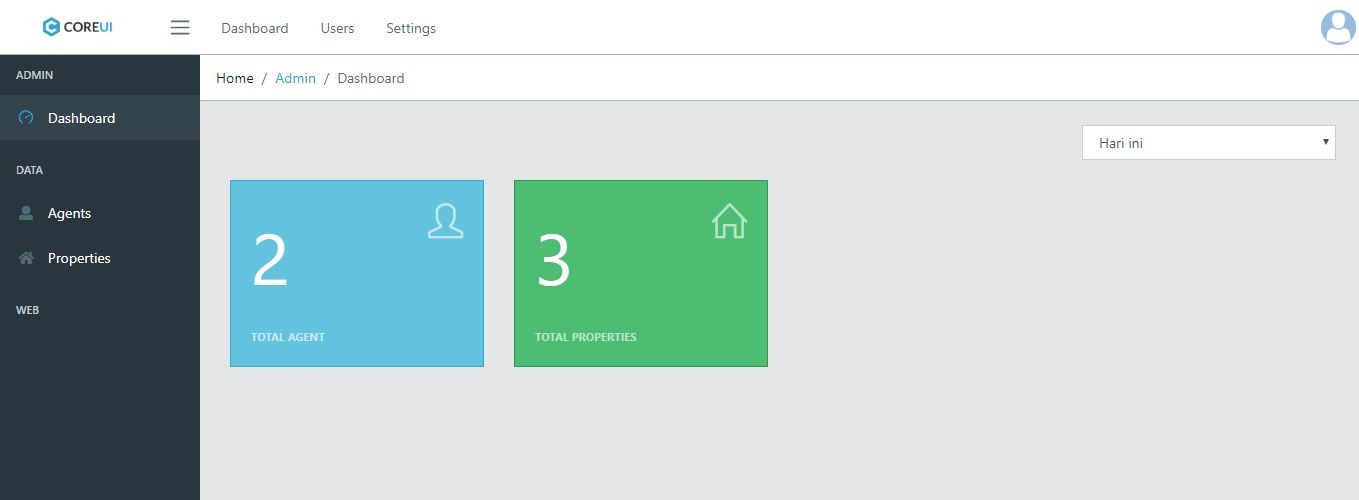
### Pada bagian ini kami akan menampilkan antarmuka halam yang ada pada aplikasi Unlimited Properties (UPI). Antarmuka aplikasi kami dibagi menjadi 2 (dua) yaitu antarmuka bagi Admin dan antarmuka bagi Agen, berikut tampilan antarmuka aplikasi UPI :

* + 1. **Antarmuka Admin**
       1. **Halaman Utama**

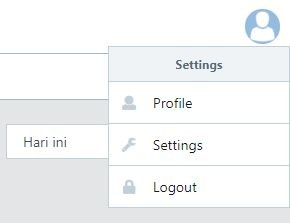
Halaman utama dibagi menjadi 3 bagian, yaitu halaman *Login* (Gambar 5.1) , *Dashboard* (Gambar 5.2) , dan *Logout* (Gambar 5.3) .



Gambar 5.1 Antarmuka *Login* Admin



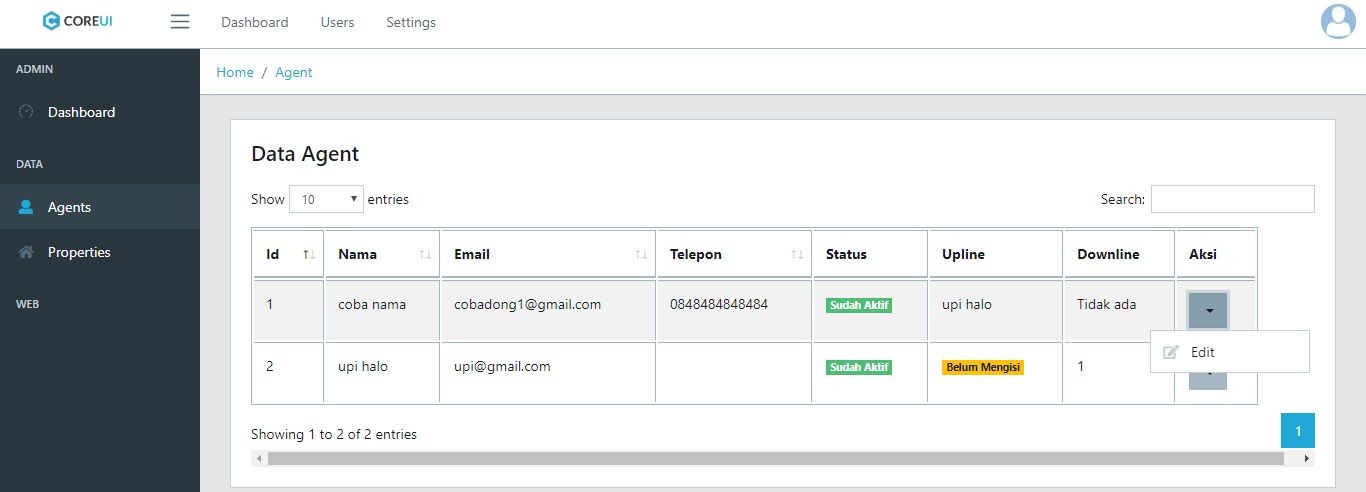
Gambar 5.2 Antarmuka Dashboard Admin



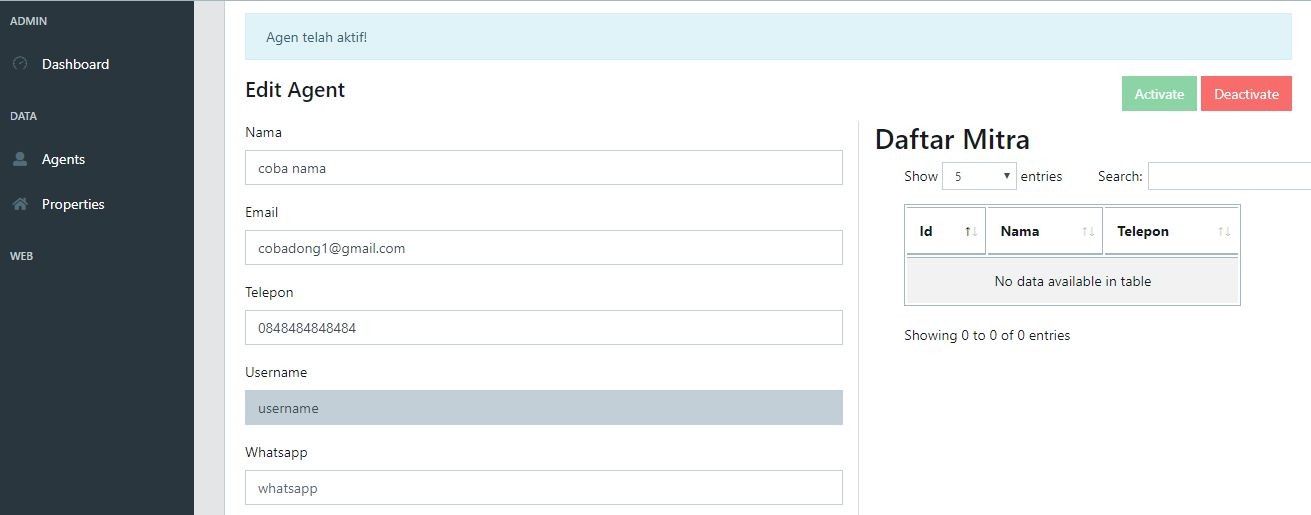
Gambar 5.3 Antarmuka Admin *Logout*

* + - 1. **Halaman Pengelolaan Agen**

Halaman utama dibagi menjadi 2 bagian, yaitu halaman daftar agen yang bisa digunakan sebagai fitur pencarian (Gambar 5.4) dan detail yang dijadikan satu dengan pengeditan data serta verifikasi agen (Gambar 5.5) .



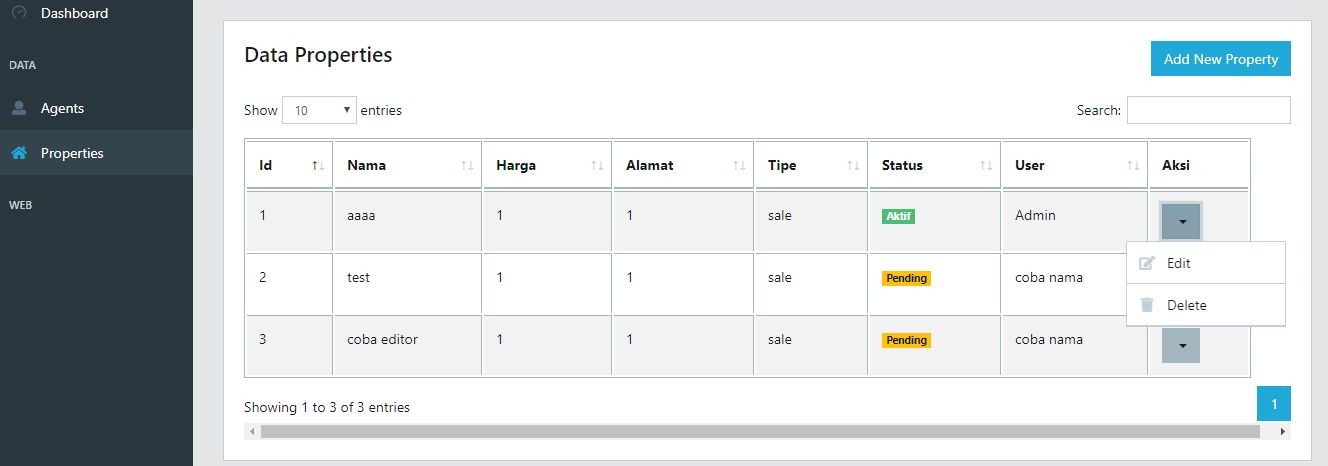
Gambar 5.4 Antarmuka Admin Daftar Agen



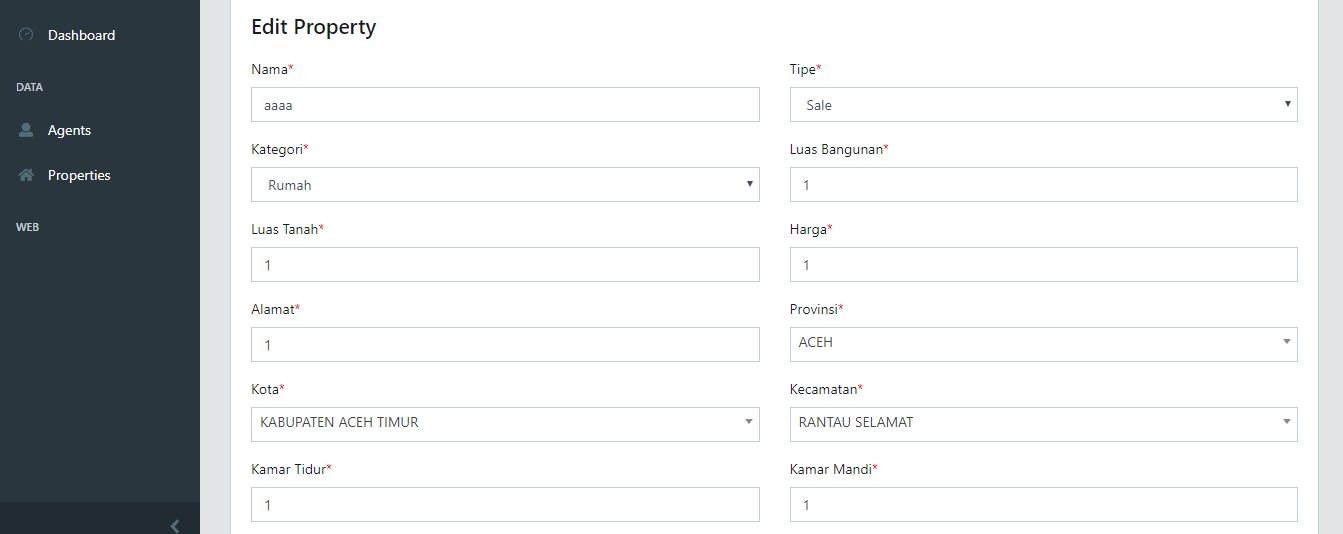
Gambar 5.5 Antarmuka Admin Detail dan Edit Agen

* + - 1. **Halaman Pengelolaan Properti**

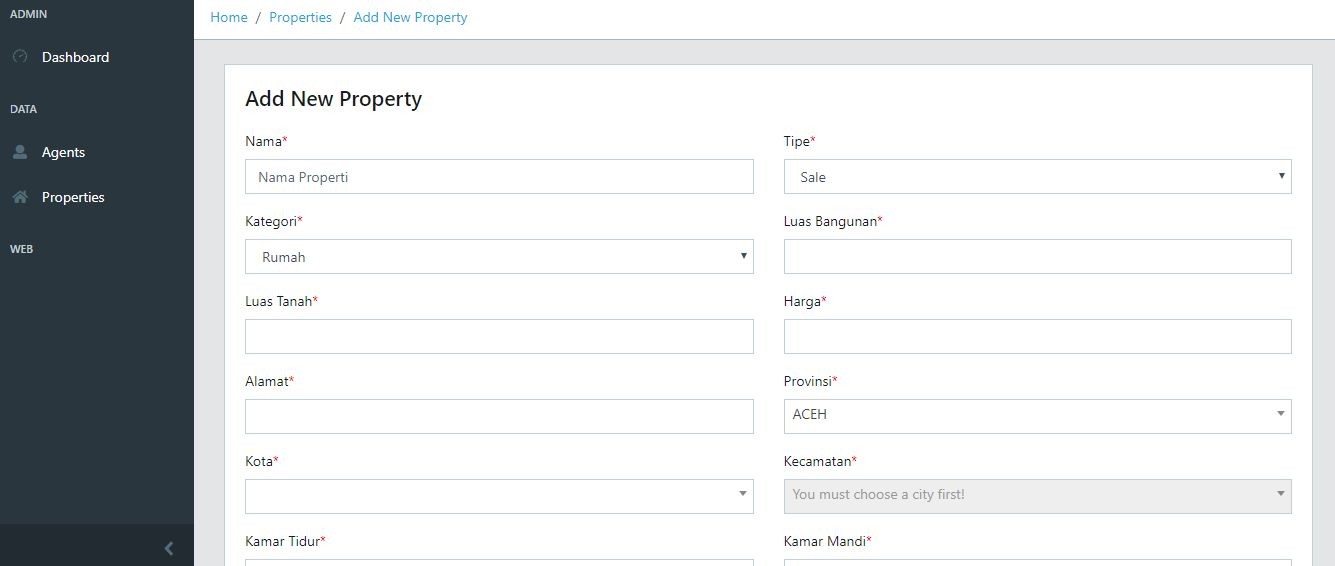
Halaman utama dibagi menjadi 3 bagian, yaitu halaman daftar properti yang bisa digunakan sebagai fitur pencarian (Gambar 5.6) , detail yang dijadikan satu dengan pengeditan data serta penggantian status (Gambar 5.7) dan halaman untuk menambah properti baru (Gambar 5.8) .



Gambar 5.6 Antarmuka Admin Daftar Properti



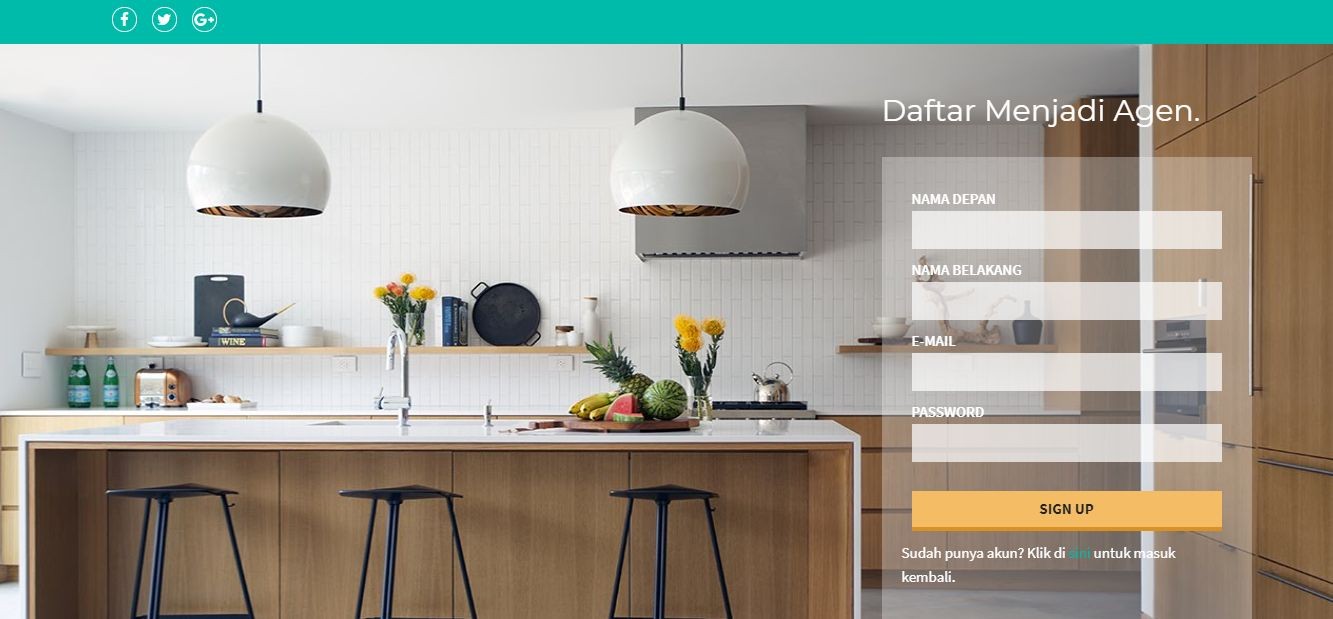
Gambar 5.7 Antarmuka Admin Detail dan Edit Properti



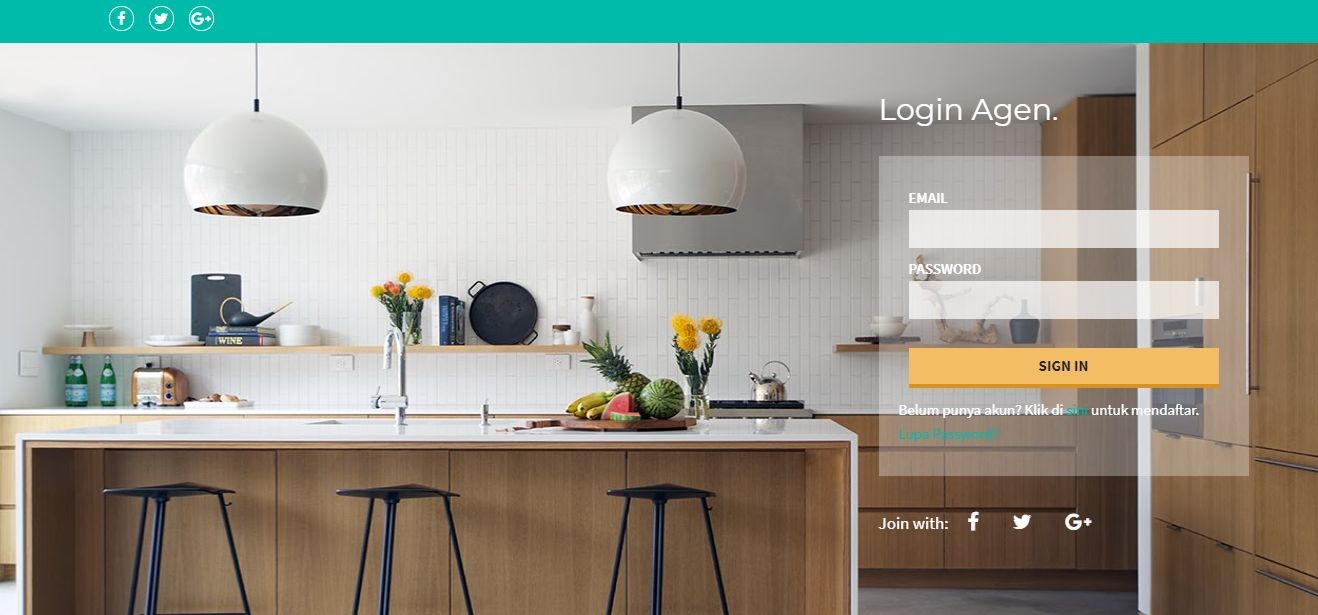
Gambar 5.8 Antarmuka Admin Tambah Properti

* + 1. **Antarmuka Agen**
       1. **Landing Page**

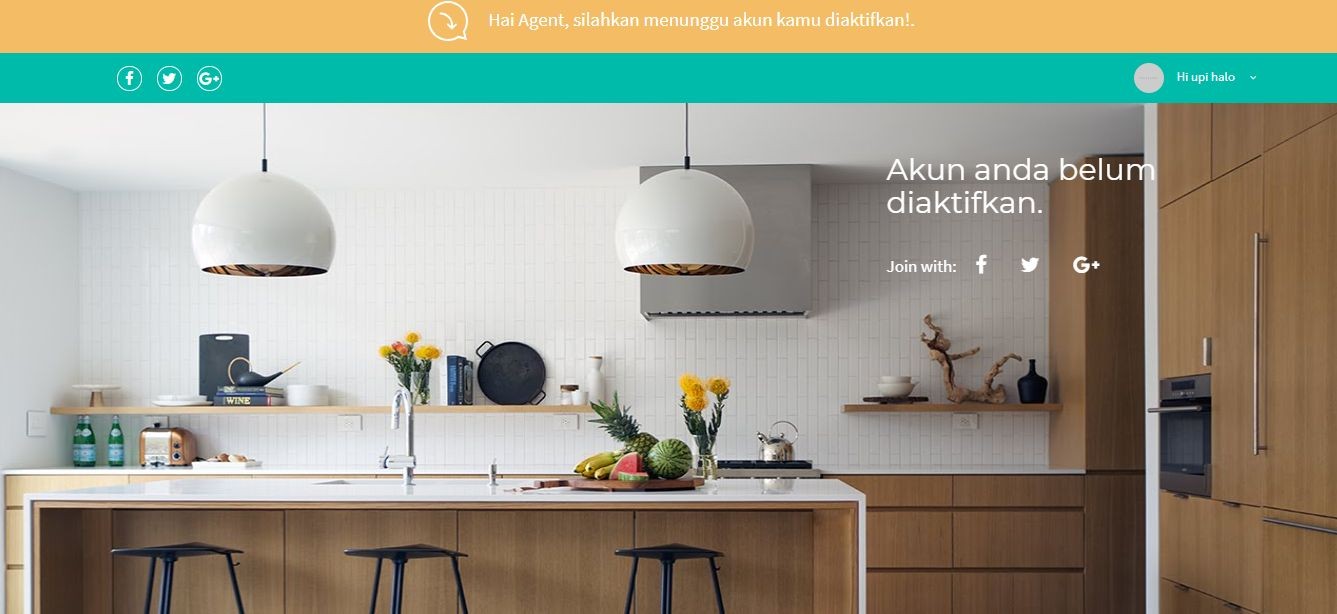
Halaman *Landing Page* pada Agen terdapat 3 bagian, yaitu halaman registrasi (Gambar 5.9) , halaman *Login* (Gambar 5.10) , dan halaman menunggu verifikasi (Gambar 5.11) .



Gambar 5.9 Antarmuka Registrasi Agen



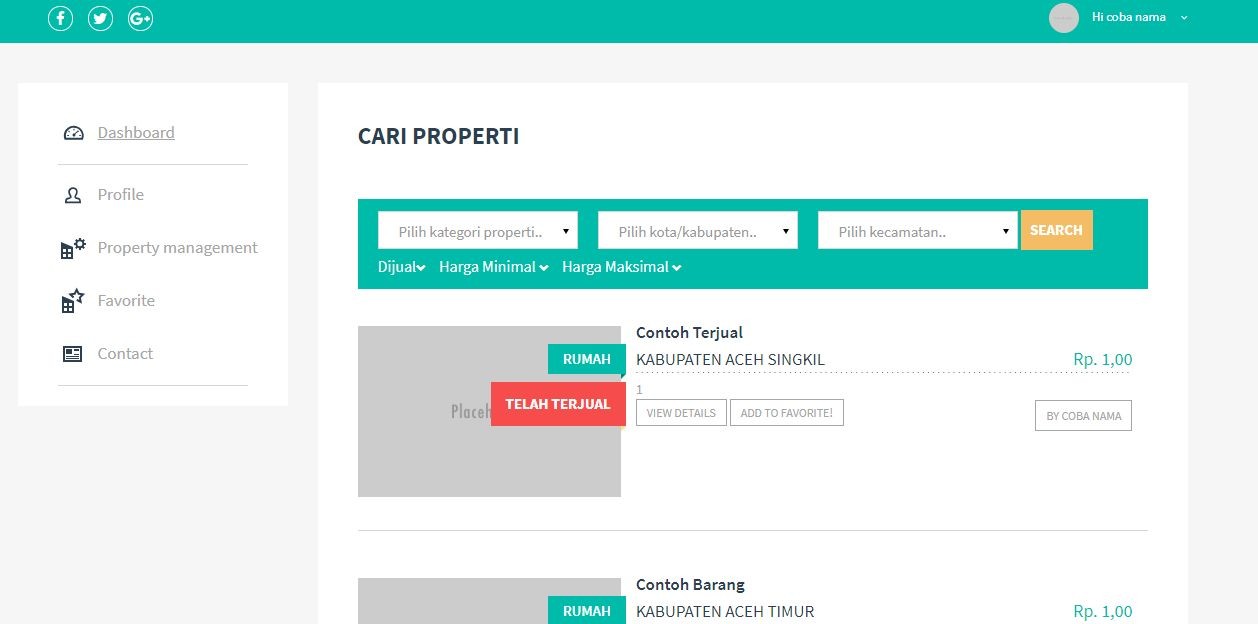
Gambar 5.10 Antarmuka *Login* Agen



Gambar 5.11 Antarmuka Tunggu Verifikasi

* + - 1. **Halaman Utama**

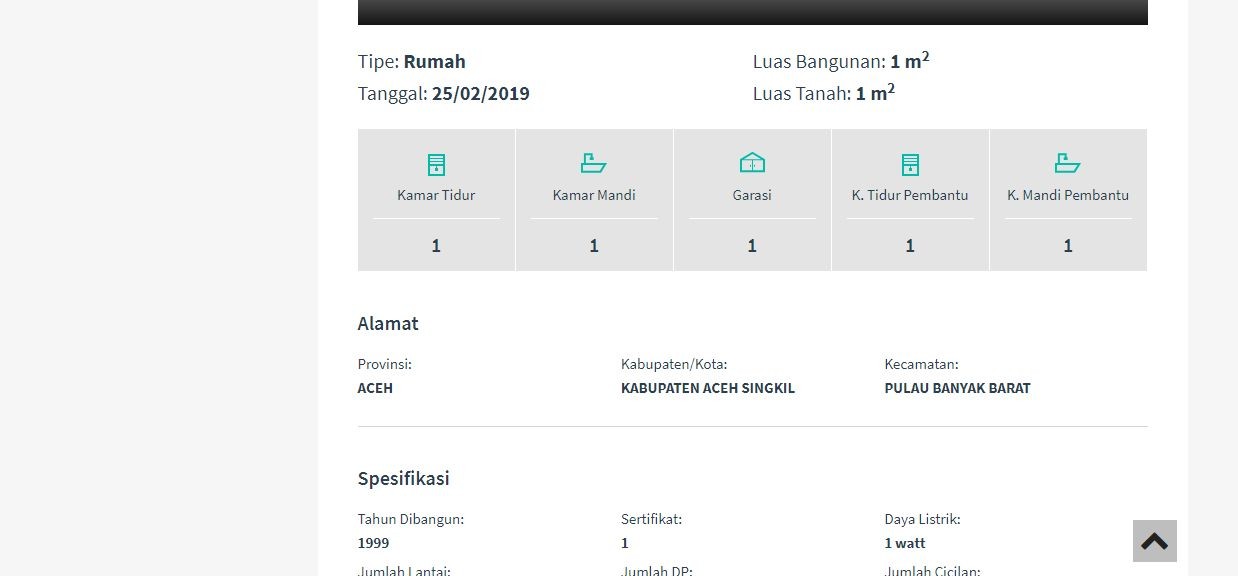
Halaman utama pada Agen adalah tampilan *dashboard* yang menampilkan daftar properti sekaligus menu pencarian (Gambar 5.12), detail properti (Gambar 5.13a dan Gambar 5.13b) ,dan setiap halaman bagi agen yang telah terverifikasi terdapat tombol navigasi serta tombol *Logout* yang dapat dilihat di Gambar 5.14



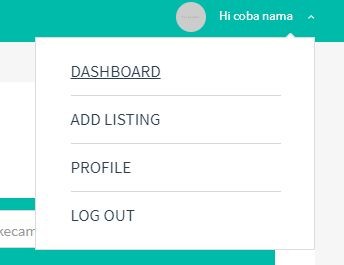
Gambar 5.12 Antarmuka *Dashboard Agen*



Gambar 5.13. a) Antarmuka Agen Detail Properti



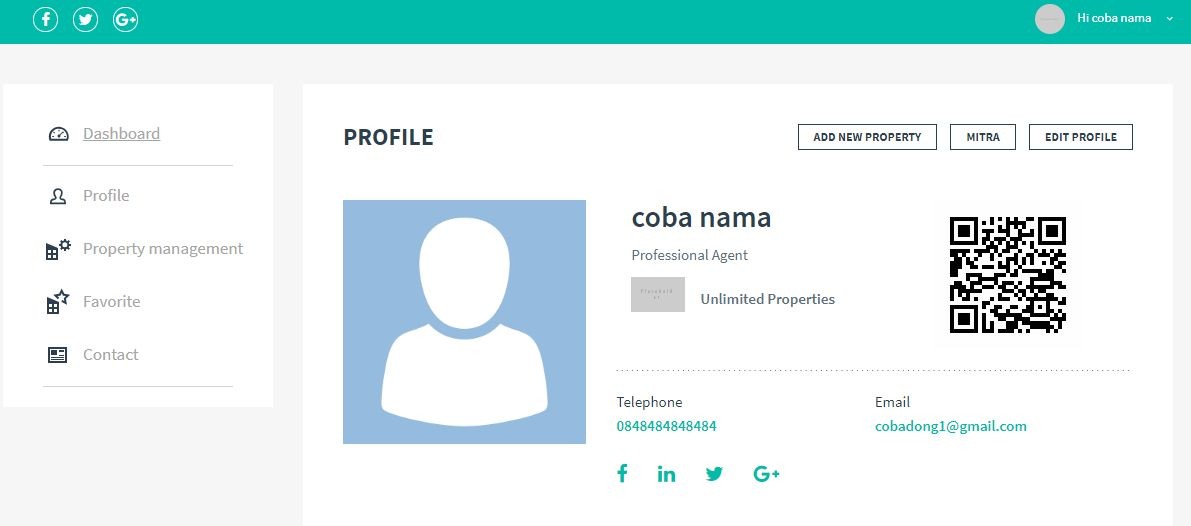
Gambar 5.13. b) AntarmukaAgen Detail Properti Lanjt.



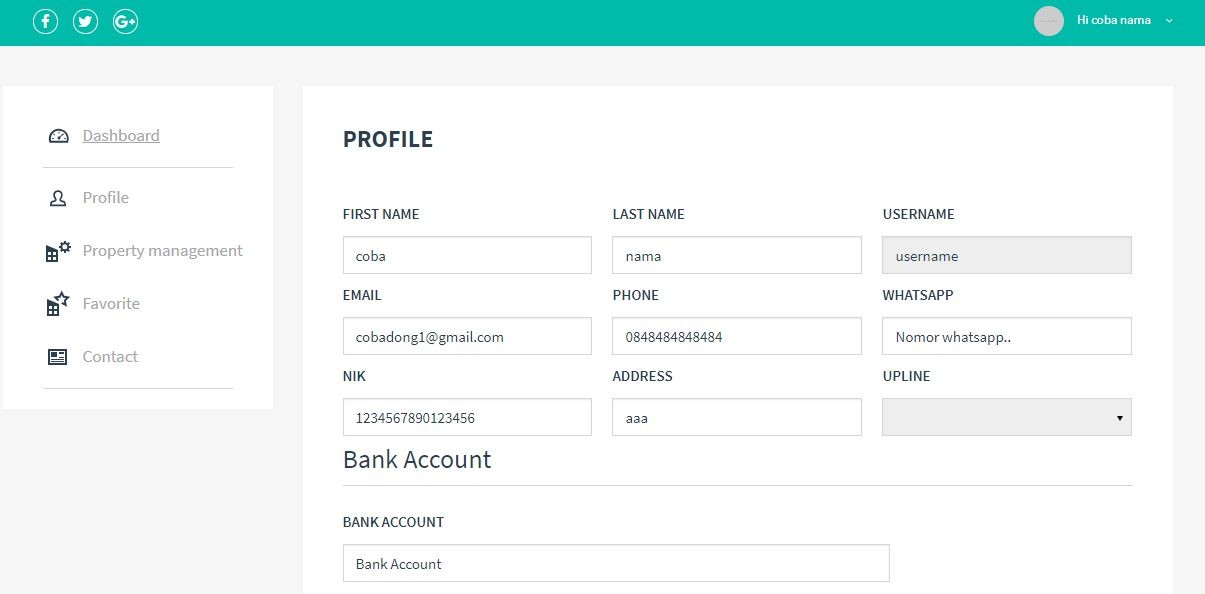
Gambar 5.14 Antarmuka Agen Navigasi dan *Logout*

* + - 1. **Halaman Profile**

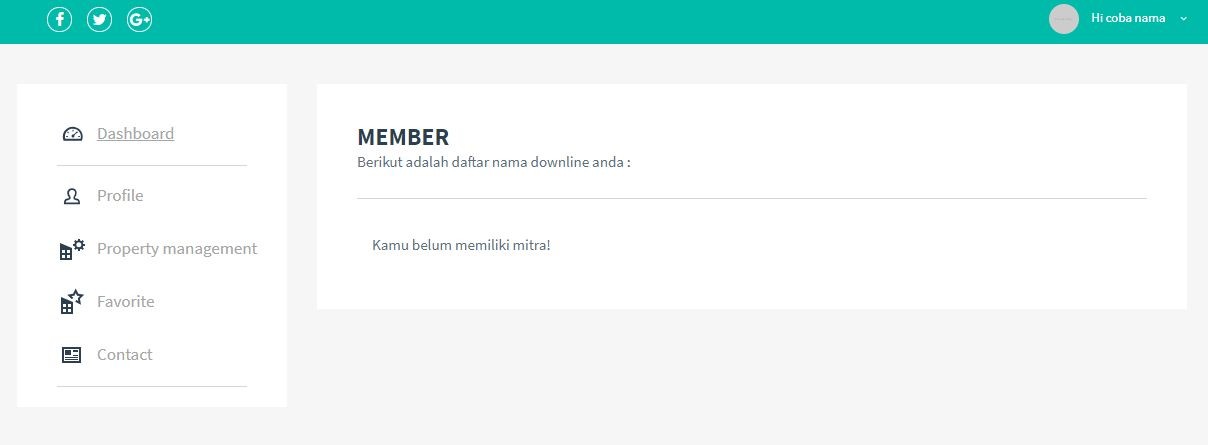
Pada bagian ini terdapat tampilan profil agen (Gambar 5.15) , tampilan pengeditan data agen (Gambar 5.16) , dan tampilan mitra agen (Gambar 5.17)



Gambar 5.15 Antarmuka Profil Agen



Gambar 5.16 Antarmuka Edit Profil Agen

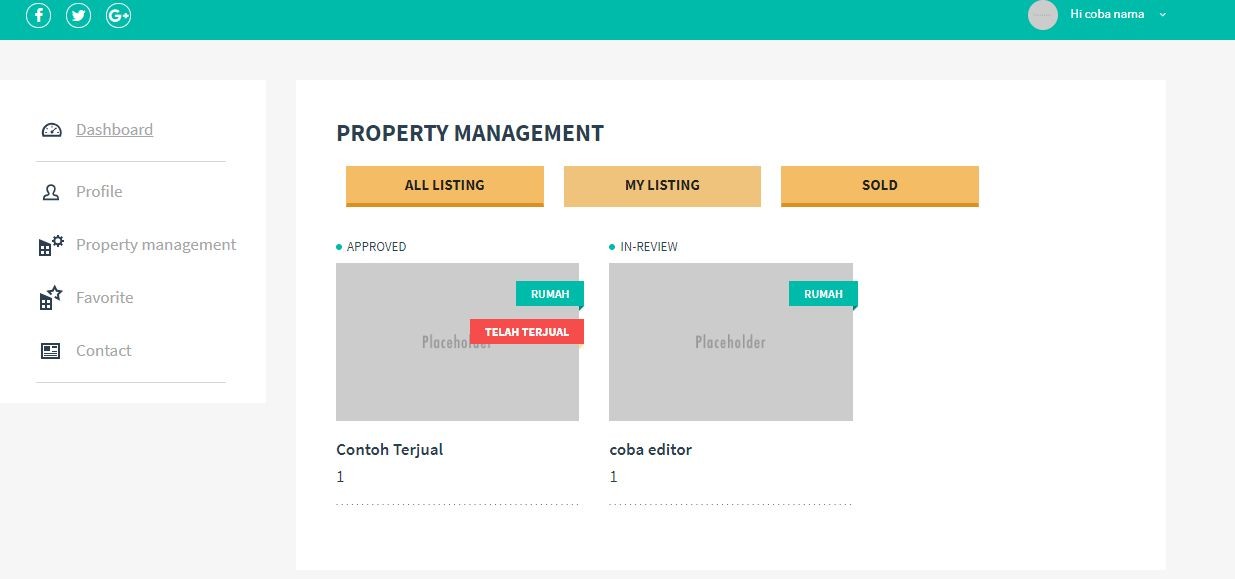


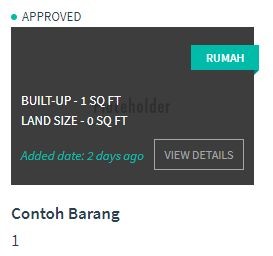
Gambar 5.17 Antarmuka Mitra Agen

* + - 1. **Halaman Property Management**

Pada bagian Property Management hanya ada terdapat halaman yang menampilkan daftar properti dengan 3 kategori yang berbeda, kategori properti milik agen baik yang telah terverifikasi atau belum , kategori properti agen yang terjual , dan semua properti yang terdaftar dalam sistem yang telah terverifikasi. Semua ketiga kategori tersebut memiliki layout yang sama seperti pada

Gambar 5.18 , dan apabila mouse diarahkan pada daftar properti akan menampilkan sebuah tombol (Gambar 5.19) yang mengarahkan ke detail gambar seperti pada Gambar 5.13 .

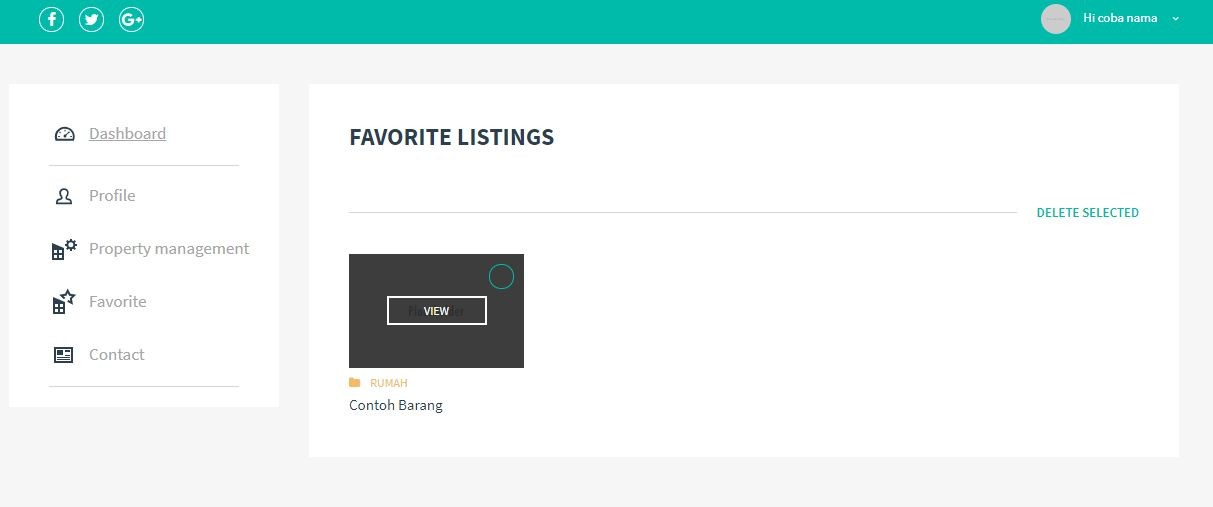


Gambar 5.18 Antarmuka Daftar Properti Agen

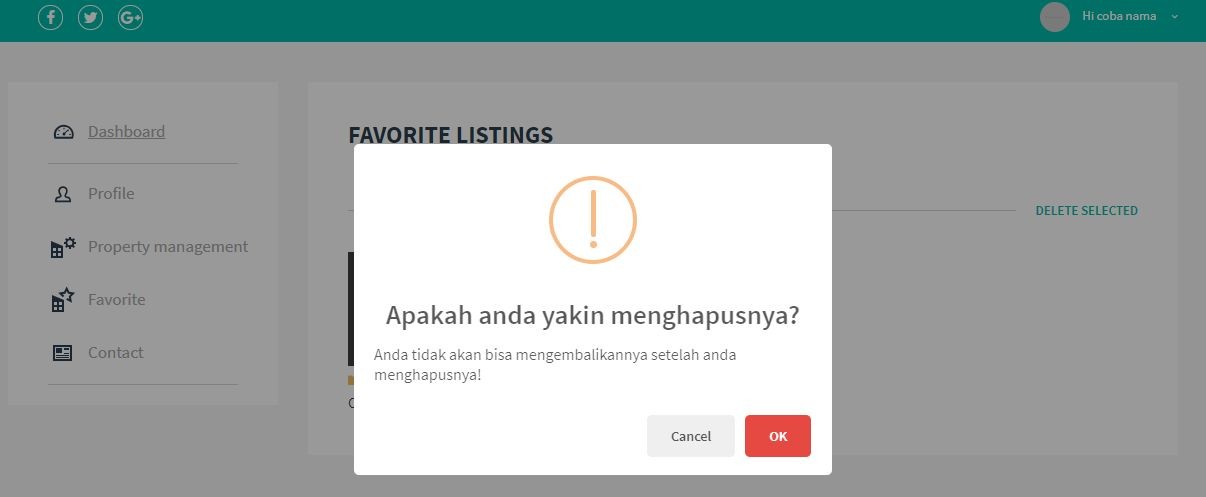
Gambar 5.19 Antarmuka Tombol Detail Properti

* + - 1. **Halaman Favorite**

Halaman favorit berisi daftar properti yang menjadi favorit agen (Gambar 5.20) dan antarmuka untuk menghapus properti dari daftar Favorit (Gambar 5.21)



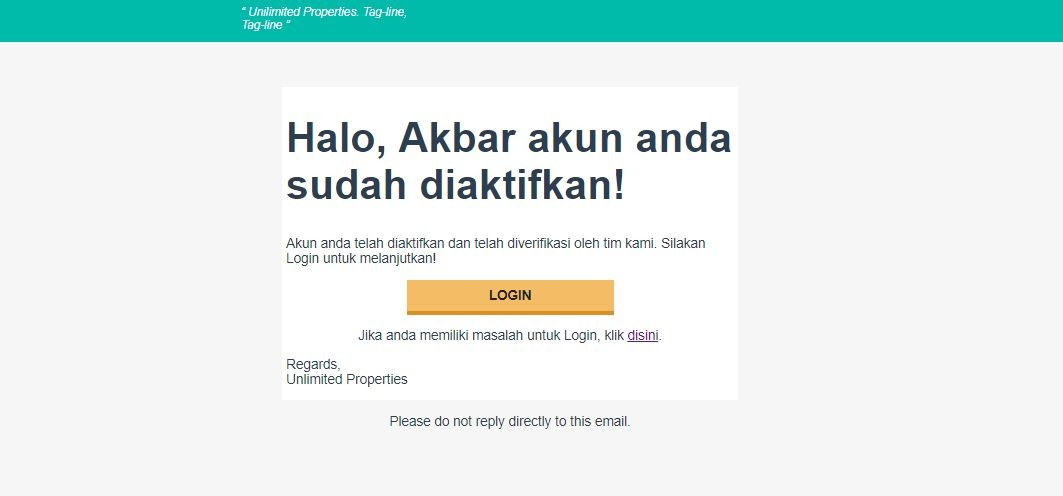
Gambar 5.20 Antarmuka Favorit



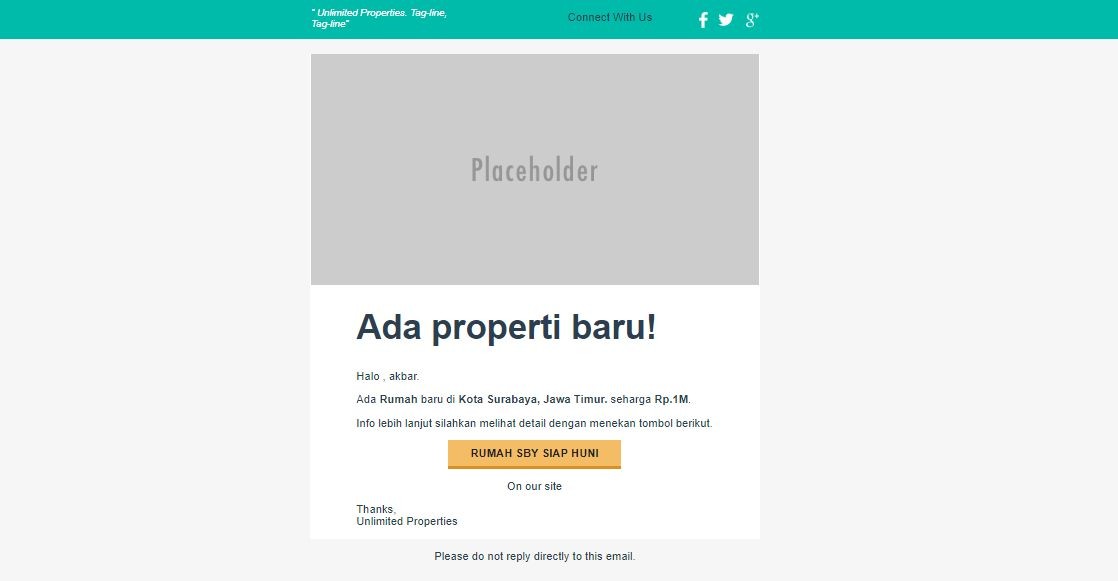
Gambar 5.21 Antarmuka Hapus Favorit

* + - 1. **Antarmuka Email**

Antarmuka email dibagi menjadi 2 bagian , yaitu email notifikasi akun terverifikasi (Gambar 5.22) dan notifikasi bahwa terdapat properti baru yang telah terverifikasi (Gambar 5.23)



Gambar 5.22 Antarmuka Email Akun Terverifikasi



Gambar 5.22 Antarmuka Email Properti Baru

*[Halaman ini sengaja dikosongkan]*

**BAB VI PENGUJIAN DAN EVALUASI**

Bab ini menjelaskan tahap uji coba terhadap aplikasi Sistem Pengelolaan Informasi dan Penjualan Properti Bangunan Unlimited Properties (UPI). Pengujian dilakukan untuk memastikan kualitas perangkat lunak yang dibangun dan kesesuaian hasil eksekusi perangkat lunak dengan analisis dan perancangan perangkat lunak.

* 1. **Tujuan Pengujian**

Pengujian dilakukan terhadap aplikasi Sistem Pengelolaan Informasi dan Penjualan Properti Bangunan Unlimited Properties (UPI) guna mengetahui beberapa hal berikut ini :

* + 1. Menguji kesesuaian dan ketepatan fungsionalitas dari seluruh sistem aplikasi.
    2. Menguji kesesuaian fitur yang dikembangkan dalam aplikasi UPI.
  1. **Kriteria Pengujian**

Penilaian atas pencapaian tujuan pengujian didapatkan dengan memerhatikan beberapa hasil yang diharapkan berikut :

* + 1. Kemampuan aplikasi untuk membedakan agen berdasarkan status kevalidannya.
    2. Kemampuan aplikasi untuk membedakan properti berdasarkan statusnya.
    3. Kemampuan aplikasi untuk memberikan hak akses kepada pengguna yang berhak.
    4. Kemampuan aplikasi memenuhi kebutuhan fungsionalitas lainnya seperti memasukkan data, mengubah data, menghapus data, melihat detil

data , melihat daftar properti dan pengguna, verifikasi agen dan properti, dan dapat mengirimkan email.

* + 1. Kesesuaian dalam memenuhi kebutuhan non-fungsionalitas aplikasi , yaitu :
       - Dapat diakses melalui internet.
       - Sistem memiliki tampilan (antarmuka) yang mudah dipahami.
  1. **Skenario Pengujian**

Skenario pengujian dilakukan dengan melakukan peran sebagai agen maupun admin yang akan menjalankan fitur-fitur dan seluruh kebutuhan fungsional dari sistem. Langkah-langkah untuk setiap kebutuhan fungsionalitas yaitu sebagai berikut :

1. Sebagai agen :
   1. Agen membuka aplikasi UPI.
   2. Agen melakukan registrasi akun.
   3. Agen menunggu konfirmasi verifikasi akun.
   4. Setelah menerima verifikasi akun melalui email, Agen melakukan *login.*
   5. Agen melengkapi data diri.
   6. Agen masuk kedalam profil, melihat data yang telah dimasukkan.
   7. Agen masuk ke *dashboard* untuk melihat daftar properti.
   8. Agen melakukan pencarian properti.
   9. Agen melihat detail properti yang dicari,
   10. Agen menambahkan properti tersebut kedalam daftar favoritnya.
   11. Agen masuk ke dalam halaman favorit.
   12. Agen menghapus satu atau lebih properti dari daftar favoritnya.
   13. Agen masuk ke dalam halaman profil untuk menambah properti.
   14. Agen menambahkan properti dengan mengisi formulir yang telah disediakan.
   15. Agen mendapatkan email notifikasi bahwa propertinya telah disetujui.
   16. Agen masuk ke halaman *property management* untuk mengedit beberapa data.

### Agen menghapus properti yang dimilikinya.

* 1. Agen *logout* dari aplikasi.

### Sebagai admin :

* 1. Admin melakukan *login*.

### Admin membuka halaman daftar Agen.

* 1. Admin membuka detail seorang Agen.
  2. Admin mengaktifkan seorang Agen.
  3. Admin menonaktifkan seorang Agen.
  4. Admin masuk ke halaman properti
  5. Admin menambahkan properti dengan mengisi formulir yang telah disediakan.
  6. Admin melakukan pencarian properti.
  7. Admin membuka detail properti.
  8. Admin menyetujui properti.
  9. Admin mengubah status properti.
  10. Admin menghapus suatu properti.
  11. Admin *logout* dari aplikasi.

# Evaluasi Pengujian

### Hasil pengujian dilakukan terhadap pengamatan mengenai perilaku sistem aplikasi UPI terhadap kasus skenario uji coba. Pengujian dilakukan oleh pihak pengembang, pengguna , dan pembimbing lapangan. Tabel 6.1 di bawah ini menjelaskan hasil uji coba terhadap aplikasi yang telah dibuat.

*Tabel 6.1. Hasil Evaluasi Pengujian*

|  |  |
| --- | --- |
| **Kriteria Pengujian** | **Hasil Pengujian** |
| Mengelola data diri sebagai agen | Terpenuhi |
| Mengelola data properti baik sebagai agen maupun admin | Terpenuhi |
| Agen dapat mengelola daftar favoritnya | Terpenuhi |
| Melakukan verifikasi agen dan penyetujuan properti sebagai Admin | Terpenuhi |
| Sistem secara otomatis mengirimkan notifikasi berupa email | Terpenuhi |
| Melakukan pencarian properti baik sebagai agen maupun admin | Terpenuhi |

*[Halaman ini sengaja dikosongkan]*

**BAB VII KESIMPULAN DAN SARAN**

* 1. **Kesimpulan**

Kesimpulan yang didapat setelah melakukan pengembangan aplikasi pada kegiatan kerja praktek di CV. Aksamedia Surabaya adalah sebagai berikut :

* + 1. Aplikasi yang dibangun telah sesuai dengan permintaan dan dapat oleh klien yang memesan jasa pembuatan *website* di Aksamedia.
    2. Dengan adanya aplikasi UPI, Agen dan Admin dapat dengan mudah memonitoring dan mengelola data untuk memaksimalkan penjualan.
    3. *Framework* yang digunakan dalam pembangunan aplikasi UPI merupakan *framework* yang umum digunakan di lingkungan kerja Aksamedia sehingga untuk *maintenance* pihak Aksamedia dapat melakukannya dengan mudah.
  1. **Saran**

Saran untuk pengembangan sistem aplikasi UPI

adalah sebagai berikut :

* + 1. Pada segi admin, perlu sebuah *view* atau halaman khusus untuk melihat detail suatu data agen ataupun properti untuk mencegah adanya perubahan yang tidak diinginkan.
    2. Perlu diadakan analisis dan diskusi bersama dengan klien untuk memastikan dan mendapatkan kebutuhan serta solusi yang tepat.

*[Halaman ini sengaja dikosongkan]*

**DAFTAR PUSTAKA**

1. "Garda Pangan," [Online]. Available : [http://gardapangan.org/tentang-kami/](%20http://gardapangan.org/tentang-kami/) [Accessed 20 January 2020].
2. "Android," [Online]. Available : <https://en.wikipedia.org/wiki/Android_(operating_system)> [Accessed 29 February 2020].
3. "Java," [Online]. Available : <https://id.wikipedia.org/wiki/Java>. [Accessed 29 February 2020].
4. "XML," [Online]. Available : https://[id.wikipedia.org/wiki/XML](https://id.wikipedia.org/wiki/XML). [Accessed 29 February 2019].
5. J. Enterprise, HTML 5 MANUAL BOOK, Jakarta: PT Elex Media Komputindo, 2014.
6. D. Bartholomew, <<MariaDB vs. MySQL,>> Dostopano, 2014,2012.

*[Halaman ini sengaja dikosongkan]*

**BIODATA PENULIS I**

Nama : Yemima Sutanto

Tempat, Tanggal Lahir : Surabaya, 28 Maret 1999

Jenis Kelamin : Perempuan

Agama : Kristen

Status : Belum Menikah

Alamat : Jln. Ploso Timur 1/68, Surabaya

Telepon : +6281357657416

Email : [yemimasutanto28@gmail.com](mailto:yemimasutanto28@gmail.com)

PENDIDIKAN FORMAL

2017-sekarang : Mahasiswa S1 Teknik Informatika ITS 2014-2017 : SMA Katolik Frateran Surabaya

2011-2014 : SMP Katolik Angelus Custos 1 Surabaya

2005-2011 : SD Santa Lorent Surabaya

KEMAMPUAN

* *Web Programmin*g (HTML, PHP, CSS)
* *Programming* (C)
* *UI/UX Design*
* *Mobile Programming* (Java)
* *Database Management* (Oracle, MySQL, PostgreSQL)

### *Software* Perkantoran (Ms. Office Word, Excel, Powerpoint)

* Bahasa (Indonesia, Inggris, Mandarin)

AKADEMIS

Kuliah : Departemen Teknik Informatika - ITS

Angkatan 2017

Semester : 6 (Enam)

IPK : 3.49 (Semester 5)

*[Halaman ini sengaja dikosongkan]*

**BIODATA PENULIS II**

Nama : Elkana Hans Widersen

Tempat, Tanggal Lahir : Surabaya, 20 Januari 1998 Jenis Kelamin : Laki-laki

Agama : Kristen

Status : Belum Menikah

Alamat : Jln. Merak III/P2-104 Rewwin, Waru

Telepon : +6289636693644

Email : elknhns[@gmail.com](mailto:akbarnotopb@gmail.com)

PENDIDIKAN FORMAL

2017-sekarang : Mahasiswa S1 Teknik Informatika ITS 2013-2016 : SMA Negeri 15 Surabaya

2010-2013 : SMP Kristen Petra 5 Surabaya

2004-2010 : SD Santo Yusup Tropodo

KEMAMPUAN

* *Web Programmin*g (HTML, PHP, CSS, Javascript)
* *Programming* (C, C++)
* *UI/UX Design*
* *Database Management* (Oracle, MySQL)

### *Software* Perkantoran (Ms. Office Word , Excel , Powerpoint)

* Bahasa (Indonesia, Inggris)

AKADEMIS

Kuliah : Departemen Teknik Informatika - ITS

Angkatan 2017

Semester : 6 (Enam)

IPK : 3.60 (Semester 5)

*[Halaman ini sengaja dikosongkan]*