IMPLEMENTASI CHATBOT PADA ORDER MANAGEMENT SYSTEM USAHA MIKRO KECIL MENENGAH

(STUDI KASUS HDKREASI)

Disusun Oleh:

N a m a : Ahmad Iswandi

NIM : 13523227

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2018

ii

HALAMAN PENGESAHAN DOSEN PEMBIMBING

IMPLEMENTASI CHATBOT PADA ORDER MANAGEMENT SYSTEM USAHA MIKRO KECIL MENENGAH

(STUDI KASUS HDKREASI)

TUGAS AKHIR

Disusun Oleh:

N a m a : Ahmad Iswandi

NIM : 13523227

Yogyakarta, 02 Februari 2018

Pembimbing 1,

(Fathul Wahid, S.T., M.Sc., Ph.D.)

Pembimbing 2,

(Septia Rani, S.T., M.Cs.)

iii

HALAMAN PENGESAHAN DOSEN PENGUJI

IMPLEMENTASI CHATBOT PADA ORDER MANAGEMENT SYSTEM USAHA MIKRO KECIL MENENGAH

(STUDI KASUS HDKREASI)

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik Informatika di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 8 Januari 2018

Tim Penguji

Septia Rani, S.T., M.Cs. _____

Anggota 1

Galang P Mahardhika, S.Kom., M.Kom. _____

Anggota 2

Almed Hamzah, S.T., M.Eng. _____

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Hendrik, S.T., M.Eng.)

iv

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Ahmad Iswandi

NIM : 13523227

Tugas akhir dengan judul:

IMPLEMENTASI CHATBOT PADA ORDER MANAGEMENT SYSTEM USAHA MIKRO KECIL MENENGAH

(STUDI KASUS HDKREASI)

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung resiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 02 Februari 2018

(Ahmad Iswandi)

HALAMAN PERSEMBAHAN

Alhamdulillah, segala puja dan puji syukur kehadirat Allah SWT yang telah memberikan nikmat, karunia, dan rahmatnya kepada penulis sehingga dapat menyelesaikan tugas akhir ini. Shalawat beserta salam tercurah kepada junjungan Nabi besar Muhammad SAW, semoga kelak akan kita dapati syafa'at darinya.

Tulisan ini penulis persembahkan untuk kedua orang tua tercinta. Ayahanda Jiman dan Ibunda Jainem. Karena berkat doa, semangat, nasehat, motivasi dan kasih sayang yang tiada henti dari mereka berdua tak mungkinlah dapat penulis capai sebuah titik pencapaian yang penulis rasakan sekarang ini. Semoga dengan pencapaian kecil dari penulis ini dapat membuat bangga orang tua tercinta di kampung halaman. Semangat kerja keras untuk menyekolahkan putra-putrinya, semoga menjadi ladang ibadah yang akan Allah balas dengan pahala yang besar kepada Ayahanda dan Ibunda penulis

HALAMAN MOTO

"Kita tak punya sayap, kita tak dapat melayang. Namun kita punya kaki untuk naik dan mendaki"

(Henry Wadsworth Longfellow)

"Jangan takut jatuh, karena yang tidak pernah memanjatlah yang tidak pernah jatuh. Yang takut gagal, kerana yang tidak pernah gagal hanyala orang-orang yang tidak pernah melangkah. Jangan takut salah, kerana dengan kesalahan yang pertama kita dapat menambah pengetahuan untuk mencari jalan yang benar pada langkah yang kedua"

(Abdul Malik Karim Amrullah a.k.a Buya Hamka)

"*Kalau energi tidak digunakan untuk kerja kerja besar, maka perhatian kita segera tercurah kepada masalah-masalah kecil*"

(Anis Matta)

Life doesn't give you seat belt, you must be prepared for all possibilites

(Batman on the lego batman movie)

KATA PENGANTAR

Allhamdulillah, penulis panjatkan kehadiran Allah SWT yang telah memberikan rahmat, hidayah, serta karunia-Nya. Tak lupa shalawat dan salam kami haturkan kepada junjungan kita Nabi Muhammad SAW dan para sahabat sehingga penulis dapat menyelesaikan laporan tugas akhir yang berjudul.”Implementasi *Chatbot* pada *Order Management System* Usaha Mikro Kecil Menengah Studi Kasus HDKreasi”.

Laporan ini disusun sebagai salah satu persyaratan yang harus dipenuhi dalam rangka menyelesaikan pendidikan pada jenjang Strata 1 di Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia

Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan rasa terima kasih kepada :

1. Bapak Hendrik, S.T., M.Eng**,** selaku Ketua Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
2. Bapak Fathul Wahid, Ph.D. selaku dosen Pembimbing 1 dalam pelaksanaan tugas akhir ini, yang telah memberi masukan kepada penulis.
3. Ibu Septia Rani, S.T, M.Cs selaku dosen Pembimbing 2 dalam pelaksanaan tugas akhir yang telah banyak memberikan koreksi terhadap kesempurnaan laporan ini, serta meluangkan waktunya untuk membimbing penulis dalam penyelesaian laporan ini.
4. Haryanto, selaku kakak dari penulis sekaligus pemilik UMKM HDKreasi, yang telah memberikan kasih sayang terhadap penulis. Beserta *stake holder* yang berada di UMKM tersebut, bang Khairi firzany, bang Iman Lukman, bang Matlaul Hadi, Widodo, terimakasih atas keramah tamahannya.
5. Mas Rahmat Gumilar, Selaku pimpinan PT. Djavis Indonesia yang merupakan mentor dengan *coding skill* di atas rata-rata yang telah memberikan banyak ilmu nya kepada penulis
6. Segenap keluarga besar teman-teman di Fakultas Teknologi Industri terutama dari Jurusan Teknik Informatika Universitas Islam Indonesia yang telah memberikan bantuan dan dukungannya.
7. Bapak dan Ibu Dosen Jurusan Teknik Informatika yang telah memberikan ilmunya kepada penulis.
8. Semua pihak yang sudah mendukung penulis, yang tidak dapat penulis sebutkan satu persatu, terima kasih atas bantuan dan do’anya.

viii

Penulis menyadari masih terdapat banyak kekurangan pada penelitian yang penulis lakukan atau di dalam laporan ini. Penulis berharap penelitian ini dapat bermanfaat bagi semua pihak.

Wassalamu'alaikum Warahmatullahi Wabarakatuh

Yogyakarta, 02 Februari 2018

(Ahmad Iswandi)

SARI

HDKreasi adalah salah satu Usaha Mikro Kecil dan Menengah (UMKM) yang bergerak di bidang industri kerajinan dan percetakan. Beragam produk dibuat oleh UMKM ini untuk memenuhi kebutuhan pelanggan di antaranya adalah *paper bag*, undangan, dan *goodie bag*. Saat ini, pengelolaan pesanan belum dilakukan secara maksimal, proses pendokumentasian pesanan masih kurang baik, pencatatan transaksi, produk dan pelanggan masih dilakukan secara manual yaitu dicatat di dalam buku catatan. Hal ini tentunya menimbulkan permasalahan terhadap usaha ini ketika suatu saat ingin mencari data pelanggan yang pernah bertransaksi dan produk apa saja yang pernah dibeli. Permasalahan lainnya yang dihadapi oleh usaha ini adalah semakin hari jumlah pelanggan yang ingin memesan produk atau sekedar bertanya mengenai produk serta pelanggan yang ingin melakukan *tracking order* (menanyakan sampai sejauh mana *proggress* produk yang dipesan) semakin banyak.

Berdasarkan permasalahaan yang telah dijelaskan di atas, penulis mencoba untuk mengembangkan sebuah OMS (*Order Management System)* dan *chatbot* yang terintegerasi. *Order Management System* ini sendiri adalah sebuah sistem yang dapat membantu usaha ini untuk melakukan pengelolaan pesanan dengan pendokumentasian yang baik. Sedangkan *chatbot* sebagai salah satu media untuk pelanggan dapat berkomunikasi dengan usaha ini secara otomatis. Pengembangan *chatbot* sendiri menggunakan sebuah *platform* pesan singkat dari Facebook yaitu *Messenger* sebagai media utama untuk menjangkau pelanggan. Agar *chatbot* dapat memahami pertanyaan yang ditanyakan oleh pelanggan, penulis menggunakan sebuah *platform* NLP (*Natural Language Processing)* yaitu API. AI atau sekarang berganti nama menjadi *DIALOGFLOW*. *Platform* inilah yang nantinya akan melakukan pemahaman terhadap setiap pertanyaan yang diberikan oleh pelanggan dan memetakan pertanyaan tersebut ke dalam sebuah *intent* atau maksud yang sudah penulis rancang di antaranya adalah penanganan percakapan untuk melakukan pesanan dan pemeriksaan pesanan.

Dari hasil penelitian yang telah dilakukan, penulis berhasil membuat sebuah sistem pengelolaan pesanan dan *chatbot* yang terintegrasi. *Chatbot* yang penulis buat mampu menjawab pertanyaan yang sesuai dengan tugas atau konteks yang telah penulis tentukan dan juga *chatbot* memiliki kemampuan untuk mengakuisisi informasi dari sistem pengelolaan pesanan yang dapat diberikan kepada pelanggan.

Kata kunci: *order management system*, *chatbot*, *nlp platform*.

GLOSARIUM

SDK	*Software Development Kit,* alat untuk pengembangan piranti lunak.
Fullfilment	Aksi untuk memenuhi sesuatu di dalam suatu keadaan.
Endpoint	Sebuah tempat di mana pengaksesan sesuatu terjadi.
Restfull api	*REpresentational State Transfer* merupakan standar arsitektur komunikasi berbasis web yang sering diterapkan dalam pengembangan layanan berbasis web. Layanan ini menyediakan sumber daya atau *resources* yang dapat diakses melalui URIs

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

HDKreasi adalah salah satu Usaha Mikro Kecil dan Menengah (UMKM) yang bergerak di bidang industri kerajinan dan percetakan. Beragam produk dibuat oleh UMKM ini untuk memenuhi kebutuhan pelanggan. Produk yang dihasilkan dalam kegiatan usaha ini di antaranya adalah *paper bag*, undangan, *goodie bag* dan produk lainnya. UMKM memiliki peranan penting dalam perekonomian Indonesia. Saat ini semakin banyak UMKM yang bermunculan. Menurut data dari Kementrian Koperasi dan Usaha Kecil Mikro Menengah Republik Indonesia terhitung pada tahun 2013 jumlah UMKM yang terdapat di Indonesia adalah sekitar 57.895.721. Hal ini merupakan tantangan baru yang harus dihadapi oleh para pelaku usaha guna dapat memenangkan persaingan usaha di antara sekian banyak usaha yang telah ada. Beragam cara dilakukan para usahawan guna bertahan di dalam persaingan diantara pelaku usaha lainnya.

HDKreasi telah memanfaatkan perkembangan teknologi di bidang informasi untuk meningkatkan kualitas usaha mereka. Dalam melaksanakan proses pemasaran, usaha ini telah menggunakan media informasi website dan juga sosial media seperti Facebook dan Instagram yang tak luput sebagai salah satu media promosi andalan. Selain itu, *platform* komunikasi juga digunakan untuk menjangkau pelanggan dari berbagai daerah seperti Line, Whatsapp, Messenger, Blackbery Messenger. Meskipun demikian, untuk memenangkan persaingan usaha, di dalam *internal* usaha HDKreasi masih banyak hal yang perlu diubah, di antaranya yaitu dari segi manajemen usaha serta peningkatan pelayanan kepada pelanggan. Saat ini, pengelolaan pesanan belum dilakukan secara maksimal, proses pencatatan pesanan masih kurang baik, pencatatan transaksi, produk dan pelanggan masih dilakukan secara manual yaitu dicatat di dalam buku catatan. Hal ini tentunya menimbulkan permasalahan terhadap usaha ini ketika suatu saat ingin mencari data pelanggan yang pernah bertransaksi dan produk apa saja yang pernah dibeli. Permasalahan lainnya yang dihadapi oleh usaha ini adalah semakin hari jumlah pelanggan yang ingin memesan produk atau sekedar bertanya mengenai produk serta pelanggan yang ingin melakukan *tracking order* (menanyakan sampai sejauh mana *progres* produk yang dipesan) semakin banyak. Akan tetapi pegawai yang bertugas menangani hal

tersebut tidak sebanding dengan permintaan yang masuk sehingga seringkali terdapat pelanggan yang diabaikan.

Untuk mengatasi permasalahan yang telah dijelaskan di atas, pada penelitian ini penulis akan membuat sebuah OMS (*Order Management System)* dan *chatbot* yang terintegrasi. *Order management system* adalah sistem perangkat lunak komputer yang digunakan di sejumlah industri untuk pencatatan dan pemrosesan pesanan. Sebuah OMS akan menyediakan pembaharuan informasi persediaan produk, serta pencatatan pelanggan dan pemrosesan pemesanan (Shetty, dkk, 2015). *Chatbot* sendiri adalah layanan yang didukung oleh sekumpulan aturan dan terkadang kecerdasan buatan, yang berinteraksi dengan pengguna melalui antarmuka obrolan. *Chatbot* juga digunakan dalam sistem dialog untuk berbagai keperluan praktis termasuk *customer services* atau akuisisi informasi (Komawar, dkk, 2015).

Penggunaan *platform* pesan singkat oleh pengguna internet sudah sangat masif digunakan untuk bertukar pesan antara satu dengan yang lainnya. Banyak sekali *platform* pesan singkat yang tersedia secara gratis di internet, seperti Messenger, Line dll. Dengan berbagai macam fungsionalitas dan fitur-fitur pendukung di dalamnya. Pada penelitian ini penulis ingin memanfaatkan *platform* pesan singkat tersebut yang akan digunakan sebagai portal untuk pelanggan berinteraksi dengan sistem pengelolaan *order* ini.

1.2 Rumusan Masalah

a. Bagaimana merancang dan membuat *Order Management System* yang dapat mengelola data di dalam basis data?
b. Bagaimana merancang *Conversational flow* untuk *chatbot* pada UMKM HDKreasi?
c. Bagaimana membangun arsitektur sistem yang terintegrasi antara *chatbot* dan *Order Management System*?

1.3 Batasan Masalah

a. *Order Management System* yang dibuat tidak memiliki fungsionalitas untuk menangani *financial processing*.
b. Penulis menggunakan layanan-layanan yang telah tersedia untuk membangun *chatbot*, untuk *platform* komunikasi penulis menggunakan layanan web Messenger dari Facebook, serta layanan pengelohan bahasa alami untuk membuat *chatbot* mengerti (API.AI).

1.4 Tujuan Penelitian

Adapun tujuan dari penelitian yang penulis lakukan adalah sebagai berikut:

a. Untuk membangun *order management system* yang dapat mengelola data yang terdapat di dalam *database*.
b. Merancang dan mengimplementasikan *conversational flow* untuk kebutuhan *chatbot* dalam menangani pertanyaan dari *user*.
c. Membuat sebuah rancangan sistem yang terintegrasi antara *chatbot* dan *order management system*, sehingga *chatbot* dapat memberikan informasi mengenai produk ataupun pesanan yang terdapat di dalam sistem sesuai dengan yang ditanyakan pelanggan.

1.5 Manfaat Penelitian

Penelitian yang dilakukan ini diharapkan akan memberi manfaat terhadap kegiatan usaha UMKM HDKreasi, di antaranya adalah sebagai berikut:

a. Membantu pemilik usaha untuk mengakomodir pengelolaan pesanan, pendokumentasian, pemantauan status pesanan dan laporan kegiatan penjualan yang pernah dilakukan.
b. Dengan adanya *chatbot*, pelanggan dapat dengan mudah untuk mendapatkan informasi produk dan informasi tentang pesanan sudah sampai sejauh mana pemrosesan pesanan yang dilakukan oleh usaha ini. Sehingga diharapkan meningkatkan kepuasan pelayanan bisnis terhadap pelanggan.
c. Memberikan kemudahan kepada usaha ini untuk memberikan notifikasi kepada pelanggan seperti notifikasi pesanan dan *invoice*.

1.6 Metodologi Penelitian

a. Studi pustaka

Pada tahapan studi pustaka ini, penulis akan mempelajari *platform* yang akan penulis gunakan untuk membuat sebuah *chatbot*, di antara yang akan penulis pelajari adalah platform layanan pesan komunikasi dari Facebook yaitu “Facebook Messenger” dan juga layanan *Natural Language Processing Platform (API.AI).* Tujuan pada tahapan ini untuk melakukan pembelajaran terhadap dokumentasi layanan dan juga penulis melakukan studi literatur dari berbagai macam referensi seperti jurnal, buku, dan penelitian yang serupa.

b. Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk mengumpulkan informasi mengenai kebutuhan akan sistem, terdapat dua analisis kebutuhan yang dilakukan yaitu analisis kebutuhan *order management system* dan juga *chatbot.* Analisis kebutuhan *order management system* ini bertujuan untuk mengumpulkan informasi apa saja yang dibutuhkan untuk mengetahui kebutuhan sistem, proses analisis dilakukan dengan melakukan observasi secara langsung di HDKreasi dan wawancara secara langsung. Analisis kebutuhan pada *chatbot* ini bertujuan untuk merancang *conversational flow* yang digunakan untuk menangani percakapan antara *chatbot* dan pelanggan.

c. Perancangan

Pada tahapan ini akan didefinisikan kebutuhan-kebutuhan sistem sesuai dengan hasil analisis pada tahapan sebelumnya, perancangan yang dibuat antara lain adalah perancangan DFD (*Data Flow Diagram), Conversational Flow, Use Case Diagram, ERD (Entity Relationship Diagram)* dan perancangan antarmuka sistem.

d. Implementasi

Pada tahapan ini akan dilakukan penerapan hasil analisis dan rancangan ke dalam kode program. Berikut akan dipaparkan beberapa teknologi yang akan digunakan dalam pembuatan *order management system* dan *chatbot.*

1. PHP digunakan untuk bahasa pemrograman di sisi *server*.
2. Codeigniter merupakan framework php yang akan penulis gunakan untuk membuat aplikasi di sisi *server*, di mana penulis akan membuat sebuah layanan *web* untuk *order management system* menggunakan *framework* ini.
3. Mysql merupakan RDBMS (*Relational Database Management System*) yang digunakan untuk menyimpan data.
4. Angularjs merupakan *frontend framework* yang berbasis *javascript* yang penulis gunakan untuk membuat *user interface*.
5. Nodejs merupakan sebuah *platform* yang dibangun di atas *Javascript Runtime Chrome*
6. Socket.io merupakan *platform websocket* di mana socket.io memperbolehkan komunikasi secara *realtime* pada aplikasi *web* maupun aplikasi *mobile.*

e. Pengujian

Pada tahapan ini akan dilakukan pengujian terhadap fungsi-fungsi tertentu dengan melakukan pengujian diharapkan hasil dari fungsi tersebut sesuai dengan keinginan. Pengujian

chatbot akan dilakukan untuk melihat seberapa jauh conversational flow yang dirancang dapat mengakomodir pertanyaan-pertanyaan yang diajukan.

1.7 Sistematika Penulisan

Sistematika penulisan ini dibuat untuk mempermudah pembaca memahami isi dari penelitian ini, maka dari itu penulis memaparkan sistematika penulisan pada penelitian ini seperti berikut :

Bab I Pendahuluan, berisi latar belakang masalah, batasan masalah, rumusan masalah, tujuan penelitian, manfaat penelitian, metode penelitian serta sistematika penulisan yang dijadikan sebagai materi laporan penelitian.

Bab II Tinjauan Pustaka, berisi tentang teori-teori yang menjadi dasar dalam penelitian yang dilakukan.

Bab III Analisis dan Perancangan Sistem, berisi tentang analisis sistem yang dibutuhkan melingkupi identifikasi masalah, rancangan solusi beserta analisis-analisis kebutuhannya, dan perancangan sistem yang berasal dari hasil analisis sistem dalam bentuk *Usecase*, *Activity Diagram, Conversation flow, ERD,* beserta desain antarmukanya.

Bab IV Implementasi dan Pengujian, berisi penjelasan dari hasil implementasi yang beserta hasil evaluasi dari pengujian yang telah dilakukan.

Bab V Kesimpulan dan Saran, pada bagian terakhir ini akan berisi kesimpulan yang ditarik dari proses, hasil maupun evaluasi selama penelitian ini berlangsung serta memberikan saran untuk kekurangan-kekurangan yang belum terakomodasi pada penelitian ini untuk dikembangkan lebih jauh ke depan.

BAB II
TINJAUAN PUSTAKA

2.1 OMS (Order Management System)

OMS atau *Order Management System* adalah sistem yang mengakomodir kebutuhan pengelolaan pesanan yang diterima oleh sebuah usaha dari pelanggan yang melakukan transaksi pembelian jasa atau produk dari usaha tersebut, OMS akan terus menyediakan informasi pembaharuan *inventory*, data pelanggan dan informasi data *order* (Shetty, dkk., 2015).

Pada dasarnya *Order Management System* digunakan untuk melakukan proses pencatatan *order* ke dalam sebuah basis data agar mempermudah pelaku usaha untuk melakukan *tracking* ataupun melakukan *reporting* tentang transaksi penjualan yang telah dilakukan. Pesanan bisa datang dari bisnis, pelanggan ataupun keduanya tergantung produk apa yang menjadi pilihan. Di dalam suatu *Order Management System* terdapat beberapa modul-modul yang digunakan untuk mendukung sistem ini di antaranya adalah:

a. Informasi Mengenai Produk. Di dalam OMS produk adalah hal yang utama, di mana transaksi penjualan akan terjadi apabila terdapat transaksi penjualan di mana pelanggan akan memilih produk ataupun jasa yang mereka inginkan untuk dibeli.
b. Informasi Pelanggan. Pencatatan informasi pelanggan juga tidak kalah penting, informasi pelanggan akan sangat berguna ke depannya bagi para pelaku usaha.
c. Pencatatan *Order*. Pencatatan *order* di dalam *Order Management System* merupakan tahapan merekam data ke dalam basis data, data-data yang direkam umumnya adalah data pelanggan beserta dengan produk apa saja yang dipilih oleh pelanggan tersebut.
d. Pemrosesan *Order*. Pemrosesan *order* pada OMS dilakukan untuk menindaklanjuti pesanan yang sebelumnya datang dari para pelanggan ataupun *user*.
e. *Invoice* merupakan dokumen yang digunakan untuk melakukan penagihan pembayaran terhadap pelanggan, pemilik usaha dapat membuat dan mencetak *invoice* untuk pelanggan.

2.2 Chatbot

Chatbot merupakan program komputer yang dirancang untuk dapat melakukan interaksi dengan manusia melalui pesan teks, maupun suara. *Chatbot* biasanya juga dibekali dengan

kecerdasan buatan dan pemrosesan bahasa alami yang membuatnya menjadi program komputer yang cerdas dan dapat menjawab pertanyaan yang diberikan oleh manusia. *Chatbot* dibangun sesuai dengan topik yang sudah dimodelkan dalam basis pengetahuan. Banyak *chatbot* yang sudah ada dibangun sesuai dengan topik dan permasalahan yang ingin dipecahkan oleh seseorang untuk keperluan pribadi ataupun keperluan bisnis, sebagai contoh adalah *chatbot* yang menangani pemesanan pizza. Di dalam *chatbot* tersebut telah ditanamkan model pengetahuan untuk menjawab pertanyaan-pertanyaan yang sesuai dengan konteks yang telah disusun.

Chatbot terdiri dari tiga kombinasi, di mana ketiga kombinasi inilah yang membentuk sebuah *chatbot* (Guzman & Ines, 2016), di antaranya adalah:

a. *User Interface*

User interface dalam *chatbot* ini sendiri adalah jembatan antara *chatbot* dan *user* saling berinteraksi. Melalui aplikasi pesan berbasis text. *User Interface* haruslah dapat memberikan pengalaman yang lebih baik kepada *user* ketika berinteraksi dengan *Chatbot*.

b. *Artificial Intelligence* (Kecerdasan Buatan)

AI atau Artificial Intelligence akan membuat *chatbot* mengerti dan memahami setiap interaksi yang terjadi dengan *user*. *Chatbot* menangani pemecahan masalah melalui aturan yang telah ditentukan sebelumnya di pohon keputusan.

c. Integrasi

Integrasi dengan sistem lainnya akan menambah kekayaan fitur yang terdapat di dalam suatu *chatbot*. Dengan mengintegrasikan *chatbot* ke sistem yang lain dapat menyediakan informasi tambahan. Dengan cara ini *chatbot* mampu memberikan informasi yang lebih kaya kepada *user*, seperti pada penelitian kali ini yang akan mengimplementasikan *chatbot* pada *order management system*.

Di keperluan bisnis *chatbot* dapat membantu pada pemecahan permasalahan yang berkaitan dengan komunikasi dengan pelanggan untuk meningkatkan pelayanan dan pengalaman dalam hal berkomunikasi, *chatbot* sangat lah efektif digunakan dalam permasalahan yang fokus dan spesifik serta dapat diprediksi (Guzman & Ines, 2016). Pada perkembangan dunia bisnis yang sangat cepat, *chatbot* menjadi salah satu alternatif solusi, ini akan memudahkan pelanggan menjangkau dan melakukan interaksi dengan bisnis.

2.3 Platform API.AI

API.AI adalah sebuah platform yang menyediakan layanan NLP (*Natural Language Processing*) dan NLU (*Natural Language Understanding*). Layanan ini digunakan untuk membuat *chatbot* lebih cerdas dan dapat memahami maksud dari apa yang ditanyakan oleh *user*. *Natural language Processing* adalah salah satu disiplin ilmu dari *Artificial Intelligence* yang fokus terhadap interaksi manusia dan komputer melalui bahasa alami yang manusia gunakan. Dalam NLP tujuan yang ingin dicapai adalah kemampuan sebuah sistem NLP memiliki pengetahuan bahasa alami baik dari susunan kalimat, arti dari kata tersebut dan maksud dari sebuah kalimat. Sedangkan NLU sendiri adalah merupakan sub bidang dari NLP, di mana fokus tujuan dari NLU itu sendiri adalah untuk melakukan pemahaman terhadap suatu kalimat dan melakukan analisis semantik. API.AI ini sendiri tidak mendukung domain pengetahuan bahasa Indonesia namun memiliki banyak sekali domain pengetahuan dalam bahasa inggris. Domain adalah koleksi pengetahuan dan struktur data. Di dalam API.AI sendiri telah banyak basis pengetahuan yang sudah tertanam di sistem layanan API.AI ini. Koleksi pengetahuan yang terdapat pada layanan ini di antaranya adalah *entitas* musisi, bandara, dll. Namun tiadanya dukungan bahasa indonesia di dalam layanan ini bukan berarti layanan ini tidak dapat digunakan oleh peneliti yang notabene berbahasa indonesia. Untuk bagaimana penulis menggunakan layanan ini akan dibahas di Bab 3 Metodologi Penelitian pada bagian perancangan *chatbot*.

2.3.1 Sejarah Singkat

API.AI yang dulunya adalah Speaktoit adalah pengembang teknologi interaksi komputer dan manusia. Fokus teknologi yang digunakan adalah dalam bidang *natural language processing* atau NLP. Perusahaan ini telah mengembangkan beberapa produk yang memiliki fitur utama melakukan tugas tanya jawab dengan bahasa alami antara manusia dan komputer.

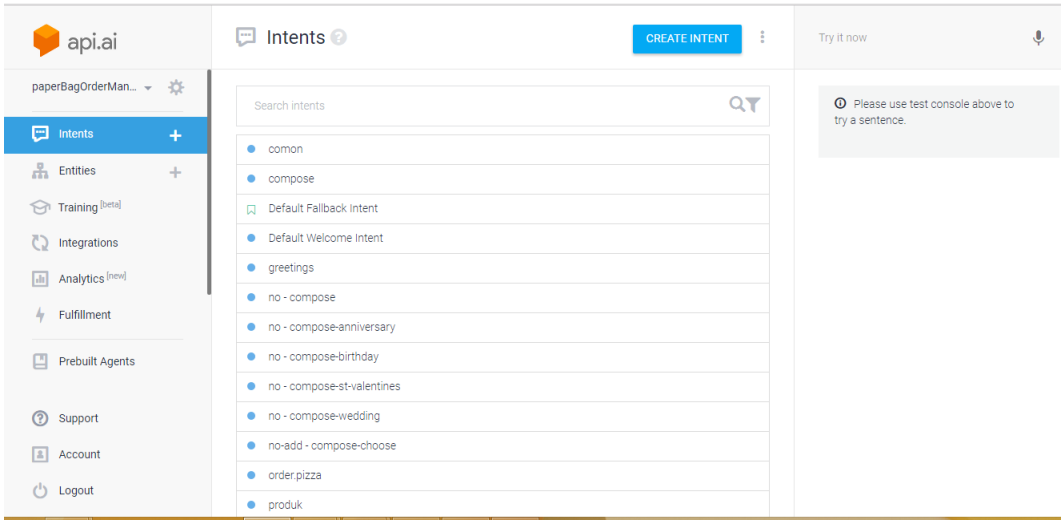
Pada bulan September 2014 Speaktoit memperkenalkan sebuah *platform* yang diberi nama API.AI. Platform inilah yang banyak digunakan oleh pengembang pihak ketiga untuk membuat sebuah *chatbot.* Selain itu API.AI menyediakan kepada pengembang fitur-fitur lainnya yaitu sebuak SDK (*Software Development Kit)* yang memiliki fungsi dapat menambahkan pengenalan suara, pemahaman bahasa alami dan konversi *text-to-speach* untuk aplikasi Android, dan IOS. API.AI juga memiliki halaman *website* yang digunakan untuk melakukan pengujian skenario yang dapat digunakan para *developer* untuk menguji skenario

yang telah dirancang sebelumnya. Pada bulan September tahun 2016 Google membeli perusahaan ini. Google menggunakan layanan ini untuk asisten *virtual* Google.

2.3.2 Lingkungan Pengembangan

Untuk dapat menggunakan layanan ini, maka penulis perlu melakukan observasi dan belajar untuk mengetahui lingkungan pengembangan dari platform API.AI. Penulis juga melihat beberapa tutorial untuk dapat membangun sebuah *chatbot* yang terintegrasi dengan platform ini.

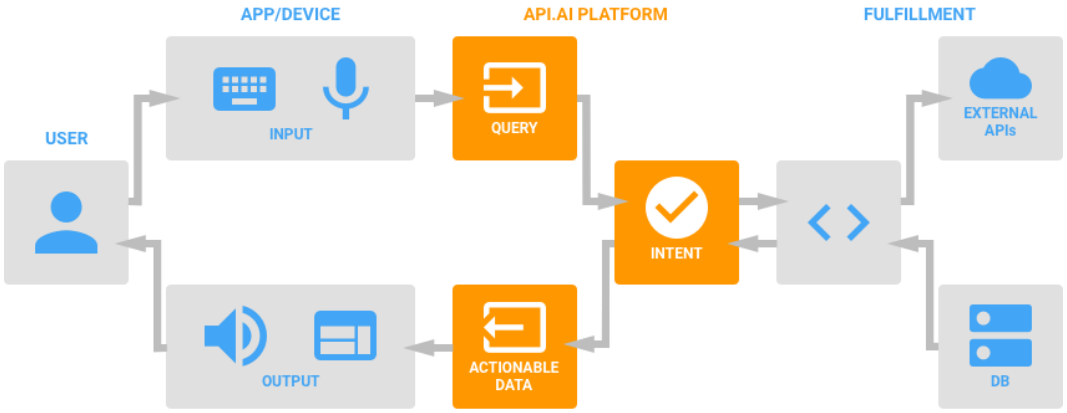
Ada beberapa hal yang perlu penulis ketahui untuk dapat menggunakan layanan ini. Langkah pertama yang penulis lakukan adalah untuk mengetahui bagaimana cara kerja dasar untuk menggunakan layanan ini. Gambar 2.1 menunjukkan halaman *dashboard* dari *platform* API.AI



Gambar 2.1 Gambar *Dashboard* Utama *Platform* API.AI

Gambar 2.1 di atas merupakan halaman dashboard utama dari platform API.AI ini, di dashboard tersebut *developer* dapat merancang *Conversational Flow* dari permasalahan yang ingin dipecahkan

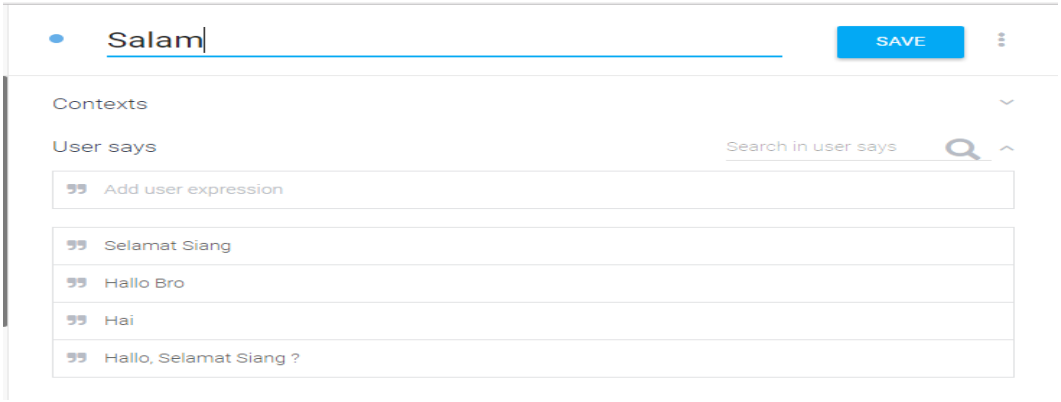
Proses awal yang dilakukan adalah dengan adanya *request* yang datangnya dari *user* dengan melakukan permohonan ke *platform*, lalu sistem didalam platform ini akan memberikan *response* ke *user* yang terkait bahwa permohonan untuk menggunakan layanan diterima, *user* disini dapat dikatakan sebagai *developer* yang merancang sebuah *chatbot*. Gambar 2.2 menunjukkan illustrasi mengenai platform ini.



Gambar 2.2 Alur Kerja *Platform* API.AI

Gambar 2.2 di atas merupakan gambaran umum mengenai cara kerja di dalam *platform* API.AI ini. Pertama-tama dimulai dari *user* yang mengirimkan *request* berupa teks ataupun suara ke dalam *platform* API.AI ini, kemudian *query* atau permintaan tadi akan diproses di dalam *intents* untuk memetakan permintaan tersebut dan tindakan apa yang harus dilakukan. Selanjutnya apabila terdapat informasi tambahan yang akan diberikan sebagai *response, platform* ini akan mengirimkan *fullfillment*, ini akan mendapatkan informasi dari sumber daya luar. Lalu *user* akan mendapatkan jawaban dari pertanyaan tersebut.

Pada langkah awal *developer* akan membuat sebuah *Agent* yang akan digunakan sebagai modul. *End-user* atau *user* yang akan memanfaatkan layanan *chatbot* yang sudah terintegrasi dengan *platform* API.AI akan menanyakan sesuatu. Permintaan dari *end-user* ini akan diteruskan ke dalam *Agent* yang telah dibuat oleh developer. Di dalam *Agent* atau modul inilah permintaan yang datang akan diproses. Agar *Agent* dapat memahami maksud dari permintaan yang dikirim, *Agent* perlu memiliki beberapa sampel yang berkaitan dengan pertanyaan tersebut. Oleh karena itu *developer* perlu mendefinisikan terlebih dahulu persamaan pertanyaan yang akan memiliki kesamaan dengan pertanyaan yang dikirim *end-user* ke dalam sebuah *Intent.* Semakin banyak variasi pertanyaan yang didefinisikan maka akan sangat membantu sistem untuk menentukan jawaban yang tepat untuk dikirimkan sebagai jawaban pertanyaan. Gambar 2.3 menunjukkan cara pendefinisian pertanyaan didalam sebuah *intent.*



Gambar 2.3 Pembuatan *Intents*

Gambar 2.3 di atas penulis mencoba membuat sebuah *intent* yang diberi isian "Salam" , lalu setelah itu penulis membuat sebuah kata-kata yaitu "Selamat Siang", "Hallo Bro", "Hai", "Hallo, Selamat Siang?". Kata-kata tersebut merupakan sebuah perkiraan pertanyaan yang memiliki kemungkinan besar *end-user* tanyakan. *Intent* ini dibuat dengan tujuan untuk menampung segala pertanyaan yang berkaitan dengan salam atau sapaan, apabila *end-user* mengirim pertanyaan salam atau sapaan, dan apabila sesuai dengan pertanyaan yang sudah ada di basis data seperti di atas maka *Agent* akan dengan mudah untuk menentukan maksud dari pertanyaan tersebut dan dapat mengirim jawaban dengan tepat. Apabila pertanyaan sapaan yang dikirim belum ada di basis data tersebut *developer* dapat mendefinisikan ke dalam daftar tersebut.

Hal yang telah dijelaskan sebelumnya belum cukup untuk membuat *Agent* dapat memberikan jawaban yang tepat, namun untuk lebih memahami lebih lanjut tentang *platform* ini maka dari itu penulis akan menjabarkan istilah-istilah yang penulis pilih dan penting di dalam *platform* ini.

a. *Agents*

Agents dapat juga disamakan dengan sebuah modul. *Agents* inilah yang akan mengelola seluruh *conversation flow*. Di dalam *Agents* ini terdapat sekumpulan *intents, dan entities*.

b. *Intents*

Merupakan sebuah tempat untuk memetakan pertanyaan apa yang dikirim oleh pengguna dan tindakan apa yang harus dilakukan. Tujuan dari *intents* adalah mendifinisikan tata bahasa percakapan dan tugas apa yang harus dilakukan saat pengguna menggunakan frasa tertentu.

c. *Entities*

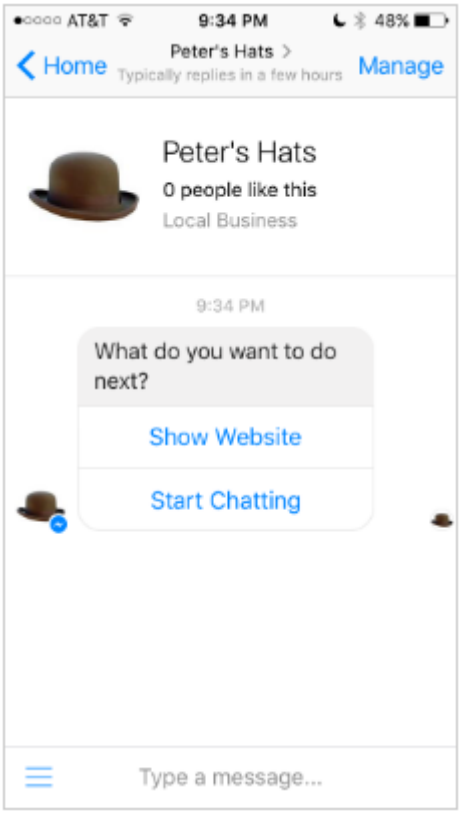
Entities adalah sebuah alat yang sangat kuat untuk mengindentifikasi nilai yang berguna dari sebuah masukan bahasa alami dalam hal ini adalah pertanyaan yang dikirim oleh *user*. Di dalam *platform* terdapat 3 jenis *entities*, yaitu *System Entities, Developer Entities*, dan *User Entities*. *System Entities* merupakan *entities* yang sudah dibenamkan di dalam sistem, artinya *entities* tersebut sudah tersedia, *Developer Entities* merupakan *entities* yang dibuat dan didefinisikan oleh *developer*, *User Entities* adalah yang didefinisikan untuk setiap *user* pada setiap permintaan.

2.4 Facebook *Messenger*

Facebook *Messenger* adalah layanan pesan instan yang dibuat oleh Facebook. Aplikasi ini memiliki beberapa fitur-fitur yang membuatnya menjadi sangat interaktif dan menarik untuk digunakan oleh *user*. Aplikasi ini dapat bertukar pesan antar sesama pengguna, bertukar foto atau video, stiker, audio, file, dan juga layanan panggilan suara dan panggilan video. Aplikasi ini telah tersedia dalam bentuk *standalone apps* untuk berbagai macam sistem operasi di antaranya Windows 10, Android, dan IOS. Tidak hanya itu aplikasi Messenger juga dapat diakses melalui *web browser*.

Messenger memiliki 600 juta pengguna pada tahun 2015. Jumlah tersebut terus meningkat sampai padai bulan April 2017 pengguna aplikasi ini mencapai 1,2 milyar pengguna. Selain digunakan untuk mengirim pesan antar satu *user* ke *user* lainnya, Messenger telah menyediakan fasilitas untuk sebuat *bot* atau robot *chat* yang dapat digunakan dan dibuat oleh para *developer* dengan memanfaatkan layanan-layanan yang ada di dalam Mesenger ini.

Aplikasi Messenger ini juga menyediakan layanan untuk membuat *user interface* pesan, sehingga tidak menimbulkan kesan monoton yang hanya diisi dengan teks melainkan berupa grafik yang menarik. Berikut ini akan dijelaskan beberapa fitur-fitur yang dapat digunakan oleh *develope*r untuk membangun sebuah *bot* menarik dengan menggunakan layanan *restfull api* yang sudah tersedia dari layanan Messenger ini. Di antaranya adalah fitur template, yang disebut sebagai *Generic Message* atau pesan terstruktur. Template yang tersedia di dalam layanan ini memiliki beragam jenis, di antaranya adalah template tombol di mana fitur ini akan menampilkan tombol di dalam antarmuka Messenger. Gambar 2.4 menunjukkan contoh template tombol.



Gambar 2.4 *Template* Tombol Messenger

Gambar 2.4 di atas merupakan contoh penggunaan template tombol yang dapat digunakan developer untuk membuat user interface yang interaktif dan menarik user.

2.5 Penggunaan API (Application Program Interface) Messenger

Untuk dapat membuat sebuah bot di Messenger terlebih dahulu penulis mempelajari bagaimana menggunakan *Application Program Interface* yang telah disediakan oleh Messenger. Diantara yang penulis pelajari adalah bagaimana mendapatkan informasi pengguna beserta bagaimana cara mengirimkan pesan kepada pengguna. Kode di bawah ini menunjukan cara untuk mengakses informasi pengguna dengan USER_ID yang telah ditentukan sebelumnya. Dengan menggunakan HTTP *request* untuk mengakses API, keluaran dari proses ini akan menampilkan informasi pengguna berupa nama depan, nama belakang, dan foto.

```
curl -X GET "https://graph.facebook.com/v2.6/<
USER_ID>?fields=first_name,last_name,profile_pic,locale,timezone,gender&access_token=PAGE_ACCESS_TOKEN"
```

Gambar 2.5 *Request* Informasi Pengguna

Untuk mengirim sebuah pesan kepada *user* menggunakan HTTP Request adalah dengan cara mengirimkan permintaan ke layanan API Messenger. Gambar 2.6 ini adalah kode untuk mengirimkan pesan. Sebelumnya kode juga harus menyertakan USER ID dan *Page Access Token.*

```
curl -X POST -H "Content-Type: application/json" -d '{
"recipient": {;
    "id": "USER_ID"
  },
  "message": {
    "text": "hello, world!"
  }
}' "https://graph.facebook.com/v2.6/me/messages?access_token=PAGE_ACCESS_TOKEN"
```

Gambar 2.6 *Request* Untuk Mengirim Pesan

2.6 Penelitian Terdahulu

Order Management System bukanlah hal yang baru di dalam sebuah bisnis. Terdapat beberapa penelitian serupa yang penulis temukan. Pada penelitian ini penulis membaca dan mempelajari penelitian-penelitian sebelumnya sebagai referensi penulis untuk mengembangkan sistem pengelolaan pesanan ini. Berikut ini adalah beberapa penelitian serupa yang penulis rangkum ke dalam beberapa paragraf.

Adipura, dkk (2015) melakukan penelitian dengan judul "Perancangan Order Management System Berbasis Web Application Pada Usaha Mikro Dan Kecil Menengah Menggunakan Metode Waterfall" untuk mengembangkan sebuah sistem di sebuah usaha mikro kecil dan menengah "Sugoimasa". Sugoimasa adalah usaha yang bergerak di bidang kuliner, usaha ini memiliki permasalahan terhadap pengelolaan pesanan yang belum benar-benar baik. Atas permasalahan tersebut Yafshil Adipura dan teman-teman melakukan penelitian untuk memecahkan permasalahan tersebut dengan mengembangkan *Order Management System* berbasis *web* dengan menggunakan metode *Waterfall, OMS* yang dikembangkan memiliki fitur-fitur melakukan pencatatan terhadap pesanan, pemantauan terhadap pesanan, dan pelaporan.

Bora dan Gupta (2012) melakukan penelitian yang berjudul "Application On Order Management System In Restaurant" untuk membuat sebuah *Food Order Management System* untuk *restaurant.* Dengan memanfaatkan *local area network* yang ada di *restaurant,* pelanggan dapat mengakses layanan pemesanan makanan dari sebuah *server* yang sudah tersedia di *restauran* tersebut melalui peralatan *mobile* ataupun computer.

Chinmay dan Himil (2012) melakukan sebuah penelitian yang berjudul "Online Sales Order Management System For Riwasa Tiles" pada sebuah perusahaan bernama "Riwasa tiles". Pada penelitian ini perusahaan belum memiliki *order management system.* Untuk informasi produk, perusahaan masih menggunakan katalog sedangkan kegiatan pemesanan barang pelanggan harus datang secara langsung untuk memesan barang. Setelah melakukan analisis, hasil dari penelitian ini adalah membuat sebuah *order management system* yang dapat menampilkan informasi produk yang tersedia secara online, pemesanan secara online, dan pelaporan penjualan.

Oupraxay dan Diego (2010) melakukan sebuah penelitian yang berjudul "Android Based Mobile Order Management System". Pada penelitian ini penulis mengungkapkan beberapa permasalahan yang terdapat pada sebuah sistem pengelolaan pesanan, di mana karyawan menggunakan sebuah perangkat komputer tablet khusus. Terlihat bahwa penggunaan komputer tablet khusus memiliki kekurangan yaitu akan sangat mahal dalam hal perawatan dan pembaharuan, untuk itu di sini penulis mengembangkan sistem pengelolaan pesanan menggunakan android. Aplikasi yang dirancang akan menggunakan konsep *Client-Server* di mana terdapat pusat data yang digunakan untuk menyimpan seluruh data yang dikirim melalui *client*.

Setelah melakukan ulasan terhadap penelitian sebelumnya penulis mendapati kesimpulan bahwa semua penelitian memiliki kesamaan konsep yaitu untuk membuat *order management system*, OMS yang dibuat pada dasarnya memiliki tujuan yang sama dengan berbagai fitur-fitur penting yang ada di dalamnya seperti pencatatan pesanan, pencatatan produk, pelanggan dan pemrosesan terhadap pesanan namun dengan studi kasus yang berbeda. Adapun dalam penelitian yang penulis lakukan sekarang ini memiliki sebuah pembeda dari penelitian yang dilakukan sebelumnya di mana penulis akan mengimplementasikan *chatbot* pada sistem pengelolaan pesanan yang akan memudahkan pelanggan untuk menanyakan produk ataupun mendapatkan informasi dan notifikasi dari sistem pengelolaan pesanan. Berikut ini merupakan tabel tambahan yang penulis buat untuk menunjukkan perbedaan antara setiap penelitian pada Tabel 2.1.

Tabel 2.1 *Summary* Penelitian Terdahulu

No	Judul Penelitian	Tahun	Penulis	Studi Kasus	Menerapkan ***Chatbot***
1	Perancangan Order Management System Berbasis Web Application Pada Usaha Mikro Dan Kecil Menengah Menggunakan Metode Waterfall	2015	Adipura dan kawan-kawan	UMKM Sugoimosa	Tidak menerapkan *chatbot*
2	Application On Order Management System In Restauran	2012	Bora dan Gupta	*Restaurant*	Tidak menerapkan *chatbot*
3	Online Sales Order Management System For Riwasa Tiles	2012	Chinmay dan Himil	Riwasa Tiles	Tidak menerapkan *chatbot*
4	Android Based Mobile Order Management System	2010	Oupraxay dan Diego	-	Tidak menerapkan *chatbot*

BAB III
METODOLOGI PENELITIAN

3.1 Analisis Sistem

Analisis dilakukan untuk melakukan penggalian informasi secara terperinci dan melakukan penguraian terhadap data-data ke dalam komponen-komponen yang dibutuhkan untuk sistem yang akan dibangun, sehingga di harapkan nantinya sistem yang akan dibangun memiliki kesesuaian dengan keinginan yang akan dicapai. Proses analisis ini penulis laksanakan dengan melakukan observasi dan wawancara langsung ke HDKreasi untuk menggali informasi yang di butuhkan.

Sistem Pengelolaan Pesanan yang mengintegrasikan *chatbot* untuk keperluan *customer service* pada studi kasus di usaha mikro kecil dan menengah HDKreasi adalah sistem yang dapat memperbolehkan usaha ini untuk dapat melakukan pengelolaan terhadap pesanan. Sistem ini akan dapat melakukan pencatatan pesanan, pemrosesan pesanan, pencatatan pelanggan dan informasi atau laporan mengenai transaksi penjualan, serta informasi-informasi lainnya. Dengan analisis yang telah dilakukan penulis dapat menentukan kebutuhan yang berkaitan dengan sistem tersebut. Mulai dari kebutuhan akan *input* sistem, proses-proses yang ada di dalam sistem dan *output* sistem. Sehingga sistem yang dibuat akan sesuai dengan yang diharapkan.

3.1.1 Analisis Kebutuhan Masukan

Kebutuhan masukan di dalam sistem ini adalah masukan yang dilakukan oleh *user.* Terdapat 3 tipe *user* yang terdiri dari *admin* , pegawai dan pelanggan.

a. Kebutuhan masukan *admin*

Masukan yang akan dilakukan oleh *admin* HDKreasi di dalam sistem ini. Data yang akan dimasukkan antara lain

1. *Password* dan *username. Admin* menggunakan *password* dan *username* untuk dapat masuk ke dalam sistem.
2. Periode dan pengaturan sistem. Data periode ini berisi data bulan dan tahun untuk mengatur sistem.
3. Pegawai. Data pegawai ini berisi data pegawai dengan berbagai atribut seperti nama username dan password.

b. Kebutuhan masukan pegawai

Kebutuhan masukan pegawai merupakan masukan yang akan dilakukan oleh pegawai di dalam sistem ini.

1. Data produk adalah data-data produk yang diperjualbelikan oleh HDKreasi. Di dalam data produk ini berisi informasi mengenai jumlah *stock* produk yang tersedia dan harga yang ditentukan.
2. Data pelanggan adalah data yang berisi informasi pelanggan yang melakukan transaksi pembelian produk.
3. Data *order* ini berisi tentang transaksi penjualan HDKreasi kepada para pelanggannnya yang pernah dilakukan dan seberapa banyak pemesanan produk oleh pelanggan
4. Data *invoice* merupakan data surat penagihan pembayaran terhadap pelanggan yang sudah melakukan transaksi dengan HDKreasi.
5. Data *Shipment* merupakan data mengenai pesanan yang sudah dalam proses pengiriman.

c. Kebutuhan masukan pelanggan

Kebutuhan masukan pelanggan merupakan masukan yang akan dilakukan oleh pelanggan di dalam sistem ini. Data yang akan dimasukkan antara lain:

1. Data pelanggan adalah data yang berisi pelanggan yang melakukan transaksi pembelian produk.

3.1.2 Analisis Kebutuhan Proses

Analisis kebutuhan proses ini bertujuan untuk mengetahui proses apa saja yang terjadi pada Sistem Pengelolaan Pesanan di HDKreasi. Berikut adalah beberapa proses yang teridentifikasi dari tahapan analisis:

a. Proses pengelolaan data produk.
b. Proses pengelolaan data pelanggan.
c. Proses pengelolaan data pesanan.oup
d. Proses pengelolaan data *invoice.*
e. Proses pengelolaan data *shipment.*
f. Proses pengelolaan periode dan system.

3.1.3 Analisis Kebutuhan Keluaran

Kebutuhan keluaran yang terjadi pada Sistem Pengelolaan Pesanan HDKreasi ini adalah:

a. Informasi data produk.
b. Informasi pelanggan.
c. Informasi pesanan.
d. Informasi pegawai.
e. Informasi invoice.
f. Informasi shipment.
g. Informasi periode.

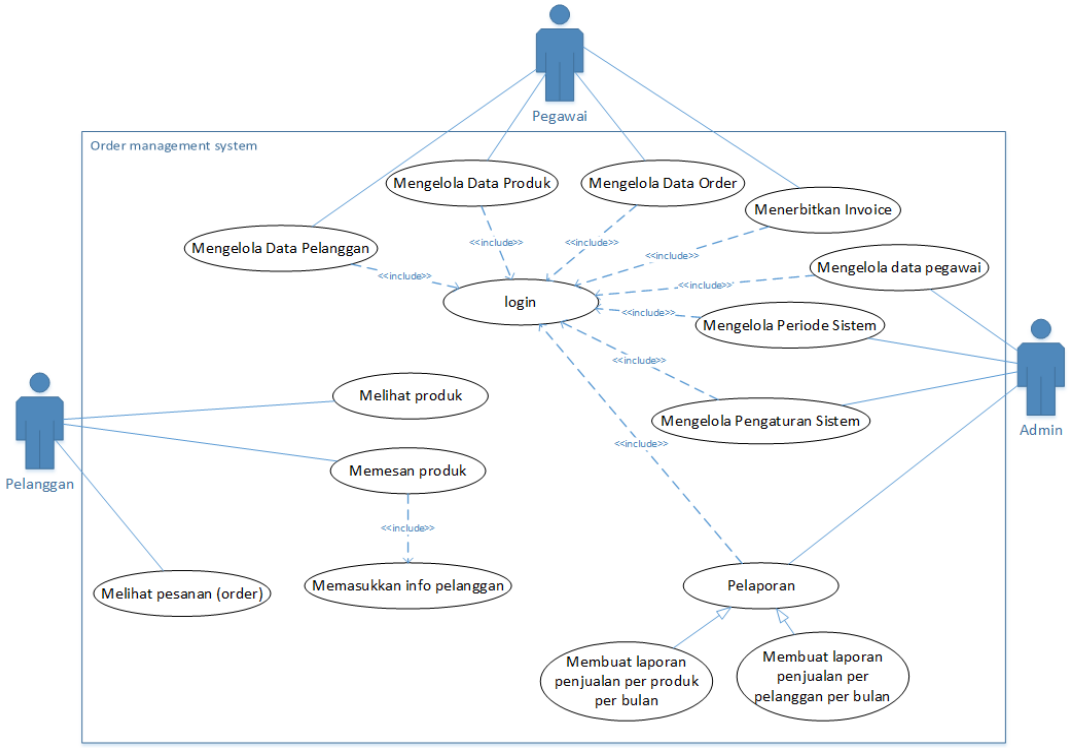
3.2 Perancangan Sistem

Dalam pembuatan perangkat lunak diperlukan adanya perancangan yang baik, agar perangkat lunak yang dibangun memiliki dokumentasi yang baik pula untuk menghasilkan sistem yang memiliki kebutuhan yang sesuai dengan yang pengguna inginkan. Pada penelitian ini perancangan dibagi menjadi 5 tahap, yaitu perancangan *use case* diagram, perancangan *activity* diagram, perancangan *database*, perancangan arsitektur sistem, perancangan *chatbot* dan perancangan antarmuka sistem.

3.2.1 Perancangan Fungsionalitas (Use Case Diagram)

Use Case Diagram atau yang sering disebut sebagai diagram perilaku digunakan untuk menggambarkan secara ringkas keterhubungan antara *case*, aktor dan sistem. *Use Case Digram juga* dapat digunakan untuk memahami bagaimana suatu sistem bekerja. Di dalam *use case diagram* itu sendiri memiliki sekumpulan aksi *(use case)* yang berkolaborasi atau memiliki hubungan dengan entitas yang berada di luar sistem yang disebut sebagai aktor.

Pada sistem yang akan dibuat ini *use case* menggambarkan 3 aktor yang terhubung dengan beberapa case yang berisi proses di sistem ini, diantara aktor tersebut adalah admin, *user*, dan pelanggan. Penjelasan gambar mengenai *use case diagram* yang terdapat di dalam sistem pengelolaan pesanan ini dapat dilihat pada Gambar 3.1.



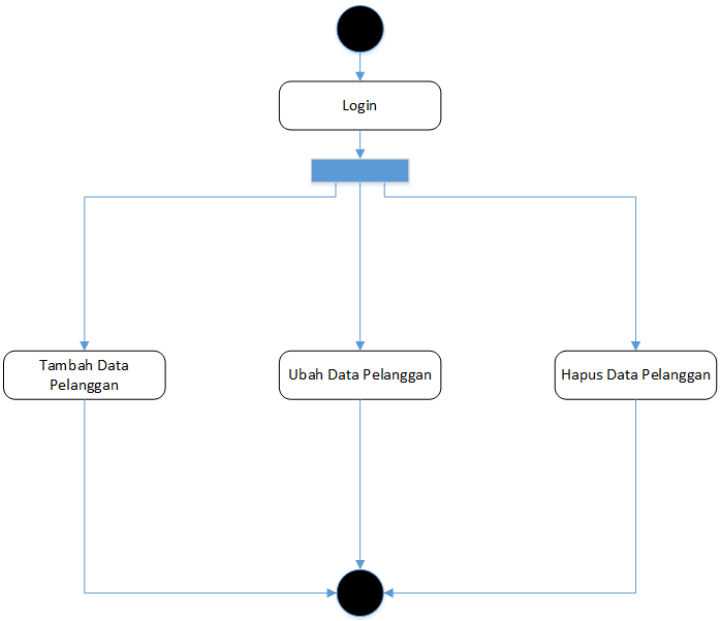
Gambar 3.1 *Use Case Diagram*

Use case diagram di atas menunjukkan terdapat 3 aktor yang ada di dalam sistem ini di antaranya adalah admin, pegawai, pelanggan. Pegawai memiliki beberapa aksi (*use case)* yang di antaranya adalah pegawai dapat melakukan pengelolaan data pelanggan, pengelolaan data produk, pengelolaan data *order*, pengelolaan data *invoice* dan *shipment*. Sedangkan *use case* yang dapat dilakukan oleh admin adalah melakukan pengelolaan data periode dan pengaturan sistem serta pelaporan penjualan. Kedua aktor tersebut yaitu admin dan pelanggan harus melakukan login terlebih dahulu untuk dapat menggunakan operasi-operasi yang ada di dalam sistem yang ditandai pada *use case* di atas dengan *use case login* dan memiliki hubungan *include.* Untuk pelanggan sendiri terdapat beberapa aksi yang dapat digunakan pelanggan yaitu melihat pesanan, melihat produk, melakukan pesanan. Pelanggan tidak melakukan *login* untuk menggunakan *use case* yang terkait dikarenakan pelanggan akan melakukan interaksi melalui *platform* pesan *Messenger*.

3.2.2 Perancangan Perilaku Sistem (Activity Diagram)

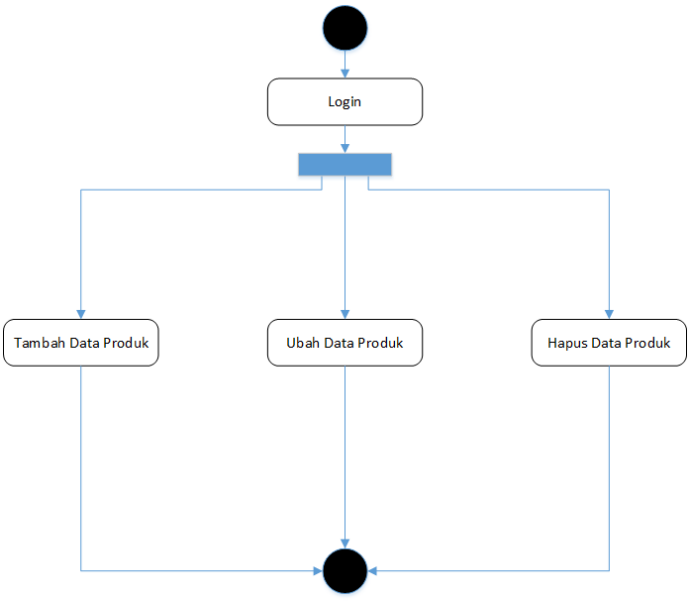
Activity Diagram merupakan aliran aktivitas yang terjadi di dalam sistem. Sebuah aktivitas dapat menggambarkan sebuah operasi yang terjadi di dalamnya dan memodelkan aksi yang akan dilakukan saat suatu operasi dijalankan, serta memodelkan hasilnya. Penjelasan

mengenai *activity diagram* untuk pegawai dalam aktifitas pengelolaan pelanggan yang ditunjukkan pada Gambar 3.2.



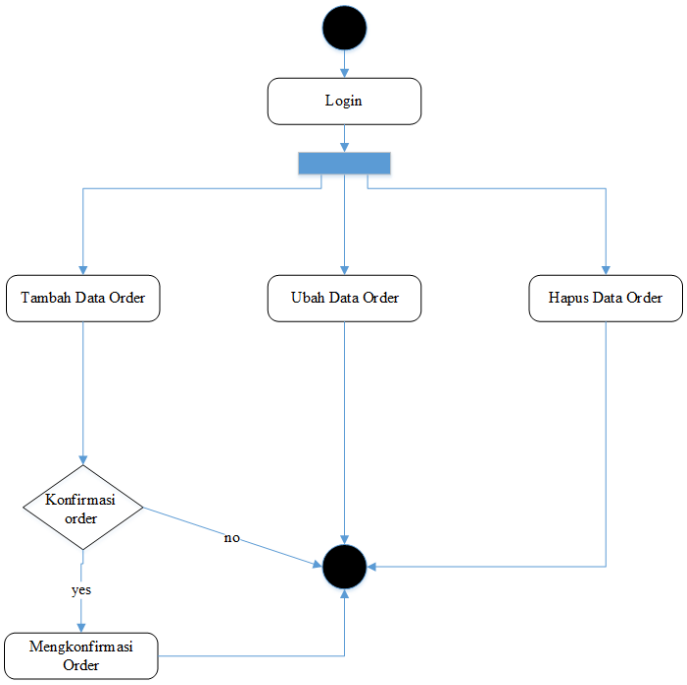
Gambar 3. 2 Diagram Aktivitas Pegawai Untuk Mengelola Pelanggan

Gambar 3.2 merupakan diagram aktifitas pegawai untuk mengelola pelanggan, disini pegawai diberi hak untuk dapat mengelola aktifitas seperti menambahkan pelanggan, mengubah data pelanggan, dan menghapus data pelanggan. Selanjutnya akan dibahas mengenai aktifitas pegawai yang lainnya yaitu aktifitas pengelolaan data produk yang ditunjukkan pada Gambar 3.3.



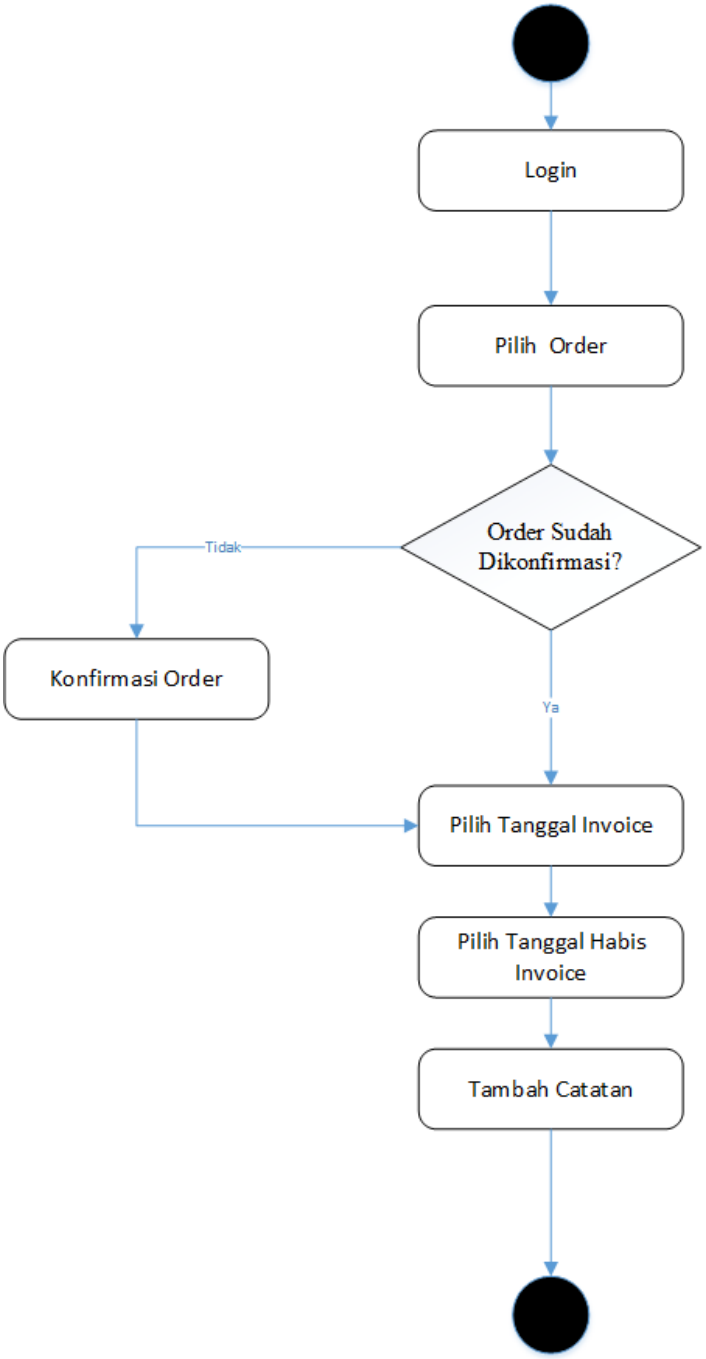
Gambar 3.3 Diagram Aktifitas Mengelola Data Produk

Setelah ditunjukkan *activity diagram* pengelolaan data produk, selanjutnya akan dibahas mengenai *activity diagram* mengenai pengelolaan data pesanan yang ditunjukkan pada Gambar 3.4.



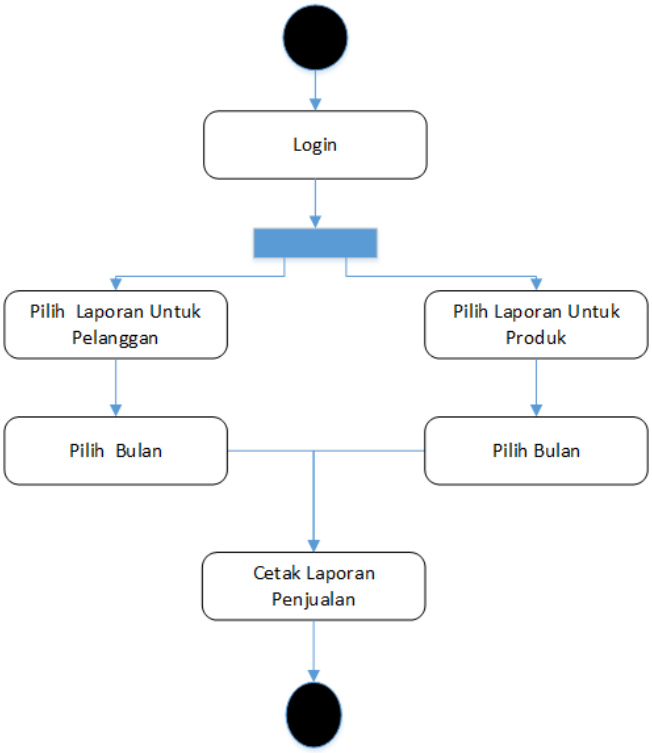
Gambar 3.4 Diagram Aktifitas Pengelolaan Data *Order*

Activity diagram Gambar 3.4 di atas merupakan aktifitas pengelolaan data *order*. Selanjutnya adalah *activity diagram* untuk aktifitas penerbitan *invoice* yang ditunjukkan pada Gambar 3.5.



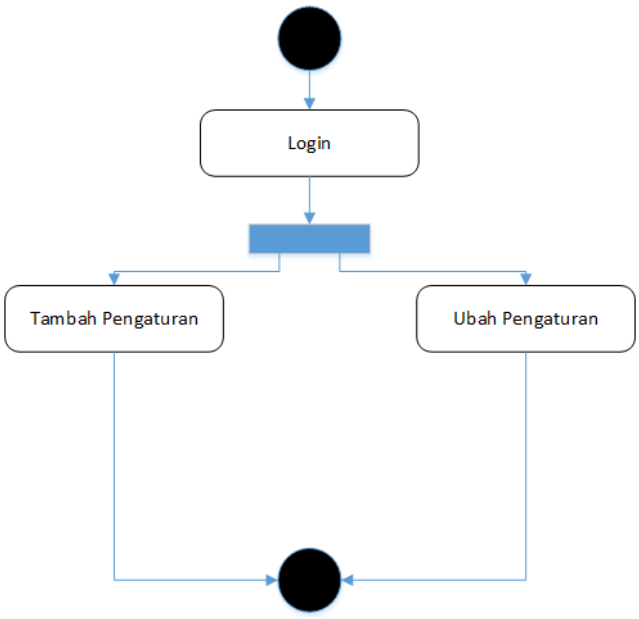
Gambar 3.5 Diagram Aktifitas Menerbitkan *Invoice*

Gambar 3.5 di atas merupakan aktifitas penerbitan *invoice*. Selanjutnya akan mulai dibahas untuk aktifitas diagram untuk admin. Yang pertama adalah aktifitas untuk mendapatkan laporan penjualan. Di dalam aktifitas ini admin dapat membuat laporan penjualan berdasarkan pelanggan ataupun produk. Gambar 3.6 di bawah ini merupakan *activity diagram* untuk pelaporan penjualan.



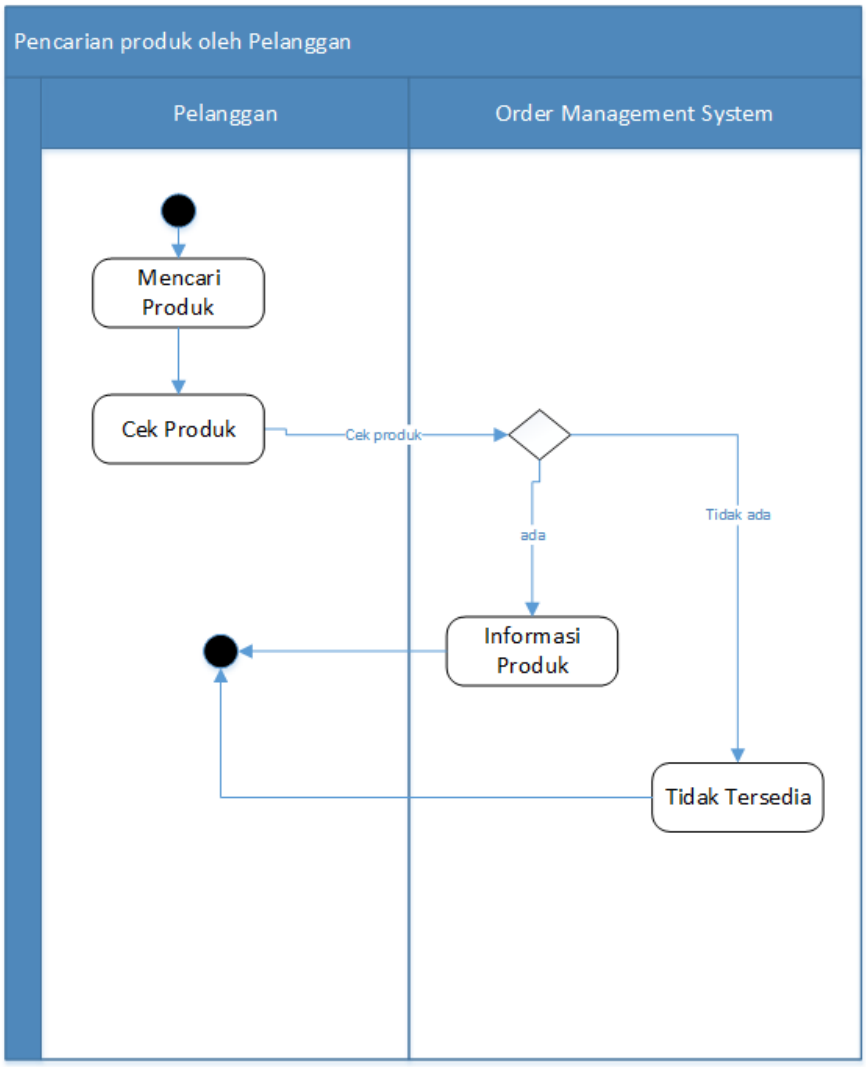
Gambar 3.6 Diagram Aktifitas Mencetak Laporan (*Admin*)

Gambar 3.6 di atas merupakan aktifitas diagram laporan. Selanjutnya adalah aktifitas diagram untuk pengaturan sistem yang ditunjukkan pada Gambar 3.7.



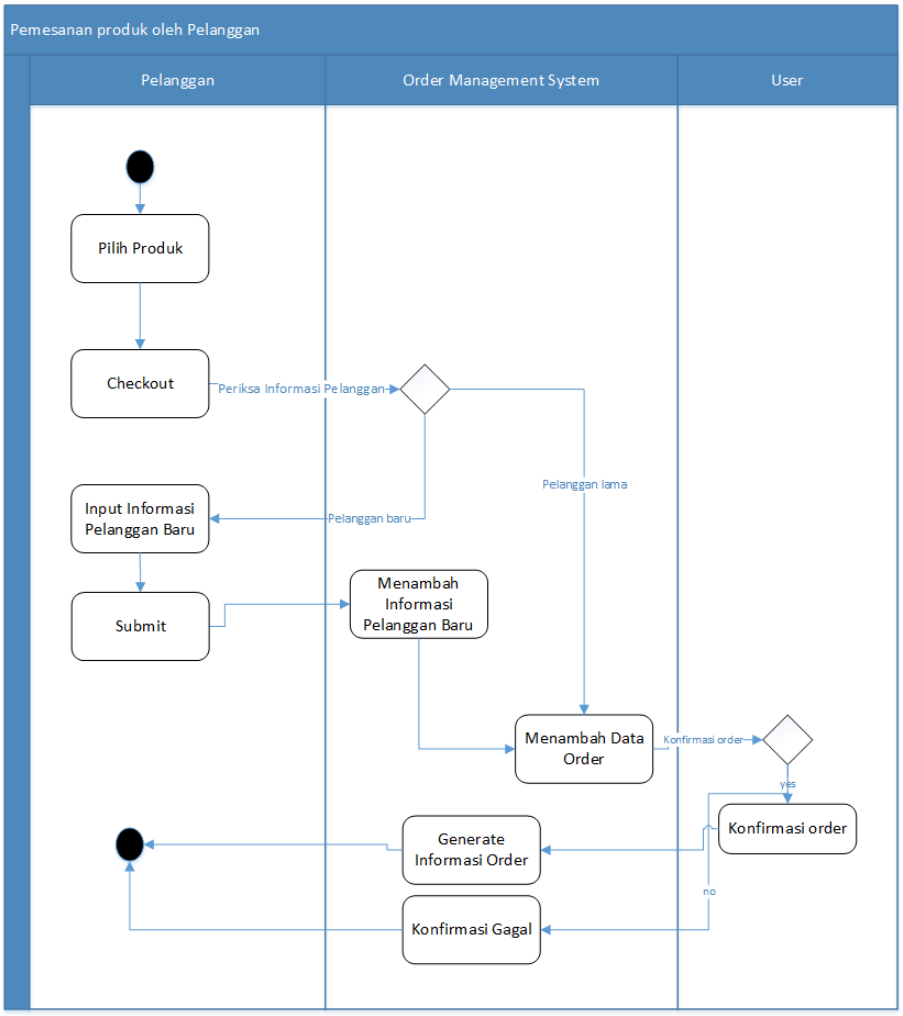
Gambar 3.7 Diagram Aktifitas Mengelola Pengaturan Sistem (*Admin*)

Gambar 3.7 di atas merupakan aktifitas pengaturan sistem. Selanjutnya adalah akan mulai dibahas untuk aktifitas diagram untuk pelanggan. Gambar 3.8 ini merupakan *activity diagram* untuk pelanggan mencari produk.



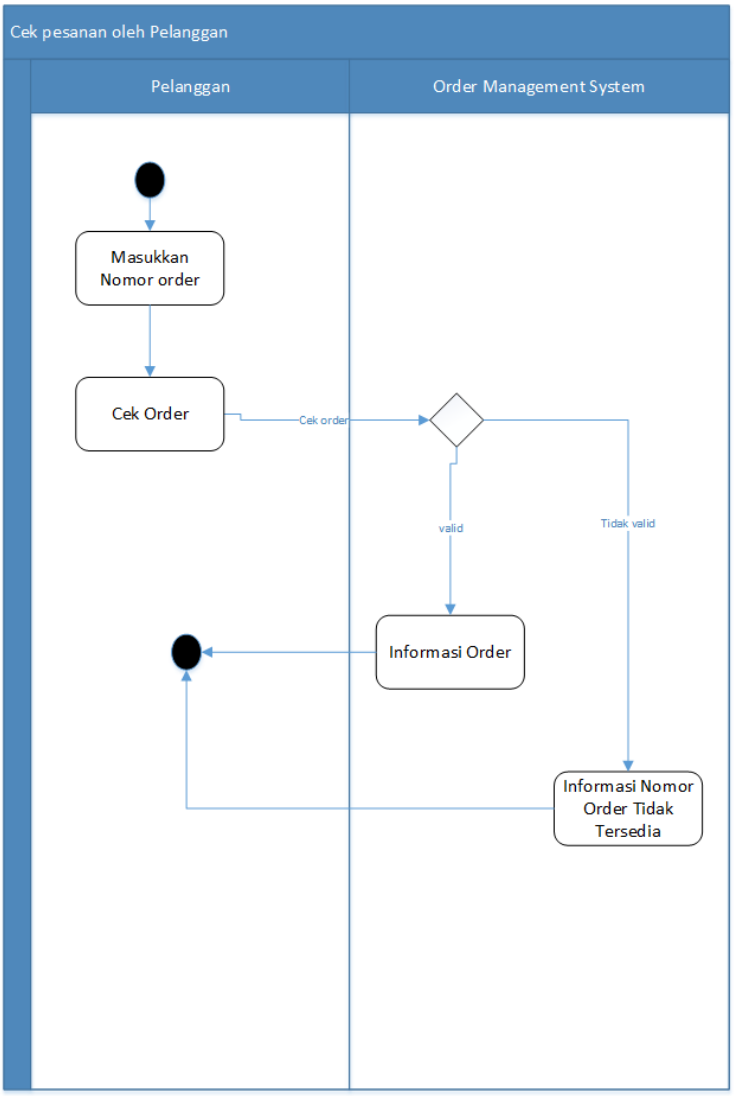
Gambar 3.8 Diagram Aktifitas Mencari Produk

Selanjutnya adalah aktifitas diagram untuk pemesanan produk yang ditunjukkan pada Gambar 3.9.



Gambar 3.9 Diagram Aktifitas Memesan Produk

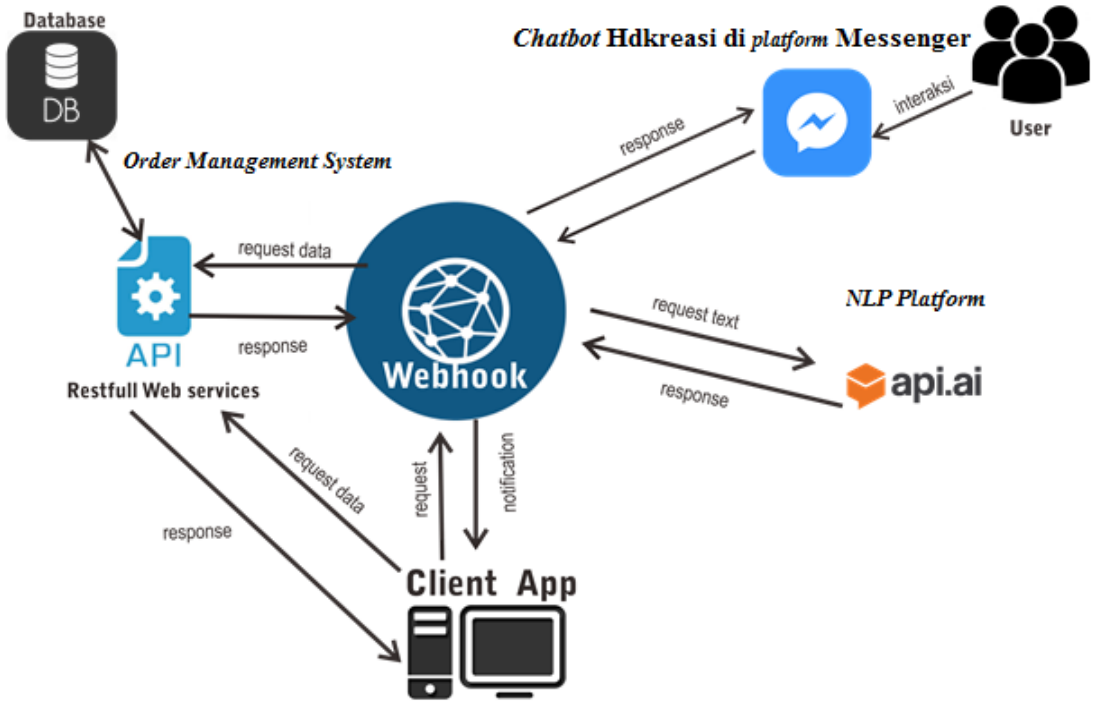
Selanjutnya adalah aktifitas diagram untuk cek pesanan yang ditunjukkan pada Gambar 3.10.



Gambar 3.10 Diagram Aktifitas Melihat Pesanan

3.2.3 Perancangan Arsitektur Sistem

Perancangan arsitektur sistem ini sendiri merupakan tahapan yang penulis lewati untuk membuat rancangan arsitektur yang terintegrasi antara sistem pengelolaan pesanan dan juga *chatbot*. Pembuatan *chatbot* akan menggunakan layanan yang sudah tersedia seperti yang sudah penulis jelaskan sebelumnya yaitu menggunakan *platfom* pesan teks dari Facebook yaitu Messenger dan layanan NLP yaitu API.AI. Agar *chatbot* dan sistem pengelolaan pesanan dapat berinteraksi maka dari itu diperlukan pula rancangan yang dapat mendukung hal tersebut. Gambar 3.11 di bawah ini merupakan gambaran secara umum arsitektur sistem yang penulis buat dan gunakan dalam hal pengembangan sistem.



Gambar 3.11 Arsitektur Sistem

Gambar 3.11 di atas merupakan rancangan arsitektur sistem pengelolaan pesanan yang terintegrasi dengan beberapa komponen lainnya seperti layanan pesan Messenger dan layanan NLP yaitu API.AI. Terdapat komponen-komponen yang berada di dalam rancangan arsitektur tersebut di antaranya adalah *restfull api web services*, *database*, *webhook*, *Messenger*, layanan API.AI, dan aplikasi di sisi user. Untuk komponen di dalam *chatbot* sendiri terdapat komponen dari *platform* atau layanan yang digunakan yaitu layanan *Messenger* dan API.AI. *Database* merupakan tempat di mana data-data disimpan, dan diolah. *Database* akan di *hosting* dan dapat diakses secara online. Untuk dapat mengelola data yang terdapat di dalam *database* maka dibutuhkan sebuah antarmuka yang dapat melakukan tugas-tugas pengelolaan data yang ada di dalam *database*. *Restfull API web service* merupakan layanan yang berbasis REST. REST (*REpresentational State Transfer)* adalah standar komunikasi berbasis web yang sering diterapkan dalam pengembangan layanan berbasis web, REST menggunakan HTTP (*Hypertext Transfer Protocol*) sebagai protocol untuk komunikasi data. Pembuatan *web services* berbasis REST ini bertujuan agar data yang terdapat di dalam *database* dapat dilakukan pengolahan dengan mengakses layanan dari *platform* apapun.

Webhook merupakan HTTP *callback.* Konsep dari *webhook* sederhana, *webhook* akan bekerja apabila terdapat suatu kejadian yang datang dari luar. Dalam hal ini adalah HTTP *post*

yang merupakan *request* yang diterima. Kelebihan dari penggunaan *webhook* untuk pengembangan aplikasi adalah dalam hal integrasi antara satu aplikasi satu dengan yang lainnya. Fungsi utama dari *webhook* itu sendiri adalah menerima data secara *realtime* dan mengirimkannya kembali. Dalam rancangan arsitektur tersebut dapat jelas dilihat bahwa *webhook* lah yang menjembatani antara *chatbot* dan sistem pengelolaan pesanan.

Dengan menggunakan *platform* pesan teks Mesenger pelanggan melakukan interaksi dengan sistem, kemudian *request text* dari pelanggan tersebut akan selanjutnya diolah di *platform* NLP API.AI. Melalui *webhook*, antara Messenger dan API.AI akan saling berinteraksi. *Request* yang berupa teks tersebut diolah di API.AI untuk dicari tahu dan diidentifikasi maksud dari *request* tersebut. Apabila misalnya *request text* tersebut merupakan pertanyaan untuk melakukan pengecekan *order* maka API.AI akan memberi identifikasi bahwa maksud dari request tersebut adalah pengecekan *order*. Layanan API tersebut akan mengirimkannya kembali ke *webhook* untuk diolah lebih lanjut. *Webhook* juga akan berkomunikasi dengan *resftull web services* untuk mendapatkan data yang dibutuhkan.

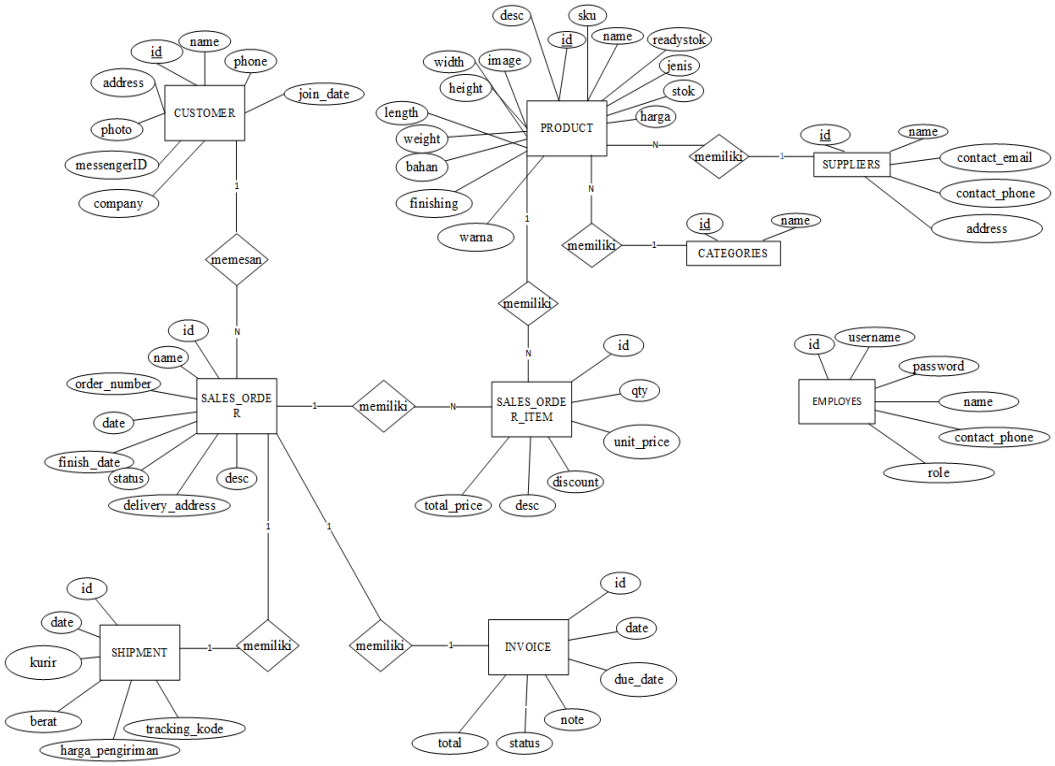
3.2.4 Perancangan Database

Perancangan *database* merupakan hal yang sangat penting dilakukan dalam penelitian ini. *Database* sebagai tempat seluruh data-data yang penting akan dikelola di dalamnya, sehingga perlu adanya perancangan yang baik agar nantinya struktur data yang terdapat di *database* sesuai dengan keinginan yang ingin dicapai.

Pada tahapan perancangan *database* ini, akan dilalui dengan beberapa tahapan yaitu dengan melakukan pemodelan struktur data yang penulis lakukan dengan menggunakan ERD (*Entitiy Relathionship Diagram)* dan selanjutnya hasil dari pemodelan tersebut penulis akan membuat model tabel-tabel yang berelasi.

a. ERD (Entity Relationship Diagram)

ERD (*Entity Relationship Diagram)* merupakan teknik yang umumnya digunakan untuk melakukan perancangan *database* dan struktur data (Li Q., 2009). ERD akan memodelkan kebutuhan data yang akan digunakan untuk kebutuhan pengembangan sistem. Pada sistem pengelolaan *order* ini, penulis telah membuat pemodelan ERD yang ditunjukkan pada Gambar 3.12 di bawah ini.

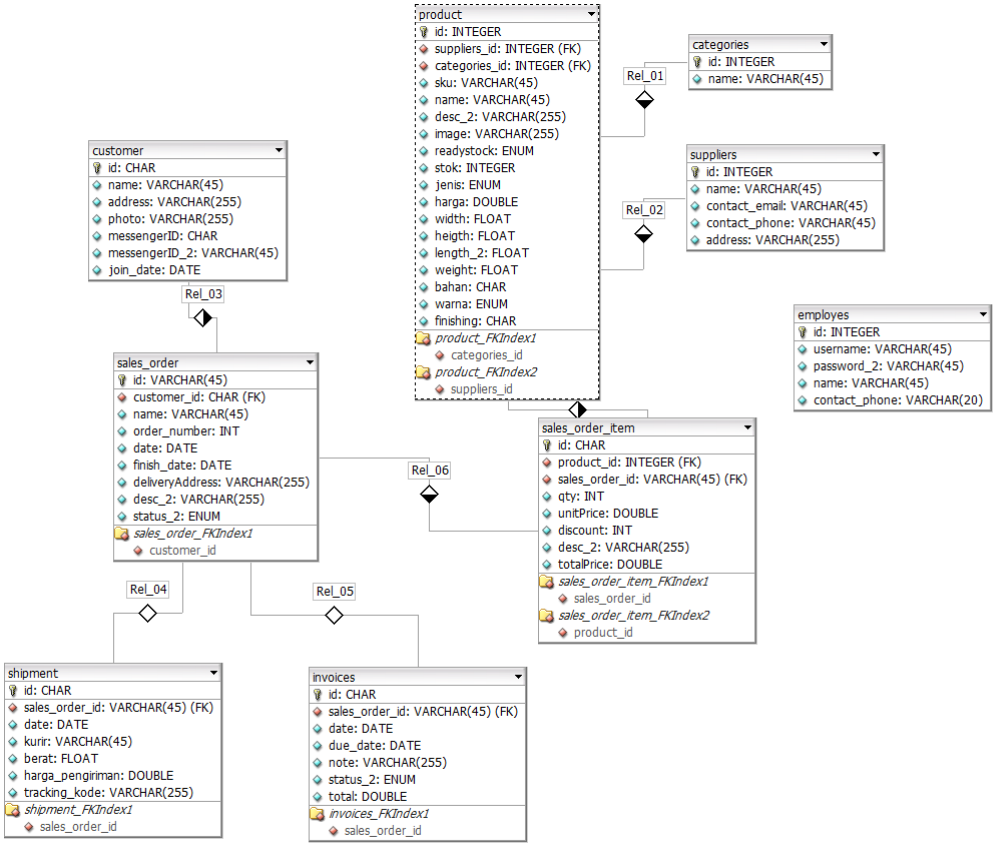


Gambar 3.12 *Entity Relationship Diagram*

Dari ERD yang ditunjukkan pada Gambar 3.12 di atas dapat diambil informasi dari entitas, atribut, *relationship* dan kardinalitas dari rancangan basisdata sistem pengelolaan pesanan. Terdapat 6 entitas yang terdapat pada ERD tersebut, yaitu *customer, product, categories, sales_order, sales_order_item, shipment, invoice*. Entitas *customer* merupakan entitas yang berisi informasi pelanggan. Entitas *product* berisi informasi produk yang disimpan di dalam *databas*e, entitas ini memiliki hubungan dengan entitas *categories* dengan kardinalitas *many to one* dengan artian bahwa entitas *product* dapat memiliki banyak categori di dalamnya. Entitas *sales_order* merupakan enttitas yang berisi data-data pesanan di dalamnya, entitas ini memiliki hubungan dengan entitas lainnya seperti *sales_order_item, shipment, invoices.* Untuk *sales_order_item* memiliki hubungan dengan *product* dengan hubungan N:1 yang berarti *product* dapat dimiliki oleh banyak *sales_order_item*.

b. Relasi Tabel

Dari ERD yang sudah dibuat sebelumnya. Penulis lalu membuat relasi tabel yang dapat digunakan untuk mengetahui alur dari *database* yang dibuat. Pada Gambar 3.13 menjelaskan mengenai relasi tabel dari sistem pengelolaan pesanan.



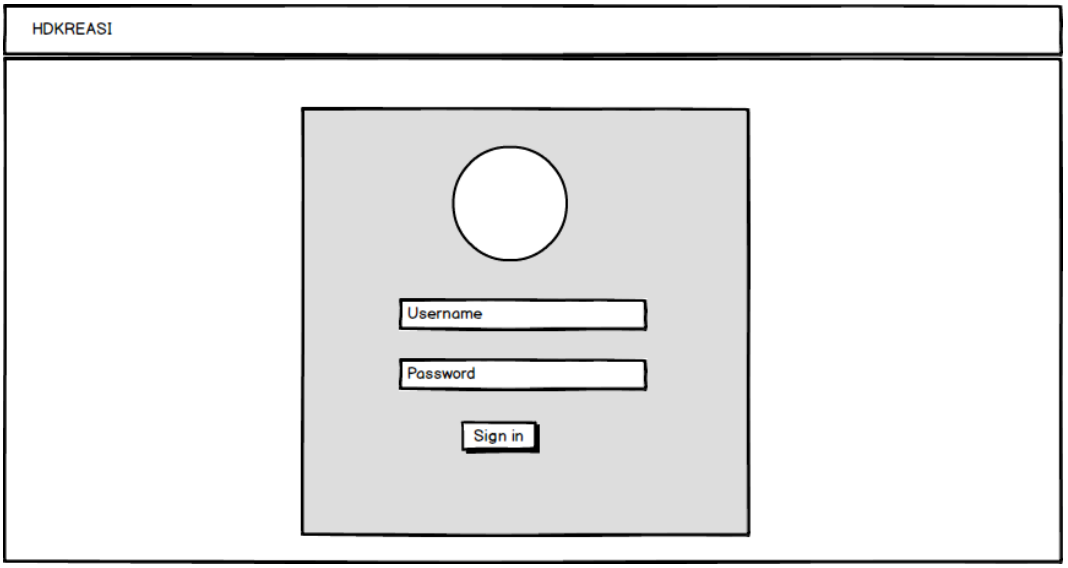
Gambar 3.13 Relasi Tabel

3.2.5 Perancangan Antarmuka

Antarmuka atau *interface* merupakan salah satu hal penting yang dapat mempermudah pengguna untuk berinteraksi dengan sistem. Pengguna akan mendapatkan informasi diinginkan pengguna dengan mudah sesuai dengan *input* interaksi yang diberikan pengguna ke dalam sistem.

a. Halaman Login

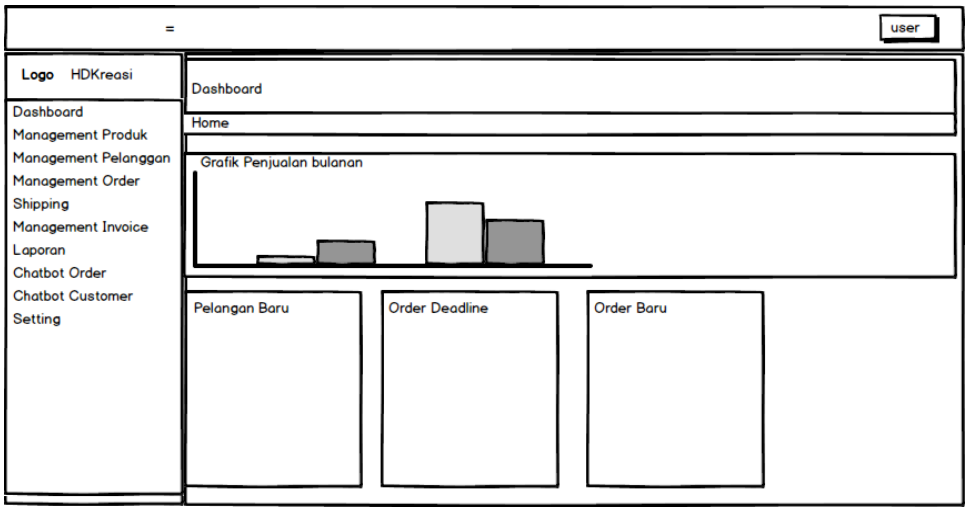
Halaman Login digunakan untuk melakukan verifikasi admin dan pegawai yang akan menggunakan dan masuk ke dalam sistem. Data yang dimasukkan kemudian dicocokkan dengan data yang terdapat di dalam *database*. Rancangan halaman login ditunjukkan pada Gambar 3.14.



Gambar 3.14 Halaman *Login*

b. Halaman Utama

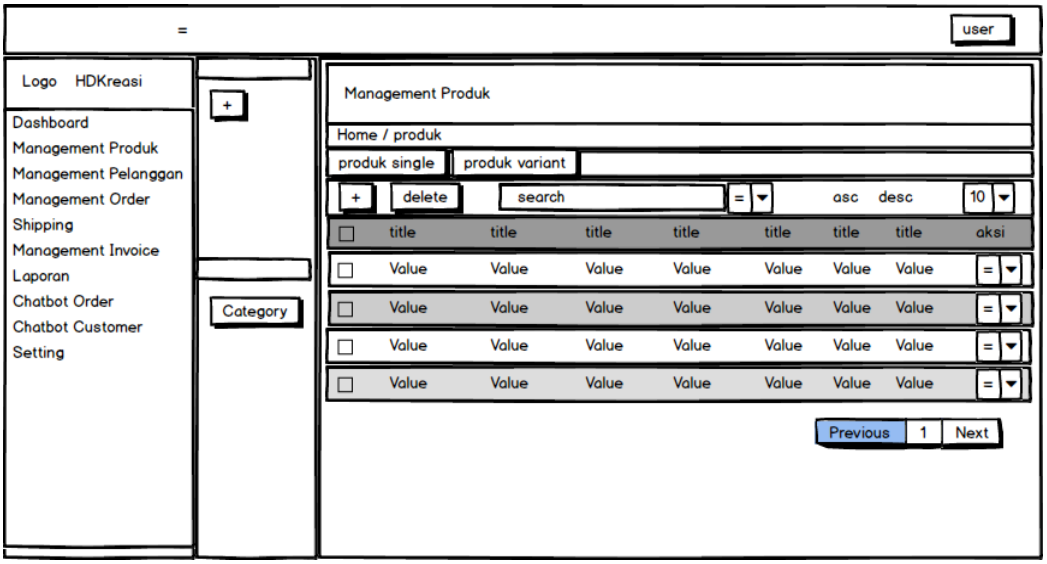
Halaman utama adalah halaman sambutan, halaman ini yang pertama kali akan dilihat oleh pengguna. Halaman ini berisi informasi-informasi umum di antaranya adalah grafik penjualan bulanan, pesanan yang sudah hampir melewati batas pengerjaan. Rancangan halaman utama ditunjukkan pada Gambar 3.15.



Gambar 3.15 Halaman *Dashboard*

c. Halaman Produk

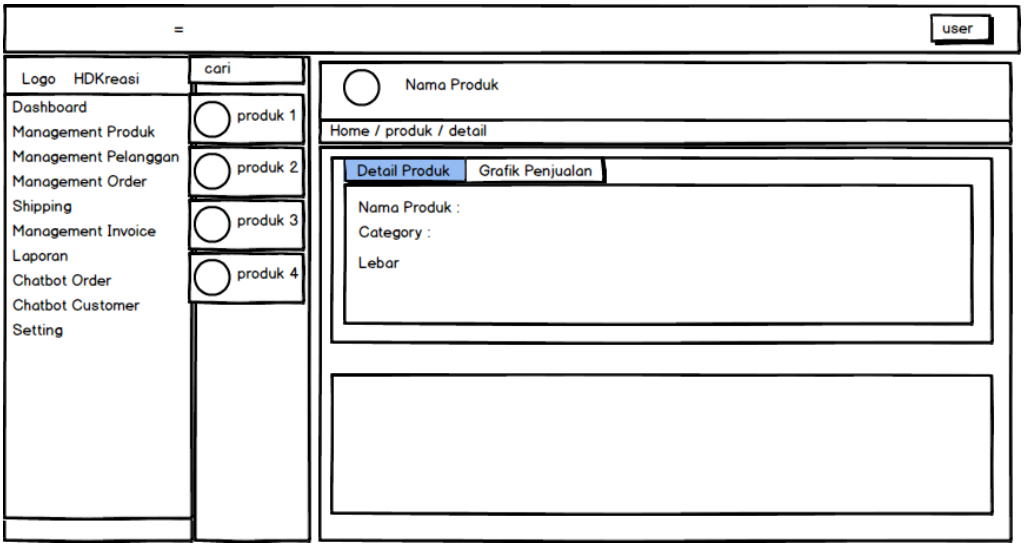
Halaman ini digunakan untuk melakukan pengelolaan terhadap produk-produk apa saja yang dijual oleh UMKM ini . Rancangan halaman produk dapat dilihat pada Gambar 3.16.



Gambar 3.16 Halaman Produk

d. Halaman Detail Produk

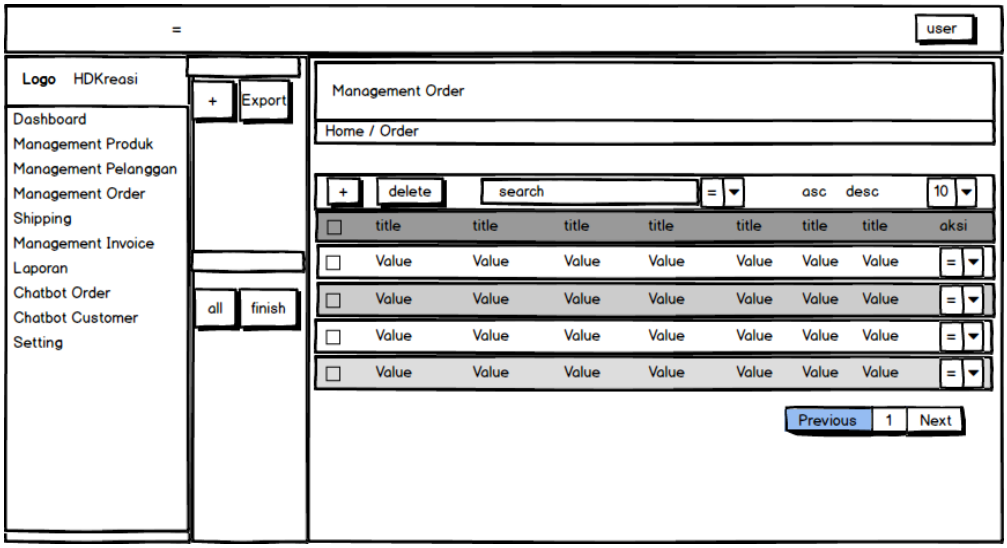
Halaman ini adalah halaman lanjutan dari produk. Halaman ini akan menampilkan informasi mengenai detail informasi produk. Rancangan halaman detail produk ditunjukkan pada Gambar 3.17.



Gambar 3.17 Halaman Detail Produk

e. Halaman Order

Halaman ini digunakan untuk melakukan pengelolaan terhadap pesanan atau *order*. Rancangan halaman dokumen dapat dilihat pada Gambar 3.18.



Gambar 3.18 Halaman *Order*

3.2.6 Perancangan Chatbot

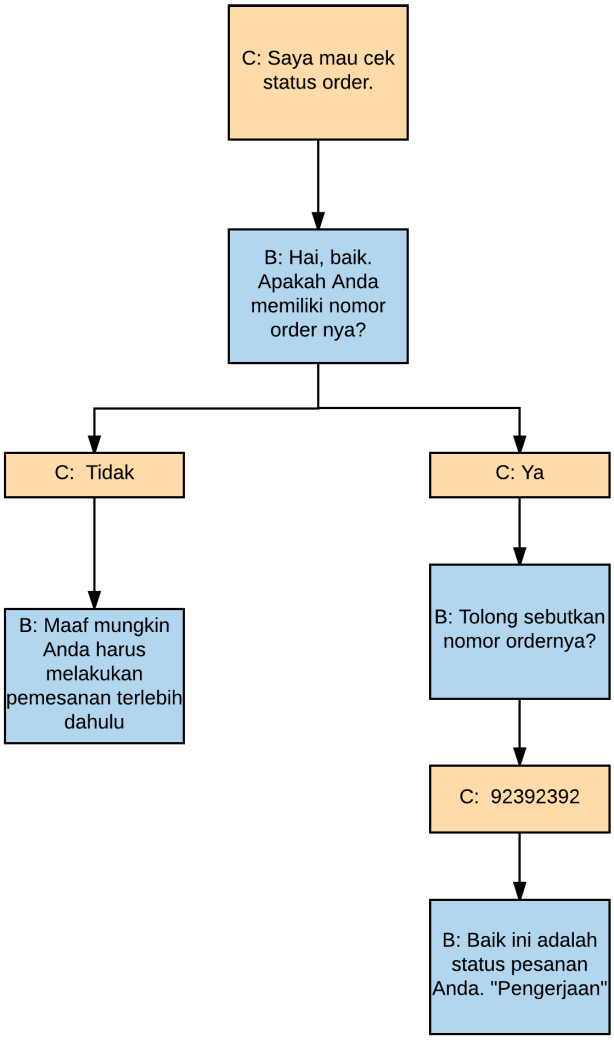
Pada bab ini akan dijelaskan tahapan perancangan *chatbot*. Penulis akan merancang *chatbot* yang bertujuan untuk memudahkan dalam proses implementasi pada bab selanjutnya. Ada beberapa perancangan yang akan dibuat dalam keperluan membangun sebuah *chatbot* di antaranya yaitu perancangan *conversational flow* dan *conversational user interface*.

Conversational Flow

Conversational flow merupakan aliran percakapan yang ada di dalam *chatbot*. Adanya *conversational flow* bertujuan agar percakapan antara *chatbot* dan pelanggan memiliki aliran dan aturan yang baku. Pelanggan akan bertanya mengenai sesuatu, namun sebelum keinginan pelanggan dipenuhi harus ada langkah-langkah yang harus dilalui. Penulis sebelumnya telah menentukan beberapa tugas yang sesuai dengan permasalahan yang akan diatasi, di antaranya adalah *conversational flow* ketika pelanggan ingin menanyakan status pesanan dan ingin memesan produk. Setelah melakukan perancangan *conversational flow* selanjutnya penulis akan melakukan perancangan di dalam *platform* API.AI. Perancangan di dalam *platform* API.AI akan menjelaskan bagaimana penggunaan layanan ini dan hal-hal apa saja yang perlu dibuat dan diperhatikan untuk mendukung *chatbot* agar dapat memahami *request* yang datangnya dari pelanggan. Untuk perancangan ini penulis mengambil referensi percakapan yang sudah terjadi antara pelanggan dan UMKM HDKreasi baik dari email ataupun layanan pesan lainnya, lalu penulis mencoba merancang aliran percakapan yang sesuai dengan tugas yang ingin diselesaikan.

a. Conversational flow status pesanan

Conversational flow status pesanan ini berfungsi merupakan penanganan percakapan ketika pelanggan akan melakukan pengecekan terhadap pesanan yang sudah dibuat pelanggan. Berikut merupakan *conversational flow* status pesanan yang ditunjukkan pada Gambar 3.19.

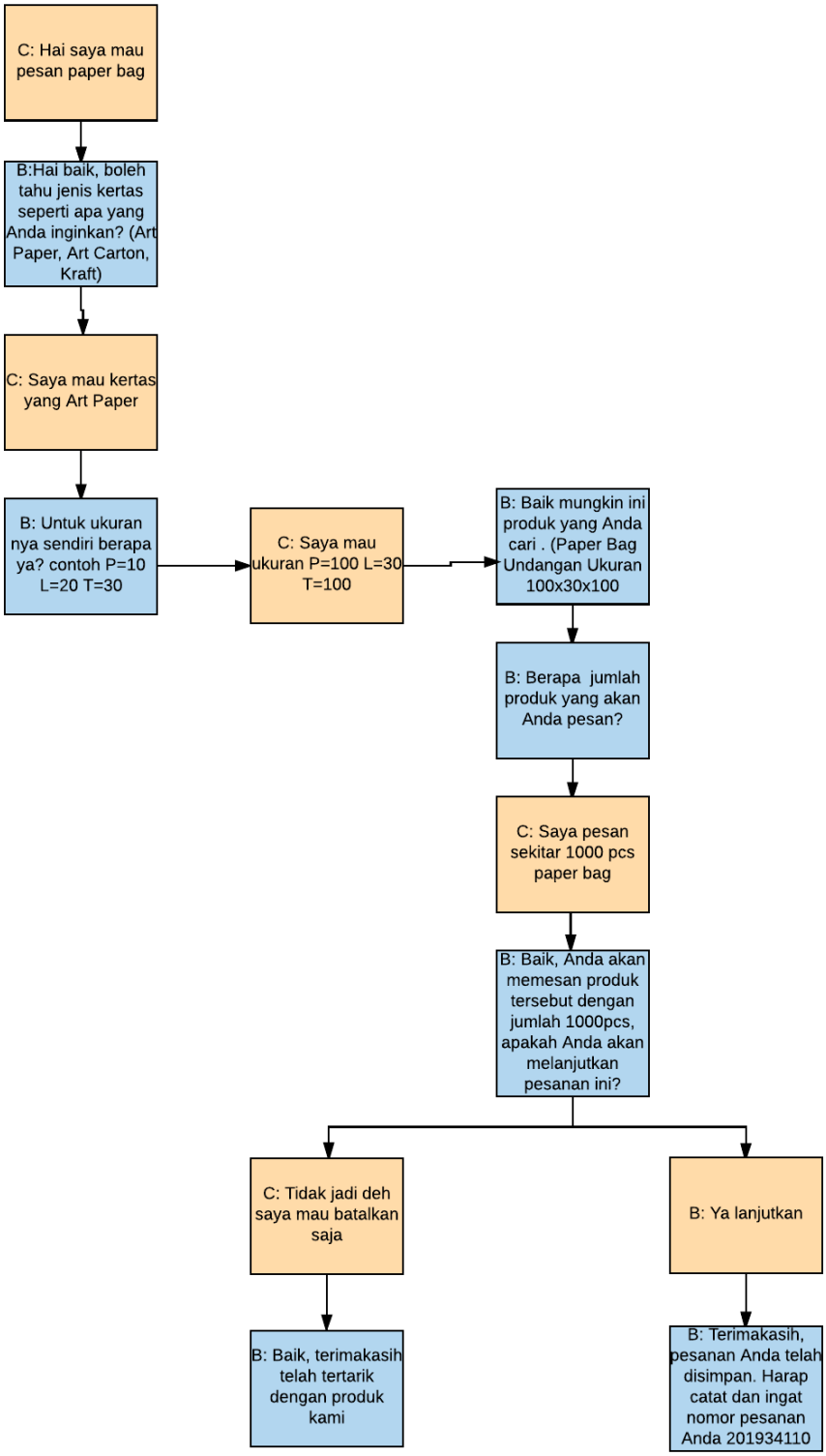


Gambar 3.19 *Conversational Flow* Cek Status Pesanan

b. *Conversational flow* tanya produk dan melakukan pesanan

Conversational flow tanya produk ini berfungsi untuk menangani percakapan ketika pelanggan menanyakan produk dan selanjutnya adalah melakukan pesanan terhadap produk

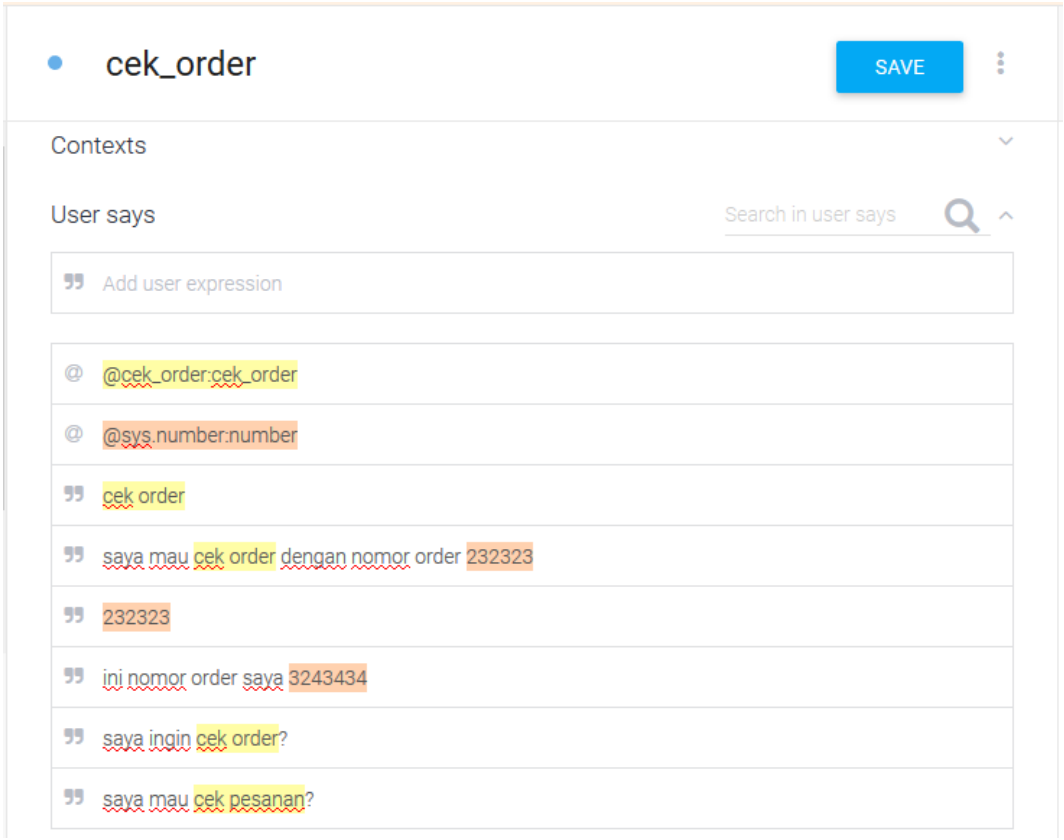
yang sudah dipilih sebelumnya. Berikut merupakan *conversational flow* tanya produk dan melakukan pesanan yang ditunjukkan pada Gambar 3.20.



Gambar 3.20 *Conversational Flow* Pesanan

Setelah melakukan perancangan pada *conversational flow* yang direpresentasikan pada diagram-diagram di atas. Selanjutnya penulis akan memaparkan penggunaan layanan API.AI ini. Pada langkah pertama yang penulis lakukan adalah membuat sebuah *Agent,* yang merupakan sebuah istilah yang terdapat di dalam *platform* ini untuk digunakan sebagai sebuah *module* tempat segala pemrosesan bahasa alami dilakukan. Sebuah *Agent* juga dapat digunakan untuk merancang dan mengelola *conversational flow* tertentu. Dari *Agent* inilah akan dilakukan pemrosesan terhadap *request text* yang datang dari pelanggan dan melakukan pemetaan terhadap *request* tersebut ke dalam bagian yang bersesuaian dengan maksud dari *request* tersebut atau di dalam *platform* ini disebut sebagai *Intent.* Sebuah *Intent* mewakili sebuah maksud dari *request text* yang datang dari pelanggan. Sebagai contoh ketika pelanggan atau *user* mengatakan "Hallo, selamat siang" kata-kata tersebut memiliki maksud sebagai sebuah kata sapaan. Di dalam *platform* inilah dilakukan NLU (*Natural Language Understanding*) yang mana *request* yang datang akan dikelompokkan sesuai dengan maksudnya. Pada contoh sebelumnya merupakan kata yang mengandung sapaan, dengan demikian layanan ini akan mengidentifikasi dan memasukkannya ke dalam golongan kata sapaan.

Sebelumnya penulis telah menentukan beberapa tugas untuk menangani pertanyaan-pertanyaan yang datang dari pelanggan. Seperti pertanyaan untuk melakukan cek pesanan dan pertanyaan untuk melakukan pemesanan. Gambar 3.19 dan Gambar 3.20 merupakan aliran percakapan yang sudah dirancang sebelumnya yang pada akhirnya penulis gunakan sebagai titik acuan untuk membangun sebuah *chatbot*. Selanjutnya penulis akan membuat sebuah *Intent* di dalam layanan ini. *Intent* yang pertama kali penulis buat adalah "cek_order", maksud dari pembuatan *Intent* ini adalah untuk menggolongkan semua *request text* dari *user* dengan kriteria *request* yang memiliki kecocokan dengan tujuan atau maksud untuk melakukan pengecekan pesanan lalu memetakan request tersebut ke dalam golongan *intent* ini. Lalu penulis akan membuat kamus atau frasa yang sekiranya dapat mewakili pertanyaan *user* yang berkaitan dengan pertanyaan untuk melakukan cek pesanan atau *order*. Berikut ini merupakan beberapa frasa atau kamus yang telah penulis buat di dalam *intent* "cek_order" ini yang ditunjukkan pada Gambar 3.22.



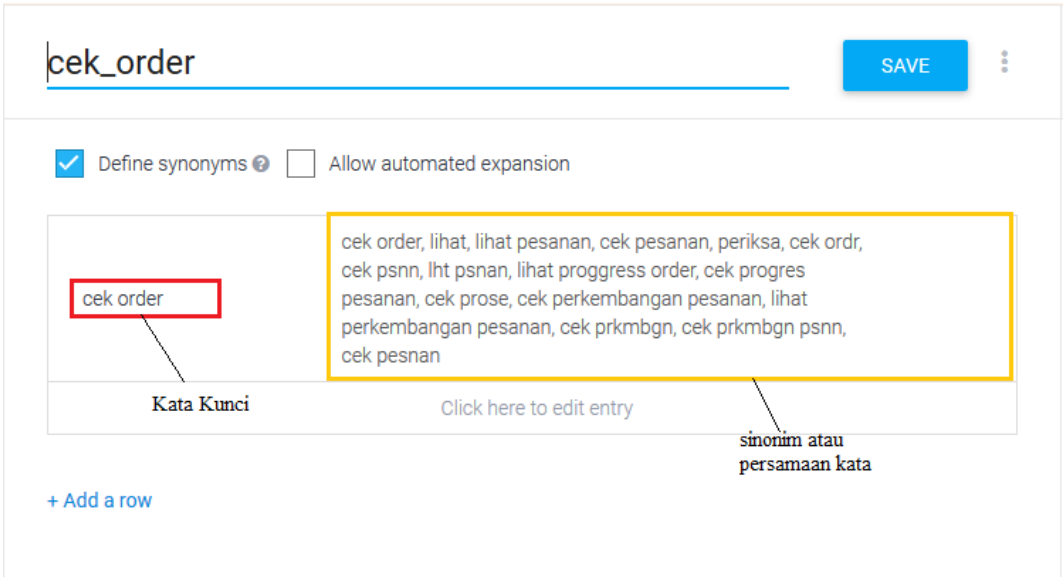
Gambar 3.21 Pendefinisian Frasa di Dalam *Intent* "Cek order"

Pada Gambar 3.21 di atas, penulis telah membuat beberapa sampel kamus atau frasa yang sekiranya mewakili pertanyaan yang diajukan oleh *user* apabila ingin melakukan pengecekan pesanan. Di dalam API.AI ini sendiri telah memiliki fungsionalitas untuk dapat memahami *request text* dari *user* yaitu dengan menggunakan NLP (*Natural Language Understanding)* yang bertujuan untuk memahami apa yang *user* katakan sesuai dengan sampel kamus atau frasa yang telah penulis buat di dalam *intent* ini, apabila *user* mengatakan pertanyaan yang memiliki kemiripan dengan frasa yang sudah didefinisikan maka dengan sendirinya layanan ini akan mengelompokkan pertanyaan tersebut ke dalam *intent* ini. Di dalam frasa atau kamus yang penulis sudah buat juga terdapat komponen lainnya yaitu *entities* dan *parameters* yang penulis gunakan untuk mengambil informasi yang berguna dari *request text* yang datang. Penulis ingin mengekstrak informasi yang datang dari *user* yaitu informasi nomor pesanan. Untuk mendapatkan informasi tersebut penulis menyisipkan sebuah parameter berupa *entites system number* ke dalam frasa atau sampel yang penulis buat.

Seperti yang telah dijelaskan pada Bab 2 Tinjauan Pustaka, *entities* adalah sebuah alat yang sangat kuat untuk mengindentifikasi nilai yang berguna dari sebuah masukan bahasa

alami dalam hal ini adalah pertanyaan yang dikirim oleh *user*. *Entities* yang penulis gunakan adalah *entities built in* yang terdapat di dalam sistem ini yaitu *system number* dan *entities* yang penulis definisikan sendiri yaitu "cek_order". Kedua *entities* yang diterapkan di dalam frasa atau kamus inilah nantinya yang akan dicocokkan dengan *request text* yang datang apakah terdapat kata yang teridentifikasi memiliki kesesuaian baik berupa *number* dan kata yang mirip dengan kata kunci cek order.

Entities "cek_order" yang penulis buat merupakan entitas yang bertujuan untuk mengidentifikasi apakah *request text* dari *user* memiliki kesamaan atau teridentifikasi memiliki kesesuaian kata kunci dengan *entities* "cek_order" ini. Di dalam *entities* "cek_order" ini harus memiliki kata kunci utama, di sini penulis mendefinisikan kata kunci utama tersebut yaitu cek order. Selain itu dari kata kunci tersebut penulis membuat daftar sinonim atau persamaan kata agar memiliki variasi. Berikut ini merupakan pembuatan *entites* "cek_order" yang ditunjukkan pada Gambar 3.22.

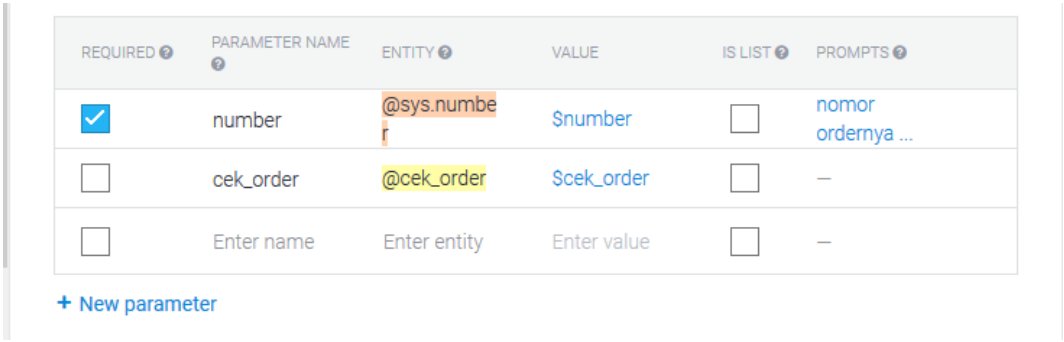


Gambar 3.22 Entities Cek_order

Pembuatan sinonim pada *entitas* "cek_order" bertujuan untuk memberikan variasi kata, karena setiap *user* akan memiliki perbedaan kata ketika ingin melakukan pengecekan pesanan mereka. Sehingga dibuatlah sinonim atau persamaan kata untuk menangani perbedaan kata tetapi memiliki arti dan tujuan yang sama. Tidak hanya itu, penulis sengaja membuat sinonim dengan singkatan kata. Di mana kemungkinan besar dengan kebiasaan menulis pesan yang disingkat akan memudahkan mesin untuk memahami maksud dari pesan atau kata yang

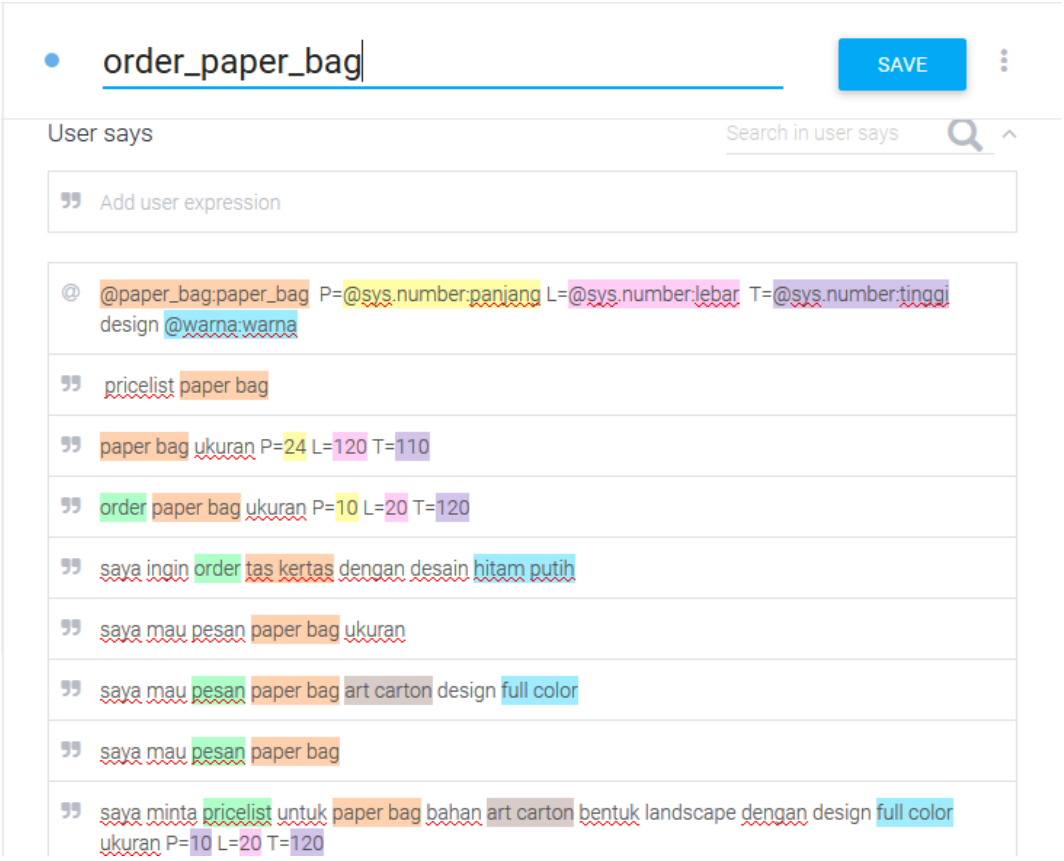
disingkat tersebut. Sebagai contoh adalah terdapat *user* yang mengetikkan cek order tetapi adapula sebagian *user* yang mengetikkan cek ordr. Hal seperti inilah yang penulis antisipasi agar pemahaman terhadap *request text* dari *user* dapat dimengerti dan dipahami serta dapat memenuhi tujuan yang diinginkan.

Untuk kebutuhan dalam pengambilan informasi di dalam *database*, penulis memerlukan informasi nomor pesanan dari pelanggan. Dalam *conversational flow* yang telah dirancang sebelumnya, *chatbot* akan menanyakan nomor pesanan pelanggan. Di dalam *intents* cek order inilah penggalian informasi nomor *order* dilakukan. Frasa atau kamus yang penulis buat memiliki parameter berupa angka, di mana apabila *user* atau pelanggan mengirim *request text* berupa angka maka akan langsung dikenali sebagai sebuah nomor *order* dan disimpan di dalam parameter untuk dikelola lebih lanjut. Gambar 3.23 merupakan parameter yang ada di dalam *intents* ini.



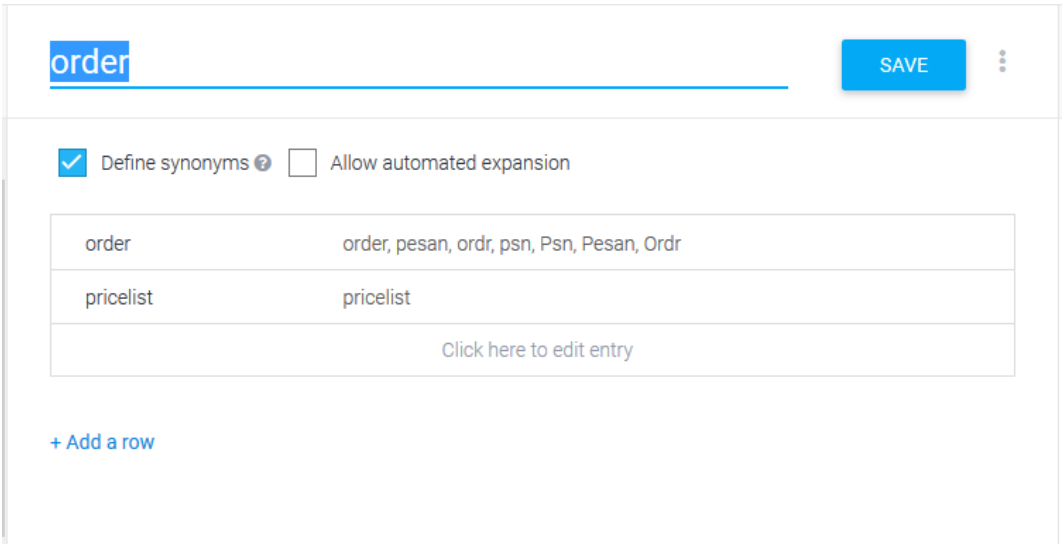
Gambar 3.23 Parameter *Intents* Cek_order

Terlihat pada Gambar 3.23 di atas terdapat nama parameter *number*. Parameter ini adalah parameter yang harus diisi ditandai tanda centang pada pojok kiri pada bagian *Required.* Apabila di dalam *request text* yang masuk tidak terindikasi angka atau *number* maka sistem ini akan menanyakan kembali nomor *order* yang diperlukan. Untuk perancangan *intents* selanjutnya adalah perancangan untuk menangani pemesanan produk atau mendapatkan informasi produk. Pada Gambar 3.24 merupakan *intents* untuk menangani hal tersebut.



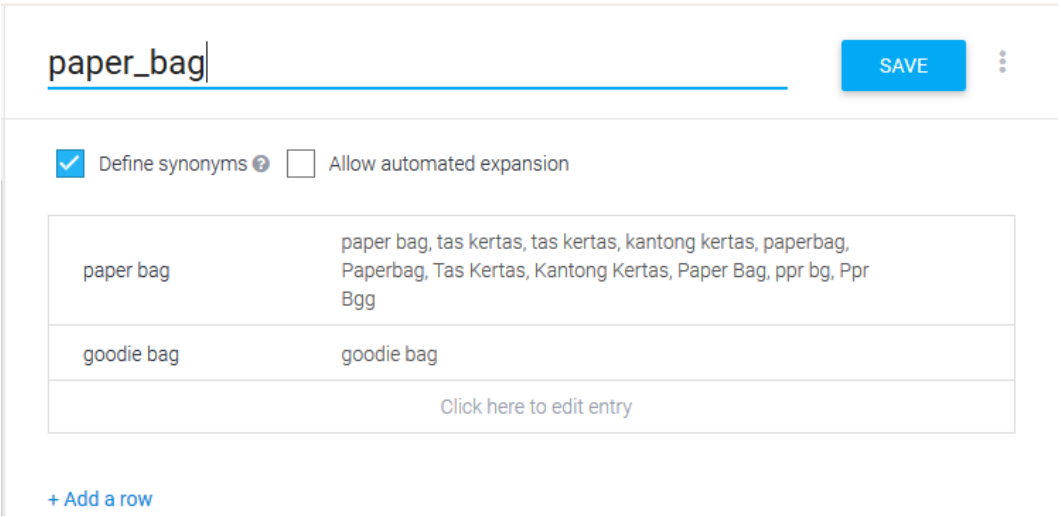
Gambar 3. 24 *Intents Order Paper Bag*

Intents "order_paper_bag" ini dibuat untuk menangani tugas ketika pelanggan ingin menanyakan produk ataupun memesan produk. Di dalam *intents* ini penulis telah membuat sekumpulan frasa atau kamus sebagai sampel untuk kata-kata yang mungkin diberikan *user* ketika ingin menanyakan produk. Pada bagian ini penulis akan mengumpulkan informasi yang dibutuhkan untuk melakukan pencarian produk dan menyimpannya di dalam parameter yang telah ditentukan seperti informasi mengenai ukuran panjang, lebar dan tinggi produk yang diinginkan, jenis kertas serta bahan dari produk tersebut. Terdapat juga beberapa *entities* yang penulis gunakan, baik *entities* yang penulis definisikan sendiri ataupun *entities built in* dari sistem. Diantara *entities* yang penulis gunakan dan buat adalah *order*, paper bag, paper type, warna. *Entities order* digunakan untuk mengidentifikasi kata-kata dalam frasa yang memiliki kesamaan kata kunci di dalam *entities* ini. Berikut merupakan *entities order* yang ditunjukkan pada Gambar 3.25.



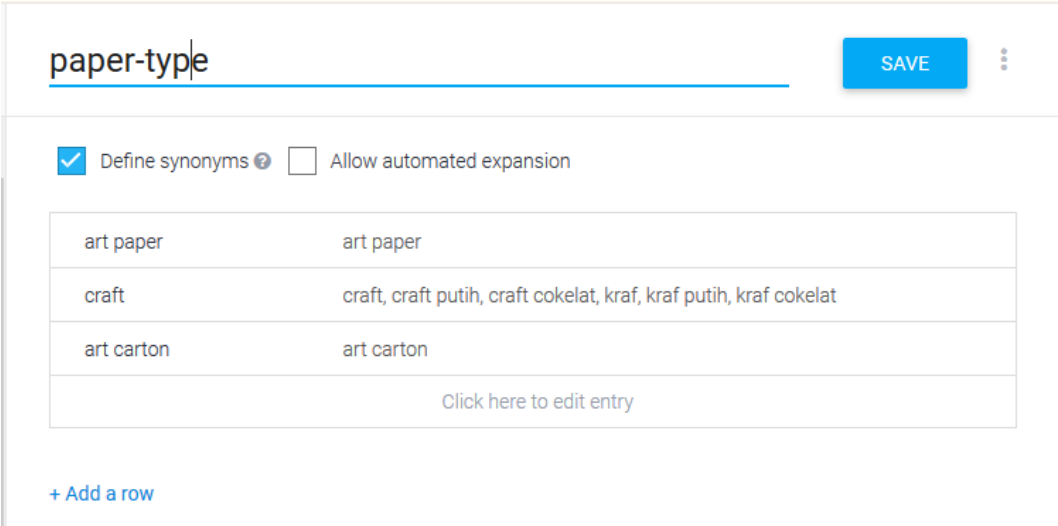
Gambar 3.25 Entitas *Order*

Pada entitas *order* ini penulis membuat kata kunci *order* dan *pricelist* beserta dengan sinonim dan singkatannya. Hal ini bertujuan jika *user* mengirimkan *request text* dan memiliki kesamaan dengan kata kunci atau sinonimnya dapat dikatakan bahwa *request text* tersebut teridentifikasi memiliki entitas *order*, dengan kata lain bahwa *request text* tersebut masuk ke dalam golongan *intents order*. Selanjutnya adalah entitas "paper_bag" yang merupakan entitas yang memuat kata kunci dan sinonim mengenai produk *paper bag*. Berikut merupakan entitas "paper_bag" yang ditunjukkan pada Gambar 3.26.



Gambar 3.26 Entitas *Paper Bag*

Selanjutnya adalah entitas “paper-type” atau jenis kertas dari produk *paper bag*. Di dalam entitas ini penulis membuat beberapa kata kunci dan sinonimnya. Berikut merupakan entitas “paper-type” yang ditunjukkan pada Gambar 3.27.



Gambar 3.27 Entitas *Paper Type*

Seluruh *intents* yang telah dibuat, akan diolah lebih lanjut di dalam *server webhook*. Ketika terjadi *request* ke dalam layanan API.AI dan dilakukan pengolahan teks maka layanan ini akan memberikan *response* kepada peminta dalam hal ini adalah *server webhook* berupa data *json* yang berisi jenis *intent*, parameter dan *value* yang terdapat di dalam *intent.* Di dalam *server webhook* inilah nantinya data json tersebut diolah lebih lanjut untuk memberikan *response* yang tepat kepada pelanggan.

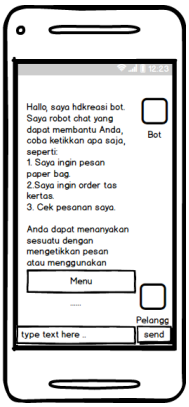
Conversational User Interface

Conversation user interface (CUI) merupakan antarmuka yang disajikan *chatbot* kepada pelanggan di dalam *platform* pesan Messenger. Di dalamnya berisi tombol-tombol yang dapat dimanfaatkan oleh pelanggan. Penulis tidak membuat aplikasi berbasis *mobile* melainkan menggunakan aplikasi Messenger yang sudah tersedia dan disediakan oleh Facebook. Penulis membuat CUI ini bertujuan agar komunikasi antara *chatbot* dan pelanggan terjadi dengan interaktif. Selain dapat berinteraksi dengan cara memasukkan pesan, pelanggan juga dapat memanfaatkan *User Interface* yang berada di *chatbot* untuk menggunakan fungsionalitas yang ada di dalamnya. Ada beberapa rancangan yang penulis buat dalam perancangan antarmuka

chatbot ini. Di antaranya adalah rancangan pembuka, rancangan menu, rancangan notifikasi *invoice,* rancangan tampilan produk.

a. Rancangan pembuka

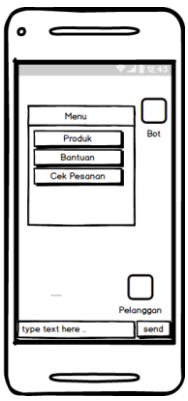
Pada rancangan antarmuka ini, *chatbot* akan menampilkan balasan pesan berupa informasi yang berbasis grafis di dalam *platform* pesan Messenger. Pada rancangan ini merupakan representasi dari antarmuka *platfrom* pesan dari Facebook Messenger. Berikut merupakan rancangan ketika pelanggan pertama kali berinteraksi dengan *chatbot* yang ditunjukkan pada Gambar 3.28.



Gambar 3.28 Rancangan Pembuka

b. Rancangan menu

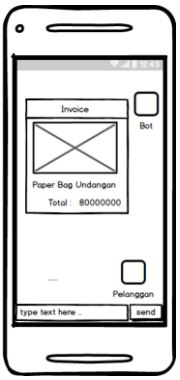
Rancangan menu *default* ini merupakan rancangan yang berisi menu-menu yang dapat pelanggan gunakan untuk menggunakan fungsionalitas yang tersedia di dalam *chatbot* ini. Berikut merupakan rancangan menu yang ditunjukkan pada Gambar 3.29.



Gambar 3.29 Rancangan Menu

c. Rancangan notifikasi *invoice*

Pada rancangan antarmuka ini, *chatbot* akan menampilkan balasan pesan berupa informasi notifikasi *invoice*. Berikut merupakan rancangan *invoice* yang ditunjukkan pada Gambar 3.30.



Gambar 3.30 Rancangan Notifikasi *Invoice* dan Pesanan

d. Rancangan tampilan produk

Pada rancangan ini akan ditampilkan sejumlah produk sesuai dengan keinginan pelanggan. Berikut merupakan rancangan tampilan produk yang ditunjukkan pada Gambar 3.31.

Gambar 3.31 Rancangan Tampilan Produk

3.3 Perancangan Pengujian

Pada tahapan perancangan pengujian ini, penulis akan membuat skenario pengujian terhadap sistem pengelolaan pesanan. Di mana terdapat beberapa pengujian yang akan dilakukan di antaranya adalah pengujian fungsionalitas dengan menggunakan metode *Black Box*, pengujian usabilitas dan pengujian *chatbot.*

3.3.1 Pengujian Black Box

Pengujian perangkat lunak pada sistem pengelolaan pesanan ini menggunakan metode pengujian *Black Box*. Pengujian ini berfokus terhadap fungsionalitas perangkat lunak yang dibuat. Diantara pengujian yang akan dilakukan adalah mengenai *output* yang akan diberikan sistem pada fungsionalitas tertentu misalnya adalah apakah sistem akan memberikan informasi pesan berhasil setelah *user* melakukan penyimpanan data dan lain sebagainya seperti validasi terhadap masukan *user*.

3.3.2 Pengujian Usabilitas

Pengujian usabilitas ini berhubungan dengan seberapa besar tingkat kenyamanan *user* dalam menggunakan sistem. Terdapat beberapa indikator yang digunakan untuk melakukan penilaian terkait dengan pengujian usabilitas ini, di antaranya adalah seperti yang ditunjukkan pada Tabel 3.1.

Tabel 3.1 Indikator Penilaian Usabilitas

Indikator	Penjelasan
Efe	Efektifitas
Efi	Efisiensi
Sat	Kepuasan dan Kenyamanan
Lea	Learnability
Fle	Fleksibilitas
Rob	Robustness
EoL	Mudah dipelajari
Rec	Mudah diingat
Pro	Produktivitas optimal
MeR	Minim kesalahan
HUS	Kepuasan dan kenyamanan pengguna

3.3.3 Pengujian Chatbot

Pengujian chatbot sendiri dilakukan untuk menilai apakah *chatbot* memiliki kemampuan untuk menjawab pertanyaan yang diajukan oleh pelanggan. Untuk dapat menguji *chatbot* itu sendiri terdapat beberapa poin-poin penting yang dimasukkan kedalam pengujian diantaranya adalah mengenai kemampuan *chatbot* dalam memhami pertanyaan dan memberi jawaban terhadap pertanyaan yang diberikan pelanggan, layanan navigasi yang disediakan *chatbot* untuk mempermudah informasi, serta kemampuan *chatbot* untuk mengakuisisi informasi dari sistem pengelolaan pesanan.

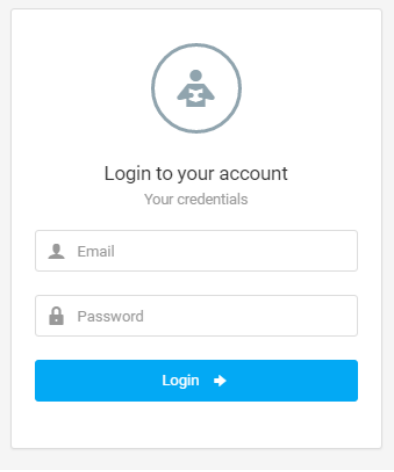
BAB IV
IMPLEMENTASI DAN PENGUJIAN

Pada bab ini menggambarkan mengenai implementasi perancangan yang sebelumnya telah dibahas pada Bab III. Bahasan dalam bab ini meliputi implementasi dan pengujian sistem. Implementasi dan pengujian sistem ini dilakukan untuk mengetahui apakah sistem yang telah dibangun sudah bekerja sesuai dengan rancangan sistem. Pengujian sistem dilakukan dengan menggunakan metode *black-box testing*. Metode ini digunakan untuk melakukan pengujian yang berfokus pada keluaran yang dihasilkan dari proses masukan yang terjadi.

Pada struktural laporan di bab ini, pengujian ditempatkan di dalam setiap bab implementasi dari perancangan. Terdapat beberapa implementasi yang penulis lakukan yaitu implementasi halaman login, halaman *dashboard*, halaman produk, halaman *order*, halaman *invoice*, halaman *shipping* yang merupakan fitur utama yang terdapat di dalam sistem pengelolaan pesanan ini dan implementasi *chatbot.*

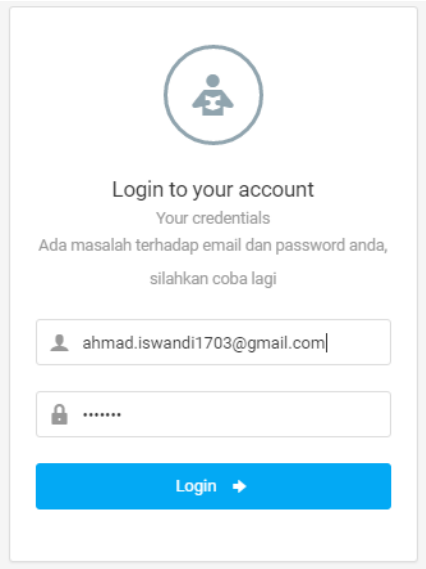
4.1 Halaman Login

Halaman ini adalah halaman yang pertama kali akan muncul pada saat sistem pengelolaan pesanan ini diakses. Untuk dapat menggunakan *fungsionalitas* yang terdapat di dalam sistem ini, pengguna harus melakukan *login* dengan memasukkan *email* dan *password* pada *form* yang telah tersedia di tampilan antarmuka sistem. Tampilan halaman *login* dapat dilihat pada Gambar 4.1.



Gambar 4.1 Halaman *login*

Gambar 4.1 di atas merupakan halaman *login* yang pertama kali akan dilihat oleh *user* ketika mengakses sistem ini. Selanjutnya adalah tampilan halaman login ketika terjadi kesalahan masukan yang dilakukan oleh *user* yang ditunjukkan pada Gambar 4.2.



Gambar 4.2 Halaman *Login* dengan Pesan Kesalahan

Gambar yang ditampilkan di atas merupakan pesan kesalahan yang terjadi pada saat terdapat kesalahan masukan yang dilakukan oleh *user*, maka akan ditampilkan peringatan bahwa terdapat masalah terhadap *email* atau *password* yang dimasukkan. Pada tahapan pengujian halaman *login* ini dapat diperoleh hasil yang ditunjukkan pada Tabel 4.1 tentang skenario pengujian yang dilakukan pada halaman *login.*

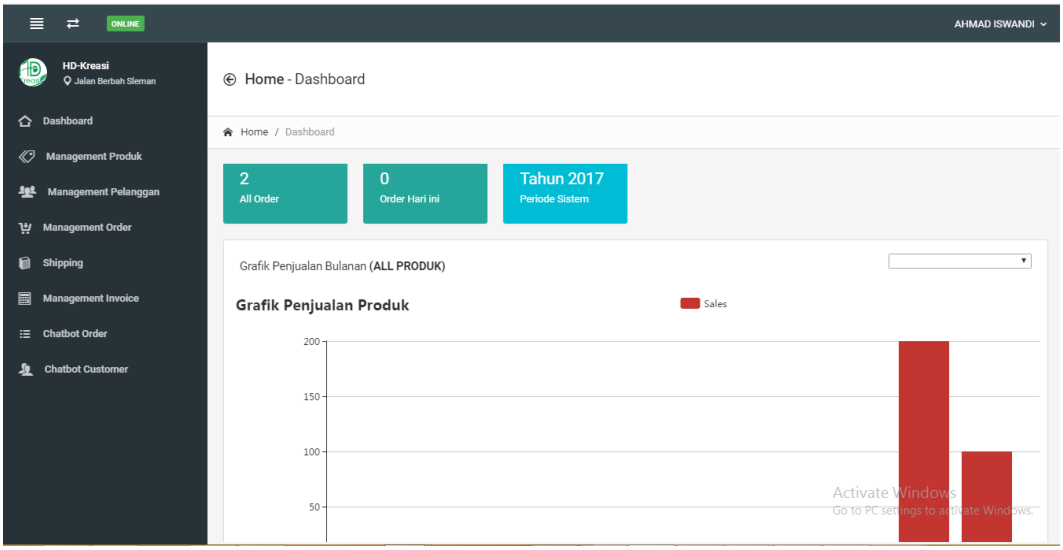
Tabel 4.1 Pengujian pada Halaman *Login*

Skenario pengujian	**Input**	**Output yang diharapkan**	**Output hasil**
Memasukkan *email* dan *password* yang benar dan sudah terdaftar di dalam sistem	*Email* dan *Password* (benar)	Halaman *dashboard*	Halaman *dashboard* (valid)
Memasukkan *email* dan *password* yang salah dan tidak terdaftar	*Email* dan *Password* (salah)	Tampil pesan kesalahan	Tampil pesan kesalahan (valid)
Mengosongkan *email* dan *password*	*Email* dan *Password* (kosong)	Tampil pesan kesalahan	Tampil pesan kesalahan (valid)

Email benar password salah	Email benar password salah	Tampil pesan kesalahan	Tampil pesan kesalahan (valid)

4.2 Halaman Dashboard

Halaman *dashboard* adalah halaman yang akan menampilkan beberapa informasi grafis yang dapat dilihat oleh *user* yang berstatus sebagai pegawai atau pun *super admin*. Di dalam halaman ini akan ditampilkan info grafik seperti grafik penjualan produk dan juga informasi mengenai sistem lainnya. Gambar 4.3 merupakan halaman *dashboard* yang dapat dilihat oleh *user*.



Gambar 4.3 Halaman Dashboard

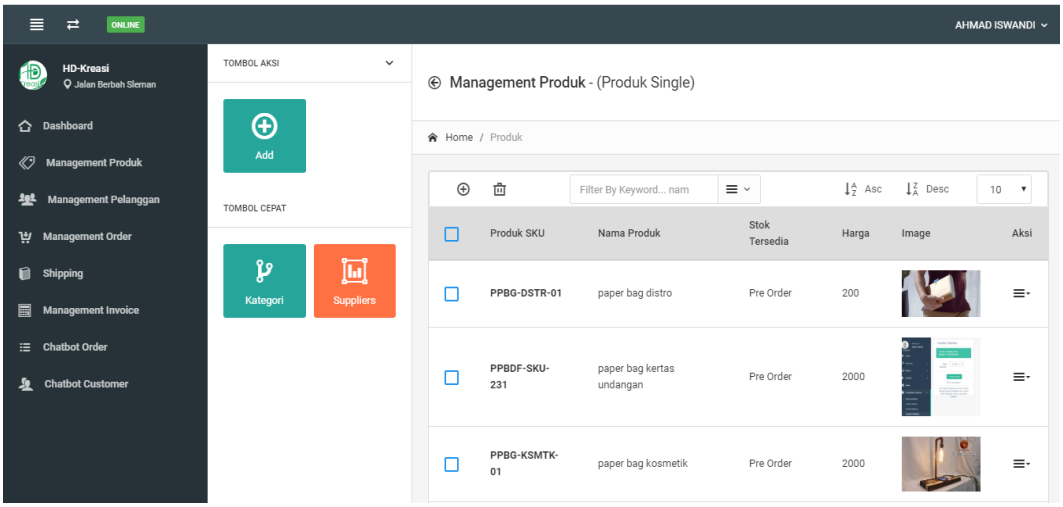
Untuk hasil pengujian pada halaman *dashboard* ini dapat ditunjukkan pada Tabel 4.2.

Tabel 4.2 Pengujian pada Halaman *Dashboard*

Skenario pengujian	Input	Output yang diharapkan	Output hasil
Mengubah data produk untuk menampilkan grafik penjualan produk tersebut (fungsionalitas)	Klik Produk dari *dropdown form*	Grafik akan berubah sesuai dengan produk yang dipilih pada *form dropdown*	Grafik berubah (valid)

4.3 Halaman Produk

Halaman produk adalah halaman yang akan menampilkan informasi mengenai produk yang terdapat di dalam *database* yang dapat dilihat oleh *user* yang berstatus sebagai pegawai. Di dalam halaman ini akan ditampilkan info produk serta dapat dilakukan penambahan, pengubahan ataupun penghapusan dari dalam halaman ini. Di halaman ini juga *user* dapat menambahkan data pegawai dan data suppliers. Gambar 4.4 merupakan halaman dashboard yang terdapat dalam sistem pengelolaan pesanan ini.



Gambar 4.4 Halaman produk

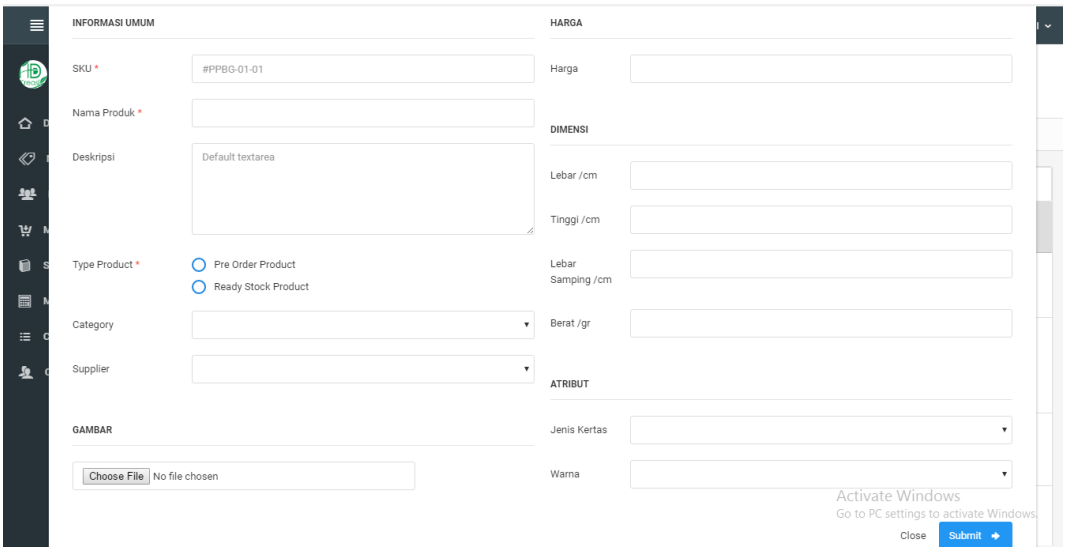
Berikut ini merupakan hasil pengujian pada halaman produk terhadap aksi atau fungsionalitas yang ditunjukkan pada Tabel 4.3.

Tabel 4.3 Hasil Pengujian pada Halaman Produk

Skenario pengujian	Input	Output yang diharapkan	Output hasil
Mengakses menu untuk menambah data (fungsionalitas)	Klik *action add*	*Modal dialog* akan muncul dan menampilkan *form*	*Modal dialog* muncul (valid)
Mengakses menu untuk mengubah data produk (fungsionalitas)	Klik *action* edit pada tabel kolom aksi	*Modal dialog* akan muncul dan menampilkan *form* yang telah terisi data sesuai dengan produk pilihan *user*	*Modal dialog* muncul dan terisi data sesuai denga produk pilihan *user* (valid)
Mengakses menu untuk melihat detail produk (fungsionalitas)	Klik *action* lihat detail	Halaman detail produk	Halaman detail produk (valid)

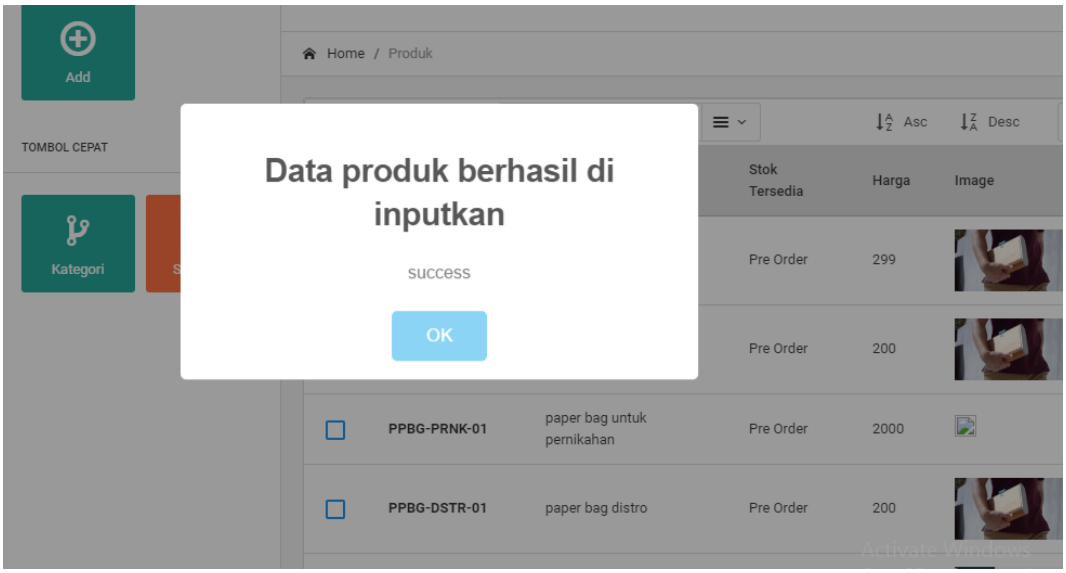
	pada tabel kolom aksi		
Mengakses menu untuk menghapus data produk	Klik *action* delete pada tabel kolom aksi	Muncul *modal dialog* untuk konfirmasi penghapusan produk	Muncul *modal dialog* konfirmasi penghapusan produk
Mengakses menu untuk menambahkan data *suppliers*	Klik *action suppliers*	Muncul *modal dialog* untuk menambah data *suppliers*	Muncul *modal dialog* untuk menambah data *suppliers*
Mengakses menu untuk menambahkan data kategori	Klik *action* kategori	Muncul *modal dialog* untuk menambah data kategori	Muncul *modal dialog* untuk menambah data kategori

Pada halaman pengelolaan produk ini terdapat pilihan aksi yang dapat digunakan oleh *user* yaitu tambah, edit, hapus, lihat detail, tambah kategori, tambah suppliers. Aksi tambah merupakan aksi yang dapat *user* gunakan untuk menambah data produk. Gambar 4.5 merupakan tampilan dari *modal dialog* yang dapat digunakan *user* untuk menambahkan data produk.



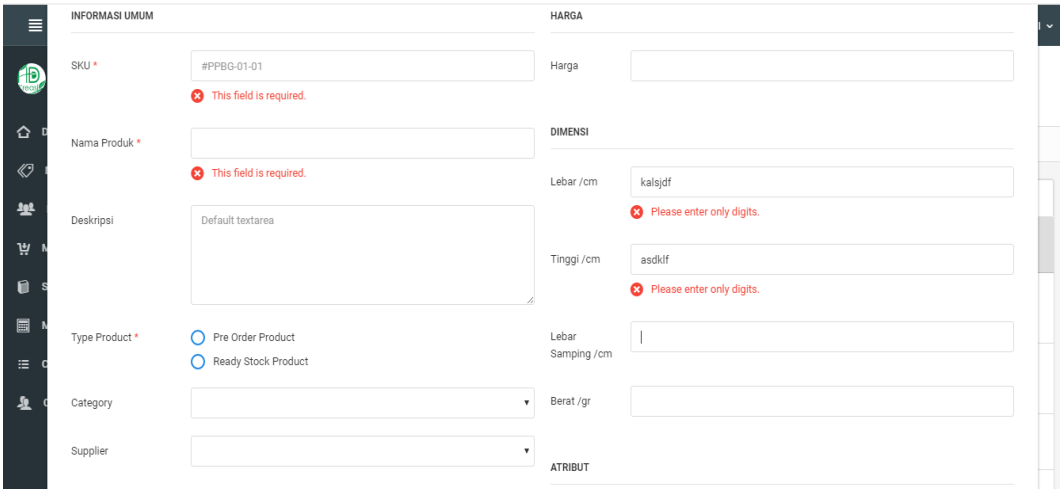
Gambar 4.5 Modal *Dialog* untuk Menambah Data Produk

Pada fungsionalitas atau aksi untuk menambahkan data produk ini akan juga menampilkan pesan bahwa proses untuk memasukkan data berhasil dilakukan yang ditunjukkan pada Gambar 4.6



Gambar 4.6 *Modal Dialog* Informasi Berhasil Memasukkan Data Produk

Dalam melakukan penambahan data produk, terdapat beberapa *form* yang wajib diisi oleh *user* dan *form* yang harus sesuai dengan ketentuan yang telah dibuat. Pada Gambar 4.7 ditunjukkan pesan kesalahan apabila terdapat *form* yang tidak diisi ataupun *form* yang isinya tidak sesuai dengan ketentuan.

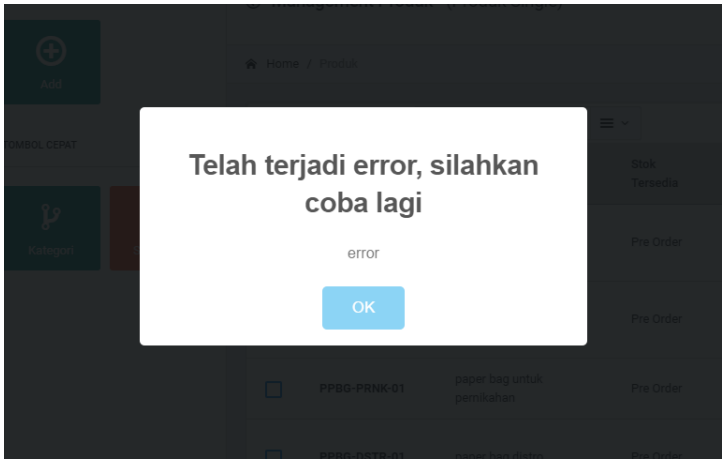


Gambar 4.7 Pesan Kesalahan Required Form

Sistem juga akan memberikan pesan kesalahan saat terjadi *error* atau kesalahan yang dikirimkan oleh *server*. Pada Gambar 4.8 merupakan pesan kesalahan yang terjadi saat terdapat duplikasi nama SKU yang dimasukkan.

Gambar 4.8 Pesan Kesalahan Duplikasi SKU

Gambar 4.8 di atas menunjukkan kesalahan bahwa terdapat duplikasi SKU atau *stock keeping unit* yang merupakan identitas produk. Selanjutnya adalah pesan kesalahan apabila terjadi *error* pada *server* ataupun masukan yang tidak benar maka akan muncul pesan seperti yang ditunjukkan pada Gambar 4.9.



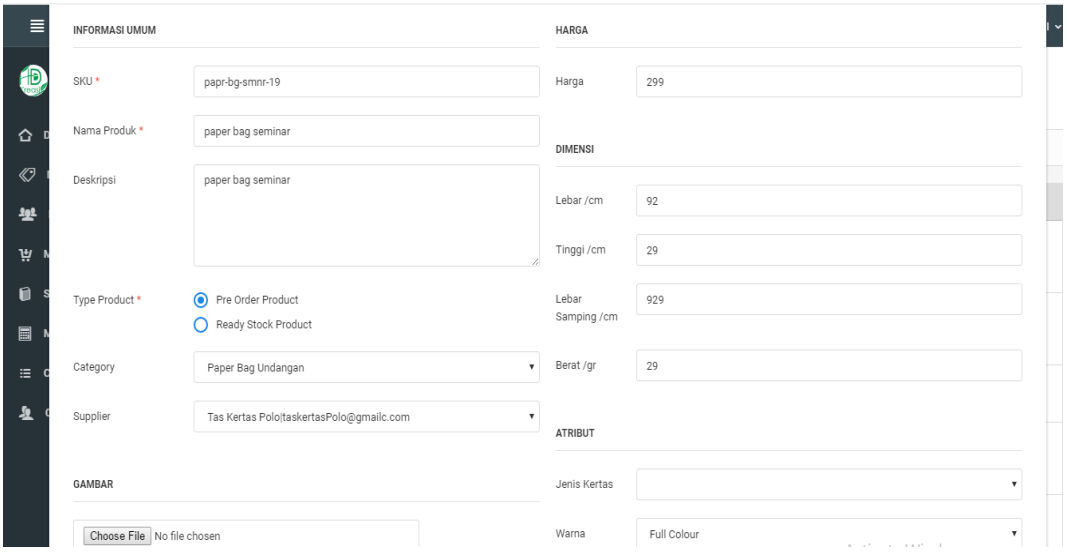
Gambar 4.9 Pesan Kesalahan Error

Berikut ini merupakan detail pengujian pada aksi untuk menambahkan produk pada halaman produk ini dapat dilihat pada Tabel 4.4.

Tabel 4.4 Pengujian terhadap aksi tambah data di halaman produk

Skenario pengujian	Input	Output yang diharapkan	Output hasil
Menambah data produk dengan data masukan valid dan benar (fungsionalitas)	*form* data produk dimasukkan dengan (benar)	*Modal dialog* akan memunculkan pesan bahwa data berhasil dimasukkan	*Modal dialog* muncul (valid)
Menambah data produk dengan data yang tidak valid (seperti *form* yang wajib diisi dan *form* yang harus diisi dengan angka). (input validasi)	*Form* data produk dimasukkan dengan data yang tidak benar	Akan muncul pesan kesalahan pada bagian bawah *form* yang diberi aturan seperti *form* yang harus diisi maka akan muncul "this field is required"	Pesan kesalahan validasi pada *form* muncul (valid)
Menambahkan data dengan SKU atau *stock keeping unit* atau identitas produk yang telah ada (fungsionalitas)	*Form* SKU diinputkan data yang sudah ada di *database*	*Modal dialog* akan memunculkan pesan bahwa SKU telah digunakan	*Modal dialog* muncul (valid)
Fungsionalitas ketika terjadi *error* di dalam *server* (fungsionalitas)	Masukan yang tidak benar dan valid	*Modal dialog* akan memunculkan pesan bahwa telah terjadi *error*	*Modal dialog* pesan kesalahan *error* muncul (valid)

Selanjutnya aksi yang terdapat pada halaman pengelolaan produk ini adalah aksi untuk melakukan pengubahan data produk, data produk akan ditampilkan di dalam *form* sesuai dengan produk yang dipilih sebelumnya. Berikut adalah tampilan untuk melakukan pengubahan data produk yang ditunjukkan pada Gambar 4.10.



Gambar 4.10 *Modal Dialog Edit* Produk

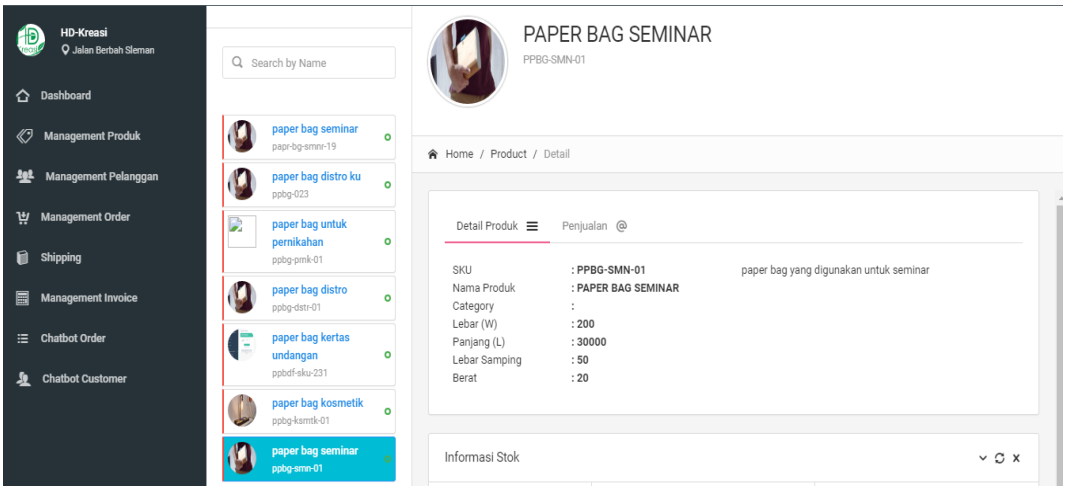
Pada saat pengubahan data selesai dilakukan maka akan muncul pesan bahwa data produk berhasil diubah dan juga ketika terdapat *form* yang tidak dimasukkan dengan benar akan muncul pesan kesalahan pada *form* seperti pada saat aksi penambahan data produk. Berikut ini adalah hasill pengujian pada aksi pengubahan data produk yang ditunjukkan pada Tabel 4.5.

Tabel 4.5 Pengujian Terhadap Aksi Pengubahan Data Produk

Skenario pengujian	**Input**	**Output yang diharapkan**	**Output hasil**
Mengubah data dengan valid (fungsionalitas)	*form* data produk dimasukkan dengan (benar)	*Modal dialog* akan memunculkan pesan bahwa data berhasil di edit	*Modal dialog* muncul dengan pesan berhasil (valid)
Ketika Edit data gagal (fungsionalitas)	Edit data gagal	Akan muncul *modal dialog* bahwa data gagal diedit	*Modal dialog* muncul dengan pesan *error* (valid)
Mengubah data *form* dengan SKU atau *stock keeping unit* atau identitas produk dengan yang telah ada (fungsionalitas)	*Form* SKU diinputkan data yang sudah ada di *database*	*Modal dialog* akan memunculkan pesan bahwa SKU telah digunakan	*Modal dialog* muncul (valid)
Menambah data produk dengan data yang tidak valid (seperti *form* yang wajib diisi dan *form*	*Form* data produk dimasukkan dengan data	Akan muncul pesan kesalahan pada bagian bawah *form* yang diberi aturan seperti *form* yang	Pesan kesalahan validasi pada

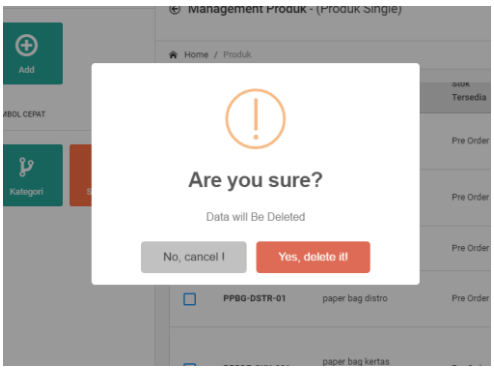
yang harus diisi dengan angka). (input validasi)	yang tidak benar	harus diisi maka akan muncul “this field is required”	*form* muncul (valid)

Aksi selanjutnya yang terdapat di dalam halaman produk ini adalah aksi untuk melihat detail dari produk yang dipilih. Pada halaman detail produk ini akan ditampilkan dari rincian produk seperti informasi stok, grafik penjualan dll. Berikut adalah tampilan dari halaman detail produk yang ditunjukkan pada Gambar 4.11.



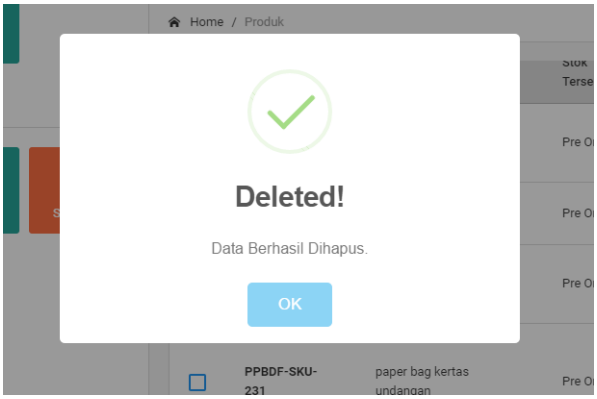
Gambar 4.11 Halaman *Detail* Produk

Pada halaman ini juga *user* dapat menghapus data produk yang dipilih, sebelum data benar-benar dihapus terlebih dahulu sistem akan memunculkan pesan konfirmasi penghapusan. Berikut ini adalah tampilan pesan konfirmasi yang ditunjukkan pada Gambar 4.12.



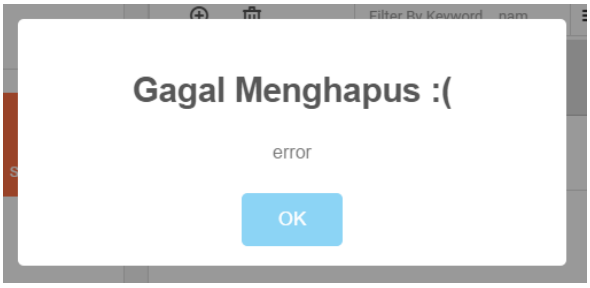
Gambar 4.12 *Modal Dialog* Konfirmasi Hapus Produk

Apabila *user* menekan tombol *delete* maka proses penghapusan dilanjutkan dan apabila *user* membatalkan maka *modal dialog* konfirmasi akan ditutup. Berikut ini merupakan tampilan ketika proses data berhasil dilakukan yang ditunjukkan pada Gambar 4.13.



Gambar 4.13 Modal Dialog Data Produk Berhasil dihapus

Apabila data gagal dihapus maka akan muncul dialog yang memberikan bahwa data produk gagal untuk dihapus. Berikut merupakan *modal dialog* untuk menampilkan bahwa data gagal dihapus yang ditunjukkan pada Gambar 4.14.



Gambar 4 14 *Modal Dialog* Data Gagal dihapus

Berikut ini merupakan detail pengujian terhadap aksi atau fungsionalitas penghapusan data produk yang ditunjukkan pada Tabel 4.6.

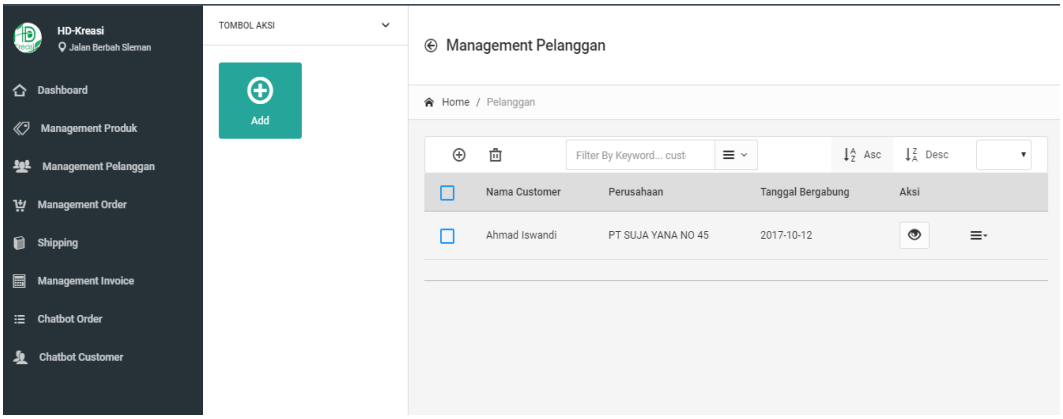
Tabel 4.6 Pengujian terhadap aksi pengubahan data

Skenario pengujian	**Input**	**Output yang diharapkan**	**Output hasil**
Hapus data produk sukses (fungsionalitas)	Klik tombol hapus dari tabel sesuai	*Modal dialog* akan muncul dan memberikan pesan bahwa data berhasil dihapus	*Modal dialog* muncul (valid)

	dengan produk		
Hapus data produk gagal	Klik tombol hapus dari tabel sesuai dengan produkn	*Modal dialog* akan muncul dan memberikan pesan bahwa data berhasil dihapus	*Modal dialog* muncul (valid)

4.4 Halaman Pelanggan

Halaman pelanggan adalah halaman yang akan menampilkan informasi mengenai pelanggan yang terdapat di dalam *database* yang dapat dilihat oleh *user* yang berstatus sebagai pegawai. Di dalam halaman ini akan ditampilkan informasi pelanggan serta dapat dilakukan penambahan, pengubahan ataupun penghapusan data pelanggan dari dalam halaman ini. Gambar 4.15 merupakan halaman halaman pelanggan yang terdapat di dalam sistem pengelolaan pesanan.



Gambar 4.15 Halaman Pengelolaan Pelanggan

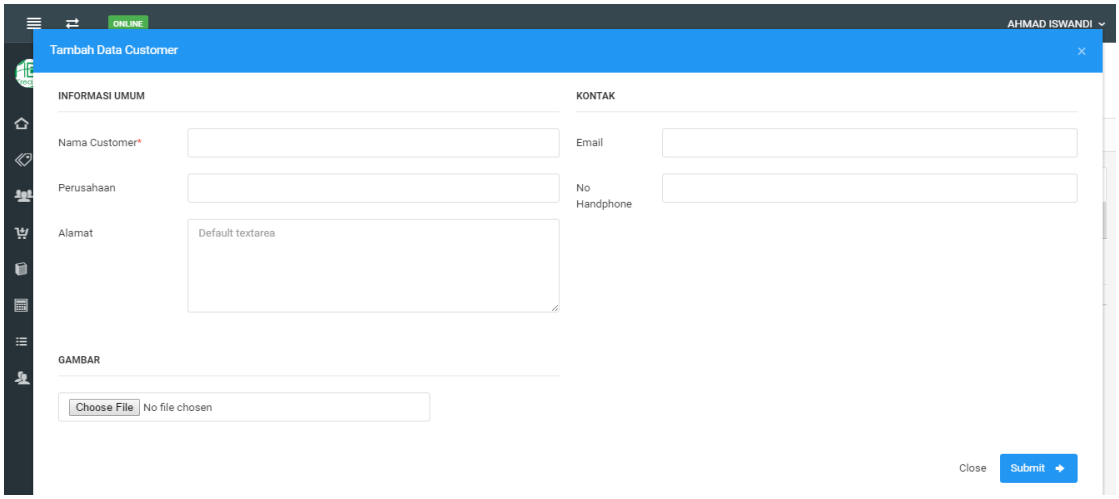
Berikut ini merupakan tabel hasil pengujian terhadap aksi atau fungsionalitas yang terdapat di halaman pelanggan yang ditunjukkan pada Tabel

Tabel 4.7 Pengujian terhadap halaman pelanggan

Skenario pengujian	Input	Output yang diharapkan	Output hasil
Mengakses menu untuk menambah data (fungsionalitas)	Klik *action add*	*Modal dialog* akan muncul dan menampilkan *form*	*Modal dialog* muncul (valid)

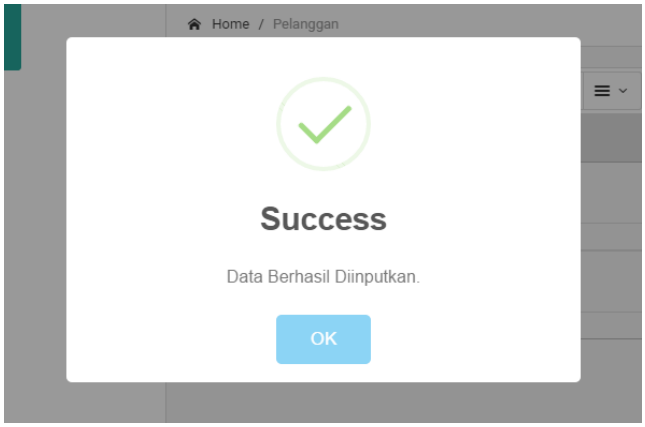
Mengakses menu untuk mengubah data pelanggan (fungsionalitas)	Klik *action* edit pada tabel kolom aksi	*Modal dialog* akan muncul dan menampilkan *form* yang telah terisi data sesuai dengan pelanggan pilihan *user*	*Modal dialog* muncul dan terisi data sesuai dengan pelanggan pilihan *user* (valid)
Mengakses menu untuk melihat detail pelanggan(fungsionalitas)	Klik *action* tombol ber simbol "eye" pada tabel kolom aksi	Tampil dropdown pada tabel	Dropdown tampil pada tabel (valid)
Mengakses menu untuk menghapus data pelanggan	Klik *action* delete pada tabel kolom aksi	Muncul *modal dialog* untuk konfirmasi penghapusan pelanggan	Muncul *modal dialog* konfirmasi penghapusan pelanggan (valid)

Pada halaman pengelolaan pelanggan ini terdapat pilihan aksi yang dapat digunakan oleh *user* yaitu tambah, edit, hapus, lihat detail. Aksi tambah merupakan aksi yang dapat *user* gunakan untuk menambah data pelanggan. Gambar 4.5 merupakan tampilan dari *modal dialog* yang dapat digunakan *user* untuk menambahkan data pelanggan.



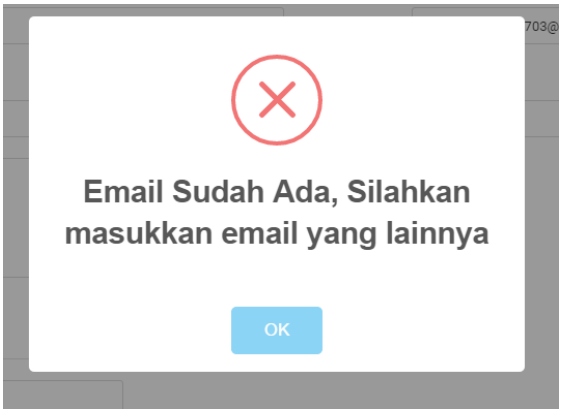
Gambar 4.16 *Modal Dialog* Tambah Data Pelanggan

Pada *fungsionalitas* atau aksi untuk menambahkan data pelanggan ini akan menampilkan pesan bahwa proses memasukkan data berhasil dilakukan yang ditunjukkan pada Gambar 4.17.



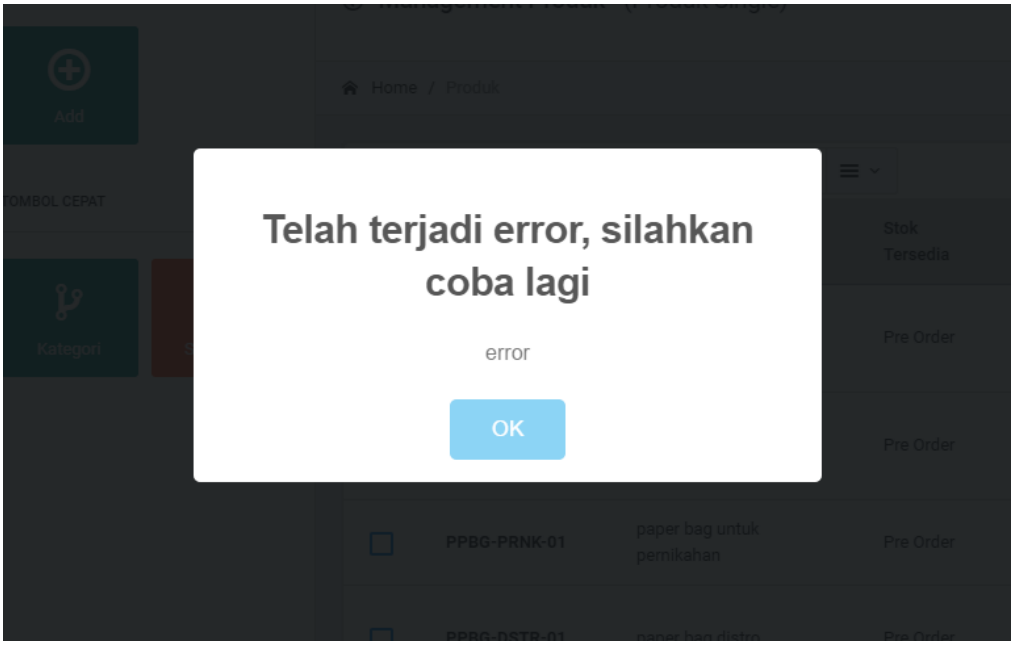
Gambar 4.17 *Modal Dialog* Data Berhasil Dimasukkan

Sistem juga akan memberikan pesan kesalahan saat terjadi *error* atau kesalahan yang dikirimkan oleh *server*. Gambar 4.18 merupakan pesan kesalahan yang terjadi saat terdapat duplikasi nama email yang di inputkan.



Gambar 4.18 Pesan Kesalahan Duplikasi Email

Gambar 4.18 di atas menunjukkan kesalahan bahwa terdapat duplikasi email yang sudah terdaftar di *database*. Selanjutnya adalah pesan kesalahan apabila terjadi *error* pada *server* ataupun masukan yang tidak benar maka akan muncul pesan seperti yang ditunjukkan pada Gambar 4.19.



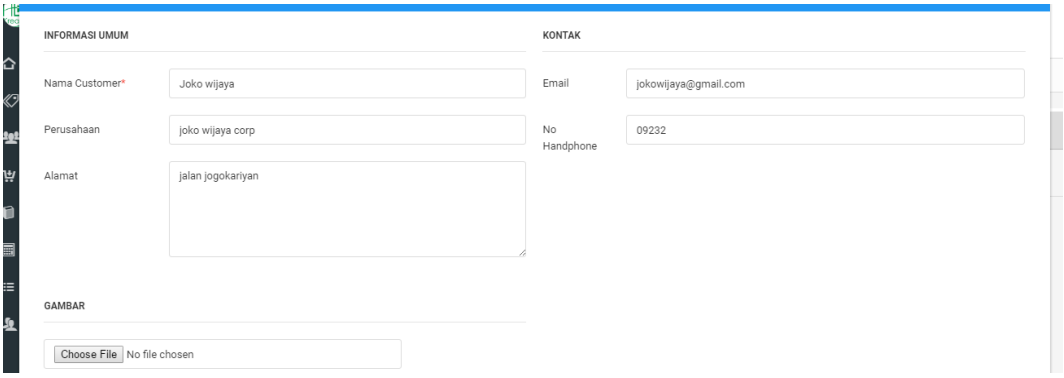
Gambar 4.19 Pesan Kesalahan *Error*

Berikut ini merupakan detail pengujian pada aksi untuk menambahkan data pelanggan yang ditunjukkan pada Tabel 4.8.

Tabel 4.8 Pengujian pada Aksi Tambah Data Pada Halaman Pelanggan

Skenario pengujian	Input	Output yang diharapkan	Output hasil
Menambah data produk dengan data masukan valid dan benar (fungsionalitas)	*form* data produk dimasukkan dengan (benar)	*Modal dialog* akan memunculkan pesan bahwa data berhasil di inputkan	*Modal dialog* muncul (valid)
Menambah data produk dengan data yang tidak valid (seperti *form* yang wajib diisi)	*Form* data produk dimasukkan dengan data yang tidak benar	Akan muncul pesan kesalahan pada bagian bawah *form* yang diberi aturan seperti *form* yang harus diisi maka akan muncul "this field is required"	Pesan kesalahan validasi pada *form* muncul (valid)
Menambahkan data dengan email yang telah ada (fungsionalitas)	*Form* email diinputkan data yang sudah ada di *database*	*Modal dialog* akan memunculkan pesan bahwa email telah digunakan	*Modal dialog* muncul (valid)
Fungsionalitas ketika terjadi *error* di dalam *server* (fungsionalitas)	Masukan yang tidak benar dan valid	*Modal dialog* akan memunculkan pesan bahwa telah terjadi *error*	*Modal dialog* pesan kesalahan *error* muncul (valid)

Selanjutnya aksi yang terdapat pada halaman pengelolaan pelanggan ini adalah aksi untuk melakukan pengubahan data pelanggan. Data pelanggan akan ditampilkan di dalam *form* sesuai dengan pelanggan yang dipilih sebelumnya. Berikut adalah tampilan untuk melakukan pengubahan data pelanggan yang ditunjukkan pada Gambar 4.20.



Gambar 4.20 *Modal Dialog* Ubah Pelanggan

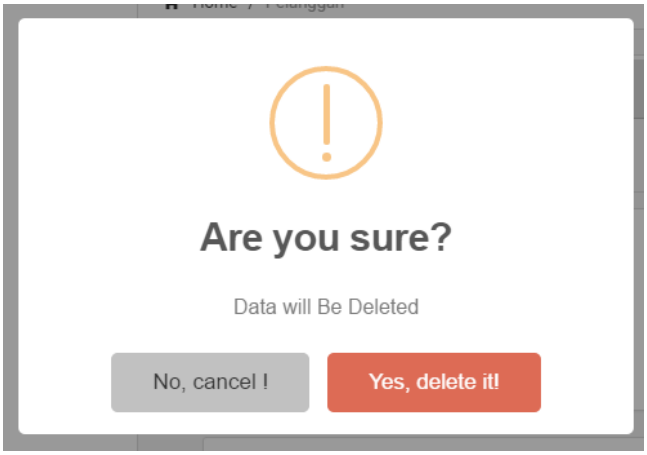
Pada saat pengubahan data selesai dilakukan maka akan muncul pesan bahwa data pelanggan berhasil diubah dan juga ketika terdapat *form* yang tidak dimasukkan dengan benar akan muncul pesan kesalahan pada *form* seperti pada saat aksi penambahan data pelanggan. Berikut ini adalah hasil pengujian pada aksi pengubahan data pelanggan yang ditunjukkan pada Tabel 4.9.

Tabel 4.9 Pengujian pada aksi ubah data pada halaman pelanggan

Skenario pengujian	**Input**	**Output yang diharapkan**	**Output hasil**
Mengubah data dengan valid (fungsionalitas)	*form* data produk dimasukkan dengan (benar)	*Modal dialog* akan memunculkan pesan bahwa data berhasil di edit	*Modal dialog* muncul dengan pesan berhasil (valid)
Ketika Edit data gagal (fungsionalitas)	Edit data gagal	Akan muncul *modal dialog* bahwa data gagal diedit	*Modal dialog* muncul dengan pesan *error* (valid)
Mengubah data *form* dengan email yang sudah tersedia di *database*	*Form* email diinputkan data yang sama dengan yang	*Modal dialog* akan memunculkan pesan bahwa email telah digunakan	*Modal dialog* muncul (valid)

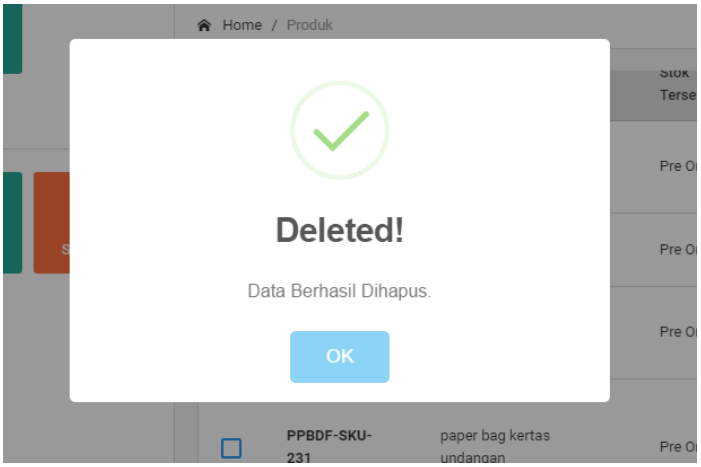
	terdapat di *database*		
Menambah data pelanggan dengan data yang tidak valid (seperti *form* yang wajib diisi). (input validasi)	*Form* data pelanggan dimasukkan dengan data yang tidak benar	Akan muncul pesan kesalahan pada bagian bawah *form* yang diberi aturan seperti *form* yang harus diisi maka akan muncul "this field is required"	Pesan kesalahan validasi pada *form* muncul (valid)

Pada halaman ini juga *user* dapat menghapus data pelanggan yang dipilih, sebelum data benar-benar dihapus terlebih dahulu sistem akan memunculkan pesan konfirmasi penghapusan. Berikut ini adalah tampilan pesan konfirmasi yang ditunjukkan pada Gambar 4.21.



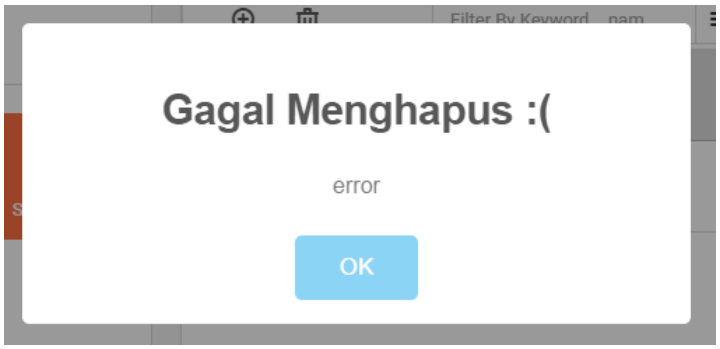
Gambar 4.21 Modal Dialog Konfirmasi Hapus Pelanggan

Apabila *user* menekan tombol *delete* maka proses penghapusan dilanjutkan dan apabila *user* membatalkan maka *modal dialog* konfirmasi akan ditutup. Berikut ini merupakan tampilan ketika proses data berhasil dilakukan yang ditunjukkan pada Gambar 4.23.



Gambar 4.22 *Modal Dialog* Data Pelanggan Berhasil dihapus

Apabila data gagal dihapus maka akan muncul dialog yang memberikan bahwa data pelanggan gagal untuk dihapus. Berikut merupakan *modal dialog* untuk menampilkan bahwa data gagal dihapus yang ditunjukkan pada Gambar 4.24.



Gambar 4.23 Modal Dialog Data Gagal dihapus

Berikut ini merupakan hasil pengujian terhadap aksi penghapusan data pelanggan yang ditunjukkan pada Tabel 4.10.

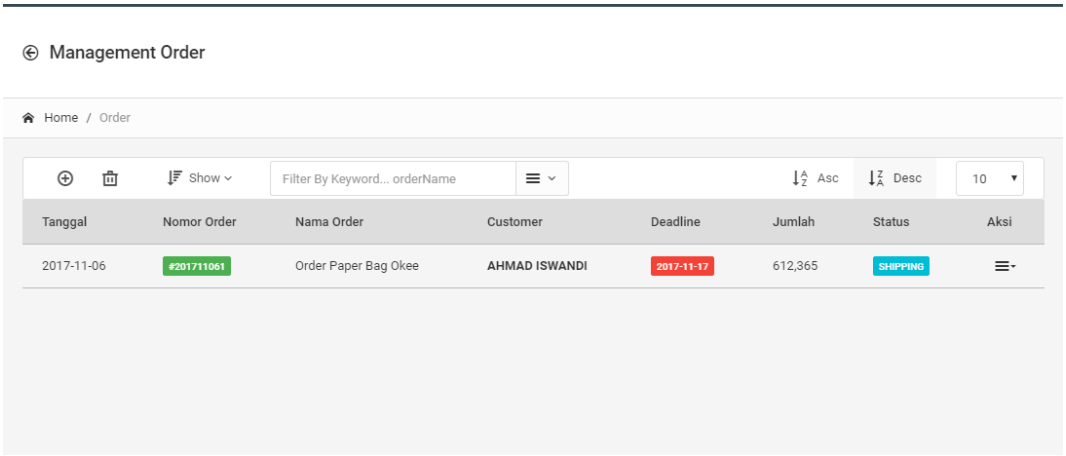
Tabel 4.10 Pengujian Pada Aksi Hapus Data Pada Halaman Pelanggan

Skenario pengujian	**Input**	**Output yang diharapkan**	**Output hasil**
Hapus data produk sukses (fungsionalitas)	Klik tombol hapus dari tabel sesuai dengan pelanggan yang dipilih	*Modal dialog* akan muncul dan memberikan pesan bahwa data berhasil dihapus	*Modal dialog* muncul (valid)

Hapus data produk gagal	Klik tombol hapus dari tabel sesuai dengan pelanggan yang dipilih	*Modal dialog* akan muncul dan memberikan pesan bahwa data berhasil dihapus	*Modal dialog* muncul (valid)

4.5 Halaman Order

Halaman *order* adalah halaman yang akan menampilkan informasi mengenai pesanan produk oleh pelanggan yang terdapat di dalam *database*, yang dapat dilihat oleh *user* yang berstatus sebagai pegawai. Di dalam halaman ini akan ditampilkan informasi pesanan. serta *user* dapat melakukan penambahan, pengubahan ataupun pengapusan data pesanan dari dalam halaman ini. Gambar 4.24 merupakan halaman *order* yang terdapat di dalam sistem pengelolaan pesanan



Gambar 4.24 Halaman Pengelolaan Pesanan

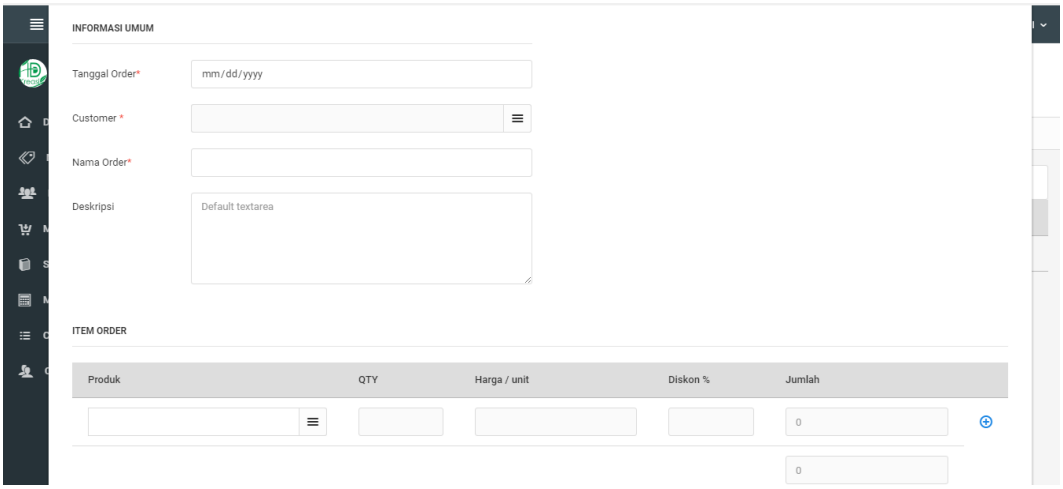
Berikut ini merupakan hasil pengujian terhadap aksi atau fungsionalitas yang terdapat di halaman pengelolaan pesanan yang ditunjukkan pada Tabel 4.11.

Tabel 4.11 Pengujian pada halaman pengelolaan pesanan

Skenario pengujian	Input	Output yang diharapkan	Output hasil
Mengakses menu untuk menambah data (fungsionalitas)	Klik *action add*	*Modal dialog* akan muncul dan menampilkan *form*	*Modal dialog* muncul (valid)
Mengakses menu untuk mengubah data	Klik *action* edit	*Modal dialog* akan muncul dan menampilkan *form* yang telah terisi data sesuai	*Modal dialog* muncul dan terisi data

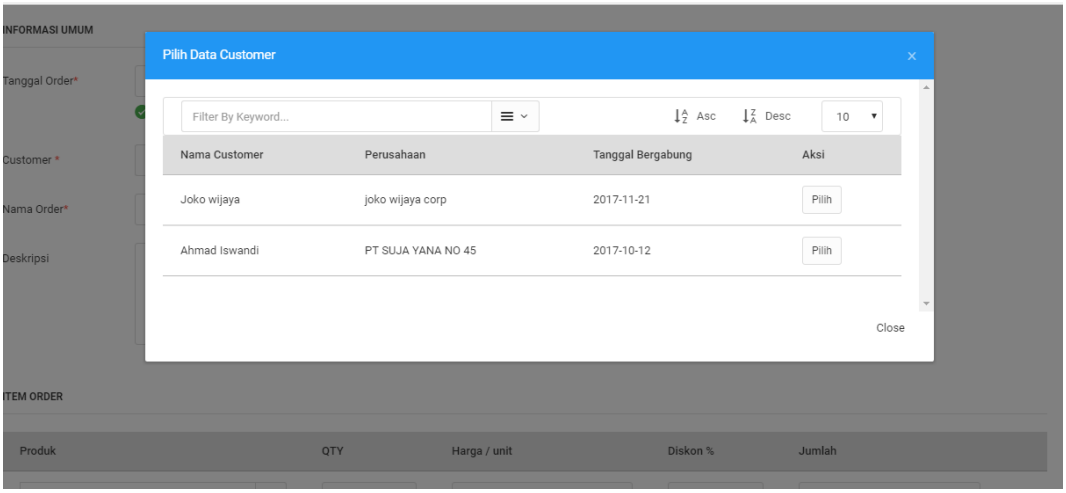
pesanan (fungsionalitas)	pada tabel kolom aksi	dengan pesanan pilihan *user*	sesuai denga pesanan pilihan *user* (valid)
Mengakses menu untuk melihat detail produk (fungsionalitas)	Klik *action* lihat detail pada tabel kolom aksi	Tampil halaman *order detail*	Halaman *order detail* tampil (valid)
Mengakses menu untuk menghapus data pesanan	Klik *action* delete pada tabel kolom aksi	Muncul *modal dialog* untuk konfirmasi penghapusan pesanan	Muncul *modal dialog* konfirmasi penghapusan pesanan (valid)

Pada halaman pengelolaan pesanan ini terdapat pilihan aksi yang dapat digunakan oleh *user* yaitu tambah, edit, hapus, dan lihat detail. Aksi tambah merupakan aksi yang dapat *user* gunakan untuk menambah data pesanan. Gambar 4.25 merupakan tampilan dari *modal dialog* yang dapat digunakan *user* untuk menambahkan data pesanan.



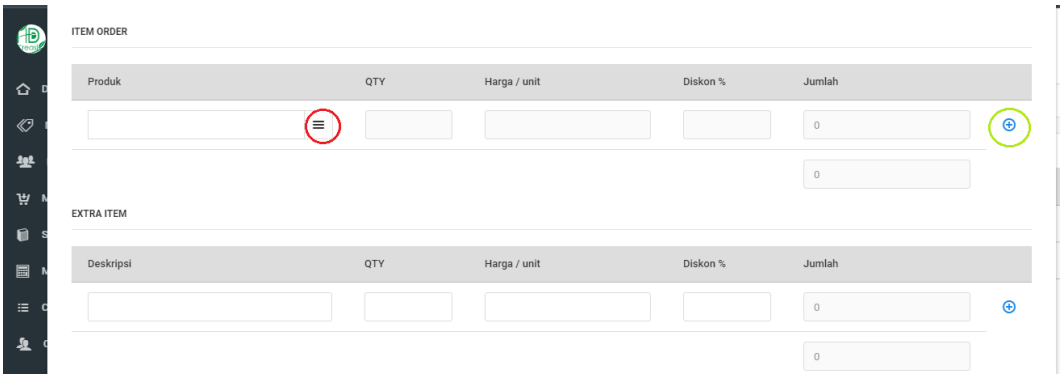
Gambar 4.25 *Modal dialog* Untuk Menambahkan Data Pesanan

Pada *fungsionalitas* atau aksi untuk menambahkan data pesanan ini, terdapat *form* yang digunakan untuk mengisi data pelanggan yang akan melakukan pesanan. *User* akan melakukan pemilihan pelanggan yang dipilih dari *modal dialog* yang muncul ketika *form customer* ditekan. Berikut merupakan *modal dialog* pilih *customer* yang ditunjukkan pada Gambar 4.26.



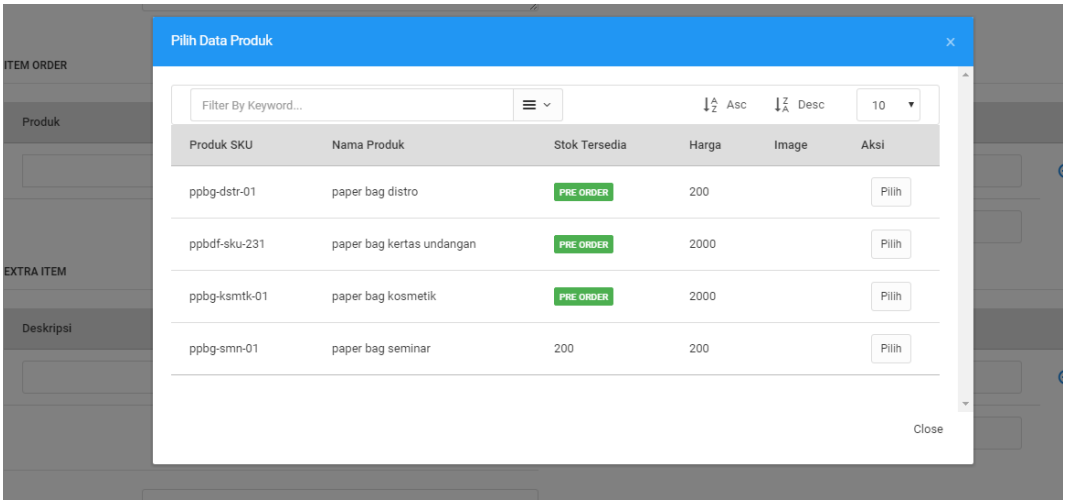
Gambar 4.26 *Modal Dialog* Pilih Data Customer

Ketika *user* menekan tombol aksi pilih pada salah satu baris tabel *customer*, data akan otomatis terisi pada *form order* seperti yang ada pada Gambar 4.25. Selanjutnya adalah pemilihan produk untuk dijadikan sebagai data *item* pesanan. *User* harus menekan tombol yang berada pada bagian *item order* untuk memunculkan *modal dialog* tabel produk. Sebelum produk dipilih maka *form* yang berkaitan seperti qty, diskon tidak dapat diisi. Berikut merupakan area di mana *user* akan memilih produk seperti yang ditunjukkan pada Gambar 4.27.



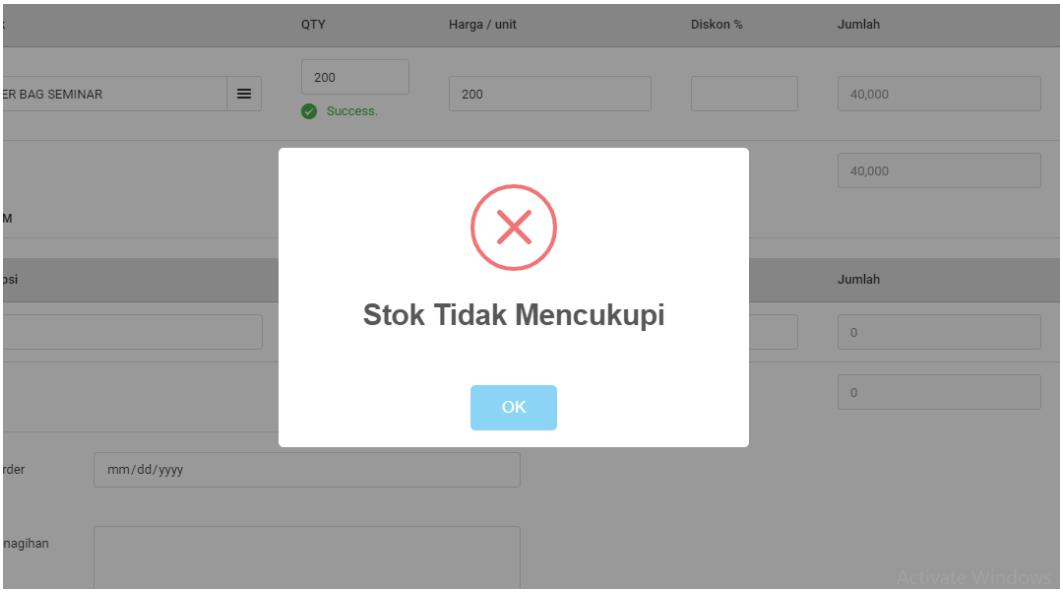
Gambar 4.27 Area *Form* Untuk Memilih Produk

Gambar 4.27 di atas terdapat sebuah tombol untuk memunculkan tabel produk dalam *modal dialog*. Sedangkan lingkaran hijau pada gambar digunakan untuk menambah data produk lainnya untuk dimasukkan sebagai *item* dalam pesanan. Berikut adalah tampilan *modal dialog* produk yang ditunjukkan pada Gambar 4.28.



Gambar 4.28 *Modal Dialog* Data Produk

Setelah *user* memilih produk dengan menekan tombol aksi pada baris tabel produk maka akan secara otomatis *form* yang terdapat di dalam area *item order* yang dipilih akan terisi nama produk tersebut. Selain itu di dalam aksi penambahan pesanan ini akan dilakukan pengecekan terhadap jumlah pesanan yang akan dipesan oleh *customer*. Apabila jumlah pesanan melebihi dari jumlah ketersediaan barang seperti yang terlihat pada Gambar 4.28 di dalam tabel pada baris paling bawah, maka sistem akan menampilkan pesan kesalahan bahwa jumlah pesanan untuk produk tersebut melewati jumlah ketersediaan barang. Berikut merupakan pesan kesalahan yang ditunjukkan pada Gambar 4.29.



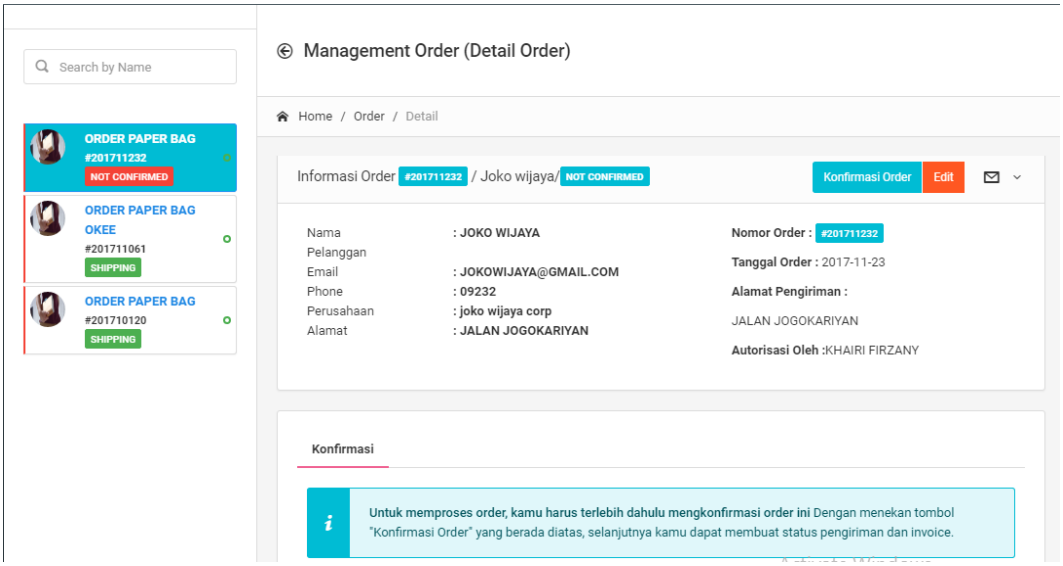
Gambar 4.29 Pesan Kesalahan Saat Stok Tidak Mencukupi

Berikut ini merupakan tabel hasil pengujian pada aksi tambah pesanan di halaman pengelolaan pesanan yang ditunjukkan pada Tabel 4.13.

Tabel 4.12 Pengujian Pada Aksi Tambah di Halaman Pesanan

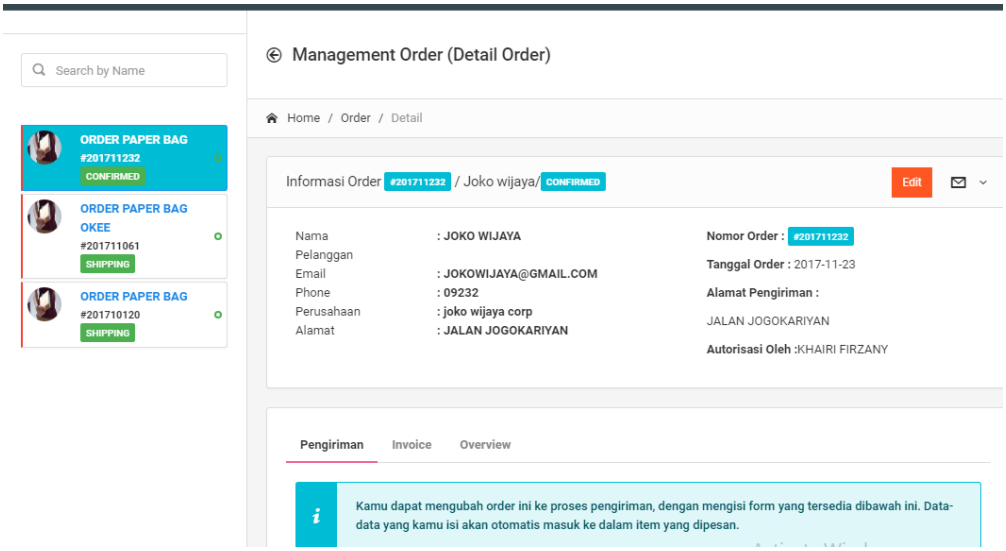
Skenario pengujian	**Input**	**Output yang diharapkan**	**Output hasil**
Menambah data produk dengan data masukan valid dan benar (fungsionalitas)	*Form* data produk dimasukkan dengan (benar)	*Modal dialog* akan memunculkan pesan bahwa data berhasil di inputkan	*Modal dialog* muncul (valid)
Menambah data produk dengan data yang tidak valid (seperti *form* yang wajib diisi dan *form* yang harus diisi dengan angka). (input validasi)	*Form* data produk dimasukkan dengan data yang tidak benar	Akan muncul pesan kesalahan pada bagian bawah *form* yang diberi aturan seperti *form* yang harus diisi maka akan muncul "this field is required"	Pesan kesalahan validasi pada *form* muncul (valid)
Menambahkan data jumlah produk yang dipesan melewati batas jumlah stok produk	Masukan jumlah produk yang dipesan melewati jumlah stok	*Modal dialog* akan memunculkan pesan bahwa stok tidak mencukupi	*Modal dialog* muncul (valid)
Fungsionalitas ketika terjadi *error* di dalam *server* (fungsionalitas)	Masukan yang tidak benar dan tidak valid	*Modal dialog* akan memunculkan pesan bahwa telah terjadi *error*	*Modal dialog* pesan kesalahan *error* muncul (valid)
Memilih data *customer*	Klik tombol pada *form customer*	*Modal dialog* pilih data *customer* akan muncul	*Modal dialog* muncul (valid)
Memilih data produk	Klik tombol pada *form item order*	*Modal dialog* pilih data produk akan muncul	*Modal dialog* muncul (valid)

Aksi selanjutnya yang terdapat di halaman ini adalah aksi yang digunakan untuk melihat detail dari pesanan. Untuk melakukan aksi tersebut *user* dapat menekan tombol aksi lihat detail di dalam tabel pesanan sesuai dengan pesanan yang akan dilihat detail informasi yang terdapat dalam pesanan tersebut. Berikut ini merupakan halaman detail pesanan yang ditunjukkan pada Gambar 4.30.



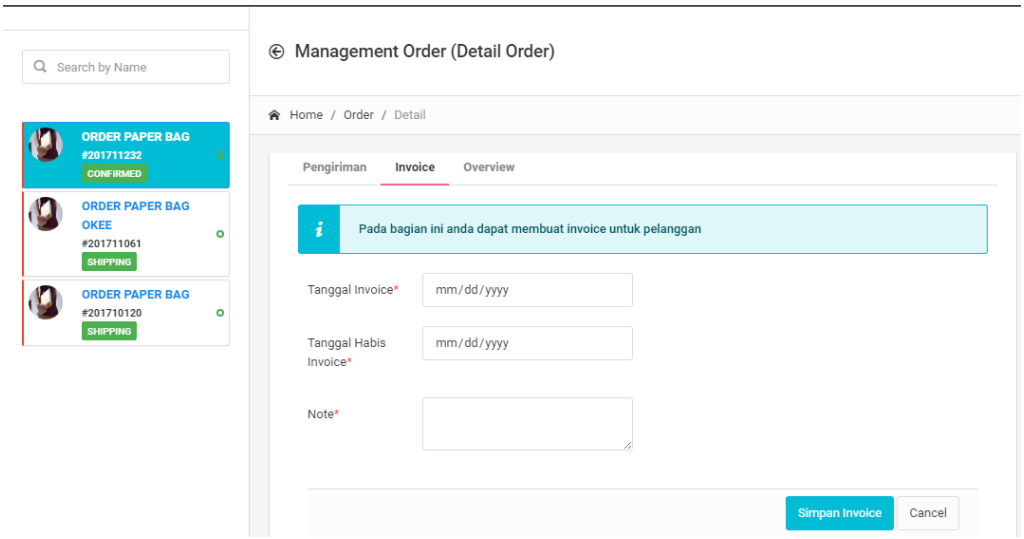
Gambar 4. 30 Tampilan Halaman Detail Pesanan

Di dalam halaman *order detail* ini terdapat beberapa aksi atau fungsionalitas yang dapat digunakan oleh *user* seperti melakukan konfirmasi *order*, mengubah data, membuat *invoice*, membuat label pengiriman, dan mengirimkan status pesanan. Aksi konfirmasi *order* adalah aksi yang digunakan untuk mengubah status pesanan dari status *not confirmed* menjadi *confirmed.* Jika konfirmasi berhasil dilakukan tampilan status pada halaman detail pesanan akan berubah. Berikut merupakan tampilan setelah pesanan berhasil dikonfirmasi yang ditunjukkan pada Gambar 4.31.



Gambar 4.31 Tampilan Detail Pesanan Setelah dikonfirmasi

Dalam halaman detail pesanan ini juga terdapat aksi untuk membuat *invoice* pada *tab invoice.* Berikut ini merupakan aksi yang digunakan untuk membuat *invoice* yang ditunjukkan pada Gambar 4.32.



Gambar 4.32 Aksi Untuk Menambah *Invoice*

Berikut ini merupakan tabel hasil pengujian pada aksi di halaman detail pesanan yang ditunjukkan pada Tabel 4.13.

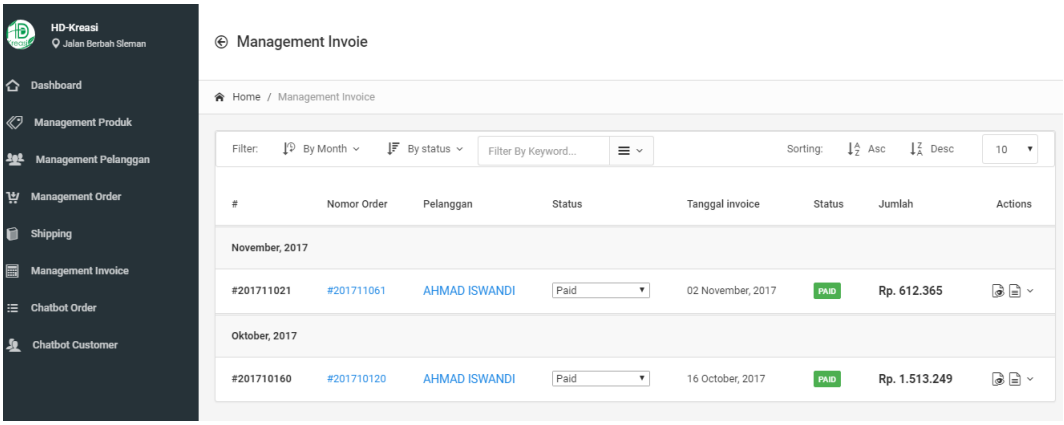
Tabel 4.13 Pengujian Pada Aksi Halaman Detail Pesanan

Skenario pengujian	**Input**	**Output yang diharapkan**	**Output hasil**
Melakukan konfirmasi *order* (fungsionalitas)	Klik tombol konfirmasi *order*	Halaman order detail akan berubah secara langsung	Halaman detail berubah (valid)
Mengubah data detail pesanan dengan menggunakan daftar pesanan yang ada di *sidebar*	Klik salah satu produk yang ada di *sidebar*	Halaman *order detail* akan berubah	Halaman berubah sesuai dengan informasi produk yang dipilih (valid)
Melakukan pengubahan informasi data pesanan (edit)	Klik tombol *edit*	*Modal dialog* akan memunculkan ubah pesanan	*Modal dialog* muncul (valid)
Melakukan pembuatan *invoice* (sukses)	*Input form* di tab *invoice*	akan muncul *layout invoice*	*Layout invoice* muncul (valid)

Melakukan pembuatan *invoice* (gagal)	*Input form* di tab *invoice*	*Modal dialog* akan memunculkan bahwa invoice gagal dibuat	*Modal dialog* muncul (valid)
Melakukan pengiriman invoice melalui *email*	Klik tombol kirim *invoice* by *email*	*Modal dialog* akan memunculkan bahwa invoice berhasil dikirim	*Modal dialog* muncul (valid)

4.6 Halaman Invoice

Halaman *invoice* merupakan halaman yang terdapat di dalam sistem pengelolaan pesanan. Di dalam halaman ini *user* dapat melakukan pengelolaan *invoice* seperti menghapus data invoice, mengubah status *invoice*, melakukan pencetakan *invoice* dan melihat detail *invoice*. Berikut ini merupakan halaman *invoice* yang ditunjukkan pada Gambar 4.33.



Gambar 4.33 Halaman Pengelolaan *Invoice*

Berikut ini merupakan tabel hasil pengujian pada aksi atau fungsionalitas yang terdapat pada halam *invoice* ini yang ditunjukkan pada Tabel 4.14.

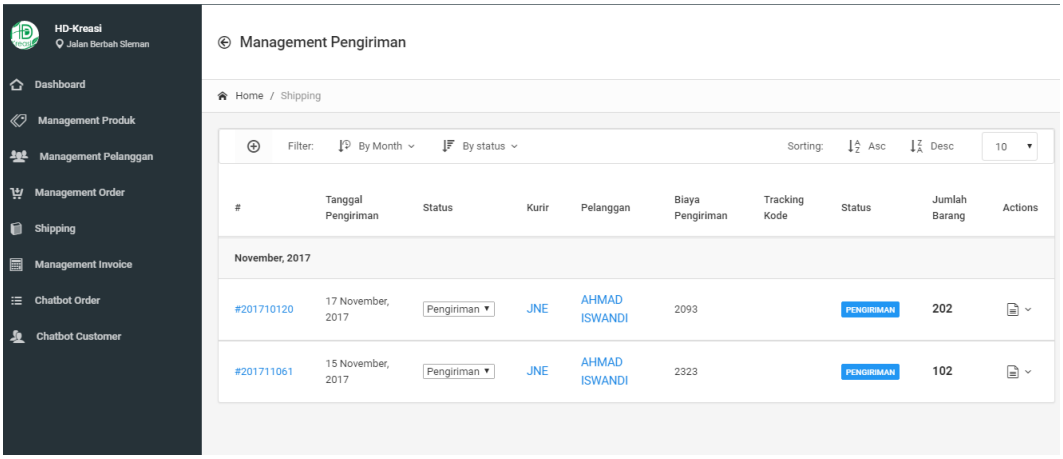
Tabel 4.14 Pengujian Pada Halaman Invoice

Skenario pengujian	Input	Output yang diharapkan	Output hasil
Melakukan pengubahan status *invoice*	Pilih status di dalam tabel	*Modal dialog* akan muncul untuk mengkonfirmasi pengubahan status	*Modal dialog* muncul (valid)
Melihat detail *invoice*	Klik tombol lihat detail di dalam tabel	*Modal dialog* akan muncul dengan detail *invoice*	*Modal dialog* muncul (valid)

Melakukan pencetakan *invoice*	Klikn tombol print	File akan terdownload dengan jenis file adalah pdf	File terdownload (valid)

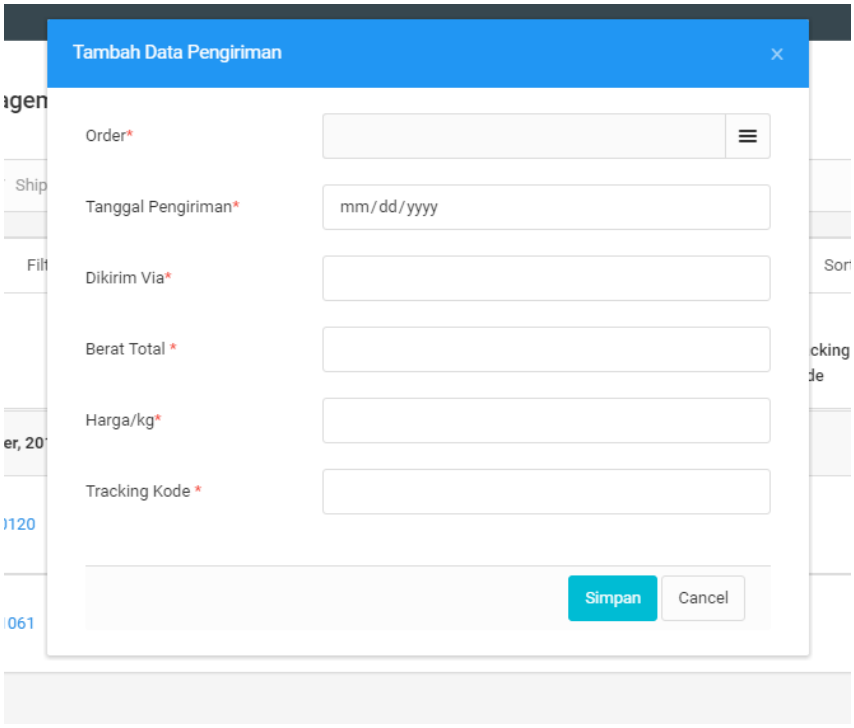
4.7 Halaman Shipping

Halaman *shipping* merupakan halaman yang terdapat di dalam sistem pengelolaan pesanan yang dapat diakses oleh pengguna yang berstatus sebagai pegawai. Di dalam halaman ini *user* dapat melakukan pengelolaan pesanan yang akan dikirim atau dalam proses pengiriman, *user* dapat melakukan tugas seperti menambah data pengiriman, menghapus data pengiriman dan melakukan pengubahan status pengiriman. Berikut ini merupakan halaman *shipping* yang ditunjukkan pada Gambar 4.34.



Gambar 4.34 Halaman Pengiriman

Di dalam halaman ini, *user* dapat melakukan beberapa tugas dari fungsionalitas yang tersedia di halaman *shipping* ini. Aksi untuk menambah data pengiriman dapat dilakukan dengan menekan tombol *plus*, lalu sistem akan memunculkan *modal dialog* yang berisi *form* yang akan diisi oleh *user*. Berikut ini merupakan *modal dialog* untuk menambah data *form* yang ditunjukkan pada Gambar 4.35.



Gambar 4.35 *Modal Dialog* Tambah Data Pengiriman

Selain itu terdapat juga aksi-aksi lainnya di dalam halaman ini seperti melakukan pengubahan data yang memiliki kesamaan dengan halaman lain saat akan melakukan pengubahan data seperti terdapat *validasi form*, dan pesan kesalahan serta pesan sukses ketika data berhasil ditambah ataupun diubah. Pengujian pada halaman ini dilakukan dengan melihat apakah fungsionalitas yang terdapat di dalam halaman ini tepat sesuai yang diinginkan. Berikut ini merupakan tabel hasil pengujian yang ditunjukkan pada Tabel 4.15.

Tabel 4.15 Pengujian Pada Halaman Shipping

Skenario pengujian	**Input**	**Output yang diharapkan**	**Output hasil**
Mengakses menu untuk menambah data (fungsionalitas)	Klik *action add*	*Modal dialog* akan muncul dan menampilkan *form*	*Modal dialog* muncul (valid)
Mengakses menu untuk mengubah data pengiriman (fungsionalitas)	Klik *action* edit pada tabel kolom aksi	*Modal dialog* akan muncul dan menampilkan *form* yang telah terisi data sesuai dengan produk pilihan *user*	*Modal dialog* muncul dan terisi data sesuai denga produk pilihan *user* (valid)
Masukan data yang tidak valid	Inputan yang tidak benar	Sistem akan memunculkan *error* mengenai validasi *form*	Muncul *error* (Valid)

	atau valid		
Mengakses menu untuk menghapus data pengiriman	Klik *action delete* pada tabel kolom aksi	Muncul *modal dialog* untuk konfirmasi penghapusan pengiriman	Muncul *modal dialog* konfirmasi penghapusan pengiriman (valid)

4.8 Implementasi Chatbot

Pada tahapan ini penulis akan menjelaskan mengenai penerapan *chatbot* di dalam sistem pengelolaan pesanan yang penulis buat. Ada beberapa tahapan penerapan *chatbot* yang penulis lakukan seperti, pembuatan dan penggunaan layanan Messenger yaitu *platform* pesan yang dimiliki oleh Facebook, pembuatan dan penggunaan layanan API.AI, menghubungkan layanan Messenger dengan API.AI melalui *webhook* untuk membangun *chatbot* yang dapat memahami masukan teks dari *user*, menghubungkan *chatbot* dengan *web services* untuk mengambil dan mengirim data dari atau ke dalam *database* melalui *webhook* dan pengujian *chatbot*.

4.8.1 Penggunaan Layanan Messenger

Penggunaan layanan Messenger harus dimulai dengan membuat akun untuk dapat menggunakan semua fungsionalitas yang terdapat di Messenger. Untuk itu terdapat persyaratan yang harus dipersiapkan untuk dapat menggunakan layanan ini. Berikut merupakan beberapa persyaratan untuk dapat menggunakan layanan Messenger ini:

a. Halaman Facebook: Halaman Facebook digunakan sebagai identitas dari sebuah *chatbot* yang akan dibuat. Pengembang harus membuat halaman Facebook yang digunakan sebagai identitas awal agar *user* dapat menemukan dan berinteraksi dengan *chatbot* yang dibuat. Halaman Facebook yang dibuat adalah halaman *fanpage* yang layaknya sebuah halaman *blog*, yang menyediakan informasi beragam sesuai dengan keinginan pemiliknya.
b. Akun pengembang Facebook: Akun ini merupakan akun yang harus dibuat untuk dapat memakai dan membuat aplikasi yang terintegrasi dengan Facebook.
c. Aplikasi Facebook:Aplikasi Facebook berisi pengaturan untuk bot Messenger, termasuk untuk mendapatkan token akses.

d. URL *Webhook*: Tindakan yang dilakukan dalam percakapan dengan bot, semua peristiwa yang terjadi di dalam *bot* akan dikirimkan ke *server webhook,* seperti pesan yang baru masuk .

4.8.2 Webhook dan Integrasi dengan Layanan Messenger

Webhook merupakan sebuah *server* yang digunakan untuk menerima *request* dan mengirimkan *response* kepada komponen-komponen lainnya yang melakukan *request* ke *server webhook* ini. Komponen lainnya yang dimaksudkan disini adalah komponen seperti layanan Messenger dan layanan API.AI agar dapat melakukan komunikasi antar satu layanan dengan layanan lainnya. Di sini penulis membuat *webhook* menggunakan *framework* node.js. Node.js penulis gunakan sebagai *server* yang akan menerima permintaan dari luar dan mengirimkan *response* kembali kepada peminta. Berikut merupakan pembuatan *server* menggunakan Node.js yang ditunjukkan pada Gambar 4.36.

```
const express = require('express')
const path = require('path')
const PORT = process.env.PORT || 5000

express()
  .use(express.static(path.join(__dirname, 'public')))
  .set('views', path.join(__dirname, 'views'))
  .set('view engine', 'ejs')
  .get('/', (req, res) => res.render('pages/index'))
  .listen(PORT, () => console.log(`Listening on ${ PORT }`))
```

Gambar 4.36 Pembuatan *Server* dengan *Framework* Node.js

Gambar 4.36 di atas adalah pembuatan sebuah *server* sederhana dengan *framework* node.js dengan port 5000. Untuk mencoba apakah *server* berhasil dibuat dan dapat digunakan penulis mengakses sebuah url dari komputer penulis dan meminta *request* ke *server* tersebut dengan alamat http://localhost:5000. Penulis mendapati *response* dari *server* berupa halaman *index* atau halaman pertama saat penulis mengakses url tersebut, sehingga penulis berasumsi bahwa berjalan dan dapat diakses. Selanjutnya lalu penulis menaruh *server* tersebut ke dalam *hosting* yang penulis buat dengan url yang dapat diakses di http://arcane-mesa-75663.herokuapp.com/.

Untuk dapat menggunakan layanan Messenger sebelumnya penulis harus mengetahui fungsionalitas apa yang penulis butuhkan untuk keperluan merancang *chatbot*. Layanan Messenger membutuhkan sebuah *webhook* yang merupakan tempat kode fungsionalitas Messenger disimpan dan juga sebagai tempat untuk menerima, memproses dan mengirimkan pesan kepada *user*. Langkah awal adalah membuat sebuah *endpoint* di dalam *server webhook* untuk menerima semua *request* yang masuk dari Messenger yang dibuat oleh *user*. Berikut

adalah contoh kode dari pembuatan *endpoint* di dalam *webhook* yang telah disediakan oleh Facebook untuk keperluan layanan Messenger yang dibuat, ditunjukkan pada Gambar 4.37.

```
app.post('/webhook', function (req, res) {
  var data = req.body;

  // Make sure this is a page subscription
  if (data.object == 'page') {
    data.entry.forEach(function(pageEntry) {
      var pageID = pageEntry.id;
      var timeOfEvent = pageEntry.time;
      pageEntry.messaging.forEach(function(messagingEvent) {
        if (messagingEvent.optin) {
          receivedAuthentication(messagingEvent);
        } else if (messagingEvent.message) {
          receivedMessage(messagingEvent);
        } else if (messagingEvent.postback) {
          receivedPostback(messagingEvent);
        } else if (messagingEvent.read) {
          receivedMessageRead(messagingEvent);
        } else {
          console.log("Webhook received unknown messagingEvent: ", messagingEvent);
        }

      });
    });
    res.sendStatus(200);
  }});
```

Gambar 4.37 *Endpoint Webhook* Untuk Layanan Messenger

Pada kode di atas memiliki tujuan untuk menerima *request* yang datang dari Messenger. Pesan yang dituliskan oleh *user* di Messenger akan dikirimkan kedalam *endpoint* ini, lalu di dalam *server webhook* ini pulalah dikirimkan *response* kepada *user* yang mengirimkan pesan tadi. Berikut ini merupakan sebuah fungsi untuk melakukan pengiriman *response user* yang ditunjukkan pada Gambar 4.38.

```
function sendTextMessage(recipientId, messageText) {
  var messageData = {
    recipient: {
      id: recipientId
    },
    message: {
      text: messageText,
      metadata: "DEVELOPER_DEFINED_METADATA"
    }
  };
  callSendAPI(messageData);
}
```

Gambar 4.38 Fungsi Untuk Mengirimkan *Response* Berupa Teks

Fungsi di atas merupakan salah satu fungsi yang dapat digunakan untuk melakukan pengiriman sebuah *response* dari *request* yang dikirim oleh *user*. Terdapat beberapa fungsionalitas lagi yang dapat digunakan seperti mengirimkan *template generic*, *invoice template* dan lainnya. Penggunaan fungsionalitas tersebut disesuaikan dengan keperluan, seperti saat akan mengirimkan daftar produk maka fungsi yang akan tepat digunakan adalah dengan mengirimkan *template generic*. Untuk dapat mengirim *response* ke *user* dapat dilakukan dengan mengirimkan *request* ke layanan Sessenger seperti yang ditunjukkan pada Gambar 4.39 .

```
function callSendAPI(messageData) {
  request({
    uri: 'https://graph.facebook.com/v2.6/me/messages',
    qs: { access_token: PAGE_ACCESS_TOKEN },
    method: 'POST',
    json: messageData

  }, function (error, response, body) {
    if (!error && response.statusCode == 200) {
      var recipientId = body.recipient_id;
      var messageId = body.message_id;

      if (messageId) {
        console.log("Successfully sent message with id %s to recipient %s",
          messageId, recipientId);
      } else {
      console.log("Successfully called Send API for recipient %s",
        recipientId);
      }
    } else {
      console.error("Failed     calling     Send     API",     response.statusCode,
response.statusMessage, body.error);
    }
  });
}
```

Gambar 4.39 Request ke Layanan Messenger Untuk Mengirimkan Pesan ke *User*

4.8.3 Integrasi Messenger dan Layanan API.AI Melalui Webhook

Salah satu tujuan dari penelitian ini di antaranya adalah membuat sebuah *chatbot* yang dapat memahami masukan yang diberikan oleh *user* yang melakukan percakapan dengan *chatbot*. Untuk dapat melakukannya disini penulis menggunakan sebuah layanan NLP (*Natural Language Processing)* dari API.AI yang akan melakukan pengolahan dan pemahaman terhadap *request text* yang dikirim oleh *user*. Sehingga hasil akhir yang akan dicapai adalah *chatbot* yang memiliki kemampuan untuk menjawab pertanyaan yang diajukan oleh *user* dengan berbagai macam maksud pertanyaan dengan tepat. Pentingnya *webhook* sebagai penghubung antara *platform* pesan teks seperti Messenger dan layanan API.AI sangatlah diperlukan, dikarenakan di *webhook* segala penerimaan *request*, pemrosesan dan pengiriman *response* dilakukan untuk setiap komponen yang terhubung. Penulis telah menjelaskan sebelumnya pada BAB III Metodologi Penelitian tentang bagaimana layanan API.AI ini bekerja. Pada tahapan ini akan dijelaskan langkah yang lebih teknis bagaimana menghubungkan Messenger dengan layanan API.AI ini. Untuk menggunakan layanan ini sebelumnya penulis harus membuat sebuah akun di layanan tersebut dan mendaftarkan alamat url *server webhook* agar setiap request dan *response* ditujukan ke *endpoint* yang tepat yang telah dibuat di dalam *server webhook*. Alamat url *server webhook* yang didaftarkan adalah https://arcane-mesa-75663.herokuapp.com/ai. Berikut adalah kode *endpoint* untuk API.AI yang bertujuan untuk menerima *request* yang datang dari layanan tersebut, yang ditunjukkan pada Gambar 4.40.

```
app.post('/ai', (req, res) => {
  console.log('*** Webhook for api.ai query ***');
  console.log(req.body.result);

  let msg="hai";
  if (req.body.result.metadata.intentName === 'welcom-message') {
        return res.json({
          speech: msg,
          displayText: msg,
          source: 'welcom-message'
        });
}
});
```

Gambar 4.40 Kode *Endpoint* untuk Layanan API.AI

Kode diatas akan menerima request yang datang dari layanan API.AI yang dikirim melalui HTTP POST. Tujuannya adalah untuk melakukan *fullfilment* atau pengisian *response* yang sesuai untuk diberikan sebagai jawaban *chatbot* atas pertanyaan yang diajukan oleh *user* di dalam sebuah percakapan. Pada baris ke-6 di dalam kode tersebut, merupakan sebuah

kondisi apabila *request* yang masuk memiliki data *intentName* adalah *welcome-message*, yang apabila kondisi terpenuhi *response* yang akan dikirim kembali ke layanan API tersebut terlihat pada baris ke-8 objek *speech* diisi dengan variable "msg" yang berisi kata "hai". Sebelumnya, layanan ini tidak akan mengirim *request* apapun ke *endpoint* tersebut jika tidak ada *request* yang masuk ke dalam layanan API.AI tersebut. Pengiriman request ke layanan tersebut dilakukan untuk mengirimkan teks yang datang dari *plaltform* Messenger untuk dapat dilakukan pemahaman atas maksud dari teks yang dikirimkan. Berikut ini adalah sebuah fungsi yang terdapat di dalam *server webhook* yang berguna untuk melakukan permintaan ke layanan API.AI untuk dilakukan pengolahan dan pemahaman teks yang ditunjukkan pada Gambar 4.41.

```
function nl_send(messengerText, senderID){
        const app = nlp("ee4b10fce9a845419f72802b05ac8257");
        var apiai = app.textRequest(messengerText, {
            sessionId: 'test_session'
        });
```

Gambar 4.41 Kode untuk Melakukan *Request* ke Layanan API.AI

Di dalam kode tersebut fungsi "nl_send" memiliki dua parameter yang berisi *messengerText* dan *senderID.* Fungsi ini akan dipanggil ketika terdapat request yang masuk dari *user* melalui *platform* Messenger. Pada baris ke-2 pada kode di atas adalah melakukan inisialisasi objek nlp dengan nomor token yang didapat saat proses pendaftaran di layanan API.AI. Penulis menggunakan SDK (*Software Development Kit)* yang sudah disediakan oleh layanan API.AI. Sehingga penulis hanya tinggal memakai fungsi yang sudah ada dan menginisialisasi objek sebelum menggunakan fungsi yang tersedia. Pada baris ke-3 merupakan kode yang digunakan untuk melakukan *request* ke layanan API.AI dengan mengirimkan teks yang akan diolah dan dipahami.

Selanjutnya, setelah melakukan *request* ke dalam layanan API.AI, layanan ini akan mulai melakukan pemahaman terhadap *request text* yang dikirimkan. Apabila sudah didapati maksud dan sudah dipetakan ke dalam *intent* maka layanan tersebut akan melakukan *fullfilment* untuk *response* yang akan dikirim kembali ke *webhook* dan dikirim ke *user* di *platform* Messenger. Pengisian *fullfilment* dilakukan di sisi *webhook* dengan mengakses *endpoint* yang sudah dibuat seperti yang ditunjukkan pada Gambar 4.40. Pengisian fullfilment merupakan pilihan *optional*, yang dapat dilakukan atau tidak sama sekali. Layanan API.AI ini menggunakan teknologi *socket* untuk berkomunikasi dan mengirimkan hasil akhir dari jawaban yang didapat sesuai

dengan maksud pertanyaan secara *realtime*. Berikut ini merupakan kode untuk menerima *response* dari layanan API.AI yang ditunjukkan pada Gambar 4.42.

```
apiai.on('response',(responsed)=>{
let aiText = responsed.result.fulfillment.speech;
let intent= responsed.result.metadata.intentName;
let action=responsed.result.action;
if(intent==="welcom-message"){
       console.log("sender IDIDIDID",senderID);
       sendButtonMessage(senderID);
}
```

Gambar 4.42 Kode untuk Menerima Respon dari Layanan API.AI

Kode di atas merupakan mekanisme untuk menerima *response* atau jawaban yang diberikan dari layanan API.AI dari *request text* yang dikirim ke layanan tersebut. Pada baris ke-2, merupakan variabel aitext yang berisi *value* dari objek yang terdapat di dalam parameter *responsed* dengan objek *speech*. *Response* dari layanan API.AI ini adalah berupa data berformat json yang berisi objek yang mewakili informasi seperti *intentName, speech, paramater* dan lainnya. Di baris ke-3 berisi variabel yang diberikan value nama *intent* dari response yang datang dari layanan API.AI. Selanjutnya penulis akan menjelaskan mengenai penanganan untuk setiap *intent* yang telah penulis rancang sebelumnya. Ada beberapa intent penting yang sudah penulis buat di dalam layanan ini yang digunakan untuk mengidentifikasi dan memetakan *request text* dari *user*, di antaranya adalah *intent* untuk menangani cek pesanan dan untuk menangani pemesanan produk. Berikut adalah kode untuk menangani *intent* dari *response* yang diberikan oleh API.AI dalam menangani cek pesanan yang ditunjukkan pada gambar 4.43.

```
if(intent==="welcom-message"){
            sendButtonMessage(senderID);
          }else   if(intent==='cek_order'   ||   intent==='cek_order   -   yes'   ||
intent==='cek_order - no'){
            if(intent==='cek_order'){
              if(responsed.result.parameters.number!='' && action==='cek'){
                var orderNumber=responsed.result.parameters.number;
                sendTextMessage(senderID,"Order    anda    sedang    dalam    proses
pengerjaan");
                web_services.cekOrder(orderNumber,function(data,status){
                  if(status){
                    sendTextMessage(senderID,"Pesananan    dengan    nomor    ini:
"+orderNumber+"\nsedang dalam proses : "+data.orderStatus);
                  }else{
                    //maaf kami tidak dapat mendapatkan data anda
                    sendTextMessage(senderID,"Maaf   kami   tidak   dapat   menemukan
informasi mengenai nomor order tersebut");
                  }
                })
```

```
                //cek order
              }else{
                sendTextMessage(senderID,aiText);
              }
            }else if(intent==='cek_order - yes'){
              if(responsed.result.parameters.numbers!='' ){
                var orderNumber=responsed.result.parameters.numbers;
                web_services.cekOrder(orderNumber,function(data,status){
                  if(status){
                    sendTextMessage(senderID,"Pesanan     dengan     nomor     ini:
"+orderNumber+"\nsedang dalam proses : "+data.orderStatus);
                  }else{
                    //maaf kami tidak dapat mendapatkan data anda
                    sendTextMessage(senderID,"Maaf   kami   tidak   dapat   menemukan
informasi mengenai nomor order tersebut");
                  }
                })
              }else           if(responsed.result.parameters.number!=''           &&
responsed.result.parameters.numbers==''){
                var orderNumber=responsed.result.parameters.number;
                web_services.cekOrder(orderNumber,function(data,status){
                  if(status){
                    sendTextMessage(senderID,"Pesanan     dengan     nomor     ini:
"+orderNumber+"\nsedang dalam proses : "+data.orderStatus);
                  }else{
                    //maaf kami tidak dapat mendapatkan data anda
                    sendTextMessage(senderID,"Maaf   kami   tidak   dapat   menemukan
informasi mengenai nomor order tersebut");
                  }
                })
              }else{
                sendTextMessage(senderID,aiText);
              }

            }else{
              sendTextMessage(senderID,aiText);
            }
          }
```

Gambar 4.43 Penanganan *Intent* cek *order*

Kode di atas merupakan penanganan yang dilakukan apabila *intent* yang terdapat pada objek di dalam *response* yang dikirim oleh API.AI adalah sama dengan ‘cek_order’. Selanjutnya tujuan dari cek *order* ini sendiri adalah untuk melakukan pemeriksaan status pesanan yang sudah dibuat oleh *user*. Untuk melakukan pengecekan *user* harus memiliki nomor pesanan untuk dilakukan pemeriksaan di *database*. Nomor pesanan sebelumnya telah didapat dari pengolahan *request text* oleh *user* di dalam *platform* API.AI. Untuk memperjelas *response* yang datang dari API.AI berupa data yang berformat json maka penulis menunjukkan sebuah *response* yang data dari layanan tersebut seperti pada Gambar 4.44.

Gambar 4.44 *Respon* JSON dari API.AI

Di dalam *chatbot* juga terdapat penangan percakapan pemesanan produk yang akan dilakukan oleh *user* atau pelanggan. Di dalam penangan percakapan ini terdapat beberapa *intent* yang digunakan di antaranya adalah *intent order_paper_bag_yes*. Kode untuk menangani apabila *intent* dari *response* yang dikirim oleh layanan API.AI adalah *order-paper_bag_yes* seperti yang ditunjukkan pada Gambar 4.45.

```
    else if(intent==='order_paper_bag_yes'){
            if(responsed.result.parameters.paper_type!=''                    &&
responsed.result.parameters.panjang!='' && responsed.result.parameters.lebar!='' &&
responsed.result.parameters.tinggi!='' && responsed.result.parameters.warna!='' ){
              //get data from server with function
              var data={"paper_type":responsed.result.parameters.paper_type,
                        "panjang":responsed.result.parameters.panjang,
                        "lebar":responsed.result.parameters.lebar,
                        "tinggi":responsed.result.parameters.tinggi,
                        "warna":responsed.result.parameters.warna};
              sendTextMessage(senderID,"Mungkin ini produk yang kamu cari");

              setTimeout(function(){
                sendTypingOn(senderID)
              },2000);
              web_services.getProdukData(data,function(datas){
                sendTypingOff(senderID);

                if(datas.status!="Error"){
                  var myTimer=setTimeout(function di(){
                     sendGenericMessage(senderID,datas)
```

```
                  },2000);
                }else{
                  sendTextMessage(senderID,"Maaf telah terjadi kesalahan di server
kami, silahkan coba beberapa saat lagi");
                }


              });
            }else if(responsed.result.parameters.paper_type==''){
              sendTextMessage(senderID,aiText);

                var myTimer = setTimeout(function di(){
                  sendQuickReplyPaperType(senderID)
                }, 2000);
            }else            if(responsed.result.parameters.warna==''            &&
responsed.result.parameters.panjang!='' && responsed.result.parameters.lebar!='' &&
responsed.result.parameters.tinggi!=''){
                sendTextMessage(senderID,aiText);
                var myTimer = setTimeout(function di(){
                  sendQuickReplyWarna(senderID)
                }, 2000);
            }else{
              sendTextMessage(senderID,aiText);
            }
          }
```

Gambar 4.45 Penaganan Pesanan Untuk Menampilkan Produk

Gambar 4.45 di atas merupakan penanganan untuk *intent order_paper_bag_yes* yang terjadi apabila *user* memberikan pertanyaan seperti “saya mau pesan paper bag”. Nantinya pertanyaan dari *user* akan diolah di dalam layanan API.AI dan layanan ini akan mengembalikan *response* yang sesuai. Apabila teridentifikasi bahwa pertanyaan yang diberikan oleh *user* adalah dikelompokkan pada *intents order_paper_bag_yes* maka kode diatas lah yang akan dijalankan. Pada bagian *intent order_paper_bag_yes* bertujuan untuk menampilkan produk yang sesuai dengan *request text* yang diberi *user*. Mulai dari jenis kertas, lebar paper bag dan informasi lainnya. Lalu data-data tadi akan dikirim ke *web service* untuk mencari data yang sesuai dengan yang di inginkan oleh *user*. Lalu di dalam kode tersebut akan dilakukan permintaan ke *web service* untuk mendapatkan informasi produk yang sesuai denga permintaan *user*. Setelah didapat maka data produk tersebut akan dikirimkan kembali ke *user* dengan menggunakan *generic template* ke *platform* Messenger.

4.8.4 Integrasi Chatbot dengan Web Service

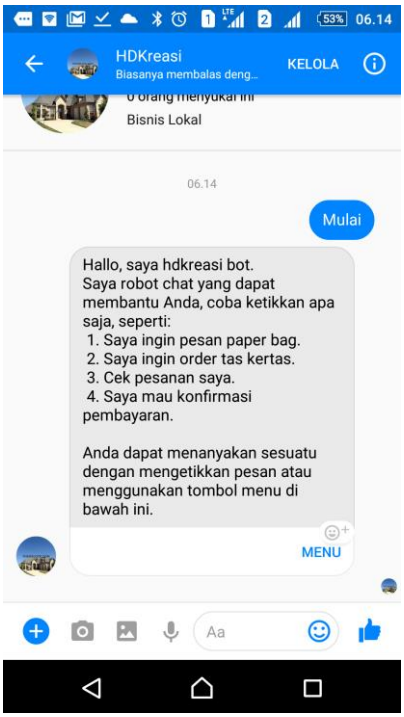
Untuk memperkaya informasi yang diberikan kepada *user* yang melakukan percakapan dengan *chatbot* seperti informasi pesanan dan produk. Maka diperlukan adanya integrasi dengan *web service* sistem pengelolaan pesanan. *Web service* ini merupakan sebuah antarmuka yang digunakan untuk mengelola informasi yang terdapat di dalam *database*. Penulis telah

membuat sebuah *web service* yang dapat diakses melalui url http://layangwacan.com. Kode-kode untuk mengakses fungsionalitas yang terdapat di dalam *web service* dari *webhook server* dirujuk pada lampiran laporan bagian "Lampiran kode untuk melakukan request ke *web service* melalui *webhook*".

Kode yang berada pada lampiran merupakan fungsi-fungsi untuk melakukan pengaksesan *web services*. Fungsi-fungsi tersebut dapat melakukan permintaan ke *web service* seperti permintaan informasi pesanan, informasi produk, dan menyimpan pesanan. Pada Gambar 4.43 merupakan sebuah contoh fungsi pengaksesan *web service* yang dipergunakan di dalam *chatbot* ini.

4.8.5 Pengujian Chatbot

Pada tahapan ini, penulis akan melakukan pengujian terhadap *chatbot* yang telah dibuat sebelumnya untuk memeriksa apakah keluaran atau *response* yang diberikan akan sesuai dengan yang telah ditentukan sebelumnya. Pengujian dilakukan ke dalam dua tugas. Di antaranya adalah kemampuan *chatbot* untuk menjawab pertanyaan dari *user* untuk melakukan pesanan produk dan melakukan pemeriksaaan pesanan. Pada saat *user* pertama kali berinteraksi dengan *chatbot*, *user* akan disambut dengan pesan pembuka di dalam aplikasi Messenger, seperti yang ditunjukkan pada Gambar 4.46.



Gambar 4.46 Pesan Pembuka

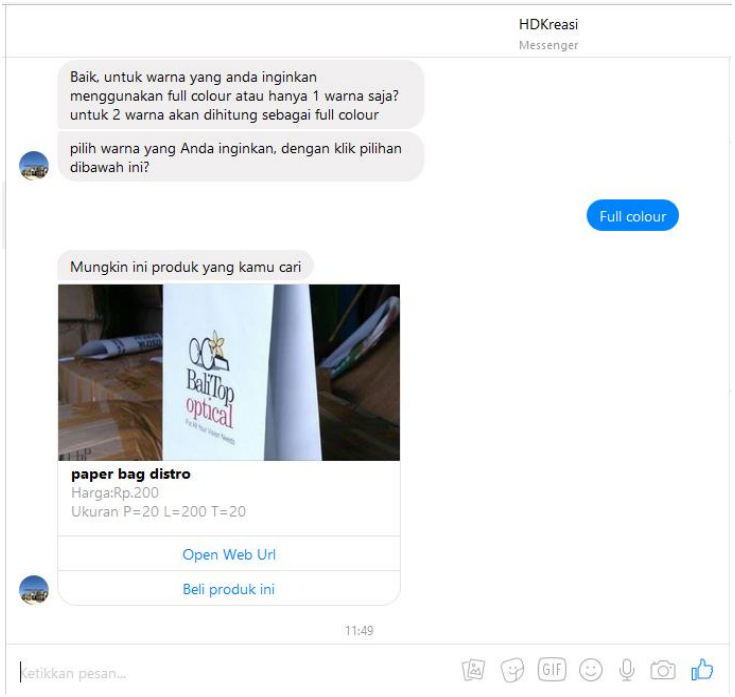
Implementasi Tugas Melakukan Pemesanan Produk

Penanganan percakapan untuk melakukan pemesanan terhadap produk adalah ketika *user* atau pelanggan bertanya hendak melakukan pemesanan produk dengan sebuah pertanyaan seperti "saya mau pesan paper bag". Pertanyaan seperti itu memiliki kata kunci "pesan" dan "paper bag". Kata kunci pesan adalah kata kunci yang dapat diinterpretasikan sebagai sebuah bentuk kata yang bermaksud untuk melakukan pemesanan, sedangkan kata kunci *paper bag* merupakan sebuah produk yang akan ditanyakan. Untuk melakukan pemesanan pertama-tama *user* akan bertanya dengan pertanyaan seperti yang sebelumnya telah disebutkan yaitu "saya mau pesan paper bag". Nantinya proses pemahaman kata tersebut akan dilakukan di layanan API.AI. Berikut ini merupakan contoh percakapan yang terjadi antara *user* dan *chatbot* untuk menangani pesanan yang ditunjukkan pada Gambar 4.47.

Gambar 4.47 Percakapan Penanganan Pesanan

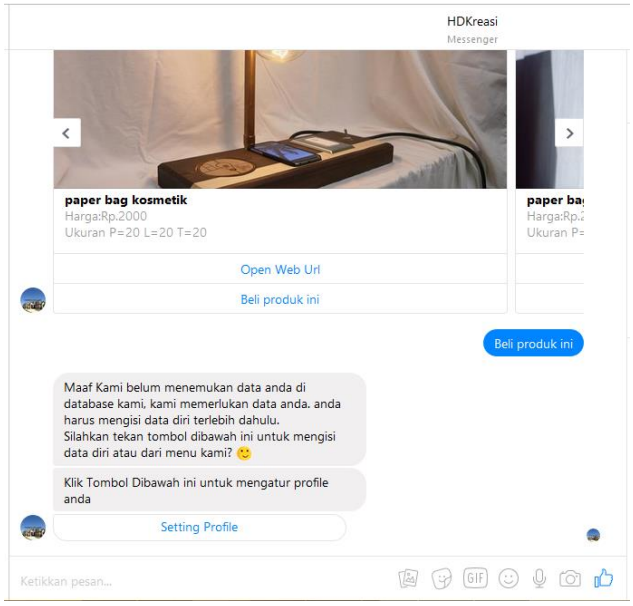
Pada percakapan yang terjadi, setelah *user* mengajukan pertanyaan untuk melakukan pesanan, jika memang pertanyaan tersebut teridentifikasi sebagai pertanyaan yang bertujuan untuk melakukan pesanan, maka *chatbot* akan memberi beberapa *response* pertanyaan kepada

user untuk mengumpulkan informasi mengenai produk seperti apa yang *user* kehendaki, seperti informasi mengenai jenis kertas dari *paper bag* yang akan dipesan, panjang area, lebar dan tinggi serta informasi lainnya. Setelah dikumpulkan mengenai informasi *paper bag* yang menjadi keinginan *user*, langkah selanjutnya adalah *chatbot* akan melakukan query ke *database* melalui *web services* untuk mencari data produk yang dikehendaki. Berikut ini merupakan tampilan produk di dalam *platform* Messenger yang akan diberikan oleh *chatbot* kepada *user* yang ditunjukkan pada Gambar 4.48.



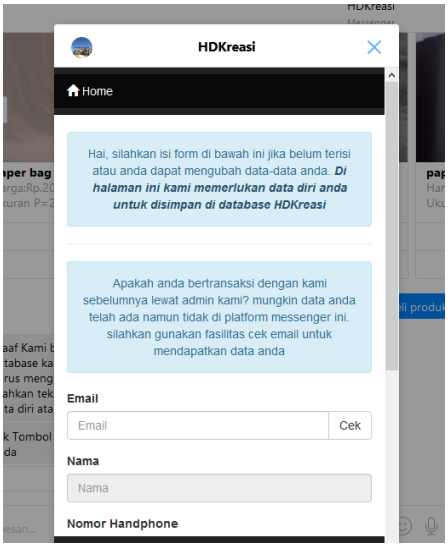
Gambar 4.48 Tampilan Daftar produk

Gambar 4.48 di atas merupakan tampilan daftar produk yang akan ditampilkan oleh *bot* Messenger. Selanjutnya *user* dapat memilih produk sesuai yang diinginkan dengan menekan tombol beli produk ini pada tampilan daftar produk. Setelah itu, maka *chatbot* akan memproses pesanan tersebut, namun terlebih dahulu akan dilakukan pengecekan terhadap *user* yang berinteraksi dengan *chatbot* apakah sudah terdaftar di dalam *database* atau belum. Jika *user* belum terdaftar maka *chatbot* akan merespon seperti yang terlihat pda Gambar 4.49.



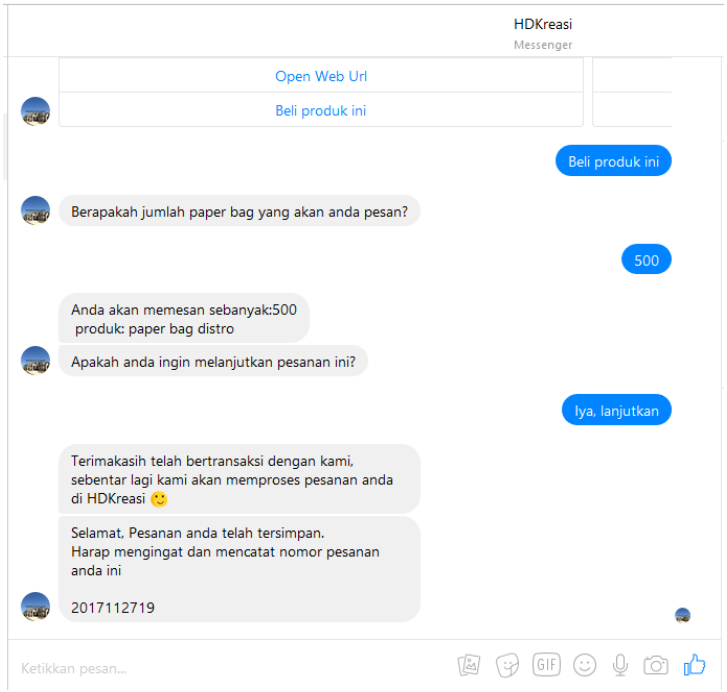
Gambar 4.49 Respon *Chatbot* saat *user* belum terdaftar

Pada saat belum terdaftar sebagai *member* di dalam *database* sistem pengelolaan pesanan, maka *chatbot* akan memberikan jawaban seperti yang terlihat pada Gambar 4.49 di atas. *User* akan dihimbau untuk melakukan pendaftaran data diri dengan menekan tombol *Setting Profil*. Lalu setelah tombol ditekan maka *user* akan diberikan *form* yang harus diisi dalam keperluan pendaftaran *member* baru. Berikut adalah *form* pendaftaran yang ditunjukkan pada Gambar 4.50.



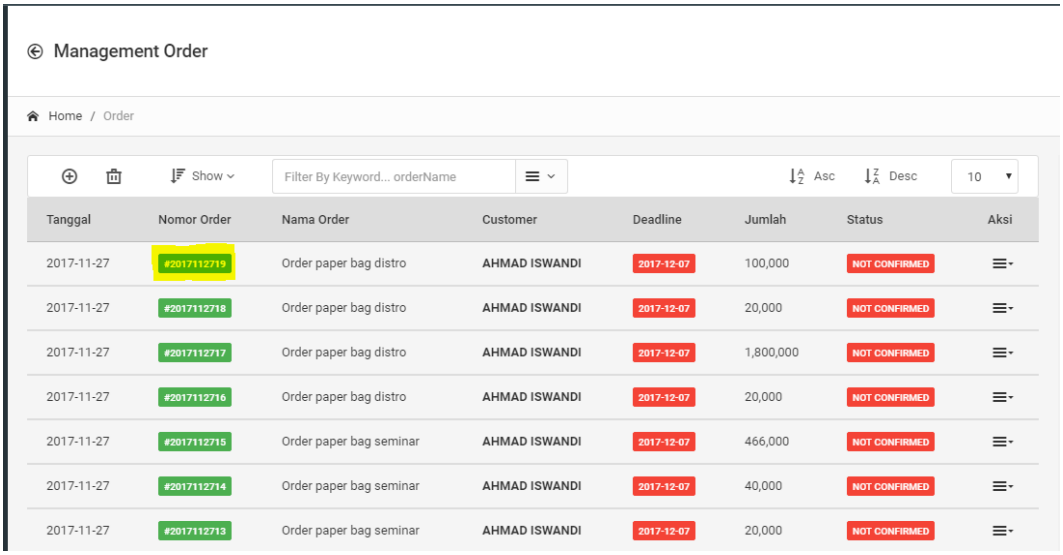
Gambar 4.50 Form Pendaftaran *Member* baru

Namun apabila pelanggan sudah terdaftar sebagai *member* maka proses pemesanan produk selanjutnya adalah *chatbot* akan menanyakan berapa jumlah produk yang akan di pesan oleh pelanggan. Gambar 4.51 merupakan proses pemesanan produk untuk menanyakan jumlah pesanan.



Gambar 4.51 Proses Pemesanan Produk Menanyakan Jumlah Pesanan

Gambar 4.51 di atas adalah proses pemesanan produk yang terakhir yaitu menanyakan jumlah yang akan dipesan kepada *user*. Selanjutnya *user* akan diberi pertanyaan oleh *chatbot* apakah pesanan yang baru saja dibuat akan dilanjutkan. Jika *user* menjawab iya, maka pesanan akan disimpan di *database* dan *chatbot* akan memberikan informasi mengenai nomor pesanan. Untuk melihat apakah pesanan berhasil disimpan di dalam *database*, pegawai HDKreasi dapat melihat di halaman sistem pengelolaan pesanan pada menu *chatbot order* yang ditunjukkan pada Gambar 4.52.



Gambar 4.52 Pemeriksaan Pesanan yang Datang dari *Chatbot*

Penulis juga melakukan pengujian fungsionalitas terhadap tugas percakapan untuk menangani pemeriksaan pesanan. Di bawah ini merupakan tabel hasi pengujian fungsionalitas pada tugas untuk melakukan pemeriksaan pesanan yang ditunjukkan pada Tabel 4.16.

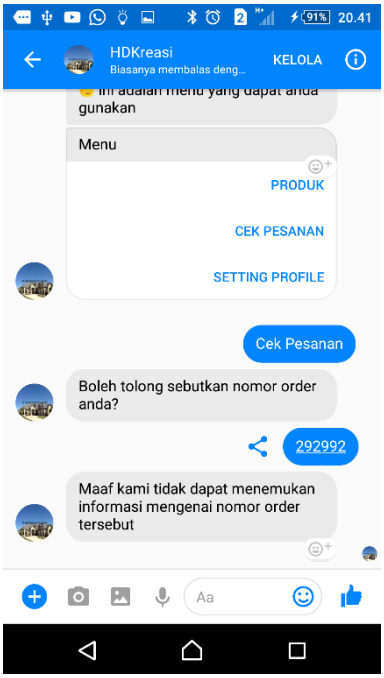
Tabel 4.16 Pengujian Pemesanan Produk pada *chatbot*

Skenario pengujian	**input**	**Output yang diharapkan**	**Output hasil**
Melakukan pemesanan produk melalui *chatbot* saat *user* belum terdaftar	*User* yang belum terdaftar tetapi langsung melakukan pemesanan	*User* akan diarahkan untuk melakukan pendaftaran	Valid (*chatbot* memberikan arahan untuk *user* baru mendaftar ke sistem)
Melakukan pemesanan (sukses)	Melakukan pemesanan produk	*Chatbot* akan merespon dengan memberikan informasi bahwa pesanan berhasil dilakukan dan mengeluarkan nomor pesanan	*Valid (chatbot* merespon)

Implementasi Tugas Pemeriksaan Pesanan

Pada tahapan ini penulis akan melakukan pengujian terhadap tugas pemeriksaan pesanan. Tugas atau fungsionalitas ini digunakan untuk menangani percakapan ketika *user* ingin menanyakan informasi pesanan. Terdapat dua cara yang dapat dilakukan *user* untuk

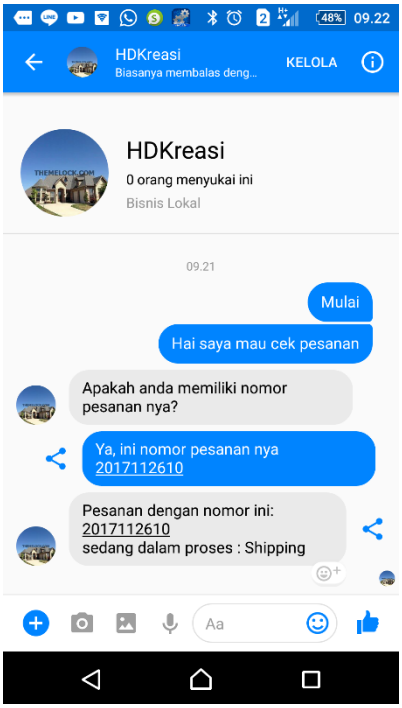
menanyakan informasi pesanan yaitu dengan menggunakan antarmuka pada tombol menu atau langsung mengetikkan pertanyaan secara langsung. Berikut ini merupakan pemeriksaan pesanan yang dilakukan melalui antarmuka atau menggunakan tombol menu yang ditunjukkan pada Gambar 4.53.



Gambar 4.53 Cek Pesanan Menggunakan Tombol *Menu*

Gambar 4.53 di atas merupakan cara yang dapat digunakan untuk berinteraksi dengan chatbot untuk memeriksa pesanan dengan menggunakan tombol menu. Setelah tombol menu ditekan lalu *chatbot* akan menanyakan nomor pesanan milik *user*. Apabila nomor pesanan terdapat di *database* maka *chatbot* akan memberikan informasi mengenai pesanan tersebut. Jika nomor pesanan tersebut tidak tersedia maka *chatbot* akan memberikan *response* seperti pada Gambar 4.53 di atas.

Untuk melakukan pengecekan terhadap pesanan, *user* juga dapat menanyakan dan mengetikkan secara langsung pertanyaan melalui papan ketik pada antarmuka *platform* Messenger Jika mengikuti *conversational flow* yang telah dirancang sebelumnya maka akan didapati hasil seperti pada Gambar 4.54.



Gambar 4.54 Percakapan Untuk Menanyakan Informasi Pesanan

User pertama-tama akan menanyakan kepada *chatbot* dengan menggunakan pertanyaan seperti ini "Hai saya mau cek pesanan". Setelah itu *chatbot* akan memberikan *response* dengan menanyakan kembali pertanyaan seperti ini "Apakah Anda memiliki nomor pesanannya?". Lalu *user* akan menjawab dengan memasukkan nomor pesanannya. Jika nomor pesanan ditemukan di dalam *database* maka *chatbot* akan membalas pesan tersebut dengan memberikan informasi status mengenai proses pesanan tersebut.

Untuk memperluas pengujian terhadap *chatbot* dan mengetahui pemahaman dari *chatbot* akan pertanyaan *user*, di sini penulis mencoba untuk bertanya kepada *chatbot* dengan beberapa pertanyaan yang berkaitan dengan tugas melakukan pemeriksaan pesanan. Tentunya dengan menggunakan kata kunci "cek order" dan berbagai macam sinonim yang berkaitan dengan kata tersebut. Tidak hanya itu, di sini penulis juga akan memberikan pertanyaan *typo* atau pertanyaan yang tidak benar penulisan katanya. Berikut ini adalah beberapa pertanyaan yang kemungkinan akan ditanyakan oleh *user* dan *response* yang diberikan oleh *chatbot* yang ditunjukkan pada Tabel 4.17.

Tabel 4.17 Pengujian Pertanyaan Untuk Pemeriksaan Pesanan

Pertanyaan *user*	**Jawaban *chatbot***
Hai, saya ingin cek order saya dong	*Chatbot* merespon dengan kembali menanyakan nomor *order* atau nomor pesanan milik pelanggan. (chatbot memahami pertanyaan)
Saya nak cek ordr dong (pertanyaan yang terdapat kata typo)	*Chatbot* merespon dengan kembali menanyakan nomor *order* atau nomor pesanan milik pelanggan. (chatbot memahami pertanyaan)
Saya mau cek pesanan saya dengan nomor order 2017112610 (pertanyaan dengan menyisipkan nomor pesanan dalam satu waktu).	*Chatbot* merespon dengan memberikan informasi mengenai status pesanan. (chatbot memahami pertanyaan)
Saya mau cek pesnan dong	*Chatbot* merespon dengan kembali menanyakan nomor *order* atau nomor pesanan milik pelanggan. (chatbot memahami pertanyaan)
Saya mau ck psn lah	*Chatbot* tidak memahami pertanyaan tersebut

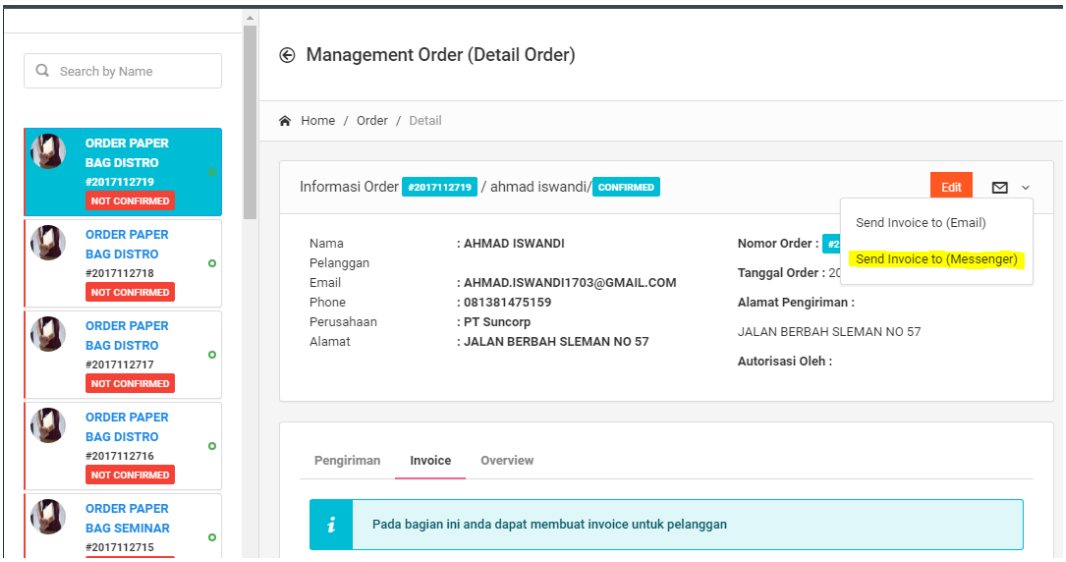
Penulis juga melakukan pengujian fungsionalitas terhadap tugas percakapan untuk menangani pemeriksaan pesanan. Di bawah ini merupakan tabel hasil pengujian fungsionalitas pada tugas untuk melakukan pemeriksaan pesanan yang ditunjukkan pada Tabel 4.18.

Tabel 4.18 Pengujian pengecekan pesanan pada *chatbot*

Skenario pengujian	**Input**	**Output yang diharapkan**	**Output hasil**
Pengecekan pesanan dengan nomor pesanan yang belum ada	Nomor pesanan belum ada	*Chatbot* memberikan respon dengan memberi tahu bahwa nomor pesanan tersebut tidak atau belum ada	*Valid (chatbot* merespon)
Pengecekan pesanan dengan nomor pesanan yang benar dan sudah ada	Nomor pesanan sudah ada	*Chatbot* akan merespon dengan memberikan informasi mengenai status pesanan dari nomor *order* yang dikirimkan	*Valid (chatbot* merespon dengan mengirimkan informasi status pesanan

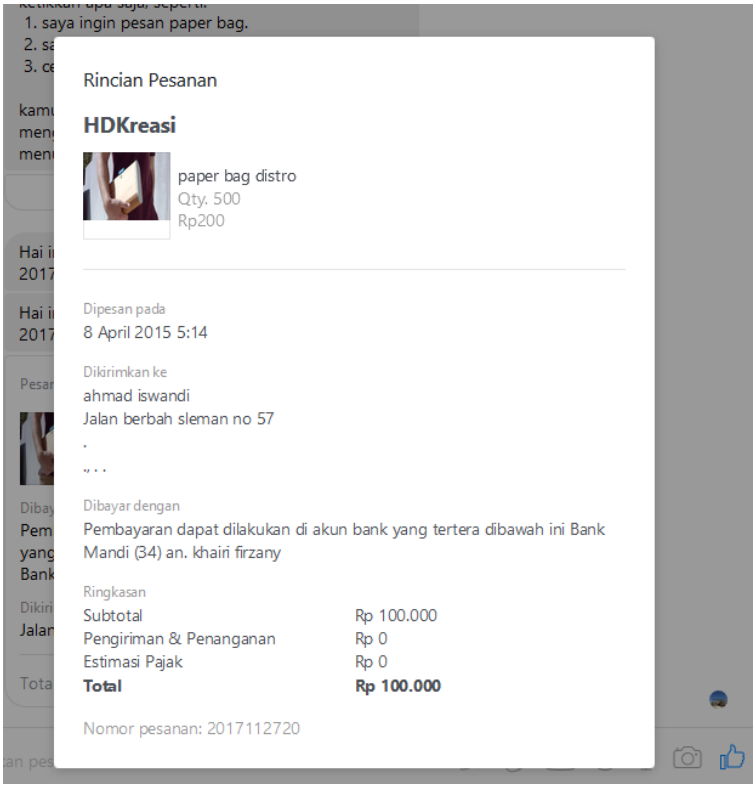
Implementasi Pengiriman Invoice

Pengiriman *invoice* dilakukan oleh pegawai, yang dilakukan secara manual di dalam halaman aplikasi pengelolaan pesanan. Pengiriman *invoice* dikirimkan ke pelanggan yang melakukan pesanan melalui *chatbot*. *Invoice* akan dikirim sesuai dengan identitas atau *messengerID* pemilik dari *member* yang melakukan pesanan. Berikut adalah halaman tempat pegawai akan mengirim *invoice* yang ditunjukkan pada Gambar 4.55.



Gambar 4.55 Pengiriman *Invoice* ke Akun Messenger *Member*

Pegawai akan memilih melalui apa *invoice* akan dikirim, dalam kasus ini *invoice* akan dikirimkan ke akun Messenger *member*. Selanjutnya apabila berhasil dikirim maka pelanggan akan menerima *invoice* di akun Messenger yang pelanggan miliki. Berikut adalah *invoice* yang berhasil dikirim ke akun Messenger milik pelanggan yang ditunjukkan pada Gambar 4.56.



Gambar 4.56 Pengiriman *Invoice* ke Akun Messenger Pelanggan

Pengujian *chatbot* di dalam bab ini merujuk kepada rancangan utama tugas penanganan percakapan yang telah dirancang sebelumnya yaitu tugas penanganan percakapan untuk melakukan pemesanan produk dan tugas percakapan untuk melakukan pemeriksaan pesanan di dalam sistem. Selanjutnya, penulis melakukan pengujian secara langsung terhadap 5 calon pelanggan mengenai kemampuan yang dimiliki oleh *chatbot* untuk menjawab pertanyaan yang diajukan oleh pelanggan beserta fitur-fitur yang terdapat di dalam *chatbot.* Berikut merupakan hasil pengujian *chatbot* kepada calon pelanggan yang ditunjukkan pada Tabel 4.19.

Tabel 4.19 Pengujian *Chatbot* Pada *User*

Pertanyaan	Jawaban
Apakah response awal *chatbot* saat anda berinteraksi memberikan penjelasan dengan baik mengenai tugas apa saja yang dapat dilakukan *chatbot*?	Dari beberapa calon pelanggan yang melakukan uji coba berinteraksi dengan *chatbot*, semuanya menjawab bahwa pada saat pertama kali berinteraksi, *chatbot* memberikan pesan perkenalan yang berisi mengenai tugas yang dapat *chatbot* lakukan.
Apakah Anda dapat dengan mudah menanyakan informasi produk?	2 dari 5 calon pelanggan menyatakan bahwa dapat dengan mudah menanyakan informasi mengenai produk, dikarenakan pelanggan ini sudah mengetahui

	spesifikasi produk yang akan dicari. Sedangkan pelanggan lainnya menyatakan agak kebingungan dikarenakan pelanggan ini belum mengetahui secara detail produk seperti apa yang ingin dicari. Sehingga pelanggan yang kebingungan ini mengharapkan rekomendasi produk dari *chatbot.*
Apakah *chatbot* memberikan jawaban sesuai dengan yang Anda tanyakan?	Calon pelanggan mengatakan untuk tugas yang dapat di tangani *chatbot* ini jawabannya sudah sesuai dengan pertanyaan yang diajukan.
Apakah tombol navigasi yang ditampilkan membantu Anda dalam berinteraksi dengan *chatbot?*	Semua calon pelanggan setuju dengan tombol navigasi yang disajikan *chatbot* memudahkan dalam interaksi percakapan.
Apakah chatbot memberikan jawaban saat pertanyaan yang Anda ajukan di luar konteks pemahaman *chatbot?*	Calon pelanggan diarahkan pada tugas yang dapat ditangani *chatbot* ketika pertanyaan yang diajukan oleh pelanggan di luar konteks tugas yang dapat ditangani oleh *chatbot.*
Apakah Anda nyaman dalam berinteraksi dengan *chatbot* daripada berinteraksi melalui *customer service?*	Calon pelanggan menyatakan bahwa mereka cukup nyaman dalam berinteraksi dengan *chatbot*.
Apakah ke depannya Anda akan berkomunikasi dengan *chatbot*?	Calon pelanggan mengatakan bahwa ke depannya komunikasi melalui *chatbot* dapat dipertimbangkan

4.9 Pengujian Usabilitas

Pada tahapan pengujian usabiltas *user* ini, penulis akan melakukan pengujian terhadap pegawai yang akan menggunakan sistem pengelolaan pesanan. Pegawai yang akan menggunakan sistem ini merupakan pegawai yang nantinya akan menjadi *admin* atau orang yang akan bertanggung jawab untuk menggunakan sistem.

Pada tahap pengujian usabilitas ini, penulis mengajukan beberapa pertanyaan kepada *admin* untuk mengetahui seberapa baik sistem yang telah dibuat oleh penulis, disini penulis memberikan pertanyaan yang memiliki nilai skala yaitu dalam rentang nilai 1-5 yang diwakilkan dengan kata “sangat tidak setuju” untuk angka 1, tidak setuju untuk angka 2, normal untuk angka 3, setuju untuk angka 4 dan sangat setuju untuk angka 5 untuk setiap pertanyaan. Dari beberapa pertanyaan, penulis memberikan beberapa indikator penilaian. Berikut adalah penjelasan mengenai beberapa indikator yang penulis gunakan pada Tabel 4.20.

Tabel 4.20 Penjelasan Indikator Penilaian

Indikator	Penjelasan
Efe	Efektifitas
Efi	Efisiensi
Sat	Kepuasan dan Kenyamanan
Lea	Learnability
Fle	Fleksibilitas
Rob	Robustness
EoL	Mudah dipelajari
Rec	Mudah diingat
Pro	Produktivitas optimal
MeR	Minim kesalahan
HUS	Kepuasan dan kenyamanan pengguna

Tabel di atas merupakan penjelasan mengenai indikator yang penulis gunakan untuk melakukan penilaian terhadap sistem. Setelah didapat indikator apa saja yang ingin digunakan langkah selanjutnya adalah melakukan wawancara. Berikut adalah hasil pengisian wawancara yang dilakukan dan ditunjukkan pada Tabel 4.21.

Tabel 4.21 Hasil Pengujian Melalui Wawancara

Indikator	Pertanyaan	Tanggapan
Efe	Sistem memudahkan pegawai untuk mengelola data di *database* ?	Pegawai memberi nilai setuju. Sistem pengelolaan pesanan memudahkan pegawai untuk mengelola data di *database*
Sat	Saya merasa puas dan nyaman saat melakukan proses penggunaan sistem pengelolaan pesanan	Pada sistem yang dibuat ini pegawai memilih nilai normal , hal ini menunjukan masih terdapat beberapa komponen atau fungsionalitas yang masih kurang baik
Lea	Saya dapat dengan cepat dan mudah memahami bagaimana cara mengoperasikan sistem tersebut	Pegawai memilih nilai normal perihal *learnability*. Sistem yang dibuat telah memiliki kemudahan untuk dapat dipelajari namun masih terdapat beberapa hal yang mungkin masih sulit dipahami dan digunakan
Rec	Saya beranggapan bahwa beberapa fitur memiliki alur yang hampir sama sehingga setiap	Nilai setuju adalah yang diberikan pegawai untuk hal ini, ini menandakan bahwasannya sistem sudah memiliki alur yang hampir sama pada tiap-tiap modul.

	fitur mudah untuk dioperasikan	
Pro	Saya beranggapan bahwa sistem sudah sangat memudahkan pegawai untuk melakukan pekerjaannya nanti	Pegawai menanggapi dengan setuju bahwa sistem dapat memudahkan pekerjaan nya nanti dalam hal pendokumentasian dan pemrosesan pesanan.
MeR	Saya beranggapan bahwa sistem sudah tidak memiliki kesalahan yang begitu banyak	Pegawai setuju dan mengatakan sudah tidak ada kendala atau kesalahan yang fatal di dalam sistem
HUS	Pada waktu ke depan, sistem ini dapat digunakan oleh pegawai	Pegawai sudah sangat setuju apabila nantinya sistem ini yang akan digunakan untuk mengelola data didalam basis data.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang dilakukan terhadap implementasi *chatbot* pada *order management system* beserta pembahasannya yang telah dilakukan, maka dapat disimpulkan bahwa:

a. Telah berhasil dibuat sistem pengelolaan pesanan untuk UMKM HDKreasi sesuai dengan kebutuhan usaha ini. Sistem pengelolaan pesanan ini dapat digunakan untuk melakukan pengelolaan pesanan yang datang dari pelanggan.
b. *Chatbot* yang dibuat untuk pelanggan dapat berkomunikasi dengan menggunakan *platform* Messenger dari Facebook telah mampu menangani beberapa tugas percakapan yang sudah penulis rancang yaitu penanganan percakapan terhadap pelanggan yang akan melakukan pemesanan produk dan juga penanganan percakapan terhadap pelanggan yang akan memeriksa pesanan yang telah dibuat.
c. *Chatbot* telah terintegrasi dengan sistem pengelolaan pesanan, di mana *chatbot* dapat mengakuisisi informasi dari sistem ini untuk diberikan kepada pelanggan. Seperti informasi mengenai produk dan juga informasi status pesanan yang dibuat oleh pelanggan.

5.2 Saran

Berdasarkan implementasi dan pengujian yang telah dilakukan, aplikasi ini masih memiliki beberapa kekurangan yang dapat diperbaiki untuk pengembangan ke depan sehingga dapat meningkatkan kinerja aplikasi menjadi lebih maksimal dalam penggunaannya. Untuk pengembangan penelitian sistem ini lebih lanjut, ada beberapa saran yang mungkin dapat diperhatikan.

a. Sistem pengelolaan pesanan dapat dikembangkan menjadi sebuah plugin yang mampu mengelola pesanan yang datang dari *e-commerce.* Sehingga sistem akan bersifat *multichannel* dalam menerima pesanan.
b. *Chatbot* yang telah dibuat hanya dapat menangani beberapa tugas percakapan di antaranya adalah penanganan percakapan untuk melakukan pemeriksaan pesanan dan pemesanan produk. Untuk kedepannya dapat dirancang *chatbot* yang memiliki kemampuan untuk

menjawab pertanyaan terkait dengan segala aktifitas usaha. Sehingga peran *customer service* di dalam usaha ini dapat digantikan dengan *chatbot.*

c. *Chatbot* dapat dikembangkan dengan menggunakan *platform* pesan singkat populer lainnya seperti line, whatsapp, telegram, dan lainnya sehingga mampu menjangkau pelanggan yang menggunakan berbagai macam *platform* pesang singkat lainnya.
d. Keamanan transaksi melalui *chatbot* belum dikembangkan secara maksimal, sehingga kedepannya dapat dibuat mekanisme yang baik untuk menunjang keamanan transaksi di dalam *chatbot.*

DAFTAR PUSTAKA

Adipura, Y., Wicaksono, R. W., & Wiyogo, M. (2015). Perancangan Order Management System Berbasis Web Application Pada Usaha Mikro (Studi Kasus Sugoimasa). *Jurnal Tugas Akhir, Fakultas Rekayasa Industri*, *2*(1), 1077–1082.

Bora, P. R., & Gupta, E. (2012). Application On Order Management System In Restaurants. *International Journal of Application or Innovation in Enggineering & Management*, *1*(2), 59–62.

Chinmay, P., & Himil, G. (2012). *Online Sales Order Management System Online Sales Order Management System For Riwasa Tiles*. Diambil kembali dari http://gnu.inflibnet.ac.in:8080/jspui/bitstream/123456789/2617/46/Online Sales Order Management System [Compatibility Mode].pdf

Guzman, Ines. (2016). Accenture-Chatbots-Customer-Service. Diambil kembali dari https://www.accenture.com/t00010101T000000__w__/br-pt/_acnmedia/PDF-45/Accenture-Chatbots-Customer-Service.pdf

Komawar, O., Thakar, P., Shetty, R., Bartakke, A., & Desai, P. M. (2015). An Internet Relay Chat Bot using AIML. *International Journal of Science and Research*, *4*(10), 2014–2016.

Li Q., C. Y. (2009). 6 Entity-Relationship Diagram. In *Entity-Relationship Diagram. In: Modeling and Analysis of Enterprise and Information Systems* (pp. 1–2). Diambil kembali dari https://link.springer.com/chapter/10.1007/978-3-540-89556-5_6

Oupraxay, A., & Diego, S. (2010, June), *Android Based Mobile Order Management System* Paper presented at 2010 Annual Conference & Exposition, Louisville, Kentucky. https://peer.asee.org/15822

Shetty, M., Shareef, W. J., Shetty, K., & Lohiya, S. (2015). B2B Order Management System. *International Journal of Computer Science and Information Technologies*, *6*(2), 1118–1122.

LAMPIRAN

a. Lampiran kode untuk melakukan *request* ke *web service* melalui *webhook*

```
var auth = "Basic " + new Buffer('admin' + ":" + '1234').toString("base64");
const config = require('config');
const request= require('request');

exports.getInvoice=function(orderID,fn){

var options = {

url: config.get('apiUrl')+'/orders/invoice_services?orderID='+orderID,

headers: {

"Authorization" : auth,

"content-type":"application/json"

}

// form:parameter

};
    request.get(options, function(error,response,body){
      if (!error && response.statusCode == 200) {
        var info = JSON.parse(body);
        // console.log("coba",info);

        return fn(info);
      }else{
        return fn(null);
      }
    });
}
//fungsi untuk mendapatkan data sesuai dengan parameter yang di inputkan oleh user
exports.getProdukData=function(parameter,fn){

    var options = {
      url: config.get('apiUrl')+'/products/produk_services',
      headers: {
        "Authorization" : auth,
        "content-type":"application/json"
      },
      form:parameter
    };
    request.post(options, function(error,response,body){
      if (!error && response.statusCode == 200) {
        var info = JSON.parse(body);
        return fn(info);
      }else{
          var dummy={"status":"Error","data":[]}
        return fn(dummy);
      }
    });
}
//untuk mendapatkan detail produk sesuai denga parameter yag di inputkan
exports.getProdukDetail=function(parameter,fn){
    var options = {
      url: config.get('apiUrl')+'/products/produk_detail?productID='+parameter,
      headers: {
        "Authorization" : auth,
        "content-type":"application/json"
```

```
      }
    };
    console.log(options.url);

    request.get(options, function(error,response,body){
      console.log(body);
      // console.log(response);
      if (!error && response.statusCode == 200) {
        var info = JSON.parse(body);
        // console.log("coba",info);

         return fn(info);

      }else{
           var dummy={"status":"Error","data":[]};
        return fn(dummy);
      }
    });
}

exports.get_order=function(orderID){
    var options = {
      url: config.get('apiUrl')+'/customers/allCust',
      headers: {
        "Authorization" : auth
      }
    };
    request(options, function(error,response,body){
      if (!error && response.statusCode == 200) {
        var info = JSON.parse(body);
        console.log("coba",info['data'][0].customerName);
        for(var key in info['data']){
          for (var en in info['data'][key]){
            console.log(info['data'][key][en]);
          }
        }

      }
    });

}
exports.sendData=function(collectData,fn){
    var auth = "Basic " + new Buffer('admin' + ":" + '1234').toString("base64");
    var options = {
      url: config.get('apiUrl')+'/orders/order_services',
      headers: {
        "Authorization" : auth
      },
      form:collectData
    };
    request.post(options, function(error,response,body){
      console.log(response);

      if (!error && response.statusCode == 200) {
        var info = JSON.parse(body);
        // for(var key in info['data']){
        //   for (var en in info['data'][key]){
        //     console.log(info['data'][key][en]);
        //   }

        // }

        console.log("-------------============",info);
        return fn(true,info);

      }else{
```

```
        var info={"status":"Error","data":{}}
        return fn(false,info);
      }
    });
}
exports.cekCustomer=function(messengerID, fn){
    var parameter={'messengerID':messengerID};
    var options = {
      url:
config.get('apiUrl')+'/customers/customer_services?messengerID='+messengerID,
      headers: {
        "Authorization" : auth,
        "content-type":"application/json"
      }
    };



    request.get(options, function(error,response,body){

      if (!error && response.statusCode == 200) {

        var info = JSON.parse(body);
        if(info.status==="Success"){
          return fn(info.data,true);
        }


      }else{
         return fn(null,false);
      }
    });
    // //dummtt
    // return fn(null,true);
}
exports.cekOrder=function(orderNumber,fn){
    var options = {
      url: config.get('apiUrl')+'/orders/cek_order?orderNumber='+orderNumber,
      headers: {
        "Authorization" : auth
      }
    };
    request(options, function(error,response,body){
      if (!error && response.statusCode == 200) {
        var info = JSON.parse(body);

        if(info.status==="Success"){


return fn(info.data,true);
        }else{


return fn(null,false);
        }
      }else{


return fn(null,false);
      }
    });
}
```