

**ANALISIS DAN PERANCANGAN *BUSINESS INTELLIGENCE*
BERBASIS *WEBAPP* MENGGUNAKAN *NOSQL DATABASE* DENGAN
METODE *KIMBALL LIFECYCLE*
(Studi kasus: *Stock Management* pada PD Intan)**

SKRIPSI

diajukan untuk menempuh ujian sarjana
pada Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Padjadjaran

**Muhamad Yusrizan
140810150041**



**UNIVERSITAS PADJADJARAN
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI S-1 TEKNIK INFORMATIKA
JATINANGOR
2019**

ABSTRAK

PD. Intan merupakan salah satu perusahaan yang bergerak dalam bidang toko kelontong jenis makanan dan minuman ringan. PD. Intan sebagai perusahaan dagang makanan ringan secara grosir yang menjual barang dari produsen atau distributor kepada konsumen dengan satuan karton maupun pak terletak di Bandung, Jawa Barat. Permasalahan yang dihadapi oleh perusahaan adalah proses penentuan pengadaan barang yang dilakukan berdasarkan perkiraan, sehingga mengakibatkan kerugian pada perusahaan. Berdasarkan masalah yang terjadi di PD. Intan perlu adanya pembangunan sistem yang melakukan analisis terhadap data transaksional yang kemudian menampilkan informasi yang telah dianalisis sehingga dapat membantu pemilik perusahaan dalam melakukan penentuan pengadaan barang. Pembangunan *business intelligence* merupakan salah satu langkah tersebut. *Business intelligence* adalah sekumpulan sistem, alat, dan proses yang berfungsi untuk mengumpulkan, mencari, dan mengolah data menjadi informasi bisnis yang dapat dijadikan suatu acuan strategi bisnis. Informasi tersebut kemudian ditampilkan kepada pengguna sehingga pengguna dapat mempertimbangkan dan melakukan strategi yang telah disajikan. Penelitian ini menggunakan metode *Kimball BI lifecycle* yang terdiri dari beberapa tahap diantaranya *project planning, business requirement definition, BI track, technology track, data track, deployment, maintenance* dan *project management*. Setelah pembangunan dilakukan lalu dilakukan pengujian terhadap sistem yang dibuat, dan dapat diambil kesimpulan bahwa *business intelligence* memudahkan pemilik dalam memantau transaksi dan persediaan yang ada di perusahaan. Selain itu juga dapat membantu dalam pembuatan keputusan untuk melakukan proses pengadaan persediaan barang.

Kata Kunci: Analisis data, *Business intelligence*, *Kimball lifecycle*, strategi bisnis

ABSTRACT

PD. Intan is one of the companies engaged in the type of food and soft drink grocery store. PD. Intan as a wholesale snack trading company that sells goods from producers or distributors to consumers in cardboard and pack units located in Bandung, West Java. The problem faced by the company is the process of determining the procurement of goods carried out based on estimates, resulting in losses to the company. Based on the problems that occur in PD. Intan needs a system development that analyzes the transactional data which then displays the information that has been analyzed so that it can help the company owner in determining the procurement of goods. The development of business intelligence is one such step. Business intelligence is a set of systems, tools, and processes that function to collect, search, and process data into business information that can be used as a reference for business strategy. The information is then displayed to the user so that the user can consider and carry out the strategies presented. This study uses the Kimball BI lifecycle method which consists of several stages including project planning, business requirement definition, BI track, technology track, track data, deployment, maintenance and project management. After the construction is carried out then a system test is made, and it can be concluded that business intelligence makes it easier for owners to monitor transactions and inventory in the company. In addition, it can also help in making decisions to carry out the inventory process.

Keywords: *Business Intelligence, Kimball lifecycle, Data analysis, Business Strategy*

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan kepada Allah SWT yang senantiasa memberikan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul “*Analisis dan Perancangan Business Intelligence Berbasis Webapp Menggunakan NoSQL Database dengan Metode Kimball Lifecycle*”, sebagai salah satu syarat dalam menempuh sarjana pada Program Studi S-1 Teknik Informatika Departemen Ilmu Komputer pada Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran.

Dalam hal ini penulis mengucapkan banyak terima kasih kepada Bapak Drs. Ino Suryana, M.Kom. selaku Dosen Wali serta Dosen Pembimbing Utama yang telah meluangkan banyak waktu untuk membimbing dan mendidik penulis selama penyusunan tugas akhir ini serta Bapak Aditya Pradana, S.T., M.Eng. selaku Dosen Pembimbing Pendamping yang telah meluangkan banyak waktu untuk membimbing dan mendidik penulis tidak hanya selama penyusunan skripsi ini.

Penulis menyadari akan keterbatasan yang dimiliki, begitu banyak bimbingan, bantuan, saran, kritik, serta motivasi yang diberikan oleh berbagai pihak dalam proses penyusunan tugas akhir ini. Oleh karena itu, ucapan terima kasih penulis juga sampaikan kepada:

1. Kedua orang tua penulis, Ayahanda Muhamad Ropandi dan Ibunda Intan Risana yang tidak henti-hentinya memberikan bantuan moral serta materiil, juga selalu mendoakan, mendidik dan memberi arahan kepada anaknya dalam menyelesaikan tugas akhir ini.

2. Prof. Dr. H. Sudradjat, MS selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran.
3. Dr. Setiawan Hadi, M.Sc.CS selaku Kepala Departemen Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran.
4. Dr. Juli Rejito, M.Kom. selaku Ketua Program Studi S-1 Teknik Informatika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran.
5. Seluruh staff pengajar dan tata usaha Departemen Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam yang sudah membantu proses perjalanan Pendidikan selama ini.
6. Dimas Azka, Ziyen Marzuq dan Bintang Rizq, adik penulis yang juga selalu memberikan semangat dan motivasi serta doanya untuk penulis selama pengerjaan tugas akhir ini.
7. Sahabat-sahabat seperjuangan Mahasiswa Teknik Informatika angkatan 2015, dan juga Keluarga Besar Himatif FMIPA Unpad yang tidak bisa disebutkan semuanya disini yang telah memberikan semangat, inspirasi, dan bantuan moral selama penulis mengerjakan tugas akhir ini.

Jatinangor, 16 Juni 2019

Penulis

DAFTAR ISI

ABSTRAK	ii
<i>ABSTRACT</i>	iii
KATA PENGANTAR	iv
DAFTAR ISI	vi
DAFTAR TABEL	x
DAFTAR GAMBAR	xii
DAFTAR LAMPIRAN	xiii
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Identifikasi Masalah	3
1.3 Batasan Masalah	3
1.4 Maksud dan Tujuan Penelitian	4
1.5 Manfaat Penelitian	5
1.6 Metodologi Penelitian	5
1.7 Sistematika Penulisan	5
BAB II TINJAUAN PUSTAKA	7
2.1 Business Intelligence	7

2.1.1	Definisi <i>Business Intelligence</i>	7
2.1.2	Jenis <i>Business Intelligence</i>	8
2.1.3	Komponen <i>Business Intelligence</i>	9
2.1.4	Arsitektur <i>Business Intelligence</i>	10
2.1.5	Manfaat <i>Business Intelligence</i>	11
2.2	<i>Data Warehouse</i>	11
2.2.1	Arsitektur <i>Data Warehouse</i>	12
2.2.2	Struktur <i>Data Warehouse</i>	12
2.2.3	Model Dimensional	15
2.2.4	<i>Extract, Transform, Load</i> (ETL)	15
2.2.5	<i>Online Analytical Processing</i> (OLAP)	16
2.3	<i>Kimball Business Intelligence Life Cycle</i>	16
2.4	<i>Website Application</i>	17
2.5	<i>JavaScript</i>	17
2.6	<i>CSS (Cascading Style Sheets)</i>	18
2.7	<i>Firebase</i>	18
2.8	<i>NodeJS</i>	20
2.9	<i>React</i>	20
BAB III ANALISIS DAN PERANCANGAN		21
3.1	<i>Analisis Perusahaan</i>	21
3.1.1	Sejarah Perusahaan	21

3.1.2	Tempat dan Kedudukan	22
3.1.3	Bentuk dan Badan Hukum	22
3.1.4	Bidang Bisnis	22
3.2	Sistematika Perancangan dan Pembangunan <i>Business Intelligence</i>	23
3.2.1	<i>Project Planning</i>	25
3.2.2	<i>Project Management</i>	27
3.2.3	<i>Business Requirement Definition</i>	30
3.2.4	<i>Technology Track</i>	35
3.2.5	<i>Data Track</i>	38
3.2.6	<i>Business Intelligence Application Track</i>	54
3.2.7	<i>Deployment, Maintenance, Growth</i>	60
A.	Deployment.....	61
B.	<i>Maintenance</i>	61
C.	<i>Growth</i>	61
BAB IV HASIL PENELITIAN DAN PEMBAHASAN		62
4.1	Implementasi Business Intelligence	62
4.1.1	Implementasi Data	62
4.1.2	Implementasi <i>Interface</i>	63
4.2	Pengujian Business Intelligence	69
4.2.1	Rencana Pengujian Developer	69
4.2.2	Rencana Pengujian Beta.....	70

4.2.3 Hasil Pengujian Developer.....	72
4.2.4 Hasil Pengujian Beta.....	73
4.2.5 Evaluasi Pengujian.....	76
BAB V SIMPULAN DAN SARAN	78
5.1 Simpulan	78
5.2 Saran.....	79
DAFTAR PUSTAKA	80
LAMPIRAN.....	83

DAFTAR TABEL

Tabel 3.1 Tujuan Pembangunan.....	26
Tabel 3.2 Timeline Pembangunan Business Intelligence	28
Tabel 3.3 Analisis Kebutuhan Bisnis	31
Tabel 3.4 Analisis Sumber Data.....	34
Tabel 3.5 Identifikasi Grain	38
Tabel 3.6 Identifikasi Dimensi.....	39
Tabel 3.7 Identifikasi Fakta.....	41
Tabel 3.8 Struktur <i>Collection users</i>	45
Tabel 3.9 Struktur <i>Collection Distributors</i>	45
Tabel 3.10 Struktur <i>Collection Customers</i>	46
Tabel 3.11 Struktur <i>Collection Products</i>	46
Tabel 3.12 Struktur <i>Collection Sales Transactions</i>	47
Tabel 3.13 Struktur <i>Collection Restocks</i>	48
Tabel 3.14 Struktur <i>Collection Purchased Transactions</i>	49
Tabel 3.15 Struktur Dimensi <i>Distributors</i>	50
Tabel 3.16 Struktur Dimensi <i>Customers</i>	50
Tabel 3.17 Struktur Dimensi Products	50
Tabel 3.18 Struktur Tabel Dimensi Waktu	51
Tabel 3.19 Struktur Tabel Fakta Pemantauan Penjualan	52
Tabel 3.20 Struktur Tabel Fakta Pemantauan Pembelian	52
Tabel 4.1 Implementasi <i>data warehouse</i>	62

Tabel 4.2 Implementasi <i>Interface</i>	63
Tabel 4.3 Rencana Pengujian Black Box pada interface	69
Tabel 4.4 Skenario Pengujian <i>Black Box</i> pada <i>interface</i>	70
Tabel 4.5 Pertanyaan Pengujian <i>Website Application</i>	71
Tabel 4.6 Rencana pengujian informasi strategis	71
Tabel 4.7 Validasi Hasil Pengujian	72
Tabel 4.8 Hasil Pengujian <i>Developer</i>	72
Tabel 4.9 Hasil Pengujian <i>Website Application</i>	73
Tabel 4.10 Hasil Pengujian Informasi Strategis.....	75

DAFTAR GAMBAR

Gambar 2.1 Arsitektur Business Intelligence.....	10
Gambar 2.2 Arsitektur data warehouse.....	12
Gambar 2.3 Struktur data warehouse.....	13
Gambar 2.4 Kimball Life Cycle.....	17
Gambar 3.1 Struktur Organisasi PD. Intan	22
Gambar 3.2 Kimball life cycle.....	24
Gambar 3.3 Cause and Effect Diagram.....	25
Gambar 3.4 Technical Architecture Design.....	36
Gambar 3.5 Diagram Analisa Pemantauan Pembelian	43
Gambar 3.6 Diagram Analisa Pemantauan Penjualan	44
Gambar 3.7 Use Case Diagram.....	55
Gambar 3.8 Rancangan Interface Dashboard Purchasing.....	57
Gambar 3.9 Rancangan Interface Dashboard Sales	58
Gambar 3.10 Rancangan Interface Dashboard Inventory	59
Gambar 3.11 Rancangan Interface Input New Employee.....	59
Gambar 3.12 Rancangan Interface Login	60
Gambar 4.1 Tampilan Login.....	64
Gambar 4.2 Tampilan Dashoard Purchasing	66
Gambar 4.3 Tampilan Dashboard Sales.....	67
Gambar 4.4 Tampilan Product List.....	68
Gambar 4.5 Hasil Google Form.....	74

DAFTAR LAMPIRAN

Lampiran 1 <i>Source Code App.js</i>	83
Lampiran 2 <i>Source Code Route</i>	85
Lampiran 3 <i>Source Code Root Reducer</i>	92
Lampiran 4 <i>Source Code Login</i>	93
Lampiran 5 <i>Source Code Register</i>	96
Lampiran 6 <i>Source Purchasing Dashboard</i>	102
Lampiran 7 <i>Source Code Sales Dashboard</i>	104
Lampiran 8 <i>Source Code Product List</i>	107
Lampiran 9 <i>Source Code Dashboard.js</i>	131
Lampiran 10 <i>Source Code Index.js</i>	131
Lampiran 11 <i>Source Code package.json</i>	133
Lampiran 12 Implementasi data warehouse dalam firebase	134
Lampiran 13 Rules pada Database	136
Lampiran 14 Implementasi Antarmuka	137
Lampiran 15 Riwayat Hidup	140
Lampiran 16 Surat Keterangan Penelitian	142

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pada dasarnya, data merupakan sumber yang sangat penting bagi suatu perusahaan untuk bertahan kompetitif. Dengan cara analisis yang tepat data tersebut dapat berfungsi sebagai peningkat maupun pengembang performa perusahaan.

PD. Intan merupakan salah satu perusahaan yang bergerak dalam bidang toko kelontong jenis makanan dan minuman ringan. Dalam hal ini PD. Intan sebagai perusahaan dagang makanan ringan secara grosir yang menjual barang dari Produsen atau distributor kepada Konsumen dengan satuan karton maupun pak sejak 11 Agustus 2005 di kota Bandung, Jawa Barat.

Berdasarkan hasil dari wawancara dengan Bapak Muhamad Ropandi selaku pemilik perusahaan, pada saat ini perusahaan memiliki beberapa permasalahan, diantaranya dalam proses penentuan jumlah pengadaan barang.

Perusahaan melakukan pengadaan berdasarkan perkiraan sehingga perusahaan mengalami kelebihan persediaan yang mengakibatkan penumpukan barang, akibatnya beberapa barang mengalami kerusakan sehingga merugikan perusahaan.

Kondisi yang lain adalah ketika kekurangan barang yang mempengaruhi penjualan sehingga menjadi tidak maksimal. Selain itu terdapat regulasi dari perusahaan produsen atau distributor yaitu adanya pembatasan kredit berdasarkan

total harga barang yang diambil sehingga perusahaan memiliki kesempatan *restock* barang yang terbatas.

Menurut Ibu Intan Risana selaku *manager marketing*, selama ini perusahaan belum memiliki sistem informasi maupun *business intelligence*, seluruh operasi masih dilakukan secara manual sehingga banyak kekeliruan yang terjadi dalam proses penjualan barang. Adanya kesalahan dalam pemberian harga ketika terjadi perubahan harga karena tidak adanya pemberitahuan informasi terhadap produk yang mengalami perubahan harga. Pada bulan lalu (Desember 2018) terdapat 30% kesalahan dari 600 transaksi yang dilakukan dikarenakan kurangnya pemberian informasi kepada pegawai yang lain. Pihak perusahaan pun tidak dapat melihat tren penjualan yang ada.

Berdasarkan permasalahan tersebut, maka perlu adanya suatu sistem yang dapat menganalisis data transaksional yang membantu mendukung pengambilan keputusan sehingga menciptakan strategi bisnis yang membantu meningkatkan produktivitas dan keuntungan perusahaan. *Business intelligence* (BI) adalah solusi dalam mengelola data mentah menjadi informasi yang berguna.

Dalam pengerjaan tugas akhir ini akan dilakukan pembangunan sistem informasi *business intelligence* pada PD. Intan yang nantinya akan menghasilkan laporan untuk dianalisis sehingga menghasilkan informasi yang dapat membantu pengambilan keputusan oleh pihak *top level management*. Hasil *business intelligence* akan dibuat dalam bentuk *Website Application* dan di *deploy* dengan menggunakan server *Google Firebase* dengan menggunakan metode *Kimball life cycle*.

1.2 Identifikasi Masalah

Berdasarkan latar belakang masalah yang disampaikan sebelumnya, maka dalam penelitian ini dapat dibuat rumusan masalah yaitu:

1. Bagaimana cara menganalisis data agar dapat menjadi informasi yang dapat mendukung pengambilan keputusan?
2. Bagaimana cara membangun *dashboard business intelligence* untuk memonitor performa penjualan, pembelian maupun pengadaan barang pada PD. Intan?
3. Bagaimana cara mengimplementasikan Kimball life cycle pada *dashboard business intelligence*?
4. Bagaimana menentukan bentuk data yang dapat dianalisis sehingga menjadi informasi yang dapat mendukung pengambilan keputusan?
5. Apa pengaruh metode Kimball *life cycle* pada pembuatan *business intelligence*?

1.3 Batasan Masalah

Dari permasalahan yang telah disampaikan di atas, penulis menentukan beberapa batasan masalah yang dibahas dalam penelitian ini, antara lain sebagai berikut:

1. Kasus yang menjadi objek penelitian adalah penjualan, pembelian dan pengadaan yang terjadi di PD. Intan

2. *Website application business intelligence* menggunakan Bahasa pemrograman HTML (*Hyper Text Markup Language*), CSS (*Cascading Style Sheet*), *Javascript*, dan *framework Node.js*.
3. Metode pembangunan BI (*business intelligence*) yang digunakan adalah *Kimball life cycle*.
4. Data yang diambil dari perusahaan PD. Intan yaitu data transaksi Januari 2019 hingga Maret 2019.

1.4 Maksud dan Tujuan Penelitian

Maksud dari penelitian ini adalah untuk menganalisis dan merancang *business intelligence* berbasis *website application* menggunakan *NoSQL database* dengan metode *Kimball life cycle*.

Setelah pemaparan maksud di atas, tujuan dari penelitian ini adalah:

1. Melakukan analisis data penjualan, pembelian, dan pengadaan barang yang terjadi di PD. Intan
2. Membuat desain dalam pembangunan *business intelligence*
3. Membangun *business intelligence* berbasis *website application*.
4. Menerapkan metode *Kimball life cycle* untuk digunakan dalam pembangunan *business intelligence*.
5. Menganalisis metode *Kimball life cycle* dalam pembangunan dan mengoptimalkan *business intelligence*.
6. Melakukan analisis terhadap penggunaan JavaScript dalam pembangunan *business intelligence*

1.5 Manfaat Penelitian

Manfaat yang ingin dihasilkan dari penelitian ini antara lain, yaitu:

1. Membantu *top level management* untuk memonitor performa penjualan, pembelian maupun pengadaan barang di perusahaan PD. Intan.
2. Mempermudah *top level management* mendapatkan informasi dengan cepat sesuai dengan informasi yang diinginkan sebagai kebutuhan analisis strategi bisnisnya.

1.6 Metodologi Penelitian

Metodologi penelitian yang digunakan dalam penelitian ini adalah:

1. Analisis permasalahan
2. Studi literatur
3. Perancangan sistem
4. Desain dan implementasi
5. Analisis dan kesimpulan

1.7 Sistematika Penulisan

Sistematika penulisan laporan penelitian tugas akhir ini berisikan materi yang akan dibahas pada setiap bab. Hal ini dilakukan untuk memberikan gambaran yang jelas mengenai penelitian yang dijalankan. Sistematika penulisan ini adalah sebagai berikut:

BAB I PENDAHULUAN

Pada bab ini akan diuraikan penjelasan mengenai latar belakang dari penelitian, identifikasi masalah yang terjadi pada objek penelitian, batasan masalah dalam pembangunan *business intelligence*, maksud dan tujuan pembangunan, manfaat dari pembangunan *business intelligence*, metodologi penelitian yang digunakan, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini berisi tentang landasan teori, sumber pustaka, dan referensi yang berhubungan dengan penelitian yaitu teori tentang pembuatan *business intelligence* dengan menggunakan metode Kimball *life cycle*.

BAB III ANALISIS DAN PERANCANGAN

Pada bab ini akan diuraikan analisis kebutuhan, analisis sistem, dan perancangan *business intelligence* yang akan dibangun sesuai dengan metode Kimball *life cycle*.

BAB IV IMPLEMENTASI DAN PENGUJIAN SISTEM

Pada bab ini akan membahas mengenai tahap implementasi yang kemudian akan dilakukan pengujian pada sistem *business intelligence* yang telah dibangun.

BAB V SIMPULAN DAN SARAN

Bab ini merupakan penutup yang berisi kesimpulan dan saran yang didapat dalam pembahasan dari penelitian yang telah dilakukan.

BAB II

TINJAUAN PUSTAKA

2.1 *Business Intelligence*

Hans Peter, seorang peneliti di IBM, memperkenalkan istilah *business intelligence* dalam sebuah artikel yang diterbitkan pada tahun 1958 (Luhn, 1958). *Business intelligence* merupakan perkembangan dari sistem pendukung keputusan atau *Decision Support Systems* (DSS) yang dimulai pada tahun 1960 hingga 1980-an. DSS sendiri dibuat untuk membantu pengambilan keputusan atau *Executive Information Systems* (EIS) dan perencanaan. Dari DSS, *data warehouse*, EIS dan OLAP, akhirnya disatukan sebagai *business intelligence*.

Kemudian pada tahun 1989, istilah *business intelligence* diusulkan oleh Howard Dresner, seorang analis dari Gartner Group, agar dipakai sebagai istilah umum untuk menggambarkan konsep dan metode untuk meningkatkan pengambilan keputusan bisnis dengan menggunakan sistem pendukung berbasis fakta (Power, 2007). Sejak akhir tahun 1990, *business intelligence* menjadi berkembang pesat hingga saat ini.

2.1.1 Definisi *Business Intelligence*

Business intelligence adalah suatu sistem yang terdiri dari proses-proses, alat dan teknik yang dibutuhkan untuk mengumpulkan data sehingga dapat diolah agar menjadi informasi, menganalisis informasi tersebut agar menjadi suatu pengetahuan, dan menampilkan pengetahuan tersebut pada *top level management*

sehingga dapat menciptakan suatu rencana yang mengendalikan suatu aksi bisnis agar dapat meningkatkan produktivitas (Maheswari, 2011). *Business intelligence* mencakup *data warehouse*, *business analytic tools*, *business intelligence dashboard* dan manajemen pengetahuan.

Hasil dari proses BI digunakan untuk menghasilkan pengetahuan, memberikan pengetahuan yang dapat ditindak lanjuti, proses yang harus dilakukan, dan orang yang tepat untuk melakukan keputusan. Tanpa adanya proses oleh orang yang tepat maka *Business intelligence* tidak akan memberikan peningkatan yang besar (Loshin, 2012).

2.1.2 Jenis *Business Intelligence*

Business intelligence terbagi ke dalam lima jenis atau kategori yaitu (Turban, Aronson, Liang, & Sharda, 2007):

1. *Enterprise Reporting*. Jenis laporan yang sangat sesuai untuk laporan operasional dan *dashboard* ini menghasilkan laporan-laporan statis yang didistribusikan ke banyak orang.
2. *Cube Analysis*. Jenis laporan ini digunakan untuk menyediakan analisis OLTP multidimensional yang ditujukan kepada manajer bisnis dalam lingkungan yang terbatas.
3. *Ad Hoc Query and Analysis*. Jenis laporan ini digunakan untuk memberikan akses kepada pengguna agar dapat melakukan *query* pada basis data dan menggali informasi sampai pada tingkat paling dasar dari informasi transaksional.

4. *Statistical Analysis and Data Mining*. Jenis laporan ini digunakan untuk melakukan analisis prediksi atau menentukan korelasi sebab akibat di antara dua matriks.
5. *Delivery Report and Alert*. Jenis laporan ini digunakan untuk mengirimkan laporan secara lengkap secara proaktif atau memberikan peringatan kepada populasi pengguna yang besar.

2.1.3 Komponen *Business Intelligence*

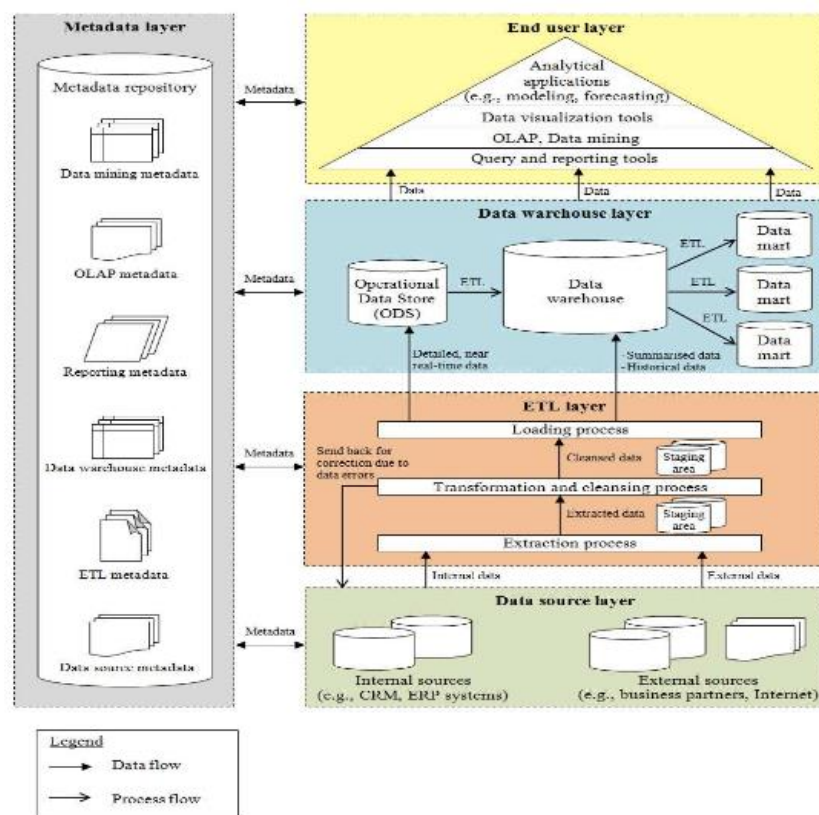
Komponen utama *business intelligence* terdiri dari OLAP (*Online Analytical Processing*), *advanced analytics*, *Corporate Performance Management* (CPM), *real-time BI*, *data warehouse* dan *data mart*, serta *data source* (Ranjan, 2009).

OLAP adalah cara dimana pengguna bisnis dapat melakukan *slice* dan *dice* data menggunakan alat-alat canggih yang memungkinkan untuk navigasi dimensi seperti waktu atau hierarki. *Advanced analytics* yang merupakan hasil dari *data mining*, *forecasting* atau analisis prediktif memanfaatkan teknik analisis statistik untuk memprediksi atau memberikan langkah-langkah kepastian pada fakta. CPM terdiri dari *portal*, *scoreboard*, dan *dashboard* yang menyediakan wadah sehingga *aggregate* dapat menggambarkan suatu cerita. *Real-time BI* adalah distribusi data yang bersifat *real-time*. *Data warehouse* menyimpan data yang diperoleh dari sumber internal maupun sumber eksternal. *Data mart* sebagaimana dijelaskan oleh Inmon adalah kumpulan bidang studi yang diorganisir untuk mendukung keputusan berdasarkan kebutuhan departemen tertentu (Inmon, 2002). Terakhir, *data source*

dapat berupa *database* operasional, data historis, data eksternal misalnya dari perusahaan riset pasar atau dari internet, atau informasi dari lingkungan *data warehouse* yang sudah ada.

2.1.4 Arsitektur *Business Intelligence*

Ada beberapa arsitektur *business intelligence* yang diajukan oleh banyak ahli. Mereka adalah Baars & Kemper (Baars & Kemper, 2008), Balaceanu (Balaceanu, 2007), Shariat & Hightower (Shariat & Hightower, 2007), Turban (Turban et al., 2007), dan Watson (Watson, 2009). Para peneliti merangkum keseluruhan desain arsitekturnya menjadi satu yang dapat diilustrasikan sebagai berikut.



Gambar 2.1 Arsitektur *Business Intelligence* (Ong, Siew, & Wong, 2011)

2.1.5 Manfaat *Business Intelligence*

Ada banyak sekali manfaat *business intelligence*, diantaranya adalah sebagai berikut (Juneja, 2019):

1. *Business intelligence* melaporkan informasi penting yang lebih cepat dan lebih akurat.
2. *Business intelligence* memfasilitasi proses pengambilan keputusan yang lebih baik dan efisien.
3. *Business intelligence* menyediakan informasi tepat waktu untuk manajemen hubungan pelanggan yang lebih baik.
4. *Business intelligence* meningkatkan profitabilitas perusahaan.
5. *Business intelligence* menyediakan fasilitas untuk menilai kesiapan organisasi dalam menghadapi tantangan bisnis baru.
6. *Business intelligence* mendukung penggunaan praktik terbaik dan mengidentifikasi setiap biaya tersembunyi.

2.2 *Data Warehouse*

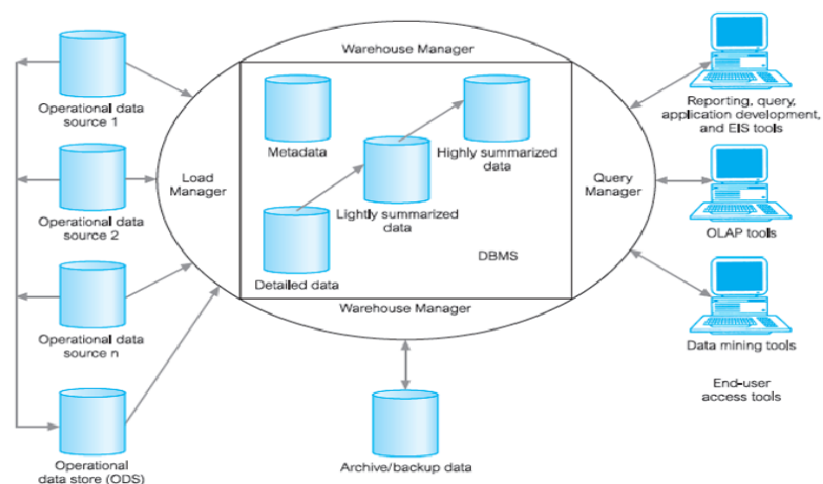
Menurut Kimball dan Ross, *data warehouse* merupakan *database* yang bersifat analisis dan *read only* yang digunakan sebagai pondasi dari sistem penunjang keputusan. *Data warehouse* mengumpulkan seluruh data yang ada pada perusahaan baik itu data masa kini ataupun data yang sudah lampau agar dapat diperoleh pandangan yang lebih baik untuk menunjang proses bisnis dan pengambilan keputusan. Tujuan utama dari *data warehouse* agar membuat informasi yang dimiliki organisasi dapat dengan mudah diakses, menyajikan

informasi yang dimiliki organisasi secara konsisten, dan memberikan landasan yang kuat dalam meningkatkan pengambilan keputusan (Kimball & Ross, 2002).

Sedangkan menurut Inmon, *data warehouse* adalah koleksi data yang bersifat *integrated*, *subject oriented*, *time-variant*, dan *non-volatile* yang membantu proses pengambilan keputusan oleh pihak manajemen (Inmon, 2002).

2.2.1 Arsitektur Data Warehouse

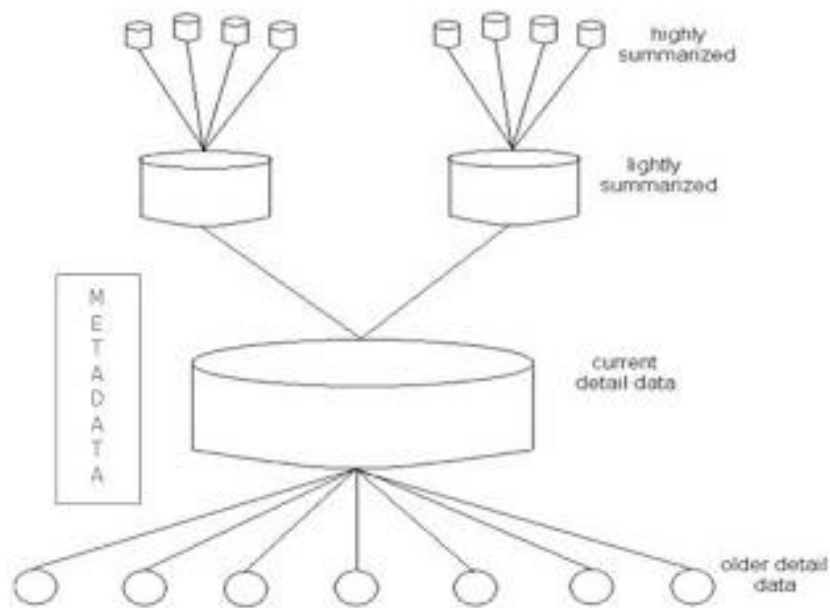
Menurut Connolly dan Begg (Connolly & Begg, 2014), *data warehouse* terdiri dari struktur dan komponen yang saling berhubungan satu sama lain dalam membangun *data warehouse*. Arsitektur *data warehouse* dapat diilustrasikan sebagai berikut.



Gambar 2.2 Arsitektur data warehouse (Connolly & Begg, 2014)

2.2.2 Struktur Data Warehouse

Menurut Indrajani (Indrajani, S.Kom., 2014), struktur *data warehouse* dapat diilustrasikan sebagai berikut.



Gambar 2.3 Struktur data warehouse (Indrajani, S.Kom., 2014)

- *Older Detail Data*
Merupakan data *back-up* (cadangan) yang jarang diakses. Biasanya disimpan pada media penyimpanan yang berbeda. Penyusunan direktorinya dilakukan berdasarkan urutan umur data sehingga data dapat tersusun rapi dan mempermudah dalam melakukan akses selanjutnya.
- *Current Detail Data*
Merupakan level terendah dalam struktur *data warehouse* yang menggambarkan detail data yang aktif pada saat ini dan keadaan yang sedang berjalan. Data jenis ini memerlukan media penyimpanan yang besar dan merupakan data yang sering diakses.
- *Lightly Summarized Data*
Merupakan rangkuman data dari *current detail data* namun data ini belum dapat digunakan untuk pengambilan keputusan karena masih belum bersifat

total summary atau bisa dibilang masih bersifat detail. Data jenis ini biasanya diakses untuk memantau kondisi yang sedang dan sudah berjalan.

- *Highly Summarized Data*

Merupakan data yang bersifat *total summary*. Data ini sangat mudah diakses terutama untuk melakukan analisis perbandingan data berdasarkan urutan waktu dan analisis menggunakan data multidimensional. Data multidimensional adalah suatu teknologi yang dirancang untuk meningkatkan efisiensi dalam *query* data sehingga menjadi media penyimpanan yang lebih baik serta memudahkan pengambilan data dalam volume besar.

- Metadata

Merupakan ilustrasi yang menggambarkan tentang struktur, isi, kunci, indeks dari data. Metadata dapat dikelompokkan ke dalam beberapa jenis yaitu:

- *Technical metadata*, suatu objek dan proses yang membentuk sistem *business intelligence* dari persepektif teknis, berisi informasi mengenai *data warehouse* untuk melakukan pengembangan *data warehouse* dan tugas-tugas manajemen oleh administrator dan perancang *data warehouse*.
- *Business metadata*, berisi informasi yang memberikan *user* informasi yang tersimpan dalam *data warehouse* menjadi suatu perspektif yang mudah dimengerti.

- *Data warehouse operational information*, yaitu seperti data *history*, *ownership*, menelusuri jejak audit, penggunaan data.

2.2.3 Model Dimensional

Model dimensional adalah struktur *database* yang digunakan untuk mengoptimalkan *query* dan sistem *data warehouse* yang terdiri dari tabel fakta dan tabel dimensi.

Fakta adalah nilai numerik yang akan dihitung atau dijumlahkan untuk tujuan bisnis. Dimensi merupakan titik awal untuk mendapatkan fakta, dimensi sendiri adalah informasi yang menarik untuk tujuan bisnis.

Model dimensional sangat cocok untuk *business intelligence* (BI) dan *data warehouse*. Model dimensional menggambarkan proses bisnis di seluruh perusahaan dan mengatur data dan strukturnya secara logis. Tujuannya adalah untuk memungkinkan pelaporan, *query*, dan analisis BI (Sherman, 2015).

2.2.4 *Extract, Transform, Load* (ETL)

Menurut Denney, ETL adalah prosedur umum untuk menyalin data dari satu sumber atau lebih ke dalam sistem tujuan yang merepresentasikan data secara berbeda dari sumbernya atau dalam konteks yang berbeda dari sumbernya (Denney, Long, Armistead, Anderson, & Conway, 2016). Proses ETL menjadi konsep populer di tahun 1970-an dan sering digunakan dalam *data warehouse*.

Ekstraksi data melibatkan pengambilan data dari sumber yang homogen atau heterogen. Transformasi data memproses data dengan pembersihan data dan

mentransformasikannya ke dalam format/struktur penyimpanan yang tepat untuk keperluan pencarian dan analisis. Terakhir, pemuatan data menggambarkan penyisipan data ke dalam *database* target akhir seperti penyimpanan data operasional, *data mart*, *data lake* atau *data warehouse*.

2.2.5 Online Analytical Processing (OLAP)

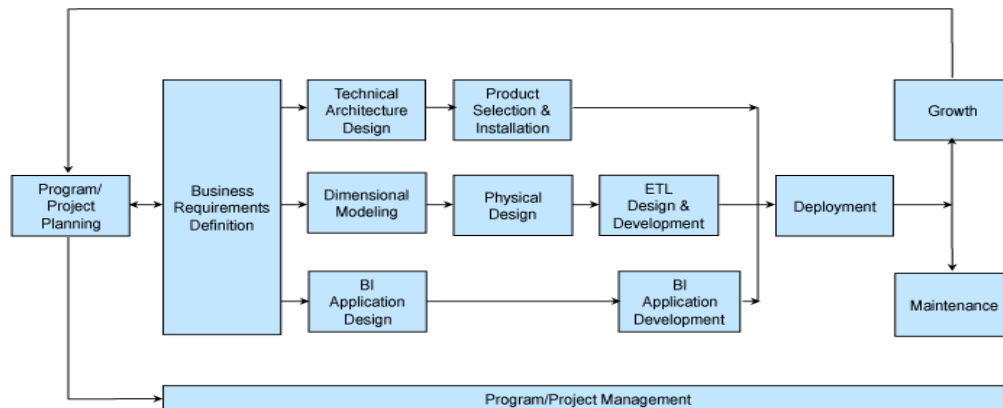
Menurut Codd (Codd, Codd, & Salley, 1993), *Online Analytical Processing* atau disingkat OLAP adalah metode pendekatan untuk menyajikan jawaban dari permintaan proses analisis yang bersifat dimensional secara cepat, yaitu desain dari aplikasi dan teknologi yang dapat mengoleksi, menyimpan, memanipulasi suatu data multidimensional untuk tujuan analisis.

Sedangkan menurut Turban, OLAP merupakan kemampuan dari memanipulasi data secara efisien dari beberapa pandangan atau perspektif. OLAP adalah bagian dari kategori yang lebih global dari *business intelligence*, yang juga merangkum hubungan antara *data reporting* dan *data mining*. Istilah OLAP merupakan perampingan dari istilah lama *database OLTP (Online Transaction Processing)* (Turban, Sharda, Aronson, & King, 2011).

2.3 Kimball Business Intelligence Life Cycle

Siklus Hidup Kimball adalah metodologi untuk mengembangkan *data warehouse*, dan telah dikembangkan oleh Ralph Kimball dan berbagai rekan kerjanya (Kimball, Ross, Thornthwaite, Mundy, & Becker, 2008).

Gambar 2.4 merupakan ilustrasi Kimball *life cycle*



Gambar 2.4 Kimball Life Cycle (Kimball, Ross, Thornthwaite, Mundy, & Becker, 2008)

2.4 Website Application

Website application adalah sebuah *website* yang memiliki sifat yang mirip dengan aplikasi desktop yaitu bersifat dinamis dan terus berkembang. *Website application* bergantung kepada interaksi pengguna, apakah itu kontribusi *content* dari *user* atau dengan mengumpulkan data dari sumber lain untuk ditampilkan kepada pengguna atau gabungan dari keduanya (Uden, 2002).

Seperti banyak *website* pada umumnya di internet, *website application* dibuat menggunakan HTML, CSS dan JavaScript. Selain itu juga menggunakan bahasa pemrograman web seperti PHP, Ruby atau Python. Sementara untuk kebutuhan yang lebih kompleks bisa menggunakan *framework*. Kebanyakan *website application* juga hampir selalu menggunakan *database*.

2.5 JavaScript

JavaScript adalah bahasa pemrograman yang paling populer untuk digunakan dalam pengembangan *website application* karena pengembang atau

developers biasanya menggunakan JavaScript untuk meningkatkan performa *web browser* menjadi lebih baik (Luanghirun & Suwannasart, 2016).

Keuntungan bahasa pemrograman JavaScript adalah JavaScript digunakan di kebanyakan *website*, mudah untuk mengakses berbagai objek dokumen, dapat memanipulasi semua objek, tersedia animasi yang menarik, tidak diperlukan *plugin* khusus untuk mengaksesnya, dan keamanannya sangat tinggi (Ahmed, 2014).

2.6 CSS (*Cascading Style Sheets*)

CSS atau *Cascading Style Sheets* adalah sebuah aturan yang biasa digunakan dalam pengembangan *website*. CSS bukan sebuah bahasa pemrograman, namun CSS menjadi komponen kunci dari *user experience* sebuah *website* karena CSS bertugas untuk membuat *website* menjadi lebih menarik, *user friendly*, dan menjadi lebih enak untuk dilihat (Theisen, 2019).

CSS dapat memanipulasi gambar, teks, warna teks, border, warna border, margin kiri, kanan, atas, bawah, dan lain-lain. Dengan adanya CSS memungkinkan kita untuk menampilkan halaman yang sama dengan format yang berbeda.

2.7 Firebase

Firebase adalah sebuah BaaS (*Backend as a Service*) yang dikembangkan oleh Google untuk memudahkan *developer* dalam membangun aplikasi tanpa mengkhawatirkan infrastruktur *backend* aplikasi tersebut (Gupta & Kapoor, 2016). Firebase memungkinkan para *developer* untuk membuat *website application* tanpa *server-side programming* yang membuat pengembangan menjadi lebih cepat dan

mudah (Kumar, Akhi, Gunti, & Reddy, 2016). Dengan menggunakan Firebase, *developer* tidak perlu membuat atau membangun REST API dengan fungsi-fungsi standar yaitu menyimpan data dan melakukan verifikasi *user*, melakukan *create*, *read*, *update*, *delete* terhadap data dalam *database*, dan fungsi – fungsi lain yang dimana fungsi-fungsi standar tersebut sudah tersedia di dalam Firebase.

Terdapat banyak fitur yang dapat digunakan dalam Firebase, diantaranya adalah sebagai berikut:

1. *Firebase Cloud Messaging*, untuk memberikan pemberitahuan dan membuat komunikasi dua arah antara perangkat.
2. *Firebase Authentication*, untuk membuat layanan otentikasi menggunakan sandi, nomor telepon, atau penyedia identitas gabungan yang populer, seperti Google, Facebook, dan Twitter, dan lain-lain.
3. *Firebase Remote Config*, untuk melakukan perubahan konfigurasi di dalam aplikasi Android/iOS, tanpa harus melakukan pembaruan aplikasi di Play Store/App Store.
4. *Firebase Real-time Database*, untuk memungkinkan data aplikasi disinkronkan untuk seluruh klien dan disimpan di *Firebase cloud*.
5. *Firebase Storage*, untuk menyimpan dan menampilkan konten buatan pengguna, seperti foto atau video dan menambahkan keamanan Google pada unggah dan unduh berkas untuk aplikasi Firebase bagaimanapun kualitas jaringannya.
6. *Firebase Hosting*, untuk melakukan hosting yang cepat dan aman untuk aplikasi web serta konten yang statis dan dinamis.

2.8 NodeJS

Node.js adalah lingkungan *runtime platform* yang awalnya dikembangkan pada tahun 2009 oleh Ryan Dahl untuk mengembangkan aplikasi sisi server. Node.js dibuat untuk mengatasi masalah pemindaian platform dengan kinerja dalam waktu komunikasi jaringan yang mendedikasikan waktu pemrosesan dan tanggapan web yang berlebihan. Node.js menggunakan model I/O yang didorong oleh peristiwa yang membuatnya ringan dan efisien, sempurna untuk aplikasi *real-time* yang berjalan di perangkat terdistribusi (Kurniawan, 2014).

2.9 React

React.js adalah sebuah library JavaScript yang dibuat oleh Facebook. Sementara itu ada juga React Native yaitu sebuah *framework* yang bisa digunakan untuk mengembangkan aplikasi Android dan iOS sekaligus dengan menggunakan bahasa JavaScript. React adalah library yang bersifat *composable user interface*, yang artinya kita dapat membuat berbagai UI yang bisa kita bagi menjadi beberapa komponen (Domes, 2017).

BAB III

ANALISIS DAN PERANCANGAN

3.1 Analisis Perusahaan

Pada bagian ini akan dijelaskan informasi mengenai PD. Intan mulai dari sejarah perusahaan, tempat dan kedudukan perusahaan, bentuk dan badan hukum perusahaan, bidang bisnis perusahaan, visi dan misi perusahaan serta struktur organisasi perusahaan.

3.1.1 Sejarah Perusahaan

PD. Intan resmi berdiri pada 11 Agustus 2005 sebagai perusahaan yang bergerak dibidang toko kelontong jenis makanan dan minuman ringan sebagai mediator antara produsen atau distributor dan konsumen dengan berjualan produk dalam satuan karton maupun pak. Berawal dari toko kelontong yang beranggota 2 orang, PD Intan terus menerus berkembang dan menjalin kerjasama dengan perusahaan lain dan berkomitmen untuk menjadi perusahaan grosir terbaik.

PD. Intan memiliki organisasi bernama AMPI (Asosisasi Motoris PD. Intan) yang terbentuk dari anggota motoris yang memiliki misi untuk membantu perusahaan dalam mencapai target, sehingga mendapatkan promo atau bonus yang membantu bagi seluruh konsumen yang berbelanja di PD. Intan, selain itu melakukan kegiatan sosial seperti melakukan *gathering* dan doa bersama untuk mempererat tali silaturahmi antar PD. Intan dan kosumen yang melakukan transaksi jual beli di PD. Intan.

3.1.2 Tempat dan Kedudukan

Perusahaan ini berlokasi di Bandung Jalan Terusan Pasirkoja No.132, Kecamatan Bojongloa Kaler, Kelurahan Jamika, Kota Bandung, Jawa Barat.

3.1.3 Bentuk dan Badan Hukum

PD. Intan diresmikan berdasarkan surat keterangan domisili perusahaan No.29/DP/VIII/2005 pada hari Kamis tanggal 11 Agustus 2005 di Bandung. Yang disahkan oleh Camat Bojongloa Kaler Sunarya S. Hidayat, dan Lurah Jamika Rustano. Dan terdaftar pada akta tanda daftar perusahaan, saat ini telah mengalami beberapa perubahan, yang disahkan oleh kepala badan pelayanan perizinan terpadu terakhir pada 24 Agustus 2014 di Bandung dan perubahan ini telah disimpan dalam *database* badan pelayanan perizinan terpadu pemerintah kota bandung.

3.1.4 Bidang Bisnis

Sesuai dengan surat izin usaha perdagangan perusahaan, ruang lingkup kegiatan meliputi penjualan barang kepada pengecer/konsumen. Barang yang dijual merupakan makanan dan minuman ringan.

Di bawah ini merupakan gambaran struktur organisasi di PD. Intan.



Gambar 3.1 Struktur Organisasi PD. Intan

3.2 Sistematika Perancangan dan Pembangunan *Business Intelligence*

Untuk melakukan pembangunan, diperlukannya alur berpikir yang logis sehingga memiliki arah yang jelas, teratur dan sistematis dalam pemecahan masalah yang sedang dihadapi. Sehingga, alur tersebut akan berguna dalam pencapaian tujuan yang telah ditetapkan. Alur tersebut akan berguna dalam pembangunan *business intelligence*.

Dalam hal ini penulis menggunakan metode Kimball *Business Intelligence Life cycle* karena metode ini membangun *data mart* sesuai subjek area proses bisnis yang kemudian dilakukan proses integrasi antar *data mart* sehingga informasi yang tersimpan pada *data warehouse* berbentuk multi dimensional dan sudah dilakukan denormalisasi. Hal ini bertujuan untuk memudahkan pengguna menelusuri *data warehouse* perusahaan.

Metode Kimball memiliki keunggulan diantaranya:

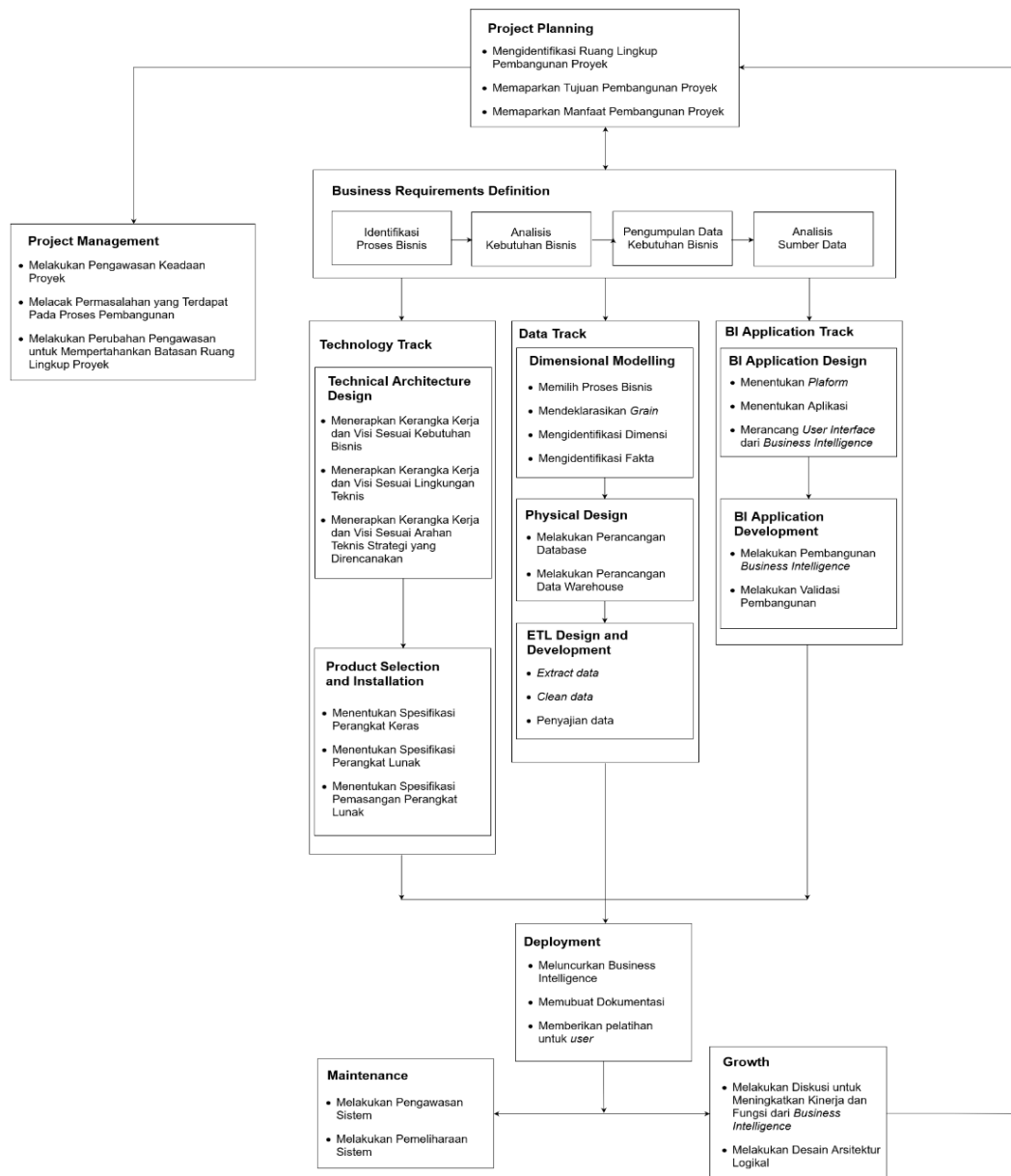
1. Sifatnya yang mudah untuk dipahami dikarenakan pembangunan *data mart* per proses bisnis
2. Waktu pembangunan yang lebih cepat
3. Tidak memerlukan sumber daya yang banyak dalam pembangunannya

Namun metode kimball memiliki kelemahan diantaranya:

1. Data yang berlebihan dapat menyebabkan anomali pembaruan
2. Menambahkan kolom ke tabel fakta dapat menyebabkan masalah kinerja

3. Integrasi data lama kedalam data warehouse menjadi proses yang kompleks

Dibawah ini merupakan Kimball *business intelligence life cycle* sebagai sistematika perancangan yang digunakan pada pembangunan BI dan di sesuaikan untuk kasus perusahaan PD. Intan.



Gambar 3.2 Kimball life cycle

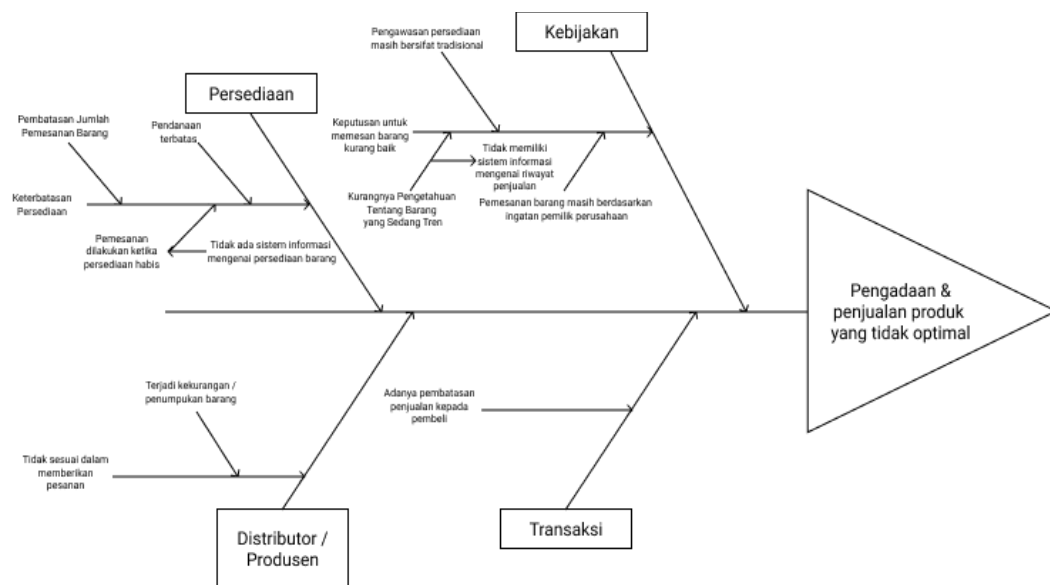
3.2.1 Project Planning

Pada tahap ini, akan dilakukan identifikasi kebutuhan yang terkait dengan pembangunan *Business Intelligence* diantaranya:

1. Mengidentifikasi Ruang Lingkup Pembangunan

Pada tahap ini dilakukan wawancara dengan pemilik perusahaan sehingga ruang lingkup yang pembangunan dapat diketahui. Berdasarkan wawancara dengan pemilik PD. Intan permasalahan yang dihadapi perusahaan adalah pengadaan dan penjualan produk yang tidak dioptimal.

Untuk memaparkan identifikasi ruang lingkup dibentuk *cause and effect diagram*.



Gambar 3.3 Cause and Effect Diagram

2. Tujuan Pembangunan

Tujuan dari pembangunan ini adalah merancang sistem *Business Intelligence* yang memudahkan pemantauan kegiatan pengadaan, penjualan dan pengelolaan persediaan dengan menyediakan informasi yang berguna

dalam proses analisis dan pengambilan keputusan sehingga dapat membantu memperbaiki perencanaan pengisian persediaan barang dan meningkatkan keuntungan yang didapat.

Tabel 3.1 Tujuan Pembangunan

Aspek	Keputusan
Pembelian (Sales)	Memantau pendanaan yang terjadi di perusahaan
	Menentukan jumlah pendanaan untuk suatu produk
	Memantau transaksi pembelian
	Memantau perubahan harga suatu produk
Pengadaan	Mengidentifikasi kriteria evaluasi dan pemilihan distributor /produsen
	Memantau perbandingan <i>restock order</i> dengan <i>purchase order</i>
	Menentukan persediaan produk yang harus tersedia
	Menghitung jumlah kebutuhan <i>replenish</i> optimal
	Menentukan <i>safety stock</i>
	Menentukan <i>reorder point</i>
	Membantu pengecekan untuk merilis <i>restock order</i>
Penjualan (Sales)	Memantau dan menganalisis profit dan pendapatan pada periode tertentu
	Memantau perbandingan profit dan pendapatan pada suatu periode tertentu
	Menganalisa tren penjualan suatu produk

3. Manfaat Pembangunan

Pada tahap ini akan dijelaskan tentang harapan dari dampak yang akan terjadi ketika pembangunan telah selesai, manfaat dari pembangunan harus berkaitan dengan masalah yang diteliti.

Manfaat yang diharapkan dalam pembangunan *Business Intelligence* ini adalah meningkatkan efektivitas pemesanan persediaan sehingga dapat meningkatkan produktivitas perusahaan.

3.2.2 *Project Management*

Project Management merupakan tahap pengawasan dan pelacakan masalah selama pembangunan *business intelligence*. hal ini dilakukan agar pembangunan tetap mengikuti sistematika *life cycle*.

Pada pengerjaannya dibuat *timeline* pengerjaan proyek dari awal perencanaan hingga peluncuran. Selain itu dibuat pula *project board* untuk memantau *progress* yang telah dan akan dilakukan dalam pembuatan *business intelligence*.

Timeline dibagi menjadi beberapa tahap sesuai dengan sistematika yang dibuat. Hal ini dikarenakan ada kegiatan yang dilakukan secara paralel atau bersamaan. Tahap – tahap dalam pengerjaan antara lain adalah tahap perencanaan, Tahap pencarian kebutuhan bisnis, Tahap technology track, Tahap data track, Tahap Business Intelligence Application track, Tahap Pembangunan Website, Tahap Penanganan *Firebase*.

3.2.3 *Business Requirement Definition*

Untuk menghasilkan inisiatif BI yang baik, diperlukan pemahaman tentang pengguna bisnis dan kebutuhannya. Tanpa adanya pemahaman tersebut inisiatif BI tersebut akan menjadi hal yang sia – sia.

Pendekatan ini bertujuan untuk mengumpulkan pengetahuan mengenai permasalahan dari objek yang difokuskan, analisis tersebut harus memahami faktor – faktor yang mendorong agar dapat merubah variabel bisnis sehingga rancangan dapat digunakan. Pendefinisian dari kebutuhan bisnis merupakan hal yang dibutuhkan, hal ini sangat penting karena merupakan dasar dari seluruh siklus hidup *business intelligence*.

Untuk mendapatkan *business requirement definition* dilakukan wawancara secara berkala dengan pemilik perusahaan PD. Intan, data tersebut kemudian akan melewati tahap – tahap analisis yang terdiri dari tahap identifikasi proses, analisis kebutuhan bisnis, pengumpulan data kebutuhan bisnis dan analisis sumber.

1. Identifikasi Proses bisnis

Dalam tahap ini dilakukan proses identifikasi proses bisnis pada bagian *purchasing*, dan *sales* mengenai proses penjualan dan pembelian yang berlangsung dari PD Intan.

Proses bisnis yang akan dijadikan fokus dalam pembuatan Business Intelligence:

- Pembelian (*Sales*)

Kebutuhan bisnis diperlukan oleh pemilik pada proses ini adalah menampilkan dan menganalisis seluruh data pembelian yang terjadi di PD.Intan.

- Penjualan (*Purchasing*)

Kebutuhan bisnis yang diperlukan oleh pemilik pada proses ini adalah menampilkan dan melakukan analisis terhadap seluruh transaksi penjualan yang terjadi di PD. Intan.

- Pengadaan

Kebutuhan yang diperlukan oleh pemilik pada proses ini adalah menampilkan seluruh produk yang terdapat di Gudang PD. Intan. Sehingga pengguna mengetahui informasi mengenai produk yang ada.

2. Analisis Kebutuhan Bisnis

Setelah proses bisnis didapatkan, maka proses tersebut akan menghasilkan kebutuhan bisnis yang digunakan sebagai acuan dan dasar dari pembangunan *Business Intelligence*. Selain kebutuhan bisnis, diperlukan pula kebutuhan informasi digunakan untuk pemilihan data yang dibutuhkan sehingga dapat mempermudah perancangan dan pembangunan database maupun data warehouse.

Tabel 3.3 Analisis Kebutuhan Bisnis

A. Pembelian	
Kebutuhan Bisnis	Kebutuhan Informasi
Berapa transaksi pengeluaran yang dilakukan oleh PD Intan dalam suatu periode?	<ul style="list-style-type: none"> • Jumlah transaksi per periode

Kebutuhan Bisnis	Kebutuhan Informasi
Berapa transaksi pengeluaran yang dilakukan dalam suatu periode?	<ul style="list-style-type: none"> • Jumlah transaksi per distributor/produsen • Jumlah transaksi per produk • Perbandingan harga per produk
Berapa jumlah pembelian yang dilakukan terhadap suatu distributor / produsen dalam suatu periode?	
Berapa jumlah pembelian per produk?	
Apa saja produk yang mengalami perubahan harga?	<ul style="list-style-type: none"> • Perubahan harga per produk
B. Penjualan	
Kebutuhan Bisnis	Kebutuhan Informasi
Berapa profit yang didapat secara keseluruhan?	<ul style="list-style-type: none"> • Jumlah Profit pada suatu periode • Jumlah Penjualan barang • Perbandingan jumlah nilai penjualan sekarang dengan periode lain
Berapa profit yang didapat pada periode tertentu?	
Berapa jumlah nilai penjualan keseluruhan?	
Berapa jumlah nilai penjualan per periode?	
Berapa jumlah nilai penjualan per customer?	
Berapa jumlah nilai penjualan per produk?	

Kebutuhan Bisnis	Kebutuhan Informasi
Berapa jumlah nilai penjualan per customer?	<ul style="list-style-type: none"> • Jumlah Profit pada suatu periode • Jumlah Penjualan barang
Bagaimana perbandingan penjualan sekarang dengan periode lain?	<ul style="list-style-type: none"> • Perbandingan jumlah nilai penjualan sekarang dengan periode lain
C. Pengadaan	
Kebutuhan Bisnis	Kebutuhan Informasi
Berapa banyak produk yang tersedia dalam Gudang?	<ul style="list-style-type: none"> • Jumlah persediaan suatu produk yang tersedia
Berapa selisih antara <i>restock order</i> dengan <i>purchase order</i> ?	<ul style="list-style-type: none"> • Jumlah barang yang masuk ke dalam Gudang • Status selisih • <i>Delivered quantity</i> • <i>Ordered quantity</i> • Selisih antara <i>delivered product</i> dengan <i>ordered product</i>

3. Pengumpulan Data Kebutuhan Bisnis

Pada tahap ini seluruh data yang didapat dari beberapa sumber dikumpulkan. Sumber data tersebut diantaranya adalah data penjualan dan pembelian PD Intan, dan wawancara dengan pemilik perusahaan.

4. Analisis Sumber Data

Data yang telah dikumpulkan sebelumnya akan dilakukan analisis kembali, tahap ini bertujuan untuk membuat data mentah menjadi data yang berkualitas yang dapat digunakan pada pembangunan *Business Intelligence*

Tabel 3.4 Analisis Sumber Data

Aspek	Kebutuhan data
Pembelian	
Memantau transaksi pembelian usaha	Tanggal transaksi
	Nama distributor / produsen
	Produk
	<i>Quantity</i> produk
	Harga beli produk
	Total pembelian
Memantau perubahan harga produk	Tanggal transaksi
	Nama distributor
	Produk
	Harga beli per produk
Penjualan	
<ul style="list-style-type: none"> Memantau dan menganalisis profit dan pendapatan pada periode tertentu Memantau perbandingan profit dan pendapatan pada suatu periode 	Tanggal transaksi
	<i>Customer</i>
	Produk yang terjual

Aspek	Kebutuhan data
Penjualan	
<ul style="list-style-type: none"> • Memantau perbandingan profit dan pendapatan pada suatu periode • Menganalisa tren penjualan suatu produk 	Tanggal transaksi
	<i>Customer</i>
	Produk yang terjual
	<i>Quantity</i> produk terjual
	Nilai transaksi
Pengadaan	
<ul style="list-style-type: none"> • Memantau persediaan produk yang dimiliki • Membantu merilis <i>restock order</i> 	Distributor/Produsen
	Produk
	<i>Quantity</i> produk

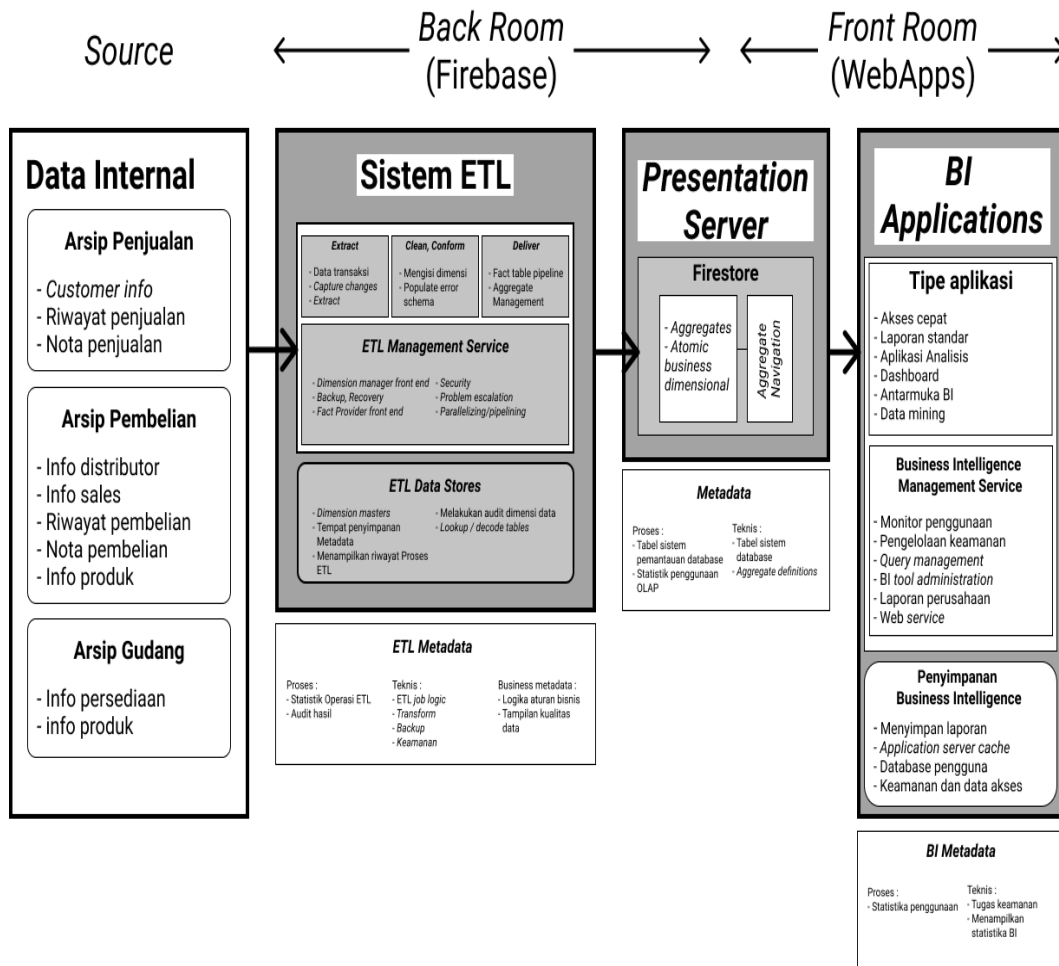
3.2.4 *Technology Track*

Pada jalur di tahap ini akan diuraikan konsep arsitektur teknis secara keluruhan, lalu memilih dan menggunakan produk yang menunjang perancangan *Business Intelligence*

1. *Technical Architecture Design*

Pada tahap ini diterapkan kerangka kerja dan visi arsitektur secara keseluruhan yang meliputi tiga faktor diantaranya kebutuhan bisnis, lingkungan teknis saat ini, dan arahan teknis strategis yang direncanakan.

Berikut merupakan *Technical Architecture Design* yang digunakan dalam pembangunan BI



Gambar 3.4 Technical Architecture Design

2. Product Selection and Installation

Setelah perencanaan arsitektur disusun, tahap selanjutnya adalah melakukan *product selection & installation* sesuai kerangka kerja yang dipilih. Pada tahap ini pula dilakukan beberapa penentuan, antara lain menentukan spesifikasi perangkat keras, menentukan spesifikasi perangkat lunak, dan melakukan spesifikasi instalasi perangkat lunak.

A. Spesifikasi Perangkat Lunak

Untuk pembangunan *business intelligence*, dibutuhkan perangkat lunak yang dapat menunjang pengembangan aplikasi ini. Berikut merupakan kebutuhan perangkat lunak yang akan digunakan untuk pembangunan.

1. Sistem operasi yang digunakan untuk mengembangkan aplikasi ini adalah Windows 10 Home Single Language Versi 1809.
2. *Database* yang digunakan untuk mengembangkan aplikasi ini adalah Firebase *Cloud* Firestore.
3. *Browser* yang digunakan untuk mengembangkan aplikasi ini adalah Google Chrome dan Mozilla Firefox.
4. *Text editor* yang digunakan untuk mengembangkan aplikasi ini adalah Visual Studio Code versi 1.36.1 dilengkapi dengan *extension* *prettier* dan ES7 React/Redux.GraphQL/JS snippets.

B. Spesifikasi Perangkat Keras

Selain membutuhkan perangkat lunak untuk mengembangkan *business intelligence*, dibutuhkan perangkat keras yang menunjang untuk pembangunan *business intelligence*. Berikut merupakan kebutuhan rincian perangkat keras komputer yang akan digunakan untuk pembangunan ini.

1. *Processor*: 1.60 GHz Intel Core i5
2. *Memory*: 8GB 2400 MHz DDR4
3. *Graphics Display*: Intel UHD Graphics 620 4021 MB graphics
4. *Harddisk*: 512GB
5. *Graphics Render*: NVIDIA GeForce MX250

3.2.5 Data Track

Pada jalur ini akan diuraikan rangkaian aktivitas yang mengikuti definisi persyaratan bisnis, dimulai dari *dimensional modelling*, *physical design*, *ETL design and development*.

1. Dimensional Modelling

Data yang didapat pada tahap *business requirement* digunakan sebagai acuan untuk proses perancangan. *Dimensional modelling* dilakukan dengan acuan *Four-Step Kimball Dimensional Design process (bottom-up)*.

Dibawah merupakan rancangan *dimensional modelling* berdasarkan *Four-step kimball*

- Memilih Aspek Bisnis

Aspek Bisnis yang diambil pada dimensional modelling ini adalah aspek pembelian dan penjualan barang yang terjadi di PD. Intan

- Mengidentifikasi Grain

Identifikasi grain yaitu, Pemaparan dari aspek bisnis untuk menghasilkan suatu informasi mengenai kebutuhan yang diperlukan oleh masing – masing aspek. Tabel 3.5 memaparkan analisis kebutuhan dari setiap aspek bisnis.

Tabel 3.5 Identifikasi Grain

Skema Model dimensional	Analisis Kebutuhan	Dimensi
Pemantauan Penjualan	Berapa profit yang didapat secara keseluruhan?	Waktu, Produk, Pelanggan

Skema Model dimensional	Analisis Kebutuhan	Dimensi
Pemantauan Penjualan	Berapa profit yang didapat pada periode tertentu?	Waktu, Produk, Pelanggan
	Berapa jumlah nilai penjualan keseluruhan?	
	Berapa jumlah nilai penjualan per periode?	
	Berapa jumlah nilai penjualan per customer?	
	Berapa jumlah nilai penjualan per produk?	
Pemantauan Pembelian	Berapa transaksi pengeluaran yang dilakukan dalam suatu periode?	Waktu, Produk, Distributor
	Berapa jumlah pembelian yang dilakukan terhadap suatu distributor / produsen dalam suatu periode?	
	Berapa jumlah pembelian per produk?	
	Apa saja produk yang mengalami perubahan harga?	

- Mengidentifikasi Dimensi

Pada bagian ini, diuraikan tabel dimensi beserta atribut yang dibutuhkan untuk menghasilkan informasi yang diinginkan. Tabel 3.6 menguraikan identifikasi dimensi dari setiap dimensi

Tabel 3.6 Identifikasi Dimensi

Dimensi	Atribut	Keterangan
Waktu	idWaktu	Berupa kode identifier dimensi waktu

	Tanggal	Berupa informasi mengenai tanggal
Dimensi	Atribut	Keterangan
Waktu	Bulan	Berupa informasi mengenai bulan
	Tahun	Berupa informasi mengenai tahun
Produk	idProduk	Berupa kode sebagai identifier dimensi produk
	namaProduk	Berupa informasi mengenai nama produk
	jenisProduk	Berupa informasi mengenai jenis produk
	hargaBeli	Berupa informasi mengenai harga beli produk
	hargaJual	Berupa informasi mengenai harga jual
Distributor	idDist	Berupa kode identifier dimensi distributor
	namaDist	Berupa informasi mengenai nama distributor
	jenisDist	Berupa informasi mengenai jenis distributor
	Alamat	Berupa informasi mengenai alamat Distributor
Pelanggan	idCust	Berupa kode identifier dimensi pelanggan
	namaCust	Berupa informasi mengenai nama pelanggan
	jenisCust	Berupa informasi mengenai jenis pelanggan
	Alamat	Berupa informasi mengenai alamat pelanggan
Petugas	idPetugas	Berupa kode identifier dimensi petugas

	namaPetugas	Berupa informasi mengenai nama petugas
--	-------------	--

- Mengidentifikasi Fakta

Untuk menghasilkan Informasi yang dapat diambil kesimpulan. Diperlukan identifikasi kebutuhan fakta. Pada bagian ini, akan diuraikan kebutuhan atribut yang diperlukan dalam membuat suatu fakta. Atribut tersebut kemudian dijelaskan secara ringkas pada bagian keterangan. Tabel 3.7 merupakan penguraian dari fakta

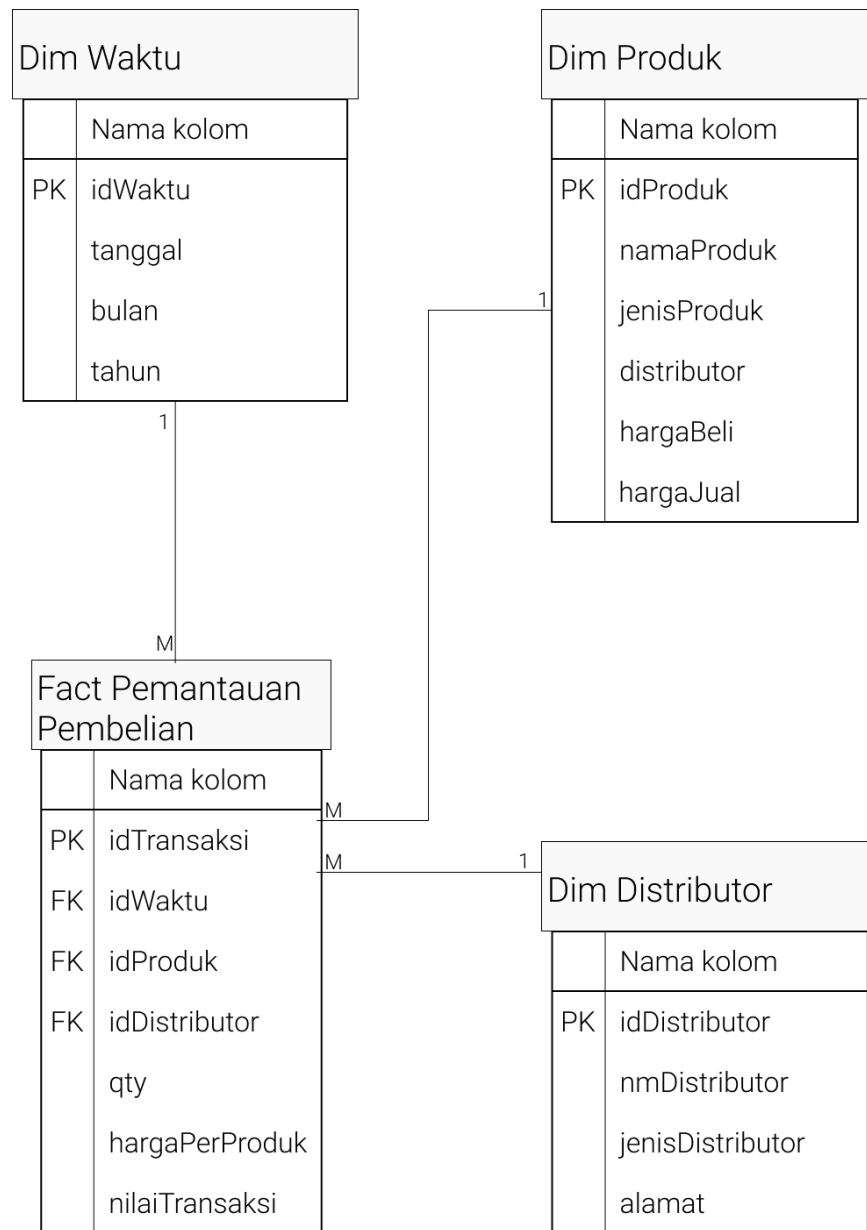
Tabel 3.7 Identifikasi Fakta

Fakta	Atribut	Keterangan
Pembelian	idTransaksi	Berupa kode identifier fakta pembelian
	idWaktu	Berupa kode identifier untuk mengakses dimensi waktu
	idProduk	Berupa kode identifier untuk mengakses dimensi produk
	idDistributor	Berupa kode identifier untuk mengakses dimensi distributor
	Qty	Berupa informasi mengenai kuantitas produk yang dibeli
	hargaPerProduk	Berupa informasi mengenai harga satuan produk yang dibeli
	nilaiTransaksi	Berupa informasi mengenai harga satuan produk yang dikalikan dengan kuantitas beli
Penjualan	idTransaksi	Berupa kode identifier fakta penjualan
	idWaktu	Berupa kode identifier untuk mengakses dimensi waktu
	idPelanggan	Berupa kode identifier untuk mengakses dimensi produk
	idProduk	Berupa kode identifier untuk mengakses dimensi distributor
	qty	Berupa informasi mengenai kuantitas produk yang dibeli

	nilaiTransaksi	Berupa informasi mengenai harga satuan produk yang dikalikan dengan kuantitas jual
--	----------------	--

A. Analisa Pemantauan Pembelian

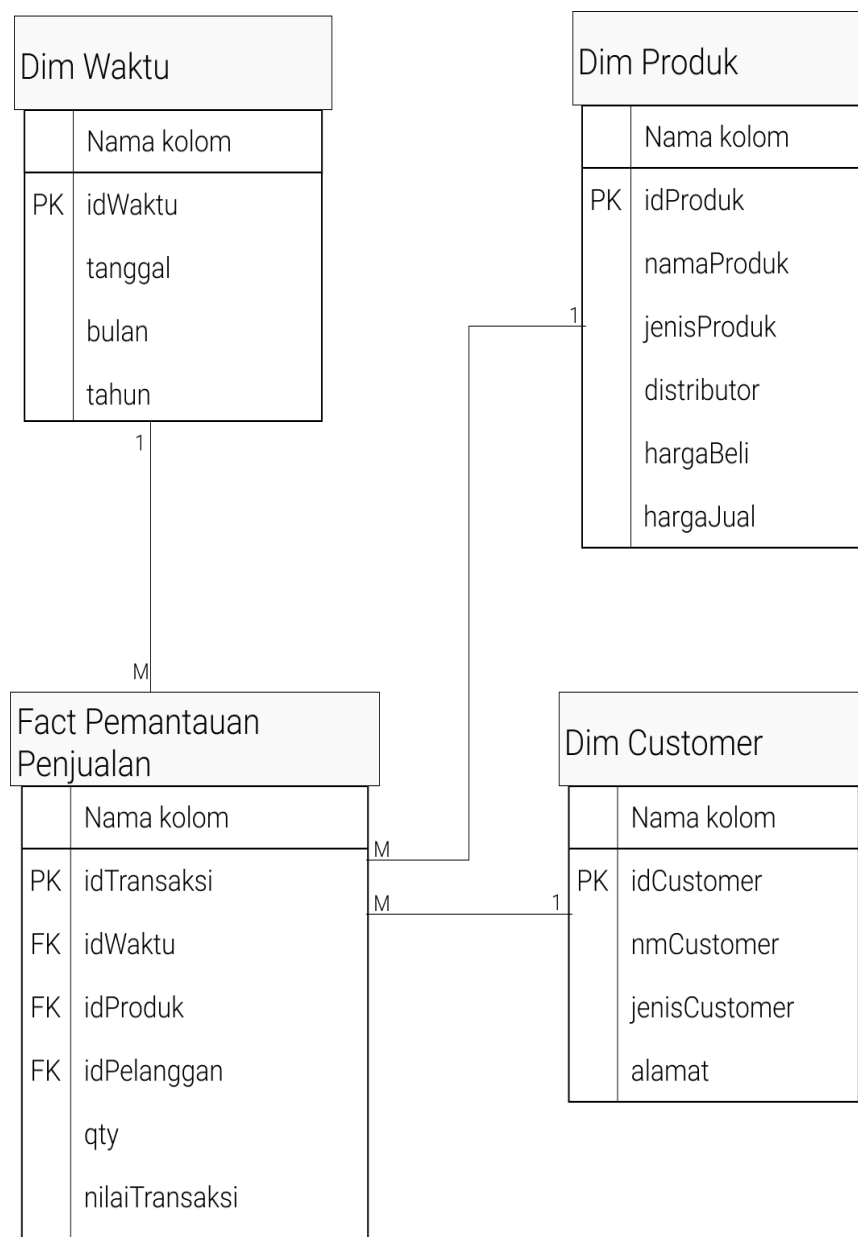
Pada bagian ini, dipaparkan hubungan antara dimensi yang dibutuhkan dalam membangun suatu fakta. Skema yang digunakan dalam pembuatan fakta ini adalah skema *star*. Gambar 3.5 merupakan ilustrasi dari Analisa pemantauan pembelian



Gambar 3.5 Diagram Analisa Pemantauan Pembelian

B. Analisa Pemantauan Penjualan

Pada bagian ini akan diilustrasikan hubungan antara dimensi dalam membangun fakta pemantauan penjualan, dimensi disesuaikan dengan kebutuhan informasi yang ingin dihasilkan. Gambar 3.6 merupakan ilustrasi dari Analisa pemantauan penjualan



Gambar 3.6 Diagram Analisa Pemantauan Penjualan

2. *Physical Design*

Rancangan yang terdapat di *dimensional modelling* lalu diterjemahkan dalam bentuk *physical design*. Fase berikut merupakan fase perancangan *database* dan data warehouse.

Tabel dibawah merupakan rancangan database yang digunakan pada pembangunan business intelligence

Tabel 3.8 Struktur *Collection users*

Nama Field	Tipe Data	Keterangan
userName	<i>String</i>	Nama <i>user</i>
firstName	<i>String</i>	Nama depan <i>user</i>
lastName	<i>String</i>	Nama belakang <i>user</i>
initials	<i>String</i>	Inisial <i>user</i>
email	<i>String</i>	Email <i>user</i>
password	<i>String</i>	Password <i>user</i>
role	<i>String</i>	Kedudukan <i>user</i> di perusahaan
joinedAt	<i>Timestamp</i>	Waktu pada saat <i>collection</i> dibuat

Tabel 3.8 diuraikan bagian dari *collection user* yang merupakan tabel yang berisi informasi pengguna *business intelligence* untuk mengakses *website application*.

Tabel 3.9 Struktur *Collection Distributors*

Nama Field	Tipe Data	Keterangan
distributorName	<i>String</i>	Nama distributor
address	<i>String</i>	Alamat Distributor
createdAt	<i>Timestamp</i>	Waktu pada saat distributor baru diinputkan kedalam <i>collection</i>

Nama Field	Tipe Data	Keterangan
createdBy	String	User yang membuat <i>collection</i>
updatedAt	Timestamp	Waktu pada saat <i>collection</i> diubah
updatedBy	String	User yang melakukan perubahan pada <i>collection</i>

Tabel 3.9 merupakan pemaparan isi dari *collection distributor*. Berisi informasi dari distributor atau supplier yang berhubungan dengan PD. Intan.

Tabel 3.10 Struktur *Collection Customers*

Nama Field	Tipe Data	Keterangan
customerName	String	Nama pelanggan
address	String	Alamat Pelanggan
createdAt	Timestamp	Waktu pada saat <i>collection</i> dibuat
createdBy	String	User yang membuat <i>collection</i>
updatedAt	Timestamp	Waktu pada saat <i>collection</i> diubah
updatedBy	String	User yang melakukan perubahan pada <i>collection</i>

Tabel 3.10 berisi kumpulan kolom yang terdapat dalam *collection customer*. *Collection* ini berisi informasi pelanggan yang membeli barang di PD. Intan

Tabel 3.11 Struktur *Collection Products*

Nama Field	Tipe Data	Keterangan
productName	String	Nama Produk
distributor	String	Distributor dari produk
sellPrice	Number	Harga jual produk
buyPrice	Number	Harga beli produk

Nama <i>Field</i>	Tipe Data	Keterangan
producer	<i>String</i>	Producer dari produk
createdAt	<i>Timestamp</i>	Waktu pada saat <i>collection</i> dibuat
createdBy	<i>String</i>	<i>User</i> yang membuat <i>collection</i>
updatedAt	<i>Timestamp</i>	Waktu pada saat <i>collection</i> diubah
updatedBy	<i>String</i>	<i>User</i> yang melakukan perubahan pada <i>collection</i>
stock	<i>Number</i>	Kuantitas persediaan produk

Tabel 3.11 Memaparkan isi dari *collection products*. *Collection* ini digunakan untuk menyimpan informasi barang yang terdapat di Gudang PD. Intan.

Tabel 3.12 Struktur *Collection Sales Transactions*

Nama <i>Field</i>	Tipe Data	Keterangan
buyList	<i>Array</i>	List yang berisi produk yang dibeli
buyList[i].productName	<i>String</i>	Nama dari produk yang dibeli
buyList[i].productId	<i>String</i>	Id dari produk yang dibeli
buyList[i].qty	<i>Number</i>	Kuantitas pembelian produk
buyList[i].totalSales	<i>Number</i>	Harga total pembelian produk berdasarkan banyaknya kuantitas
customerName	<i>String</i>	Nama dari pelanggan yang melakukan transaksi
salesName	<i>String</i>	Nama pegawai yang melayani pelanggan
status	<i>String</i>	Status dari pesanan pembelian

Nama <i>Field</i>	Tipe Data	Keterangan
subtotal	<i>Number</i>	Total pembelian dari seluruh transaksi
createdAt	<i>Timestamp</i>	Waktu pada saat <i>collection</i> dibuat
processedAt	<i>Timestamp</i>	Waktu pada saat <i>collection</i> diproses
processedBy	<i>Number</i>	<i>User</i> yang melakukan perubahan pada <i>collection</i>

Tabel 3.12 Merupakan penguraian dari *collection sales transaction*. Berisi informasi mengenai penjualan barang yang dilakukan oleh *sales* dengan pelanggan.

Tabel 3.13 Struktur *Collection Restocks*

Nama <i>Field</i>	Tipe Data	Keterangan
restockList	<i>Array</i>	List yang berisi produk yang akan di <i>restock</i>
restockList[i].productName	<i>String</i>	Nama dari produk yang akan di <i>restock</i>
restockList[i].productId	<i>String</i>	Produk id barang
restockList[i].qty	<i>Number</i>	Kuantitas <i>restock</i> barang
distributor	<i>String</i>	Distributor target <i>restock</i>
Status	<i>String</i>	Status pemesanan <i>restock</i>
createdBy	<i>String</i>	Waktu pada saat <i>collection restock</i> dibuat
createdAt	<i>Timestamp</i>	<i>User</i> yang membuat <i>collection</i>
processedAt	<i>Timestamp</i>	Waktu pada saat <i>collection restock</i> diproses
processedBy	<i>String</i>	<i>User</i> yang melakukan perubahan pada <i>collection</i>

Tabel 3.13 Berisi gabungan *field* yang membangun *collection restock*.
Collection ini digunakan menampung informasi *restock* yang dibutuhkan PD. Intan.

Tabel 3.14 Struktur *Collection Purchased Transactions*

Nama <i>Field</i>	Tipe Data	Keterangan
<code>purchasedList</code>	<i>Array</i>	List dari barang yang dibeli
<code>purchasedList[i].productName</code>	<i>String</i>	Nama dari produk
<code>purchasedList[i].productId</code>	<i>String</i>	Produk id barang
<code>purchasedList[i].qty</code>	<i>Number</i>	Kuantitas produk yang dibeli
<code>purchasedList[i].buyPrice</code>	<i>Number</i>	Harga beli dari produk yang dibeli
<code>purchasedList[i].sellPrice</code>	<i>Number</i>	Harga jual dari produk yang dibeli
<code>purchasedList[i].disc</code>	<i>Number</i>	Diskon yang didapat per produk
<code>purchasedList[i].total</code>	<i>Number</i>	Total pembelian produk berdasarkan kuantitas
<code>purchasedList[i].producer</code>	<i>String</i>	Nama producer per produk
<code>status</code>	<i>String</i>	Status pembelian barang
<code>Subtotal</code>	<i>Number</i>	Total pembelian produk keseluruhan
<code>isExistDist</code>	<i>Boolean</i>	Parameter untuk memastikan distributor ini adalah distributor yang baru
<code>distributor</code>	<i>String</i>	Nama distributor pemasok persediaan
<code>createdAt</code>	<i>Timestamp</i>	Waktu pada saat <i>collection</i> dibuat
<code>createdBy</code>	<i>String</i>	<i>User</i> yang membuat <i>collection</i>
<code>processedAt</code>	<i>Timestamp</i>	Waktu pada saat diproses
<code>processedBy</code>	<i>Number</i>	<i>User</i> yang melakukan perubahan

Tabel 3.14 Merupakan pemaparan dari *collection purchased transaction*.
Collection ini berisi informasi mengenai daftar barang yang dibeli oleh PD. Intan kepada distributor/supplier lain.

Tabel 3.15 Struktur Dimensi *Distributors*

Nama <i>Field</i>	Tipe Data	Keterangan
idDistributor	<i>String</i>	Data berisi kode unik sebagai <i>identifier</i> distributor, data ini merupakan <i>primary key</i> dari dimensi distributor
distributorName	<i>String</i>	Nama distributor
jenisDistributor	<i>String</i>	Jenis produk yang dijual oleh distributor
alamat	<i>String</i>	Tempat distributor berada

Tabel 3.15 Merupakan pemaparan dari dimensi distributor. Yang akan digunakan untuk pembangunan data warehouse dalam business intelligence

Tabel 3.16 Struktur Dimensi *Customers*

Nama <i>Field</i>	Tipe Data	Keterangan
idCustomer	<i>String</i>	Data berisi kode unik sebagai <i>identifier</i> pelanggan
customerName	<i>String</i>	Nama pelanggan
jenisCustomer	<i>String</i>	Jenis penjualan pelanggan
alamat	<i>String</i>	Tempat penjualan pelanggan

Tabel 3.16 merupakan struktur dimensi *customers* yang digunakan dalam pembangunan data warehouse dalam business intelligence

Tabel 3.17 Struktur Dimensi *Products*

Nama <i>Field</i>	Tipe Data	Keterangan
idProduct	<i>String</i>	Sebagai <i>identifier</i> pelanggan, data ini merupakan <i>primary key</i> dimensi <i>products</i>
productName	<i>String</i>	Berisi dengan nama produk

Nama <i>Field</i>	Tipe Data	Keterangan
productType	<i>String</i>	Berisi jenis produk
distributor	<i>String</i>	Nama Produk
buyPrice	<i>Number</i>	Harga beli dari suatu produk
sellPrice	<i>Number</i>	Harga jual dari produk

Tabel 3.17 Merupakan pemaparan struktur tabel dimensi produk, dimana berisi informasi mengenai produk yang terdapat di PD. Intan.

Tabel 3.18 Struktur Tabel Dimensi Waktu

Nama <i>Field</i>	Tipe Data	Keterangan
idTanggal	<i>String</i>	Berisi kode unik gabungan dari tanggal, bulan dan tahun sebagai primary key dari dimensi waktu
tanggal	<i>String</i>	Berisi informasi mengenai tanggal
bulan	<i>String</i>	Berisi informasi mengenai bulan
tahun	<i>String</i>	Berisi informasi mengenai tahun

Tabel 3.18 Adalah penjelasan dan pemaparan dari tabel dimensi waktu yang akan digunakan di data warehouse dalam pembangunan business intelligence. Informasi yang terdapat dalam tabel ini adalah kumpulan tanggal terjadinya transaksi penjualan maupun pembelian di PD. Intan. Tabel ini juga terhubung dengan fakta Analisa pembelian maupun penjualan.

Tabel 3.19 Struktur Tabel Fakta Pemantauan Penjualan

Nama Field	Tipe Data	Keterangan
idTransaksi	String	Data berisi kode unik sebagai <i>identifier</i> pelanggan
idWaktu	String	Berisi kode unik sebagai <i>identifier</i> waktu
idProduk	String	Berisi kode unik sebagai <i>identifier</i> produk
idPelanggan	String	Berisi kode unik sebagai <i>identifier</i> pelanggan
Qty	Number	Berisi kuantitas dari produk berdasarkan transaksi
nilaiTransaksi	Number	Berisi nilai transaksi berdasarkan harga jual dan kuantitas barang

Tabel 3.19 merupakan pemaparan dari struktur tabel fakta pemantauan penjualan. Tabel ini akan digunakan untuk proses OLAP yang kemudian akan ditampilkan dalam bentuk grafik.

Tabel 3.20 Struktur Tabel Fakta Pemantauan Pembelian

Nama Field	Tipe Data	Keterangan
idTransaksi	String	Data berisi kode unik <i>identifier</i> pelanggan
idWaktu	String	Berisi kode unik <i>identifier</i> dimensi waktu
idProduk	String	Berisi kode unik <i>identifier</i> dimensi produk
idDistributor	String	Berisi kode unik sebagai <i>identifier</i> pelanggan, data ini merupakan <i>foreign key</i> yang menghubungkan tabel fakta dengan dimensi <i>distributor</i>
Qty	Number	Berisi kuantitas dari produk yang dibeli berdasarkan transaksi
hargaPerProduk	Number	Berisi nilai total harga berdasarkan harga beli dan kuantitas barang
nilaiTransaksi	Number	Berisi nilai dari jumlah harga keseluruhan pembelian dari suatu transaksi

Tabel 3.20 adalah uraian dari tabel fakta pemantauan pembelian. Tabel ini berisi data pembelian yang dilakukan oleh PD. Intan, namun telah mengalami proses ETL.

3. *ETL Design and Development*

Pada tahap ini seluruh sumber data yang terkait tentang topik akan dikumpulkan dalam satu *data warehouse* yang akan dilakukan standarisasi data. Tahap ini dilakukan dimulai dari pengumpulan seluruh data mentah, melakukan perubahan terhadap data yang dikumpulkan, penyajian data, dan pengelolaan sistem. Pada tahap ini terdapat 4 proses penting, yaitu:

1. Ekstrak Data

Proses ini merupakan proses pengumpulan seluruh data mentah yang berhubungan dengan tujuan pembangunan *business intelligence* dari sumber data yang berbeda.

2. Pembersihan Data

Pada tahap ini sumber data akan dikirimkan dan di proses kembali oleh sistem ETL, hal tersebut dilakukan untuk meningkatkan kualitas data yang diterima dari sumber. Data yang telah di proses kemudian digabungkan dengan data dari sumber yang lain untuk memperoleh kualitas dimensi yang lebih berkualitas.

3. Penyajian Sistem

Proses yang dilakukan pada tahap ini antara lain menyusun dan memuat data yang telah mengalami proses ETL ke dalam model dimensi yang akan disajikan pada server presentasi.

4. Pengelolaan Sistem

Tahap ini merupakan tahap untuk mengelola sistem dan proses yang terkait lingkungan ETL.

3.2.6 *Business Intelligence Application Track*

Jalur ini berfokus pada penguraian mengenai aplikasi *Business intelligence*. Pada jalur ini akan membahas *Business Intelligence Application Design* dan *Business Intelligence Application Development*.

1. *Business Intelligence Application Design*

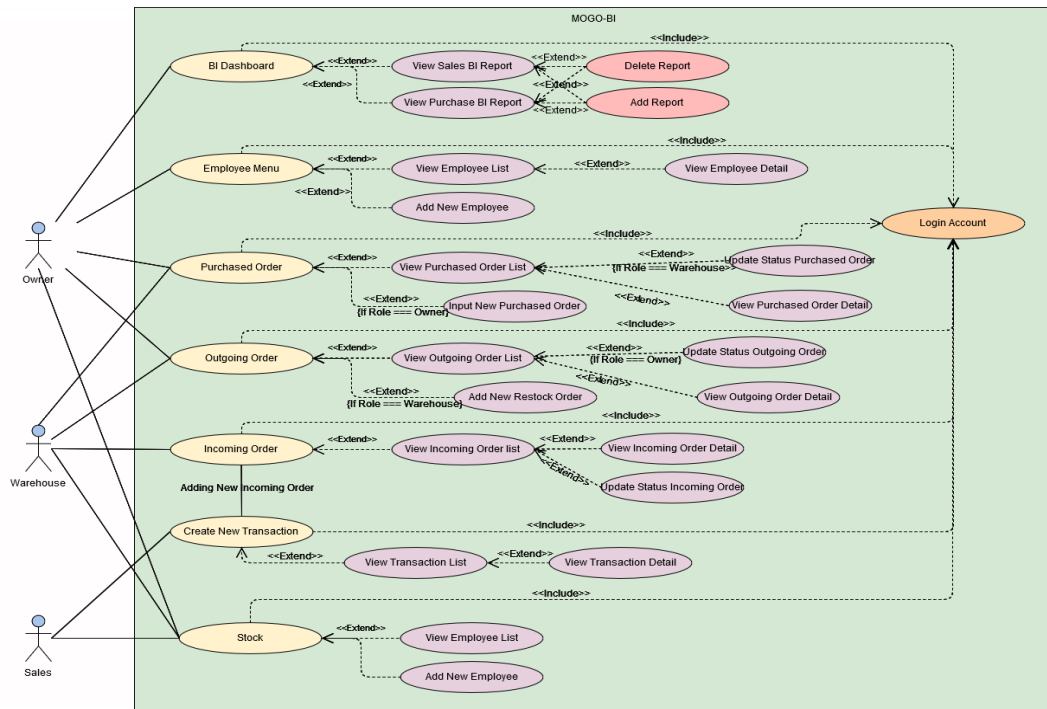
Tahap ini melakukan perancangan dan pemilihan aplikasi atau *platform* yang akan digunakan dalam mendukung kebutuhan bisnis. Aplikasi atau *platform* tersebut merupakan penghubung yang menampilkan *business value* dari solusi yang diberikan *Business Intelligence*.

Pada tahap ini penulis memilih platform NodeJs dengan menggunakan *library* ReactJs sebagai antarmuka untuk menampilkan hasil dari data yang telah diolah, selain itu dibuat pula sistem Informasi OLTP sebagai penunjang memasukkan data kedalam *business intelligence*.

Sistem Informasi OLTP yang terdapat pada *business intelligence* adalah fungsi memasukkan produk baru kedalam sistem, melakukan transaksi pembelian barang kepada *supplier* maupun penjualan barang kepada *customer*, melakukan pemantauan persediaan produk pada PD. Intan, membuat *restock order*, memasukkan *purchased order* dan fungsi – fungsi lain untuk memantau transaksi yang terjadi di PD. Intan.

Dalam perancangan *website application* fungsi dari *business intelligence* hanya dapat digunakan oleh *top-level-management* atau pengguna yang memiliki *role* “owner”, namun untuk kasus *inventory* seluruh pengguna *website application* dapat mengakses laman tersebut.

Gambar 3.7 merupakan rancangan UML yang diterapkan pada sistem.



Gambar 3.7 Use Case Diagram

Sistem yang dibangun memiliki 3 *role* yang terdiri dari *Owner*, *Warehouse*, *Sales* berdasarkan bidang yang terdapat di PD.Intan. *Role Sales* berfungsi sebagai pelaku dalam melayani *customer* yang akan berbelanja, Pesanan dari *customer* akan dimasukkan oleh *sales* dan pesanan tersebut akan diproses oleh pihak *warehouse*.

Role Warehouse berfungsi sebagai pelaku dalam mempersiapkan pesanan yang dikirim oleh *sales*, *Warehouse* juga berfungsi sebagai pelaku yang membuat *restock order* yang kemudian akan di pesan oleh *owner* kepada *supplier* produk.

Role Owner memiliki akses dalam memantau penjualan, pembelian dan persediaan yang terdapat di perusahaan, memproses *restock* yang dibuat oleh *warehouse*, dan memasukkan barang yang dikirim oleh *supplier* dalam bagian *purchased order* yang kemudian akan diproses oleh *warehouse*.

A. Perancangan *User Interface*

Perancangan *user interface* sangat penting dalam mengembangkan sebuah *website application*. Dalam perancangan *user interface*, diharapkan *user interface* yang akan dikembangkan dapat dibuat dengan efektif. Pada subbab ini akan dirancang beberapa rancangan *user interface* untuk digunakan sebagai *interface business intelligence*. Berikut beberapa rancangan *user interface* pada *business intelligence*.

Rancangan yang ditampilkan diantaranya adalah rancangan *Dashboard Purchasing*. Halaman ini hanya dapat diakses oleh akun yang memiliki *role owner*. Konten dari halaman ini diantaranya informasi pengeluaran yang diperoleh PD. Intan, Informasi pembelian produk dan *supplier* yang dipilih. Selain itu terdapat grafik yang berisi informasi tentang pengeluaran yang dilakukan oleh perusahaan.

Rancangan berikutnya merupakan rancangan *Dashboard Sales* berisi informasi dari seluruh pendapatan yang diperoleh perusahaan, Terdapat informasi keuntungan yang didapat oleh perusahaan, selain itu dimunculkan grafik yang berisi tren produk yang terjual di perusahaan.

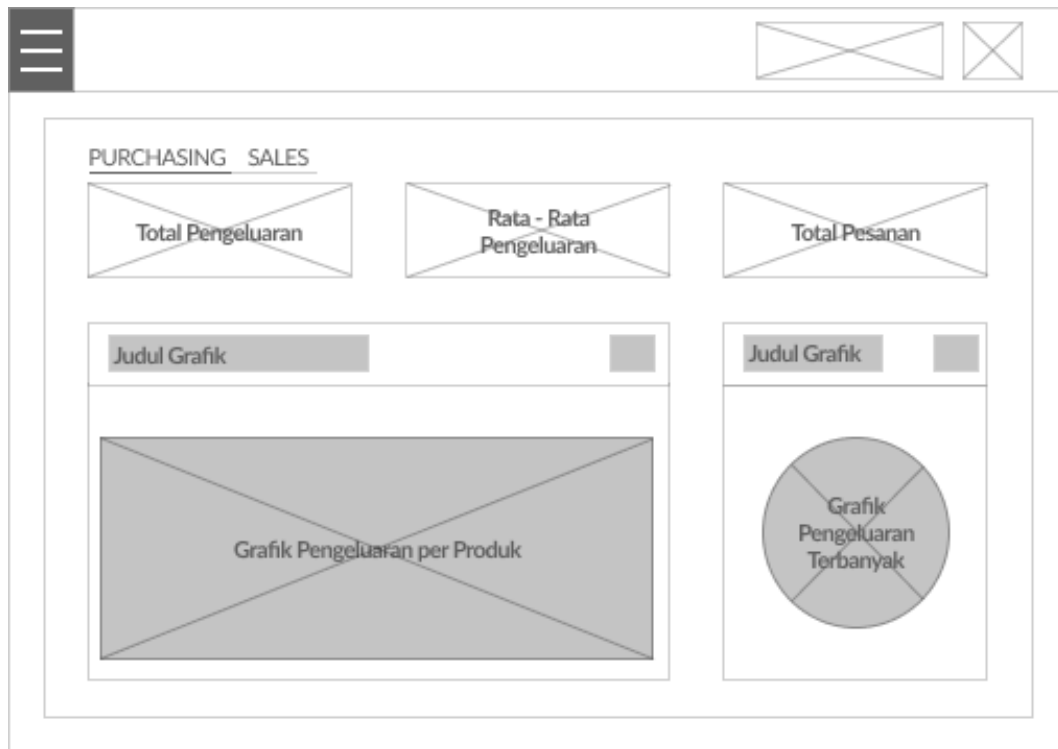
Rancangan *Dashboard Inventory* dapat diakses oleh seluruh akun, halaman ini berisi informasi mengenai produk yang dimiliki perusahaan.

Rancangan Input new employee merupakan halaman yang hanya dapat diakses oleh owner, halaman ini berfungsi untuk memasukkan akun baru untuk anggota baru.

Rancangan login berfungsi sebagai halaman utama sebelum melakukan seluruh aktivitas pada sistem.

- *Dashboard Purchasing*

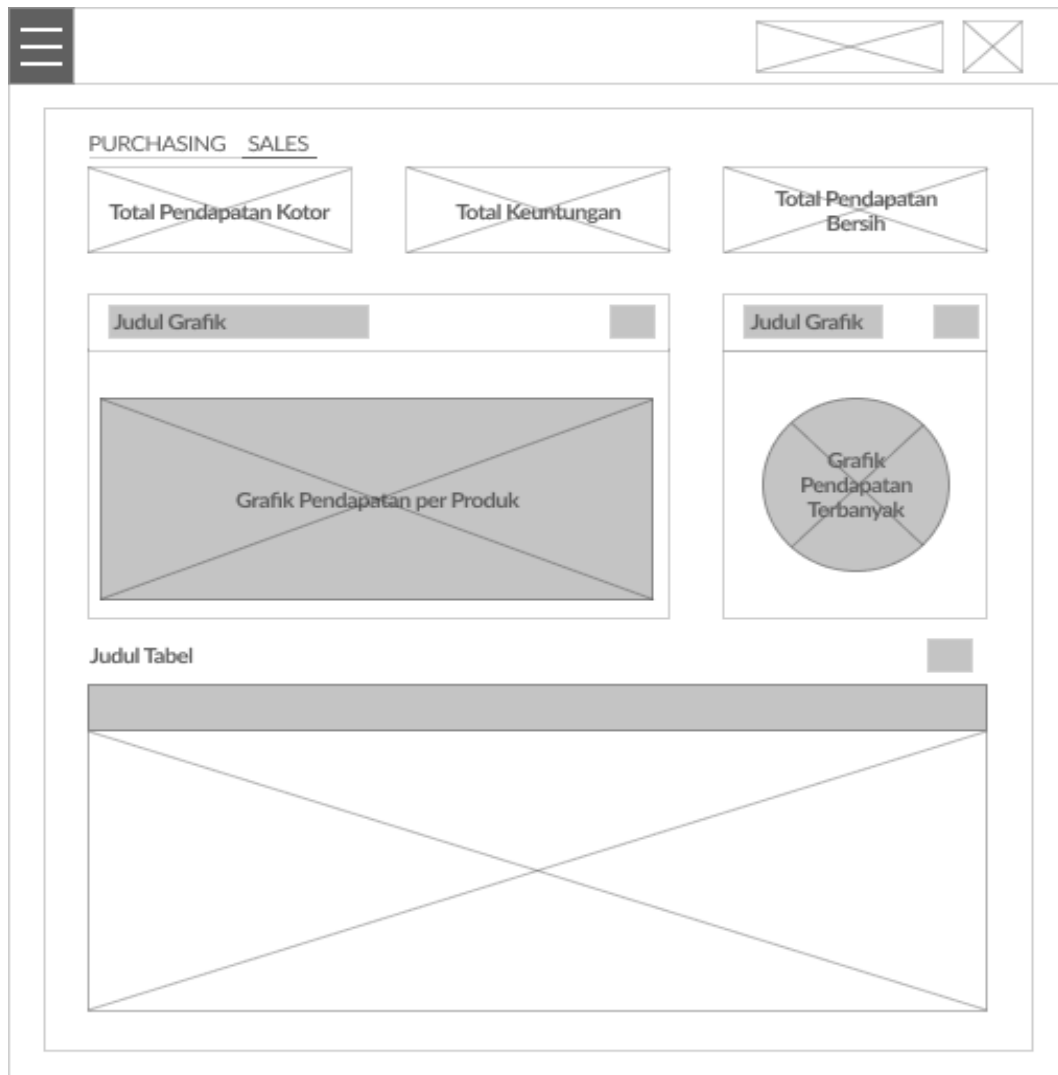
Berikut merupakan *Dashboard Purchasing* yang digunakan untuk menampilkan report mengenai pembelian barang, informasi yang ditampilkan merupakan uraian dari Analisa pemantauan pembelian.



Gambar 3.8 Rancangan Interface Dashboard Purchasing

- *Dashboard Sales*

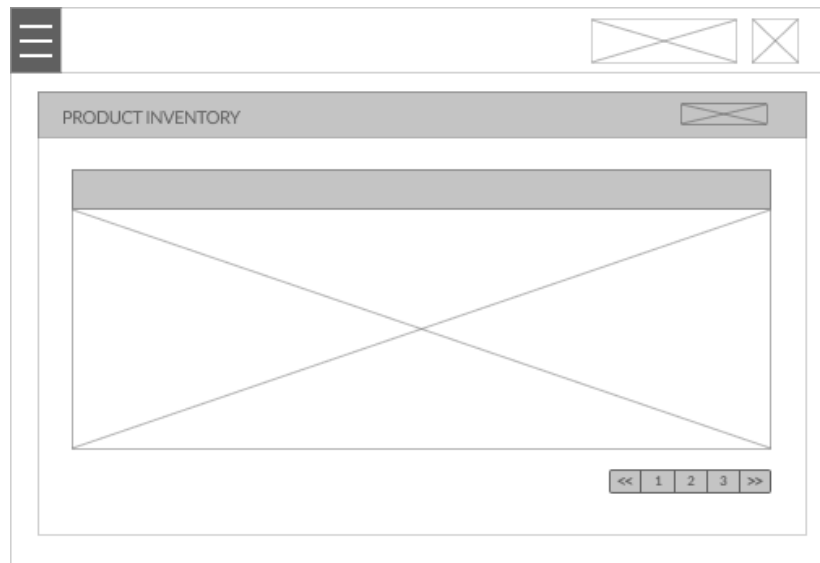
Dashboard Sales berfungsi untuk menampilkan report mengenai penjualan dan tren suatu produk, informasi yang ditampilkan merupakan uraian dari Analisa pemantauan penjualan. Halaman ini hanya dapat diakses oleh akun yang memiliki role *owner* karena berisi informasi yang berguna untuk owner dalam mengambil keputusan untuk melakukan *restock* barang.



Gambar 3.9 Rancangan Interface Dashboard Sales

- *Dashboard Inventory*

Dibawah ini merupakan rancangan *user interface dashboard inventory*, pada halaman ini akan diuraikan keseluruhan produk yang dimiliki oleh perusahaan, selain itu untuk *user* yang berstatus *owner* memiliki hak akses untuk melakukan *create, read, update, delete (CRUD)* dalam *website application* dan terhubung dengan *database* yang berisi data dari keseluruhan produk yang dimiliki.



Gambar 3.10 Rancangan Interface Dashboard Inventory

- *Input new employee*

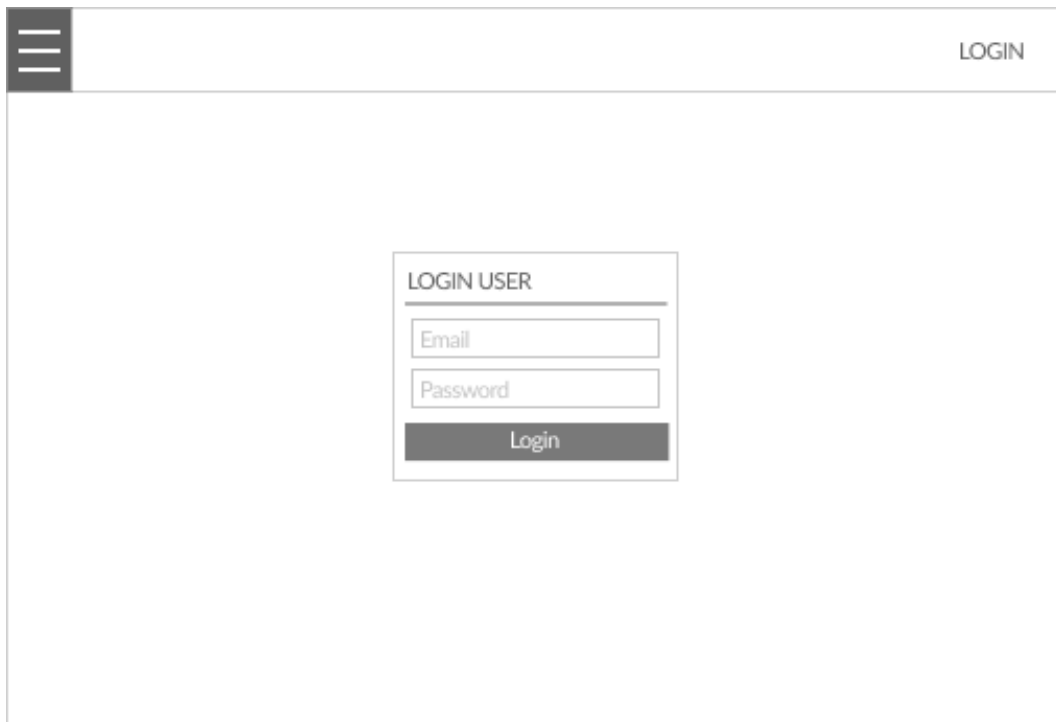
Halaman *input new employee* merupakan halaman yang hanya dapat diakses oleh pengguna berstatus *owner*. Fungsi dari halaman ini yaitu mendaftarkan / memasukkan pegawai baru kedalam *database* sehingga pengguna baru dapat mengakses *website application*.

 A wireframe of a form titled "Input Pegawai Baru". The form contains several input fields: "Email", "Nama Depan", "Nama Belakang", "Divisi" (with a dropdown arrow), "Password", and "Konfirmasi Password". At the bottom of the form are two buttons: "Submit" and "Cancel". The form is enclosed in a box with a header bar and window control icons at the top.

Gambar 3.11 Rancangan Interface Input New Employee

- *Login*

Halaman *Login* digunakan untuk melakukan proses *login* atau masuk ke dalam sistem dengan menggunakan email dan password.



Gambar 3.12 Rancangan Interface Login

2. *Business Intelligence Application Development*

Pada tahap ini dilakukan konstruksi serta validasi untuk membangun *Business Intelligence*.

3.2.7 *Deployment, Maintenance, Growth*

Setelah terbentuknya *business intelligence* perlu adanya lanjutan proses yang menunjang business intelligence agar tetap digunakan dan terus berkembang sesuai dengan kebutuhan bisnis dari suatu perusahaan.

A. Deployment

Tahap ini dilakukan peluncuran aplikasi Business Intelligence selain itu, pada tahap ini juga dilakukan validasi data, dokumentasi pembangunan Business Intelligence, dan memberikan pelatihan untuk *user*.

Business intelligence website application akan diluncurkan menggunakan jasa hosting firebase, dan dapat diakses pada www.mogo-bi.firebaseio.com

B. Maintenance

Setelah sistem *Business Intelligence* di produksi, diperlukan suatu teknis operasional untuk menjaga agar sistem bekerja secara optimal, operasi teknis tersebut antara lain adalah pemantauan, penggunaan, penyempurnaan kinerja, pemeliharaan indeks, dan cadangan sistem.

C. Growth

Growth merupakan tahap yang dilakukan untuk meningkatkan kinerja atau fungsi dari Business Intelligence mendatang.

BAB IV

HASIL PENELITIAN DAN PEMBAHASAN

4.1 Implementasi Business Intelligence

Pada tahap ini dilakukan realisasi pembangunan dari hasil analisis dan perancangan. Tahap implementasi Kimball *life cycle* pada *business intelligence* di PD. Intan meliputi implementasi data, dan implementasi *interface*.

4.1.1 Implementasi Data

Seluruh *query* yang digunakan pada saat pembangunan *database* maupun *data warehouse* yang menunjang pembangunan *business intelligence* akan dijelaskan pada bagian ini.

Untuk pembuatan *database* dan *data warehouse* menggunakan firebase, diperlukan konfigurasi *console* yang terdapat pada website firebase. Konfigurasi yang dilakukan antara lain:

- Membuat proyek firebase
- Mengaktifkan *cloud firestore*
- Menghubungkan firebase dengan *website application* yang dibangun

Keseluruhan pembuatan *database* dan *data warehouse* dapat dilihat di lampuran. Berikut daftar data warehouse yang dibuat pada firebase

Tabel 4.1 Implementasi *data warehouse*

Nama <i>Collection</i>	Deskripsi
dimTime	Berisi kumpulan data mengenai waktu transaksi

Nama <i>Collection</i>	Deskripsi
dimEmployee	Berisi kumpulan data mengenai petugas
dimCustomer	Berisi kumpulan data mengenai pelanggan
dimDistributor	Berisi kumpulan data mengenai distributor
dimProduct	Berisi kumpulan data mengenai produk
factPurchased	Berisi kumpulan data mengenai pembelian
factSales	Berisi kumpulan data mengenai penjualan

4.1.2 Implementasi *Interface*

Implementasi *interface* merupakan proses pembangunan antarmuka *business intelligence* yang digunakan untuk memudahkan *input/output* data serta menampilkan isi dari data warehouse. Tabel 4.2 merupakan tabel daftar *interface* yang dibuat. Gambar 4.1 – 4.10 merupakan tampilan *interface* yang dibuat

Tabel 4.2 Implementasi *Interface*

Menu	Deskripsi	Nama File
<i>Login</i>	Digunakan untuk menangani proses <i>login</i> , halaman ini adalah halaman utama untuk masuk kedalam <i>business intelligence</i>	SignIn.js
<i>Purchase Dashboard</i>	Digunakan untuk menampilkan informasi yang telah diproses terkait pembelian	Purchasing.js
<i>Sales Dashboard</i>	Digunakan untuk menampilkan informasi yang telah diproses terkait penjualan	Sales.js

Menu	Deskripsi	Nama File
<i>Product List</i>	Digunakan untuk menampilkan informasi mengenai seluruh produk yang ada di perusahaan	ProductList.js
<i>Register</i>	Digunakan untuk menambahkan pengguna baru kedalam database <i>website application</i>	SignUp.js

- **Login**

```
export const signIn = credentials => { //fungsi sign in
  return (dispatch, getState, { getFirebase }) => {
    const firebase = getFirebase();
    firebase
      .auth()
      .signInWithEmailAndPassword(credentials.email,
credentials.password)
      .then(() => {
        dispatch({ type: "LOGIN_SUCCESS" });
      })
      .catch(err => {
        dispatch({ type: "LOGIN_ERROR", err });
      });
  };
};
```

The image shows a login form with a light gray border. At the top, the word 'Login' is centered. Below it, there are two input fields: the first is labeled 'Email' with a person icon, and the second is labeled 'Password' with a lock icon. Below these fields is a prominent blue button with the text 'Log in' in white. At the bottom of the form, there is a link that says 'Or register now!' in blue text.

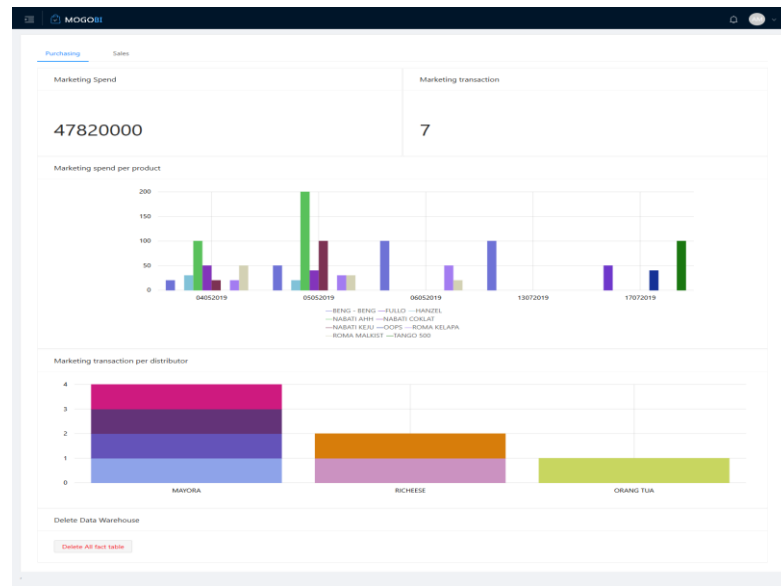
Gambar 4.1 Tampilan Login

Bagian ini merupakan tampilan awal ketika pengguna mengakses *website application*. Terdapat 2 kotak isian untuk mengisi email dan password. ketika tombol *log in* di klik maka fungsi akan di panggil. Jika email dan password terdaftar. Maka pengguna dapat mengakses *website application*.

- **Purchase Dashboard**

```
export const etlFactPurchasing = data => {
  return (dispatch, getState, { getFirestore, getFirestore }) => {
    const firestore = getFirestore();
    firestore
      .collection("factPurchased")
      .add({
        transactionId: data.id,
        timeId: data.timeId,
        productId: data.productId,
        distId: data.distId,
        qty: data.qty,
        pricePerProduct: data.pricePerProduct,
        subtotal: data.subTotal
      })
      .then(dispatch({ type: "ADD_NEW_FACT_PURCHASED" }))
      .catch(err => {
        dispatch({ type: "ADD_NEW_FACT_PURCHASED_ERROR", err });
      });
  };
};
```

Fungsi diatas merupakan fungsi untuk melakukan ETL terhadap tabel fakta pembelian. TimeId, productId, distId merupakan variabel yang didapat dari identifier masing – masing dimensi yang terhubung dengan fakta pembelian. Setelah data diambil. Lalu data akan disimpan kedalam *collection fact purchased* yang kemudian akan diolah kembali sebelum ditampilkan dalam bentuk grafik. Grafik yang ditampilkan merupakan keseluruhan data dari transaksi pembelian barang yang dilakukan oleh PD. Intan.



Gambar 4.2 Tampilan Dashoard Purchasing

Gambar diatas merupakan hasil dari pembuatan tampilan *Dashboard Purchasing*, terdapat fungsi *delete fact table* untuk melakukan penghapusan seluruh data dalam fakta pembelian.

- **Sales Dashboard**

```
export const etlFactSales = data => {
  return (dispatch, getState, { getFirestore, getFirestore }) => {
    const firestore = getFirestore();
    firestore
      .collection("factSales")
      .add({
        transactionId: data.id,
        timeId: data.timeId,
        productId: data.productId,
        qty: data.qty,
        subtotal: data.subTotal
      })
      .then(dispatch({ type: "ADD_NEW_FACT_SALES" }))
      .catch(err => {
        dispatch({ type: "ADD_NEW_FACT_SALES_ERROR", err });
      });
  };
};
```

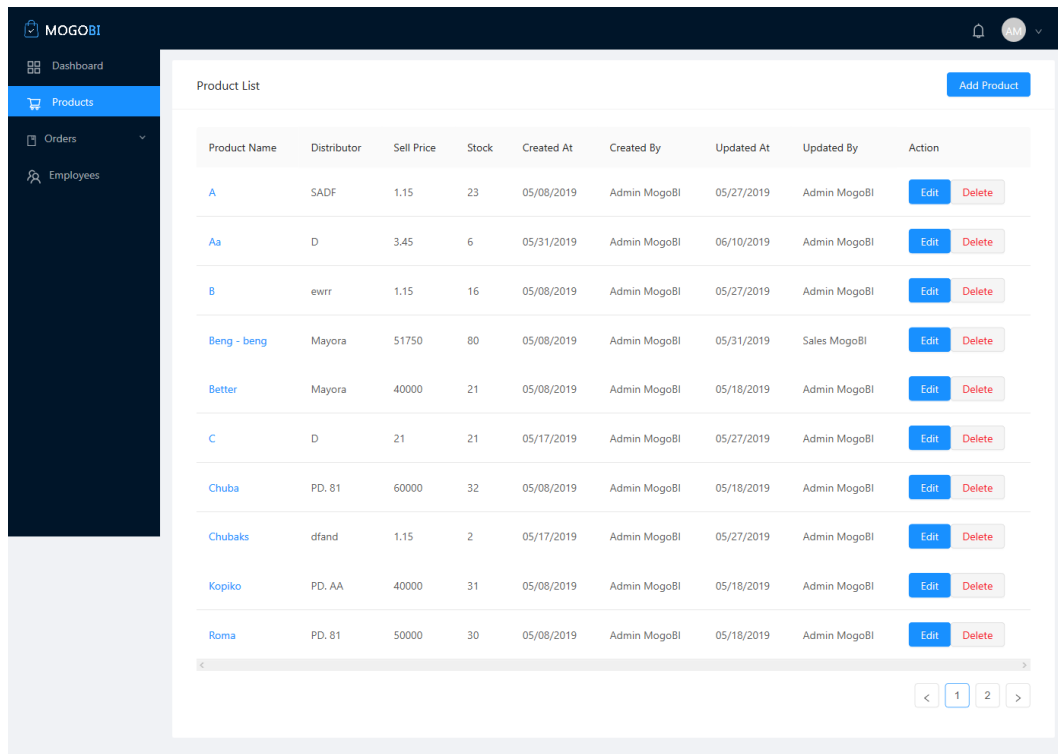


Gambar 4.3 Tampilan Dashboard Sales

Gambar 4.3 merupakan hasil dari pembuatan tampilan *dashboard sales*, berisi 1 grafik yang menampilkan hasil dari pendapatan produk dari keseluruhan penjualan.

- **Product List**

```
export const getProducts = () => {
  return (dispatch, getState, { getFirestore, getFirestore }) => {
    const firestore = getFirestore();
    firestore.collection("products").get()
      .then(querySnapshot => {
        let list = [];
        querySnapshot.forEach(doc => {
          list.push(doc.data());
        });
        dispatch({ type: "GET_PRODUCTS", data: list });
      })
      .catch(err => {
        dispatch({ type: "GET_PRODUCTS_ERROR", err });
      });
  };
};
```

Product List

Add Product

Product Name	Distributor	Sell Price	Stock	Created At	Created By	Updated At	Updated By	Action
A	SADF	1.15	23	05/08/2019	Admin MogoBI	05/27/2019	Admin MogoBI	Edit Delete
Aa	D	3.45	6	05/31/2019	Admin MogoBI	06/10/2019	Admin MogoBI	Edit Delete
B	ewrr	1.15	16	05/08/2019	Admin MogoBI	05/27/2019	Admin MogoBI	Edit Delete
Beng - beng	Mayora	51750	80	05/08/2019	Admin MogoBI	05/31/2019	Sales MogoBI	Edit Delete
Better	Mayora	40000	21	05/08/2019	Admin MogoBI	05/18/2019	Admin MogoBI	Edit Delete
C	D	21	21	05/17/2019	Admin MogoBI	05/27/2019	Admin MogoBI	Edit Delete
Chuba	PD. 81	60000	32	05/08/2019	Admin MogoBI	05/18/2019	Admin MogoBI	Edit Delete
Chubaks	dfand	1.15	2	05/17/2019	Admin MogoBI	05/27/2019	Admin MogoBI	Edit Delete
Kopiko	PD. AA	40000	31	05/08/2019	Admin MogoBI	05/18/2019	Admin MogoBI	Edit Delete
Roma	PD. 81	50000	30	05/08/2019	Admin MogoBI	05/18/2019	Admin MogoBI	Edit Delete

< 1 2 >

Gambar 4.4 Tampilan Product List

Gambar 4.4 merupakan hasil pengimplementasian desain *product list*. Berisi kumpulan informasi mengenai produk yang terdapat di PD. Intan. Pada awal *rendering* halaman. Dilakukan pemanggilan fungsi *getProducts* yang berfungsi untuk mengambil seluruh data yang terdapat di dalam database. Kemudian akan ditampilkan dalam bentuk grafik. Untuk pengguna yang memiliki jabatan sebagai *owner*, maka pengguna tersebut dapat melakukan fungsi *Create*, *Read*, *Update*, *Delete*. Dimana pengguna bisa melakukan manipulasi data untuk merubah maupun menghapus suatu data jika mengalami kesalahan. Pengguna pun dapat menambahkan produk baru kedalam database. untuk jabatan lain, hanya dapat melakukan akses membaca data.

4.2 Pengujian Business Intelligence

Pengujian dilakukan untuk memeriksa kinerja *business intelligence* yang telah dibangun untuk diimplementasikan di PD. Intan. Tujuan utama dari pengujian adalah memastikan komponen dalam *business intelligence* berfungsi dengan benar. Pengujian yang dilakukan yaitu pengujian sistem oleh *developer* dan pengujian sistem oleh *user* dengan strategi *Beta Testing*.

4.2.1 Rencana Pengujian Developer

Pada tahap ini dibuat perencanaan pengujian yang dilakukan oleh *developer* dengan menggunakan teknik pengujian *black box*. Pengujian *black box* merupakan pengujian fungsional yang melihat hasil berdasarkan *input* dan *output*. Berikut merupakan rincian rencana pengujian *black box* serta metode yang digunakan.

Tabel 4.3 Rencana Pengujian Black Box pada interface

Menu	Pengujian	Metode
<i>Login</i>	Skenario Normal	<i>Sample Testing</i>
	Skenario Alternatif	
<i>Purchase Dashboard</i>	Skenario Normal	<i>Behaviour Testing</i>
<i>Sales Dashboard</i>	Skenario Normal	<i>Behaviour Testing</i>
<i>Product List</i>	Skenario Normal	<i>Behaviour Testing</i>
<i>Register</i>	Skenario Normal	<i>Sample Testing</i>
	Skenario Alternatif	

Berikut merupakan penjelasan mengenai proses – proses pengujian yang ada pada Tabel 4.3

Tabel 4.4 Skenario Pengujian *Black Box* pada *interface*

Menu	Pola Pengujian	Pengujian
<i>Login</i>	Mengisi <i>input box email</i> dan <i>password</i>	Skenario Normal Pengguna memasukkan <i>email</i> dan <i>password</i> yang benar
		Skenario Alternatif Pengguna memasukkan <i>email</i> dan <i>password</i> yang salah
<i>Purchase Dashboard</i>	Melihat tampilan <i>purchase dashboard</i>	Skenario Normal Sistem menampilkan halaman <i>purchase dashboard</i>
<i>Sales Dashboard</i>	Melihat tampilan <i>sales dashboard</i>	Skenario Normal Sistem menampilkan <i>sales dashboard</i>
<i>Product List</i>	Melihat tampilan <i>product list</i>	Skenario Normal Sistem menampilkan halaman <i>product list</i>
<i>Register</i>	Mengisi <i>input box</i>	Skenario Normal Pengguna memasukkan data sesuai dengan tipe data yang diminta
		Skenario Alternatif Pengguna memasukkan data tidak sesuai dengan tipe data yang diminta

4.2.2 Rencana Pengujian Beta

Rencana pengujian beta merupakan rencana pengujian sistem yang dinilai oleh pengguna PD. Intan dengan menggunakan strategi pengujian beta. Hal ini dilakukan untuk mendapatkan *feedback* penggunaan *website application*. Daftar pertanyaan yang dipersiapkan untuk wawancara yang akan dilakukan kepada pemilik perusahaan dan perwakilan pegawai dari setiap divisi ditampilkan pada Table 4.4

Tabel 4.5 Pertanyaan Pengujian *Website Application*

No.	Pertanyaan	Jawaban		
		Ya	Tidak	Komentar
1.	Apakah <i>Website Application</i> yang diimplementasikan bermanfaat untuk Anda?			
2.	Apakah tampilan yang ditawarkan dalam <i>website application</i> menarik dan mudah dimengerti?			
3.	Apakah kebutuhan informasi di PD. Intan sudah terlihat dalam sistem <i>business intelligence</i> ini?			
4.	Apakah perlu ada fitur tambahan untuk pengembangannya ke depannya?			
5.	Apakah ada kritik dan saran untuk pengembangan <i>business intelligence</i> ke depannya?			

Tabel 4.6 Rencana pengujian informasi strategis

No.	Kebutuhan Informasi Strategis	Perbandingan		Kesimpulan
		Hasil yang diinginkan	Hasil yang ditampilkan	
1.	Informasi transaksi yang dilakukan dalam suatu periode			
2.	Informasi jumlah pembelian yang dilakukan terhadap suatu distributor dalam periode tertentu			
3.	Informasi persediaan barang yang tersedia dalam Gudang			
4.	Informasi mengenai profit dan pendapatan yang didapat dalam periode tertentu			
5.	Informasi nilai penjualan per produk			

Keterangan penilaian pada tabel 4.6 dapat dilihat pada tabel 4.7

Tabel 4.7 Validasi Hasil Pengujian

Penilaian Ke-	Hasil yang ditampilkan	Kesimpulan
1	[√] Value sesuai dengan hasil yang diinginkan [] Tampilan sesuai dengan hasil yang diinginkan	Memadai
2	[√] Value sesuai dengan hasil yang diinginkan [√] Tampilan sesuai dengan hasil yang diinginkan	Terpenuhi
3	[] Value sesuai dengan hasil yang diinginkan [] Tampilan sesuai dengan hasil yang diinginkan	Tidak Memadai

4.2.3 Hasil Pengujian Developer

Berdasarkan rencana dan skenario pengujian yang telah dibuat, hasil pengujian *black box* pada *interface*

Tabel 4.8 Hasil Pengujian *Developer*

Menu	Pengujian	Hasil Pengujian	Keterangan
<i>Login</i>	Skenario Normal Pengguna memasukkan <i>email</i> dan <i>password</i> yang benar	Pengguna masuk kedalam sistem	[√] Diterima [] Ditolak
	Skenario Alternatif Pengguna memasukkan <i>email</i> dan <i>password</i> yang salah	Pengguna tidak dapat masuk kedalam sistem	
<i>Purchase Dashboard</i>	Skenario Normal Sistem menampilkan <i>purchase dashboard</i>	Informasi berhasil di tampilkan	[√] Diterima [] Ditolak
<i>Sales Dashboard</i>	Skenario Normal Sistem menampilkan <i>sales dashboard</i>	Informasi berhasil di tampilkan	[√] Diterima [] Ditolak
<i>Product List</i>	Skenario Normal Sistem menampilkan <i>product list</i>	Informasi berhasil di tampilkan	[√] Diterima [] Ditolak
<i>Register</i>	Skenario Normal Pengguna memasukkan data sesuai dengan tipe data yang diminta	Pengguna dapat menambahkan <i>user</i> baru	[√] Diterima [] Ditolak
	Skenario Alternatif Pengguna memasukkan data tidak sesuai dengan tipe data yang diminta	Pengguna tidak dapat menambahkan <i>user</i> baru	

4.2.4 Hasil Pengujian Beta

Berdasarkan rencana pada pengujian beta yang telah dibuat, hasil pengujian beta dapat dilihat pada tabel 4.9 dan tabel 4.10

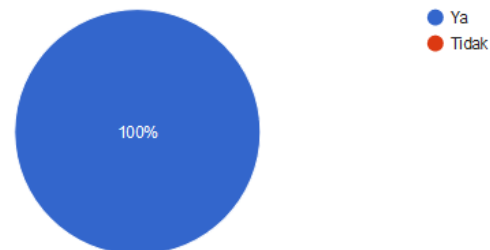
Tabel 4.9 Hasil Pengujian *Website Application*

No.	Pertanyaan	Jawaban		
		Ya	Tidak	Komentar
1.	Apakah <i>Website Application</i> yang diimplementasikan bermanfaat untuk Anda?	√ 7	0	Bermanfaat, bisa dipahami dan mudah digunakan
2.	Apakah tampilan yang ditawarkan dalam <i>website application</i> menarik dan mudah dimengerti?	√ 5	2	Tampilan masih memiliki permasalahan
3.	Apakah kebutuhan informasi di PD. Intan sudah terlihat dalam sistem <i>business intelligence</i> ini?	√ 7	0	Sudah, terdeskripsi
4.	Apakah perlu ada fitur tambahan untuk pengembangan ke depannya?	√ 5	2	Menanggulangi permasalahan yang ada
5.	Apakah ada kritik dan saran untuk pengembangan <i>business intelligence</i> ke depannya?	√ 5	2	Masih adanya permasalahan penampilan data

Data diatas berdasarkan hasil pertanyaan kepada pemilik perusahaan dan perwakilan pegawai dari setiap divisi dengan cara mengisi *google form* yang disediakan.

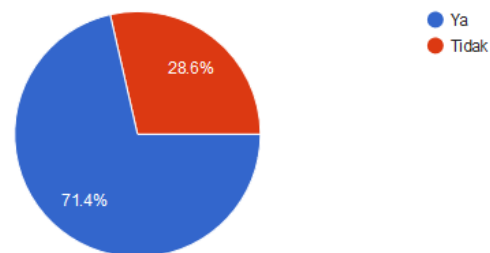
Apakah Website Application yang diimplementasikan bermanfaat untuk Anda?

7 responses



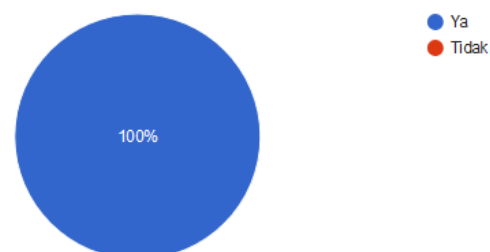
Apakah tampilan yang ditawarkan dalam website application menarik dan mudah dimengerti?

7 responses



Apakah kebutuhan informasi di PD. Intan sudah terlihat dalam sistem business intelligence ini?

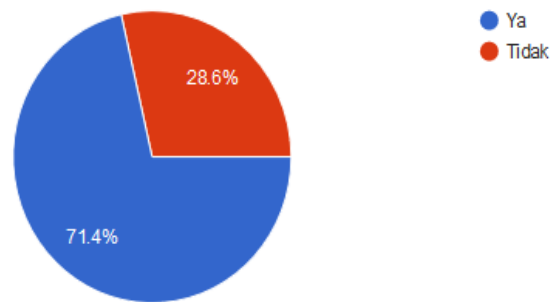
7 responses



Gambar 4.5 Hasil Google Form

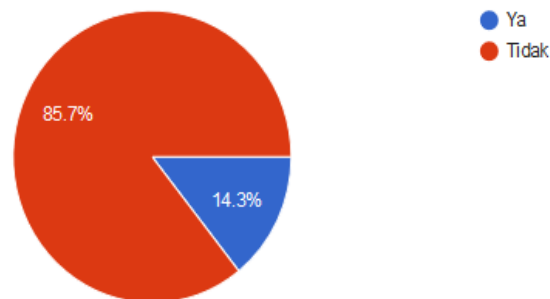
Apakah perlu ada fitur tambahan untuk pengembangan ke depannya?

7 responses



Apakah ada kritik dan saran untuk pengembangan business intelligence ke depannya?

7 responses



Tabel 4.10 Hasil Pengujian Informasi Strategis

No.	Kebutuhan Informasi Strategis	Perbandingan		Kesimpulan
		Hasil yang diinginkan	Hasil yang ditampilkan	
1.	Informasi transaksi yang dilakukan dalam suatu periode	Menampilkan total transaksi	Menampilkan total transaksi	Terpenuhi

No.	Kebutuhan Informasi Strategis	Perbandingan		Kesimpulan
		Hasil yang diinginkan	Hasil yang ditampilkan	
2.	Informasi total pembelian yang dilakukan terhadap suatu distributor	Menampilkan total pembelian perusahaan terhadap suatu distributor dalam periode tertentu	Menampilkan total pembelian perusahaan terhadap suatu distributor keseluruhan	Memadai
3.	Informasi persediaan barang yang tersedia dalam Gudang	Menampilkan informasi barang yang tersedia dalam gudang	Menampilkan informasi barang yang tersedia dalam gudang	Terpenuhi
4.	Informasi mengenai profit dan pendapatan yang didapat dalam periode tertentu	Menampilkan informasi pendapatan dan profit dalam periode tertentu	Menampilkan informasi pendapatan dan profit secara keseluruhan	Memadai
5.	Informasi nilai penjualan per produk	Menampilkan informasi penjualan per produk	Menampilkan nilai penjualan Per produk	Terpenuhi

4.2.5 Evaluasi Pengujian

Evaluasi hasil pengujian sebelumnya adalah sebagai berikut:

- Berdasarkan hasil pengujian hasil *black box* dapat disimpulkan bahwa *website application* yang dibangun sudah sesuai dengan kebutuhan fungsional
- Berdasarkan hasil pengujian data pada *website application* dapat disimpulkan bahwa data sudah sesuai dengan data yang ada di PD. Intan.
- Berdasarkan hasil pengujian wawancara kepada pemilik perusahaan PD. Intan dapat disimpulkan bahwa *website application* yang dibangun memudahkan pemilik dalam menganalisa bisnisnya di PD. Intan dan membantu pengadaan barang, serta informasi yang ditampilkan oleh aplikasi sudah sesuai dengan yang diharapkan.

BAB V

SIMPULAN DAN SARAN

5.1 Simpulan

Berdasarkan hasil analisis dan pembangunan *business intelligence* yang diimplementasikan di PD. Intan, maka dapat diambil simpulan sebagai berikut:

- Analisis informasi yang dihasilkan oleh *Business Intelligence* dapat membantu *top-level management* di PD. Intan dalam memantau perkembangan dan rencana bisnis dalam hal pengelolaan pengadaan barang.
- Implementasi *Business Intelligence* dapat menyelesaikan permasalahan yang ada pada PD. Intan. Dengan menampilkan informasi kebutuhan bisnis yang terkait dengan penjualan, pembelian dan pengadaan barang pemilik perusahaan memiliki data yang kuat untuk melakukan pengadaan barang. Sehingga produktivitas dari perusahaan meningkat
- Metode Kimball Lifecycle memudahkan dalam merancang daur hidup *Business Intelligence*, perancangan fokus dalam pengelolaan data pada Data Warehouse
- Terdapat beberapa tujuan pembangunan yang tidak tercapai dikarenakan perlu adanya wawancara lebih lanjut dengan perwakilan supplier serta izin untuk melakukan wawancara kepada supplier barang.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan oleh penulis, terdapat beberapa hal yang harus diperhatikan dan penulis sarankan agar sistem ini dapat dikembangkan lebih daripada penelitian sebelumnya.

- Keamanan pada sistem *business intelligence* perlu ditingkatkan dengan cara melakukan enkripsi terhadap data yang dikirimkan menuju database maupun data warehouse.
- Melakukan optimalisasi konten pada tampilan antarmuka *business intelligence*
- Meningkatkan fungsi dari *business intelligence* berdasarkan kebutuhan bisnis sesuai dengan kebutuhan kedepannya.
- Melakukan *maintenance* berkala dan berdiskusi dengan pemilik perusahaan dalam mengembangkan business intelligence
- Melakukan wawancara tingkat lanjut kepada supplier yang berhubungan dengan PD. Intan untuk menghasilkan business intelligence yang lebih optimal.

DAFTAR PUSTAKA

- Baars, H., & Kemper, H.-G. (2008). Management Support with Structured and Unstructured Data—An Integrated Business Intelligence Framework. *Information Systems Management*, 25(2).
<https://doi.org/10.1080/10580530801941058>
- Balaceanu, D. (2007). Components of a Business Intelligence Software Solution. *Informatica Economica*, 2(42).
- Codd, E. F., Codd, S. B., & Salley, C. T. (1993). Providing OLAP to User-Analysts: An IT Mandate. *E. F. Codd & Associates*.
- Connolly, T., & Begg, C. (2014). *Database Systems: A Practical Approach to Design, Implementation, and Management: Global Edition* (6th ed., p. 1440).
<https://doi.org/10.1007/978-1-4842-1191-5>
- Denney, M. J., Long, D. M., Armistead, M. G., Anderson, J. L., & Conway, B. N. (2016). Validating the extract, transform, load process used to populate a large clinical research database. *International Journal of Medical Informatics*, 94.
<https://doi.org/10.1016/j.ijmedinf.2016.07.009>
- Domes, S. (2017). *Progressive Web Apps with React* (1st ed., p. 403). Birmingham: Packt Publishing Ltd.
- Gupta, S., & Kapoor, B. (2016). Firebase in App Development. *International Research Journal of Engineering and Technology (IRJET)*, 3(12).
- Indrajani, S.Kom., MM. (2014). *Database Design All in One: Theory, Practice, and Case Study* (p. 326). Jakarta: PT. Elex Media Komputindo.

- Inmon, W. H. (2002). *Building the Data Warehouse* (3rd ed., p. 428; R. Elliott, Ed.). New York: John Wiley and Sons.
- Juneja, P. (2019). Business Intelligence - Architecture, Components and its Benefits.
- Kimball, R., & Ross, M. (2002). *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (Second Edition)* (2nd ed.; R. Elliott, Ed.). New York: John Wiley and Sons.
- Kimball, R., Ross, M., Thornthwaite, W., Mundy, J., & Becker, B. (2008). *The Data Warehouse Lifecycle Toolkit* (2nd ed., p. 672). Wiley.
- Kumar, K. N. M., Akhi, K., Gunti, S. K., & Reddy, M. S. P. (2016). Implementing Smart Home Using Firebase. *International Journal of Research in Engineering and Applied Sciences (IJREAS)*, 6573(10).
- Kurniawan, A. (2014). *Node.js Succinctly*. New York: Syncfusion Inc.
- Loshin, D. (2012). *Business intelligence: the savvy manager's guide / David Loshin*.
- Luhn, H. P. (1958). A Business Intelligence System. *IBM Journal of Research and Development*, 2(4). <https://doi.org/10.1147/rd.24.0314>
- Maheswari, A. K. (2011). *Business Intelligence and Data Mining* (pp. 91–101; M. Ferguson, Ed.). Business Expert Press.
- Ong, I., Siew, P., & Wong, S. (2011). A Five-Layered Business Intelligence Architecture. *Communications of the IBIMA*, (May). <https://doi.org/10.5171/2011.695619>
- Power, D. J. (2007). A Brief History of Decision Support Systems.

- Ranjan, J. (2009). Business Intelligence: Concept, Components, Techniques and Benefits. *Journal of Theoretical and Applied Information Technology*.
- Shariat, M., & Hightower, R. (2007). Conceptualizing Business Intelligence Architecture. *Marketing Management Journal*.
- Sherman, R. (2015). Business Intelligence Guidebook. In *Business Intelligence Guidebook*. <https://doi.org/10.1016/c2012-0-06937-2>
- Theisen, K. J. (2019). Programming Languages in Chemistry: A Review of HTML5/JavaScript. *Journal of Cheminformatics*, 11(1). <https://doi.org/10.1186/s13321-019-0331-1>
- Turban, E., Aronson, J. E., Liang, T.-P., & Sharda, R. (2007). *Decision Support and Business Intelligence Systems* (8th ed., p. 850). New Jersey: Prentice Hall.
- Turban, E., Sharda, R., Aronson, J. E., & King, D. (2011). *Business intelligence: A Managerial Approach* (1st ed.). Pearson Education Limited.
- Uden, L. (2002). Design process for Web Applications. *IEEE Multimedia*, 9(4). <https://doi.org/10.1109/MMUL.2002.1041948>
- Watson, H. J. (2009). Tutorial: Business Intelligence – Past, Present, and Future. *Communications of the Association for Information Systems*, 25. <https://doi.org/10.17705/1CAIS.02539>

LAMPIRAN

Lampiran 1 *Source Code* App.js

```
import React, { Component } from "react";
import { BrowserRouter } from "react-router-dom";
import { connect } from "react-redux";
import Navbar from "../components/layout/Navbar";
import Sidebar from "../components/layout/Sidebar";
import { Layout, Icon } from "antd";
import Routes from "../Route";
import { compose } from "redux";
import { firestoreConnect } from "react-redux-firebase";
import _ from "lodash";

import loadingIco from "../assets/img/load.png";
const { Content } = Layout;

class App extends Component {
  state = {
    collapsed: true
  };
  handleCollapse = collapsed => {
    this.setState({
      collapsed: !this.state.collapsed
    });
  };

  componentDidMount() {}
  componentDidUpdate(prevProps) {
    if (!_isEqual(prevProps, this.props)) {}
  }

  render() {
    const {
      auth,
      notifications,
      profile,
      products,
      employees
    } = this.props;
    const links = auth.uid ? (
      <Sidebar
        isCollapsed={this.handleCollapse}
```



```

        collapsed={this.state.collapsed}
        auth={auth}
      />
    ) : (
      ""
    );
    if (notifications) {
      return (
        <React.Fragment>
          <BrowserRouter>
            <div>
              { " " }
              <Layout style={{
minHeight: "100vh" }}>
                {links}
                <Layout>
                  <Navbar
isCollapsed={this.handleCollapse}
collapsed={this.state.collapsed}
notifications={notifications}
                                auth={auth}
profile={profile}
                                />
                                <Content
                                  style={{
margin:
"24px 16px",
marginTop: 70,
minHeight: 280
                                  }}
                                >
                                <Routes
profile={profile}
products={products}
employees={employees}
auth={auth}
                                />

```

```

        </Content>
      </Layout>
    </Layout>
  </div>
</BrowserRouter>
</React.Fragment>
);
} else {
  return (
    <div>
      <Icon type="loading"
className="splash-loading" />
    </div>
  );
}
}
}

const mapStateToProps = state => {
  return {
    auth: state.firebase.auth,
    notifications:
state.firestore.ordered.notifications,
    profile: state.firebase.profile
  };
};

const mapDispatchToProps = dispatch => {
  return {};
};

export default compose(
  connect(
    mapStateToProps,
    mapDispatchToProps
  ),
  firestoreConnect([
    { collection: "notifications", limit: 3,
orderBy: ["time", "desc"] }
  ])
)(App);

```

Lampiran 2 Source Code Route

```

import React, { Component } from "react";
import { Route, Switch } from "react-router-dom";

```

```

import Dashboard from
"./components/dashboard/Dashboard";
import ProductList from
"./components/products/ProductList";
import ProductDetails from
"./components/products/ProductDetails";
import EmployeeList from
"./components/employees/EmployeeList";
import EmployeeDetails from
"./components/employees/EmployeeDetails";
import IncomingList from
"./components/orders/incoming/IncomingList";
import IncomingDetails from
"./components/orders/incoming/IncomingDetails";
import OutgoingList from
"./components/orders/outgoing/OutgoingList";
import OutgoingDetails from
"./components/orders/outgoing/OutgoingDetails";
import RestockOrder from
"./components/orders/outgoing/RestockOrder";
import SalesReportList from "./components/sales-
report/SalesReportList";
import SalesReportDetails from "./components/sales-
report/SalesReportDetails";
import SignIn from "./components/auth/SignIn";
import SignUp from "./components/auth/SignUp";
import Playground from "./components/playground";
import ErrorPage from "./components/errorPage";
import TransactionList from
"./components/transactions/TransactionList";
import TransactionDetails from
"./components/transactions/TransactionDetails";
import SellProducts from
"./components/transactions/SellProducts";
import PurchasedList from
"./components/orders/purchase/PurchasedList";
import PurchasedDetails from
"./components/orders/purchase/PurchasedDetails";
import InputPurchased from
"./components/orders/purchase/InputPurchased";

class Routes extends Component {
  constructor(props) {
    super(props);
    this.state = {};
  }
  render() {

```

```

const { profile, auth } = this.props;

return (
  <Switch>
    <Route
      exact
      path="/"
      render={props => {
        return (
          <Dashboard
            auth={auth}
            profile={profile}
            {...props}
          />
        );
      }}
    />
    <Route
      exact
      path="/products"
      render={props => {
        return (
          <ProductList
            profile={profile}
            auth={auth}
            {...props}
          />
        );
      }}
    />
    <Route exact path="/products/:id"
component={ProductDetails} />
    <Route
      exact
      path="/employees"
      render={props => {
        return (
          <EmployeeList
            auth={auth}
            profile={profile}
            {...props}
          />
        );
      }}
    />
    <Route
      exact

```

```

        path="/employees/:id"
        render={props => {
            return (
                <EmployeeDetails
                    auth={auth}
                    profile={profile}
                    {...props}
                />
            );
        }}
    />
    <Route
        exact
        path="/orders/outgoing/"
        render={props => {
            return (
                <OutgoingList
                    auth={auth}
                    profile={profile}
                    {...props}
                />
            );
        }}
    />{" "}
    <Route
        exact
        path="/order-products/"
        render={props => {
            return (
                <RestockOrder
                    auth={auth}
                    profile={profile}
                    {...props}
                />
            );
        }}
    />
    <Route
        exact
        path="/orders/outgoing/:id"
        render={props => {
            return (
                <OutgoingDetails
                    auth={auth}
                    profile={profile}
                    {...props}
                />
            );
        }}
    />

```

```

    );
  }}
/>
<Route
  exact
  path="/orders/incoming/"
  render={props => {
    return (
      <IncomingList
        auth={auth}
        profile={profile}
        {...props}
      />
    );
  }}
/>
<Route
  exact
  path="/orders/incoming/:id"
  render={props => {
    return (
      <IncomingDetails
        auth={auth}
        profile={profile}
        {...props}
      />
    );
  }}
/>
<Route
  exact
  path="/orders/purchased/"
  render={props => {
    return (
      <PurchasedList
        auth={auth}
        profile={profile}
        {...props}
      />
    );
  }}
/>
<Route
  exact
  path="/orders/purchased/:id"
  render={props => {
    return (

```

```

        <PurchasedDetails
          auth={auth}
          profile={profile}
          {...props}
        />
      );
    }}
  />
  <Route
    exact
    path="/purchased-order"
    render={props => {
      return (
        <InputPurchased
          auth={auth}
          profile={profile}
          {...props}
        />
      );
    }}
  />
  <Route
    exact
    path="/sales-report/"
    render={props => {
      return (
        <SalesReportList
          auth={auth}
          profile={profile}
          {...props}
        />
      );
    }}
  />
  <Route
    exact
    path="/sales-report/:id"
    render={props => {
      return (
        <SalesReportDetails
          auth={auth}
          profile={profile}
          {...props}
        />
      );
    }}
  />

```

```

<Route
  exact
  path="/sell-products"
  render={props => {
    return (
      <SellProducts
        auth={auth}
        profile={profile}
        {...props}
      />
    );
  }}
/>
<Route
  exact
  path="/transactions"
  render={props => {
    return (
      <TransactionList
        auth={auth}
        profile={profile}
        {...props}
      />
    );
  }}
/>
<Route
  exact
  path="/transactions/:id"
  render={props => {
    return (
      <TransactionDetails
        auth={auth}
        profile={profile}
        {...props}
      />
    );
  }}
/>
<Route
  exact
  path="/login"
  render={props => {
    return <SignIn auth={auth}
{...props} />;
  }}
/>

```



```

        <Route
          exact
          path="/register"
          render={props => {
            return <SignUp auth={auth}
{...props} />;
          }}
        />
        <Route
          exact
          path="/playground"
          render={() => {
            return <Playground />;
          }}
        />
        <Route component={ErrorPage} />
      </Switch>
    );
  }
}

export default Routes;

```

Lampiran 3 *Source Code Root Reducer*

```

import authReducer from "./authReducer";
import productReducer from "./productReducer";
import { combineReducers } from "redux";
import salesReportReducer from
"./salesReportReducer";
import employeeReducer from "./employeeReducer";
import orderReducer from "./orderReducer";
import distributorReducer from
"./distributorReducer";
import transactionReducer from
"./transactionReducer";
import { firebaseReducer } from "react-redux-
firebase";
import { firestoreReducer } from "redux-firestore";
import restockReducers from "./restockReducer";
import purchasedReducer from "./purchasedReducer";
const rootReducer = combineReducers({
  auth: authReducer,
  product: productReducer,
  salesReport: salesReportReducer,
  transaction: transactionReducer,
  employee: employeeReducer,

```

```

    order: orderReducer,
    distributor: distributorReducer,
    restock: restockReducers,
    purchased: purchasedReducer,
    firebase: firebaseReducer,
    firestore: firestoreReducer
  });

export default rootReducer;

```

Lampiran 4 *Source Code Login*

```

import React, { Component } from "react";
import { Form, Icon, message, Input, Button,
Checkbox, Card } from "antd";
import { Redirect } from "react-router-dom";
import { connect } from "react-redux";
import { signIn } from
"../../redux/actions/authActions";
import _ from "lodash";
const FormItem = Form.Item;

class SignInForm extends Component {
  constructor(props) {
    super(props);
    this.state = {
      notification: "",
      warning: ""
    };
  }

  error(authError) {
    message.error(authError);
  }

  handleChange = e => {
    this.setState({
      [e.target.id]: e.target.value
    });
  };

  x;

  async handleSubmit(e) {
    e.preventDefault();
    this.props.form.validateFields((err, values)
=> {
      if (!err) {
        this.props.signIn(values);

```

```

    }
  });
}

render() {
  const { getFieldDecorator } =
this.props.form;
  const { auth, authError } = this.props;
  if (auth.uid) return <Redirect to="/" />;
  return (
    <React.Fragment>
      <div
        style={{
          background: "#ECECEC",
          paddingTop: "100px",
          height: "85vh"
        }}
      >
        <Card
          title="Login"
          bordered={false}
          className="mr-auto ml-auto
mt-10"
          style={{ width: 300 }}
        >
          <Form
            onSubmit={this.handleSubmit.bind(this)}
            className="login-form"
          >
            <Form.Item>
              {getFieldDecorator("email", {
                rules: [
                  {
                    required:
true,
                    message:
"Pleas input your email!"
                  }
                ]
              }) (
                <Input
                  prefix={
                    <Icon
type="user"

```

```

style={{
color: "rgba(0,0,0,.25)"
}}
/>
}

placeholder="Email"
/>
})
</Form.Item>
<Form.Item>

{getFieldDecorator("password", {
rules: [
{
required:
message:
"Please input your Password!"
}
]
})) (
<Input
prefix={
<Icon

type="lock"
style={{
color: "rgba(0,0,0,.25)"
}}
/>
}

type="password"
placeholder="Password"
/>
})
</Form.Item>
<Form.Item>
<Button
type="primary"

```

```

                                htmlType="submit"
                                className="login-
form-button"

                                >
                                Log in
                                </Button>
                                Or <a
href="/register">register now!</a>
                                </Form.Item>
                                <div
                                className="text-
center"
                                style={{ color: "red"
}}
                                >
                                {authError}
                                </div>
                                </Form>
                                </Card>
                                </div>
                                </React.Fragment>
                                );
                                }
                                }
const mapDispatchToProps = dispatch => {
  return {
    signIn: creds => dispatch(signIn(creds))
  };
};

const mapStateToProps = state => {
  return {
    authError: state.auth.authError
  };
};

const SignIn = Form.create() (SignInForm);
export default connect(
  mapStateToProps,
  mapDispatchToProps
) (SignIn);

```

Lampiran 5 Source Code Register

```
import React, { Component } from "react";
```

```

import { Form, message, Input, Button, Card, Select,
Icon, Upload } from "antd";
import { connect } from "react-redux";
import { signUp } from
"../../redux/actions/authActions";
import { Redirect } from "react-router-dom";
const { Option } = Select;

class SignUpForm extends Component {
  constructor(props) {
    super(props);
    this.state = {
      confirmDirty: false,
      autoCompleteResult: []
    };
  }

  handleSubmit = e => {
    e.preventDefault();
    this.props.form.validateFieldsAndScroll((err,
values) => {
      if (!err) {
        console.log(values);
        this.props.signUp(values);
      }
    });
  };

  handleConfirmBlur = e => {
    const value = e.target.value;
    this.setState({ confirmDirty:
this.state.confirmDirty || !!value });
  };

  compareToFirstPassword = (rule, value, callback)
=> {
    const form = this.props.form;
    if (value && value !==
form.getFieldValue("password")) {
      callback("Two passwords that you enter is
inconsistent!");
    } else {
      callback();
    }
  };
};

```

```

    validateToNextPassword = (rule, value, callback)
=> {
    const form = this.props.form;
    if (value && this.state.confirmDirty) {
        form.validateFields(["confirm"], { force:
true });
    }
    callback();
};

render() {
    const { getFieldDecorator } =
this.props.form;
    const { auth, authError } = this.props;
    const formItemLayout = {
        labelCol: {
            xs: { span: 24 },
            sm: { span: 8 }
        },
        wrapperCol: {
            xs: { span: 24 },
            sm: { span: 16 }
        }
    };
    const tailFormItemLayout = {
        wrapperCol: {
            xs: {
                span: 24,
                offset: 0
            },
            sm: {
                span: 16,
                offset: 8
            }
        }
    };

    if (auth.uid) return <Redirect to="/" />;
    return (
        <div
            style={{
                background: "#ECECEC",
                height: "104vh",
                paddingTop: "35px"
            }}
        >
            <Card
                title="Login"

```

```

        bordered={false}
        className="mr-auto ml-auto"
        style={{ width: "70vw" }}
      >
        <Form
onSubmit={this.handleSubmit}>
          <Form.Item
{...formItemLayout} label="E-mail">
{getFieldDecorator("email", {
                                rules: [
                                  {
                                    type:
"email",
                                    message:
"The
input is not valid E-mail!"
                                  },
                                  {
                                    required:
true,
                                    message:
"Please input your E-mail!"
                                  }
                                ]
                              })(<Input />)}
          </Form.Item>
          <Form.Item
{...formItemLayout} label="First Name">
{getFieldDecorator("firstName", {
                                rules: [
                                  {
                                    required:
true,
                                    message:
"Please input your first name"
                                  }
                                ]
                              })(<Input />)}
          </Form.Item>
          <Form.Item
{...formItemLayout} label="Last Name">
{getFieldDecorator("lastName", {
                                rules: [
                                  {

```



```

true,
    required:
    message:
    "Please input your last name"
    }
  ]
  })(<Input />)}
</Form.Item>

<Form.Item
{...formItemLayout} label="Role">
{getFieldDecorator("role", {
    rules: [
    {
      required:
      message:
      "Please select your role"
    }
  ]
  })(<Select>
    <Option
value="owner">Owner</Option>
    <Option
value="warehouse">Warehouse</Option>
    <Option
value="sales">Sales</Option>
    </Select>
  )}
</Form.Item>
<Form.Item
{...formItemLayout} label="Password">
{getFieldDecorator("password", {
    rules: [
    {
      required:
      message:
      "Please input your password!"
    },
    {
      validator:
      this.validateToNextPassword
    }
  ]
  })(<Input />)}
</Form.Item>

```

```

    ]
    })(<Input type="password"
/>)}
    </Form.Item>
    <Form.Item
    {...formItemLayout} label="Confirm Password">
    {getFieldDecorator("confirm", {
        rules: [
            {
                required:
true,
                message:
"PPlease confirm your password!"
            },
            {
                validator:
this.compareToFirstPassword
            }
        ]
    }) (
        <Input
            type="password"

onBlur={this.handleConfirmBlur}
        />
    )}
    </Form.Item>

    <div className="red-text
center">
        {authError ?
        <p>{authError}</p> : null}
        </div>
        <Form.Item
        {...tailFormItemLayout}>
            <Button type="primary"
htmlType="submit">
                Register
            </Button>
        </Form.Item>
    </Form>
</Card>
</div>
    );
}
}

```

```

const mapStateToProps = state => {
  return {
    auth: state.firebase.auth,
    authError: state.auth.authError
  };
};

const mapDispatchToProps = dispatch => {
  return {
    signUp: newUser => dispatch(signUp(newUser))
  };
};

const SignUp = Form.create({ name: "register"
})(SignUpForm);
export default connect(
  mapStateToProps,
  mapDispatchToProps
)(SignUp);

```

Lampiran 6 Source Purchasing Dashboard

```

import randomString from "random-string";

export const createPurchasedOrder = order => {
  return (dispatch, getState, { getFirestore,
getFirestore }) => {
    const firestore = getFirestore();
    firestore
      .collection("purchased")
      .add({
        ...order,
        key: randomString(),
        createdAt: new Date(),
        processedBy: "Pending",
        status: "pending"
      })
      .then(() => {
        dispatch({ type:
"CREATE_PURCHASED_ORDER" });
      })
      .catch(err => {
        dispatch({ type:
"CREATE_PURCHASED_ORDER_ERROR", err });
      });
  };
};

```

```

};

export const updateStatusPurchased = order => {
  return (dispatch, getState, { getFirestore,
    getFirestore }) => {
    const firestore = getFirestore();
    firestore
      .collection("purchased")
      .doc(order.id)
      .update({
        status: order.status,
        processedBy: order.processedBy,
        processedAt: new Date()
      })
      .then(() => {
        dispatch({ type:
"UPDATE_STATUS_PURCHASED" });
      })
      .catch(err => {
        dispatch({ type:
"UPDATE_STATUS_PURCHASED_ERROR", err });
      });
  };
};

export const getPurchasedOrder = () => {
  return (dispatch, getState, { getFirestore,
    getFirestore }) => {
    const firestore = getFirestore();
    firestore
      .collection("purchased")
      .get()
      .then(querySnapshot => {
        let list = [];
        querySnapshot.forEach(doc => {
          list.push(doc.data());
        });
        dispatch({ type:
"GET_PURCHASED_ORDER", data: list });
      })
      .catch(err => {
        dispatch({ type:
"GET_PURCHASED_ORDER_ERROR", err });
      });
  };
};

```

Lampiran 7 Source Code Sales Dashboard

```

import React, { Component, Fragment } from "react";
import { Col, Card, Row, Icon, Button, Tabs, Select,
DatePicker } from "antd";
import _ from "lodash";
import DropperToCsv from "../component/DropperToCsv";
import { connect } from "react-redux";
import { getPurchasedOrder } from
"../../../../../redux/actions/purchasedActions";
import { getTransactionOrder } from
"../../../../../redux/actions/transactionActions";
import { getRestockOrder } from
"../../../../../redux/actions/restockActions";
import ValueCard from "../component/ValueCard";
import BarSeriesCard from
"../component/BarSeriesCard";
import TableCard from "../component/TableCard";
import AreaSeriesCard from
"../component/AreaSeriesCard";
import LineSeriesCard from
"../component/LineSeriesCard";
import RadialChartCard from
"../component/RadialChartCard";
class Sales extends Component {
  constructor(props) {
    super(props);
    this.state = {
      cardLoading: true,
      spendPerProdPeriod: "alltime",
      transactionPerDistPeriod: "alltime",
      counterListPurchased: [],
      counterListResctock: [],
      counterListTransaction: [],
      counterPurchased: []
    };
  }

  render() {
    const { spendPeriod, transactionPeriod } =
this.state;

    return (
      <Fragment>
        <div className="gutter-example">
          <Row gutter={1}>

```



```

                                <AreaSeriesCard />
                                </Card>
                                </Col>
                                <Col className="gutter-row"
sm={24} md={24} lg={24}>
                                <Card title="Product
price comparation">
                                    <TableCard />
                                </Card>
                                </Col>
                                <Col className="gutter-row"
sm={24} md={24} lg={24}>
                                    <Card>
                                        <DropperToCsv />
                                    </Card>
                                </Col>
                                </Row>
                            </div>
                        </Fragment>
                    );
                }
            }

const mapStateToProps = state => {
    return {
        listPurchased: state.purchased.listPurchased,
        listTransaction:
state.transaction.listTransaction,
        listRestock: state.restock.listRestock
    };
};

const mapDispatchToProps = dispatch => {
    return {
        getPurchasedOrder: () =>
dispatch(getPurchasedOrder()),
        getTransactionOrder: () =>
dispatch(getTransactionOrder()),
        getRestockOrder: () =>
dispatch(getRestockOrder())
    };
};

export default connect(
    mapStateToProps,
    mapDispatchToProps
)(Sales);

```

Lampiran 8 *Source Code Product List*

```

import React, { Component } from "react";
import { connect } from "react-redux";
import {
  Modal,
  Table,
  Button,
  Card,
  Col,
  Input,
  Row,
  InputNumber,
  Icon,
  Skeleton,
  Form
} from "antd";
import moment from "moment";
import _ from "lodash";
import {
  createProduct,
  deleteProduct,
  getProduct,
  updateProduct,
  getProducts
} from "../../redux/actions/productActions";
import { compose } from "redux";
import { Redirect } from "react-router-dom";
import { firestoreConnect } from "react-redux-firebase";
const confirm = Modal.confirm;
const formItemLayout = {
  labelCol: { span: 6 },
  wrapperCol: { span: 16, offset: 1 }
};
const formPriceLayout = {
  labelCol: { span: 12 },
  wrapperCol: { span: 10, offset: 2 }
};
const formSellLayout = {
  labelCol: { span: 10, offset: 1 },
  wrapperCol: { span: 10, offset: 1 }
};

class ProductListUnWrapped extends Component {
  constructor(props) {
    super(props);
  }

```



```

        this.state = {
            ModalText: "Content of the modal",
            visibleAdd: false,
            visibleEdit: false,
            confirmLoading: false,
            tempId: null,
            tempProd: null,
            tempList: null
        };
        this.handleOk = this.handleOk.bind(this);
        this.handleOkEdit =
this.handleOkEdit.bind(this);
        this.showModalEdit =
this.showModalEdit.bind(this);

        this.detailedCollumn = [
            {
                title: "Created At",
                dataIndex: "createdAt",
                key: "createdAt",
                render: dataIndex => [
                    moment(dataIndex.toDate())
                        .subtract(10, "days")
                        .calendar()
                ]
            },
            {
                title: "Created By",
                dataIndex: "createdBy",
                key: "createdBy"
            },
            {
                title: "Updated At",
                dataIndex: "updatedAt",
                key: "updatedAt",
                render: dataIndex =>
[moment(dataIndex.toDate()).calendar()]
            },
            {
                title: "Updated By",
                dataIndex: "updatedBy",
                key: "updatedBy"
            },
            {
                title: "Action",
                dataIndex: "id",
                key: "action",

```



```

    ) }
    </Form.Item>
  </Row>
  <Row style={{
paddingLeft: "10px" }}>
    <Form.Item
    {...formItemLayout}
    label="Producer"
    >
    {getFieldDecorator("producer", {
rules: [
    {
required: true,
message:
"Please input the producer"
    }
  ]
    }) (
    <Input
    style={{ width: "98%" }}
    placeholder={
tempProd.producer
    }
    name="producer"
    onChange={this.onChange}
    />
    ) }
    </Form.Item>
  </Row>
  <Row style={{
paddingLeft: "10px" }}>
    <Form.Item
    {...formItemLayout}

```

```

label="Distributor"
                                >

{getFieldDecorator("distributor", {
rules: [
                                {
required: true,
message:
"Please input the distributor"
                                }
                                ]
                                }) (

<Input
style={{ width: "98%" }}
placeholder={
tempProd.distributor
                                }

name="distributor"
onChange={this.onChange}
                                />
                                )}
                                </Form.Item>
                                </Row>
                                <Row style={{
paddingLeft: "10px" }}>
                                <Col
span={12}>

<Form.Item
{...formPriceLayout}
label="Buy Price"
                                >

{getFieldDecorator("buyPrice", {

```

```

rules: [
  {
    required: true,
    message:
      "invalid price"
  }
]

<InputNumber
width="80%"
placeholder={
tempProd.buyPrice
}
parser={value =>
value.replace(
/\$\s?|(\,*)/g,
""
)
}
onChange={this.onChangeNumber.bind(
this,
"buyPrice"
)}
min={1}

```

```

/>
                                                    ) }

</Form.Item>
                                                    </Col>
                                                    <Col
span={12}>
<Form.Item
{...formSellLayout}
label="Sell Price"
                                                    >

{getFieldDecorator(
"sellPrice",
                                                    {
rules: [
{
required: true,
message:
"invalid price"
}
]
                                                    }
                                                    ) (

<InputNumber
width="100%"
placeholder={
tempProd.sellPrice
}

```



```

"Invalid Number"
    }
  ]
  }) (

<InputNumber
placeholder={tempProd.stock}
parser={value =>
value.replace(
/\$\s?|(\,*)/g,
""
)
}

onChange={this.onChangeNumber.bind(
this,
"stock"
)}
min={1}
/>
)}
</Form.Item>
</Row>
</Form>
</div>
);
return (
  <span key="action">
    <Button
loading={this.state.confirmLoading}
onClick={this.showModalEdit.bind(
  this,
  dataIndex,
  this.props

```



```

        type="primary"
      >
        Edit
      </Button>
    <Modal
      title="Edit Product"
      visible={this.state.visibleEdit}
      onOk={this.handleOkEdit}
      confirmLoading={this.confirmLoading}
      onCancel={this.handleCancel}
    >
      {skeletonEdit}
    </Modal>
    <Button
      onClick={this.showDelete.bind(
        this,
        dataIndex,
        this.props
      )}
      type="danger"
    >
      Delete
    </Button>
  </span>
);
}
}
];
this.columns = [
  {
    title: "Product Name",
    dataIndex: "productName",
    key: "productName ",
    render: text => <a
href="javascript:;>{text}</a>
  },
  {
    title: "Distributor",
    dataIndex: "distributor",
    key: "distributor"
  },

```

```

        {
            title: "Sell Price",
            dataIndex: "sellPrice",
            key: "sellPrice"
        },
        {
            title: "Stock",
            dataIndex: "stock",
            key: "stock"
        }
    ];
}

async componentDidMount() {}
async componentDidUpdate(prevProps) {
    const {
        selectedProduct,
        getProduct,
        productList,
        getProducts
    } = this.props;
    if (!_isEqual(prevProps.selectedProduct,
selectedProduct)) {
        if (this.state.tempId !== null) {
            getProduct(this.state.tempId);
            if (selectedProduct !== null) {
                this.setState({
                    tempProd: {
                        productName:
selectedProduct.productName,
                        distributor:
selectedProduct.distributor,
                        producer:
selectedProduct.producer,
                        buyPrice:
selectedProduct.buyPrice,
                        sellPrice:
selectedProduct.sellPrice,
                        stock:
selectedProduct.stock,
                        key: selectedProduct.key
                    }
                });
            }
        }
    }
}

```

```

        if (!_.isEqual(prevProps.productList,
productList)) {
            getProducts();
            if (productList !== null) {
                this.setState({
                    tempList: {
                        ...productList
                    }
                });
            }
        }
    }

    async showDelete(id, props) {
        confirm({
            title: "Are you sure delete this
product?",
            okText: "Yes",
            okType: "danger",
            cancelText: "No",
            onOk() {
                props.deleteProduct(id);
            },
            onCancel() {}
        });
        await setTimeout(() => {}, 2000);
    }

    onChangeEdit = e => {
        this.setState({
            tempProd: {
                ...this.state.tempProd,
                [e.target.name]: e.target.value
            }
        });
    };

    onChangeNumberEdit = (name, e) => {
        this.setState({
            tempProd: {
                ...this.state.tempProd,
                [name]: e
            }
        });
    };

    onChange = e => {

```

```

        this.setState({ [e.target.name]:
e.target.value });
    };

    showModalAdd = () => {
        this.setState({
            visibleAdd: true
        });
    };
    async showModalEdit(dataIndex, props) {
        const { getProduct, selectedProduct } =
props;
        this.setState({
            tempId: dataIndex,
            confirmLoading: true
        });
        if (selectedProduct !== null) {
            this.setState({
                tempProd: {
                    productName:
selectedProduct.productName,
                    producer:
selectedProduct.producer,
                    distributor:
selectedProduct.distributor,
                    buyPrice:
selectedProduct.buyPrice,
                    sellPrice:
selectedProduct.sellPrice,
                    stock: selectedProduct.stock,
                    key: selectedProduct.key
                }
            });
        }

        await setTimeout(() => {
            this.setState({
                visibleEdit: true
            });
        }, 1000);

        getProduct(dataIndex);
    }

    async handleOk(e) {
        const { profile, createProduct, form } =
this.props;

```

```

        e.preventDefault();
        form.validateFields((err, values) => {
            if (!err) {
                const data = {
                    ...values,
                    createdBy: profile.employeeName
                };
                createProduct(data);
                this.setState({
                    confirmLoading: true
                });
                setTimeout(() => {
                    this.setState({
                        visibleAdd: false,
                        confirmLoading: false
                    });
                    form.resetFields();
                }, 2000);
            }
        });
    }

    async handleOkEdit(e) {
        const { tempId, tempProd } = this.state;
        const { profile, updateProduct, form } =
this.props;
        console.log(tempProd);
        e.preventDefault();
        form.validateFields((err, values) => {
            if (!err) {
                const data = {
                    ...values,
                    id: tempId,
                    key: tempProd.key,
                    updatedBy: profile.employeeName
                };
                this.setState({
                    confirmLoading: true
                });

                updateProduct(data);
                this.setState({
                    confirmLoading: true
                });

                setTimeout(() => {
                    this.setState({

```

```

        visibleEdit: false,
        confirmLoading: false,
        tempProd: null,
        tempId: null
    });
    }, 2000);
    }
    });
}
onChangeNumber = (name, e) => {
    this.setState({ [name]: e });
};
buttonEdit = () => {};

handleCancel = () => {
    this.setState({
        visibleAdd: false,
        visibleEdit: false,
        confirmLoading: false,
        productName: "",
        sellPrice: "",
        buyPrice: "",
        stock: "",
        distributor: "",
        tempProd: null,
        tempId: ""
    });
};

render() {
    const { products, auth, profile } =
this.props;
    const { visibleAdd, confirmLoading } =
this.state;
    const { getFieldDecorator } =
this.props.form;

    const ownerColumns =
this.columns.concat(this.detailedCollumn);
    const listTable =
        profile.role === "owner" ? ownerColumns :
this.columns;
    const loading = !products ? true : false;

    const links =
        profile.role === "sales" ? null : (
        <div>

```

```

        <Button type="primary"
onClick={this.showModalAdd}>
            Add Product
        </Button>
        <Modal
            title="Add New Product"
            visible={visibleAdd}

onOk={this.handleOk.bind(this)}

confirmLoading={confirmLoading}
            onCancel={this.handleCancel}
        >
            <Form>
                <Row style={{
paddingLeft: "10px" }}>
                    <Form.Item

{...formItemLayout}
                        label="Product
Name"
                    >

{getFieldDecorator("productName", {
                                rules: [
                                    {
required: true,
message:
"Please input the Product Name"
                                }
                            ]
                        }) (
                            <Input
                                style={{
width: "98%" }}
                                placeholder="Product Name"
                                name="productName"
                                onChange={this.onChange}
                            />
                        ) }
                    </Form.Item>

```



```

    "Please input the distributor"

    }
  ]
  }) (
    <Input
      style={{
width: "98%" }}
placeholder="Distributor"
name="distributor"
onChange={this.onChange}

      />
    )}
  </Form.Item>
</Row>

  <Row style={{
paddingLeft: "10px" }}>
    <Col span={12}>
      <Form.Item
        {...formPriceLayout}
        label="Buy
Price"
      >
        {getFieldDecorator("buyPrice", {
          rules: [
            {
              required: true,
              message: "invalid price"
            }
          ]
        }) (
          <InputNumber
            width="80%"
            placeholder="Rp.XXXXXXX"
            parser={value =>

```

```

value.replace(
/\$\s?|(|,*)/g,
""
)
}

onChange={this.onChangeNumber.bind(
this,
"buyPrice"
)}

min={1}
/>
)}
</Form.Item>
</Col>
<Col span={12}>
<Form.Item

{...formSellLayout}
label="Sell
Price"
>

{getFieldDecorator("sellPrice", {
rules: [
{
required: true,
message: "invalid price"
}
]
}) (

<InputNumber
width="100%"
placeholder="Rp.XXXXXXX"
parser={value =>

```

```

value.replace(
/\$\s?|(|,*)/g,
""
)
}

onChange={this.onChangeNumber.bind(
this,
"sellPrice"
)}

min={1}
/>
)}
</Form.Item>
</Col>
</Row>
<Row style={{
paddingLeft: "10px" }}>
<Form.Item
{...formItemLayout} label="Stock">
{getFieldDecorator("stock", {
rules: [
{
required: true,
message: "Invalid Number"
}
]
}) (
<InputNumber
placeholder="XX"
parser={value =>
value.replace(/\$\s?|(|,*)/g, "")
}
onChange={this.onChangeNumber.bind(

```

```

this,

"stock"

    })
    min={1}
  />
  })
  </Form.Item>
</Row>
</Form>
{ /* <Row style={{ padding:
"10px 20px" }}>
    <Col span={8}>Product
Name :</Col>
    <Col span={16}>
      <Input
placeholder="Product Name"
name="productName"
value={this.state.productName}
onChange={this.onChange}
      />
    </Col>
  </Row>
  <Row style={{ padding:
"10px 20px" }}>
    <Col
span={8}>Distributor</Col>
    <Col span={16}>
      <Input
placeholder="Distributor"
name="distributor"
value={this.state.distributor}
onChange={this.onChange}
      />
    </Col>
  </Row>
  <Row style={{ padding:
"10px 20px" }}>

```

```

Price:</Col>
                                <Col span={8}>Buy
                                <Col span={16}>
                                  <InputNumber
placeholder="Rp.XXXXXXX"
                                parser={value
=>
                                value.replace(/\$\s?|(,*)/g, "")
                                }
                                onChange={this.onChangeNumber.bind(
                                this,
                                "buyPrice"
                                )}
                                value={this.state.buyPrice}
                                min={1}
                                />
                                </Col>
                                </Row>

                                <Row style={{ padding:
"10px 20px" }}>
                                <Col span={8}>Sell
                                <Col span={16}>
                                  <InputNumber
placeholder="Rp.XXXXXXX"
                                parser={value
=>
                                value.replace(/\$\s?|(,*)/g, "")
                                }
                                onChange={this.onChangeNumber.bind(
                                this,
                                "sellPrice"
                                )}
                                value={this.state.sellPrice}
                                min={1}
                                />

```

```

        </Col>
      </Row>

      <Row style={{ padding:
"10px 20px" }}>
        <Col
span={8}>Stock:</Col>
        <Col span={16}>
          <InputNumber
placeholder="XX"
parser={value
=>
value.replace(/\$\s?|(,*)/g, "")
          }
onChange={this.onChangeNumber.bind(
this,
"stock"
)}
value={this.state.stock}
min={1}
      />
    </Col>
  </Row>*/}
</Modal>
</div>
);
if (!auth.uid) return <Redirect to="/login"
/>;
if (!products) {
  return (
    <div>
      <Icon type="loading"
className="splash-loading" />
    </div>
  );
}

const paginations = products.length <= 10 ?
false : "";
return (
  <React.Fragment>
    <Card
      title="Product List"

```

```

        bordered={false}
        style={{ width: "100%" }}
        extra={links}
      >
        <Table
          className="mobile-table"
          columns={listTable}
          dataSource={products}
          scroll={{ x: "100%" }}
          loading={loading}
          pagination={paginations}
        />
      </Card>
    </React.Fragment>
  );
}
}

const ProductList = Form.create({ name: "ProductList"
})(ProductListUnWrapped);

const mapDispatchToProps = dispatch => {
  return {
    createProduct: product =>
    dispatch(createProduct(product)),
    deleteProduct: id =>
    dispatch(deleteProduct(id)),
    getProduct: id => dispatch(getProduct(id)),
    updateProduct: product =>
    dispatch(updateProduct(product)),
    getProducts: () => dispatch(getProducts())
  };
};

const mapStateToProps = state => {
  return {
    products: state.firestore.ordered.products,
    selectedProduct: state.product.product,
    productList: state.product.listProduct
  };
};

export default compose(
  connect(
    mapStateToProps,
    mapDispatchToProps
  ),

```

```

    firestoreConnect([
      { collection: "products", orderBy:
["productName", "asc"] }
    ])
  )(ProductList);

```

Lampiran 9 Source Code Dashboard.js

```

import React, { Component } from "react";
import { Row, Col, Card } from "antd";
import { connect } from "react-redux";
import { Redirect } from "react-router-dom";

import Managerboard from "../dashchild/Managerboard";
import Warehouseboard from
"../dashchild/Warehouseboard";
import Salesboard from "../dashchild/Salesboard";

import Incomes from "../statistic/Incomes";
import Profits from "../statistic/Profits";
import UpDownPrice from "../statistic/UpDownPrice";
import LowestStock from "../statistic/LowestStock";
class Dashboard extends Component {
  state = {};
  render() {
    const { auth, profile } = this.props;
    const child =
      profile.role === "owner" ? (
        <Managerboard />
      ) : profile.role === "warehouse" ? (
        <Warehouseboard />
      ) : (
        <Salesboard />
      );
    if (!auth.uid) return <Redirect to="/login"
/>;
    return
<React.Fragment>{child}</React.Fragment>;
  }
}
const mapStateToProps = state => {
  return { auth: state.firebase.auth };
};
export default connect(mapStateToProps)(Dashboard);

```

Lampiran 10 Source Code Index.js


```

import React from "react";
import ReactDOM from "react-dom";
import "bootstrap/dist/css/bootstrap.css";
import "antd/dist/antd.css";
import "./index.css";
import App from "./App";
import * as serviceWorker from "./serviceWorker";
import { createStore, applyMiddleware, compose } from
"redux";
import rootReducer from
"./redux/reducers/rootReducer";
import { Provider } from "react-redux";
import thunk from "redux-thunk";
import { reduxFirestore, getFirestore } from "redux-
firestore";
import { reactReduxFirebase, getFirebase } from
"react-redux-firebase";
import fbConfig from "./config/fbConfig";

const store = createStore(
  rootReducer,
  compose(
    applyMiddleware(thunk.withExtraArgument({
getFirestore, getFirestore })),
    reduxFirestore(fbConfig),
    reactReduxFirebase(fbConfig, {
      useFirestoreForProfile: true,
      userProfile: "users",
      attachAuthIsReady: true
    })
  )
);
store.firebaseAuthIsReady.then(() => {
  ReactDOM.render(
    <Provider store={store}>
      <App />
    </Provider>,
    document.getElementById("root")
  );
});

// If you want your app to work offline and load
faster, you can change
// unregister() to register() below. Note this comes
with some pitfalls.
// Learn more about service workers:
http://bit.ly/CRA-PWA

```

```
serviceWorker.unregister();
```

Lampiran 11 *Source Code package.json*

```
{
  "name": "mogo-bi-web",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "antd": "^3.13.2",
    "bootstrap": "^4.3.1",
    "csv": "^5.1.1",
    "firebase": "^5.8.1",
    "immutability-helper": "^3.0.0",
    "lodash": "^4.17.11",
    "moment": "^2.24.0",
    "random-string": "^0.2.0",
    "react": "^16.8.1",
    "react-dom": "^16.8.1",
    "react-redux": "^5.1.1",
    "react-redux-firebase": "^2.2.6",
    "react-router": "^4.3.1",
    "react-router-dom": "^4.3.1",
    "react-scripts": "^3.0.1",
    "react-vis": "^1.11.7",
    "recharts": "^1.5.0",
    "redux": "^4.0.1",
    "redux-firestore": "^0.6.4",
    "redux-thunk": "^2.3.0",
    "thunk": "0.0.1"
  },
  "engines": {
    "node": "10.15.3"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": "react-app"
  },
  "browserslist": [
    ">0.2%",
    "not dead",
```

```

    "not ie <= 11",
    "not op_mini all"
  ]
}

```

Lampiran 12 Implementasi data warehouse dalam firebase

/dimProduct

ID Dokumen ?

9nw94Yyq052tHMfE5Zu9

Kolom	Jenis
productName	= string
productType	= string
distributor	= string
buyPrice	= number
sellPrice	= number

/dimCustomer

ID Dokumen ?

mQ1WemyqrALMIFrsq24

Kolom	Jenis
custName	= string
custType	= string
custAddress	= string

/dimTime

ID Dokumen ?

FZMcAn7Zn6ZgvBoMA4Nk

Kolom	Jenis
date	= string
week	= string
month	= string
year	= string

/dimDistributor

ID Dokumen ?

ju3JQlvcdzda3idRyy8W

Kolom	Jenis
distName	= string
distType	= string
distAddress	= string

/dimEmployee

ID Dokumen ?

CEjxQ0okLyFOCpae5xHO

Kolom

Jenis

employeeName

=

string



/factPurchased

ID Dokumen ?

vketJ0kdSBYdPQt1fuAd

Kolom

Jenis

timeId

=

string



Kolom

Jenis

productId

=

string



Kolom

Jenis

distId

=

string



Kolom

Jenis

qty

=

number



Kolom

Jenis

pricePerProduct

=

number



Kolom

Jenis

subtotal

=

number



/factSales

ID Dokumen ?

wGXXzvNIsl7RYPZSc05f

Kolom

Jenis

timeId

=

string



Kolom

Jenis

productId

=

string



Kolom

Jenis

distId

=

string



Kolom

Jenis

qty

=

number



Kolom

Jenis

subtotal

=

number



Lampiran 13 Rules pada Database

```

service cloud.firestore {
  match /databases/{database}/documents {
    match /products/{product} {
      allow read, write:if request.auth.uid !=null
    }
    match /users/{userId}{
      allow create
      allow read : if request.auth.uid !=null
      allow write : if request.auth.uid == userId
    }
    match /notifications/{notification} {
      allow create, read,write
    }
    match /transactions/{transaction}{
      allow create
      allow read
      allow write
    }
    match /restocks/{restock}{
      allow create
      allow read
      allow write
    }
    match /purchased/{purchase}{
      allow create
      allow read
      allow write
    }
    match /distributors/{distributor}{
      allow create
      allow read
      allow write
    }
    match /transactions/{transaction}{
      allow create
      allow read
      allow write
    }
  }
  match /dimCustomer/{data}{
    allow create
    allow read
    allow write
  }
}

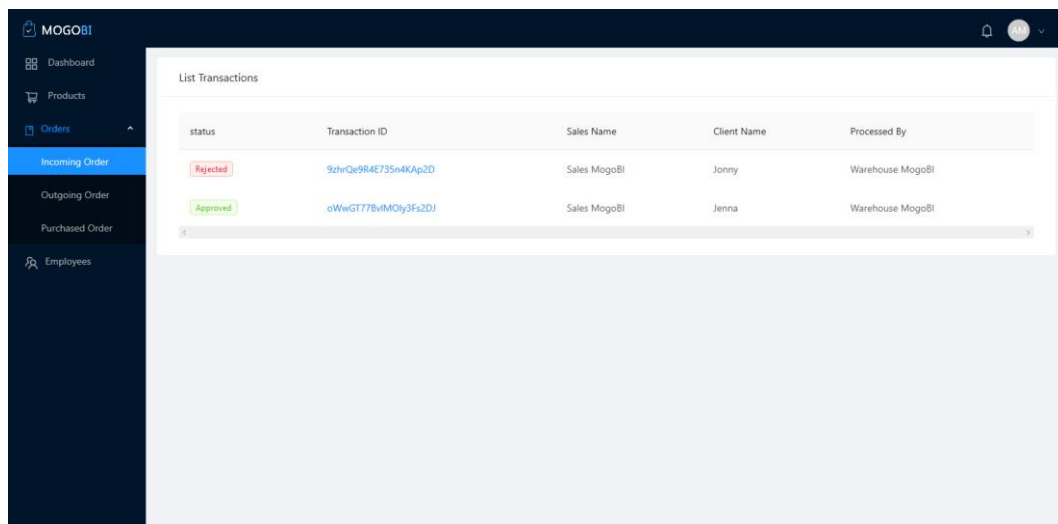
```

```

match /dimDistributor/{data}{
    allow create
    allow read
    allow write
}
match /dimEmployee/{data}{
    allow create
    allow read
    allow write
}
match /dimTime/{data}{
    allow create
    allow read
    allow write
}
match /dimProduct/{data}{
    allow create
    allow read
    allow write
}
}

```

Lampiran 14 Implementasi Antarmuka



MOGOBI

Dashboard
Products
Orders
Employees

Incoming Order
Outgoing Order
Purchased Order

Purchased Order List Input Order

Status	Order ID	Created At	Created By	Action
Approved	0b27AWeY7nz28LnCLFDn	05/27/2019	Admin MogoBI	Approved
Approved	3072eGne5Bu45k6jiz	06/10/2019	Admin MogoBI	Approved
Approved	9R4Xwd8cWgQpMmjLVOX	05/27/2019	Admin MogoBI	Approved
Approved	B4sVn30GPHVgsVPzfMyl	05/27/2019	Admin MogoBI	Approved
Pending	DAKXBcbxs5tBm4yz59Y	05/27/2019	Admin MogoBI	Approve Reject
Approved	DovjRBMf9e78G7F30Ze	05/27/2019	Admin MogoBI	Approved
Approved	JTGdYbO7NqPZokpPld	05/27/2019	Admin MogoBI	Approved
Approved	LNQId461EDihPhiuVo	05/27/2019	Admin MogoBI	Approved
Pending	MeRdXZNaIfwDL5faT	05/27/2019	Admin MogoBI	Approve Reject
Approved	Mj8uEeG4d9b9sQFI55	06/10/2019	Admin MogoBI	Approved

< 1 2 3 >

MOGOBI

Dashboard
Products
Orders
Employees

Incoming Order
Outgoing Order
Purchased Order

Product List Add Product

Add New Product

* Product Name:

* Producer:

* Distributor:

* Buy Price: * Sell Price:

* Stock:

Cancel OK

Product Name	Distributor	Stock	Created At	Updated By	Action
A	SADF		2019	Admin MogoBI	Edit Delete
Aa	D		2019	Admin MogoBI	Edit Delete
B	ewrr		2019	Admin MogoBI	Edit Delete
Beng - beng	Mayora		2019	Sales MogoBI	Edit Delete
Better	Mayora	40000	05/08/2019	Admin MogoBI	05/16/2019 Admin MogoBI Edit Delete
C	D	21	05/17/2019	Admin MogoBI	05/27/2019 Admin MogoBI Edit Delete
Chuba	PD. B1	60000	05/08/2019	Admin MogoBI	05/18/2019 Admin MogoBI Edit Delete
Chubaks	brand	1.19	05/17/2019	Admin MogoBI	05/27/2019 Admin MogoBI Edit Delete
Kopiko	PD. AA	40000	05/08/2019	Admin MogoBI	05/18/2019 Admin MogoBI Edit Delete
Roma	PD. B1	50000	05/08/2019	Admin MogoBI	05/18/2019 Admin MogoBI Edit Delete

< 1 2 >

MOGOBI

Dashboard

Products

Orders

Incoming Order

Outgoing Order

Purchased Order

Employees

Input Purchase Order

Go back

* Distributor:

PD. 81

* Products:

Beng - beng

2

40000

5

76000

⊖

Richeese

50

30000

10

1350000

⊖

1426000

+ Add field

Submit

MOGOBI

Dashboard

Products

Orders

Incoming Order

Outgoing Order

Purchased Order

Employees

Create Orders

Go back

* Distributor:

PD. AA

* Products:

Kopiko

5

31

⊖

Select Product

qty

Stock

⊖

Aa (6)

Teh Botol (32)

C (21)

Lampiran 15 Riwayat Hidup

RIWAYAT HIDUP

DATA PRIBADI

Nama : Muhamad Yusrizan
NPM : 140810150041
Tempat, Tanggal : Bandung, 03 Maret 1996
Lahir
Jenis Kelamin : Laki-Laki
Agama : Islam
Alamat : Jalan Terusan Pasirkoja no.132 Bandung
Telepon : 085759944047
Email : myusrizan@gmail.com

RIWAYAT PENDIDIKAN

2002-2008 : SDPN Pajagalan 58 Bandung
2008-2011 : SMP Negeri 3 Bandung
2011-2014 : SMA Negeri 4 Bandung
2015-2019 : Teknik Informatika FMIPA Unpad

RIWAYAT ORGANISASI

- 2017 : Staff Departemen Keprofesian Himatif FMIPA
Unpad
- 2016 : Staff Departemen Keprofesian Himatif FMIPA
Unpad

RIWAYAT KEPANITIAAN

- 2018 : Staff Divisi Acara Festival Olahraga dan Seni
UNPAD
- 2017 : Ketua Pelaksana *Informatics Festival 2017*
- 2017 : Staff Divisi Humas INSTAGRAM HIMATIF
FMIPA UNPAD
- 2016 : Koordinator Divisi Marketing *Informatics Festival*
HIMATIF FMIPA UNPAD
- 2016 : Koordinator Divisi Acara *TECHNOPRENEUR*
HIMATIF FMIPA UNPAD
- 2016 : Ketua Pelaksana *Prepare For Future* HIMATIF
FMIPA UNPAD

Lampiran 16 Surat Keterangan Penelitian

TOKO KELONTONG PD INTAN
Jl. Terusan Pasirkoja 132 Bandung
Telp. 022-6005851

Bandung, 05 Agustus 2019

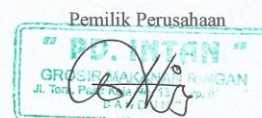
SURAT KETERANGAN

Yang bertanda tangan di bawah ini Pemilik dari PD. Intan, menerangkan bahwa :

Nama : Muhamad Yusrizan
NPM : 140810150041
Fakultas : Matematika dan Ilmu Pengetahuan Alam
Jurusan : Teknik Informatika
Judul Skripsi : "ANALISIS DAN PERANCANGAN *BUSINESS INTELLIGENCE*
BERBASIS *WEBAPP* MENGGUNAKAN *NOSQL DATABASE*
DENGAN METODE *KIMBALL LIFECYCLE*"

Yang bersangkutan telah melakukan penelitian di PD. Intan pada tanggal 27 Desember 2018
s.d. 6 Mei 2019.

Surat keterangan ini diberikan agar dapat dipergunakan sebagaimana mestinya



Muhamad Ropandi