# Pengembangan Sistem Manajemen Akuntansi Keuangan Perusahaan Untuk Toko Roti Amaya Berbasis Website Menggunakan React JS

SKRIPSI

Untuk memenuhi sebagian persyaratan   
memperoleh gelar Sarjana Komputer

Disusun oleh:

Farah Hanifah Izzati

NIM: 165150201111146



TEKNIK INFORMATIKA

TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2020/2021

# PERSETUJUAN

Pengembangan Sistem Manajemen Akuntansi Keuangan Perusahaan Untuk Toko Roti Amaya Berbasis Website Menggunakan React JS

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan

memperoleh gelar Sarjana Komputer

Disusun Oleh :

Farah Hanifah Izzati

NIM: 165150201111146

Telah diperiksa dan disetujui oleh:

|  |  |
| --- | --- |
| Dosen Pembimbing I  Nuruddin Santoso, S.T., M.T.  NIK : 197409162000121001 | Dosen Pembimbing 2  Faizatul Amalia, S.Pd., M.Pd.  NIK: - |

Mengetahui

Ketua Jurusan Teknik Informatika

Nama Ketua Jurusan

NIP: 123456789

# PERNYATAAN ORISINALITAS

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

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

Jakarta, 1 Desember 2020

­

Farah Hanifah Izati

NIM: 165150201111146

# PRAKATA

Puji syukur kehadirat Allah SWT telah melimpahkan rahmat, taufik dan hidayah-Nya sehingga skripsi yang berjudul “Pengembangan Sistem Manajemen Akuntansi Keuangan Perusahaan Untuk Toko Roti Amaya Berbasis Website Menggunakan React JS” ini dapat terselesaikan.

Penulis menyadari bahwa laporan ini tidak akan berhasil tanpa bantuan dari beberapa pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terima kasih kepada:

1. Allah SWT atas karunia-NYA penulis diberikan kesehatan, kemampuan dan ilmu dalam menyelesaikan skripsi ini.
2. Kedua orang tua, Andy Susanto dan Erliningsih yang telah memberikan doa yang tiada henti serta dukungan baik moril dan materil.
3. Kakak-kakak tersayang, Zahra Sativani dan Riza Pahlawi atas dukungan dan doa.
4. Bapak Nuruddin Santoso., S.T. M.T dan Ibu Faizatul Amalia selaku Dosen Pembimbing skripsi atas semua waktu, arahan dan kesabaran yang telah diberikan.
5. Bapak Hera selaku narasumber skripsi, semoga ALLAH SWT senantiasa memberikan kesehatan untuk bapak dan keluarga.
6. Seluruh civitas akademika Teknik Informatika Universitas Brawijaya yang telah memberikan bantuan dalam menyelesaikan skripsi.
7. Rizka, Liony, Adinda, Alma dan Dias sebagai sahabat dan keluarga yang menjadi tempat berkeluh kesah dan selalu memberikan motivasi dan hiburan dari jarak jauh.
8. Citra, Yurdha, Benita dan Nisrina sebagai teman sekaligus keluarga baru selama menempuh perkuliahan.
9. Seluruh pihak yang telah membantu kelancaran pengerjaan skripsi yang tidak dapat disebutkan satu persatu.

Jakarta, 1 Desember 2020

Farah Hanifah Izzati

Farahhanifah18@gmail.com

# ABSTRAK

Farah Hanifah Izzati, Pengembangan Sistem Manajemen Akuntansi Keuangan Perusahaan Untuk Toko Roti Amaya Berbasis Website Menggunakan React JS

Pembimbing: Nuruddin Santoso, S.T., M.T dan Faizatul Amalia

Toko Roti Amaya merupakan perusahaan menengah di Kota Malang yang bergerak di bidang produksi dan penjualan roti serta kue. Toko Roti Amaya menggunakan *Microsoft Excel* untuk membantu pelaporan keuangan, permasalahan muncul ketika beberapa proses dalam pelaporan keuangan dilakukan secara manual sebagai contoh proses pelaporan keuangan, validasi transaksi oleh akuntan, pengiriman bukti tagihan dilakukan secara manual melalui *email*. Berdasarkan permasalahan yang ada, peneliti bertujuan untuk mengembangkan sistem pelaporan keuangan otomatis yang terintegrasi dengan proses produksi, rekapitulasi bahan baku dan *stock opname* produk.

Pengembangan sistem dilakukan menggunakan metode SDLC *waterfall* dengan mengimplementasikan *library react js* dan *restfull API* serta *MySQL* sebagai basis data. Sistem menggunakan *framework* *MDB React*, *material-ui* serta *lumen* sebagai *framework* untuk membangun API. Sistem juga menerapkan *react hooks* dan *stateless component* untuk manajemen *state react js*. Pengujian sistem dilakukan pada level pengujian unit dan sistem dengan menggunakan strategi pengujian kotak putih dan kotak hitam.

Kata kunci: laporan keuangan, toko roti, *website*, react js, react hooks

# ABSTRAC

Farah Hanifah Izzati, Development Financial Accounting Management System For Amaya Bakery Based on Website Uses React JS

Pembimbing: Nuruddin Santoso, S.T., M.T dan Faizatul Amalia

*Amaya bakery is one of the middle companies based in Malang City who specialized in sell and produce cake and bread. Amaya bakery uses Microsoft Excel to help them make a financial report, but the problem is some of the processes are still manually for example generated financial report, transaction validation by an accountant, and invoice send manually via email. Based on the problem, the researcher wants to develop an automatic financial report system that is integrated with the production process, product recapitulation, and stock opname.*

*The Development system uses SLDC waterfall methods with react js library and restfull API along with MySQL as the database. The system uses MDB React, material-UI as a frontend framework, and lumen as a framework to build API. The system also uses react hooks and stateless components to manage state in react js. Software testing uses on unit and system testing level with the black box and white box strategy.*

Kata kunci: *financial report, bakery, website, react js, react hooks*

DAFTAR ISI

[Pengembangan Sistem Manajemen Akuntansi Keuangan Perusahaan Untuk Toko Roti Amaya Berbasis Website Menggunakan React JS i](#_Toc57212588)

[PERSETUJUAN ii](#_Toc57212589)

[PERNYATAAN ORISINALITAS iii](#_Toc57212590)

[PRAKATA iv](#_Toc57212591)

[ABSTRAK v](#_Toc57212592)

[ABSTRAC vi](#_Toc57212593)

[DAFTAR ISI vii](#_Toc57212594)

[DAFTAR TABEL x](#_Toc57212595)

[DAFTAR GAMBAR xii](#_Toc57212596)

[DAFTAR LAMPIRAN xiv](#_Toc57212597)

[BAB 1 PENDAHULUAN 1](#_Toc57212598)

[1.1 Latar Belakang 1](#_Toc57212599)

[1.2 Rumusan Masalah 2](#_Toc57212600)

[1.3 Tujuan 2](#_Toc57212601)

[1.4 Manfaat 2](#_Toc57212602)

[1.5 Batasan Masalah 2](#_Toc57212603)

[1.6 Sistematika Pembahasan 3](#_Toc57212604)

[BAB 2 LANDASAN KEPUSTAKAAN 4](#_Toc57212605)

[2.1 Siklus Akuntansi 4](#_Toc57212606)

[2.1.1 Laporan Keuangan 4](#_Toc57212607)

[2.1.2 Jurnal 7](#_Toc57212608)

[2.1.3 Harga Pokok Produksi 8](#_Toc57212609)

[2.2 Teknologi Pengembangan Sistem 9](#_Toc57212610)

[*2.2.1 Hypertext Markup Language* 9](#_Toc57212611)

[2.2.2 MySQL 9](#_Toc57212612)

[2.2.3 Restful API 10](#_Toc57212613)

[2.2.4 React JS 11](#_Toc57212614)

[2.2.5 MDB React 13](#_Toc57212615)

[*2.2.6 Material-UI* 14](#_Toc57212616)

[2.2.7 Lumen 14](#_Toc57212617)

[*2.3 Software Development Life Cycle* 14](#_Toc57212618)

[2.3.1 Model Waterfall 15](#_Toc57212619)

[2.4 UML (*Unified Modeling Language*) 16](#_Toc57212620)

[2.4.1 Use Case Scenario 16](#_Toc57212621)

[2.4.2 *Use Case* Diagram 16](#_Toc57212622)

[*2.5 Structured Analisys and Design* 18](#_Toc57212623)

[*2.5.1 Data Flow Diagram* 18](#_Toc57212624)

[*2.5.2 Structure Chart* 20](#_Toc57212625)

[*2.5.3 Entity Relationship Diagram* 20](#_Toc57212626)

[2.6 Pengujian Perangkat Lunak 21](#_Toc57212627)

[2.6.1 *White Box Testing* 22](#_Toc57212628)

[*2.6.2 Black Box Testing* 23](#_Toc57212629)

[*2.6.3 System Testing* 23](#_Toc57212630)

[BAB 3 METODOLOGI PENELITIAN 25](#_Toc57212631)

[3.1 Studi Literatur 25](#_Toc57212632)

[3.2 Rekayasa Kebutuhan 26](#_Toc57212633)

[3.3 Perancangan 26](#_Toc57212634)

[3.4 Implementasi 26](#_Toc57212635)

[3.5 Pengujian 26](#_Toc57212636)

[3.6 Kesimpulan 27](#_Toc57212637)

[BAB 4 REKAYASA KEBUTUHAN 28](#_Toc57212638)

[4.1 Deskripsi Sistem 28](#_Toc57212639)

[4.2 Identifikasi Aktor 28](#_Toc57212640)

[4.3 Identifikasi Kebutuhan 29](#_Toc57212641)

[4.3.1 Aturan Penomoran Kebutuhan Fungsional dan Non Fungsional 29](#_Toc57212642)

[4.3.2 Daftar Spesifikasi Kebutuhan Fungsional 29](#_Toc57212643)

[4.3.3 Daftar Spesifikasi Kebutuhan Non Fungsional 31](#_Toc57212644)

[4.4 Pemodelan Kebutuhan 31](#_Toc57212645)

[4.4.1 *Use Case* Diagram 31](#_Toc57212646)

[4.4.2 Use Case Scenario 33](#_Toc57212647)

[4.5 Analisis Data 39](#_Toc57212648)

[BAB 5 PERANCANGAN DAN IMPLEMENTASI 40](#_Toc57212649)

[5.1 Perancangan 40](#_Toc57212650)

[5.1.1 Perancangan Struktur Analisis 40](#_Toc57212651)

[5.1.2 Perancangan Struktur Desain 43](#_Toc57212652)

[5.1.3 Perancangan Basis Data 44](#_Toc57212653)

[5.1.4 Perancangan Antarmuka 45](#_Toc57212654)

[5.2 Implementasi 51](#_Toc57212655)

[5.2.1 Spesifikasi Pengembangan Sistem 51](#_Toc57212656)

[5.2.2 Implementasi Basis Data 51](#_Toc57212657)

[5.2.3 Implementasi Algoritma 52](#_Toc57212658)

[5.2.4 Implementasi Antar Muka 55](#_Toc57212659)

[BAB 6 PENGUJIAN SISTEM 57](#_Toc57212660)

[6.1 Pengujian Kondisi Jalur 57](#_Toc57212661)

[6.2 Pengujian Validasi 58](#_Toc57212662)

[6.3 Pengujian Kesesuaian 60](#_Toc57212663)

[BAB 7 KESIMPULAN 61](#_Toc57212664)

[7.1 Kesimpulan 61](#_Toc57212665)

[7.2 Saran 61](#_Toc57212666)

[DAFTAR REFERENSI 62](#_Toc57212667)

[LAMPIRAN A PANDUAN WAWANCARA 65](#_Toc57212668)

[A.1 Daftar Pertanyaan Wawancara 65](#_Toc57212669)

[A.2 Hasil Wawancara 65](#_Toc57212670)

DAFTAR TABEL

[Tabel 4.1 Identifikasi Aktor Sistem Manajemen Keuangan 29](#_Toc57212123)

[Tabel 4.2 Penomoran Spesifikasi Kebutuhan Fungsional 29](#_Toc57212124)

[Tabel 4.3 Tabel Spefikasi Kebutuhan Fungsional 29](#_Toc57212125)

[Tabel 4.4 Tabel Spesifikasi Kebutuhan Non Fungsional 31](#_Toc57212126)

[Tabel 4.5 *Use Case Scenario* Mencatat Transaksi 33](#_Toc57212127)

[Tabel 4.6 *Use Case Scenario* Mengubah Transaksi 33](#_Toc57212128)

[Tabel 4.7 *Use Case Scenario* Menampilkan List Transaksi 34](#_Toc57212129)

[Tabel 4.8 *Use Case Scenario* Menghapus Transaksi 34](#_Toc57212130)

[Tabel 4.9 *Use Case Scenario* Menambahkan Produk 34](#_Toc57212131)

[Tabel 4.10 *Use Case Scenario* Menampilkan List Produk 35](#_Toc57212132)

[Tabel 4.11 *Use Case Scenario* Menambahkan Bahan Baku 35](#_Toc57212133)

[Tabel 4.12 *Use Case Scenario* Rekapitulasi Bahan Baku 36](#_Toc57212134)

[Tabel 4.13 *Use Case Scenario* Rekapitulasi Produksi 36](#_Toc57212135)

[Tabel 4.14 *Use Case Scenario* Review Transaksi 36](#_Toc57212136)

[Tabel 4.15 *Use Case Scenario* Validasi Transaksi 37](#_Toc57212137)

[Tabel 4.16 Use Case Scenario *Review* Jurnal 37](#_Toc57212138)

[Tabel 4.17 *Use Case Scenario* *Posting* Jurnal 38](#_Toc57212139)

[Tabel 4.18 *Use Case Scenario* Generate Laporan Keuangan 38](#_Toc57212140)

[Tabel 4.19 *Use Case Scenario Generate* Laporan Keuangan 38](#_Toc57212141)

[Tabel 4.20 *Use Case Scenario* Mencetak Laporan Keuangan 39](#_Toc57212142)

[Tabel 4.21 *Use Case Scenario* Mengunduh Laporan Keuangan 39](#_Toc57212143)

[Tabel 5.1 Informasi Elemen Halaman Mencatat Transaksi Penjualan 46](#_Toc57212144)

[Tabel 5.2 Informasi Elemen Halaman Mencatat Transaksi Kebutuhan Perusahaan 47](#_Toc57212145)

[Tabel 5.3 Implementasi Basis Data *Review* Jurnal 52](#_Toc57212146)

[Tabel 5.4 Implementasi Basis Data PostingJurnal 52](#_Toc57212147)

[Tabel 5.5 Implementasi Algoritma Review Jurnal Perhari (Component.js) 53](#_Toc57212148)

[Tabel 5.6 Implementasi Algoritma Review Jurnal (Context.js) 54](#_Toc57212149)

[Tabel 6.1 *Pseudocode* fungsi komponen reviewJurnal 57](#_Toc57212150)

[Tabel 6.2 Pengujian Validasi Sistem 59](#_Toc57212151)

DAFTAR GAMBAR

[Gambar 2.1 Siklus Akuntansi 4](#_Toc57212089)

[Gambar 2.2 Contoh Laporan Laba Rugi 6](#_Toc57212090)

[Gambar 2.3 Contoh Laporan Ekuitas Pemilik 6](#_Toc57212091)

[Gambar 2.4 Contoh Neraca 6](file:///D:\SKRIPSI\skripsi%20lengkap%20(Autosaved).docx#_Toc57212092)

[Gambar 2.5 Contoh Laporan Arus Kas 7](#_Toc57212093)

[Gambar 2.6 Contoh Jurnal Umum 2 Kolom 7](#_Toc57212094)

[Gambar 2.7 Desain Arsitekur *Restful API* 10](#_Toc57212095)

[Gambar 2.8 Siklus Hidup *React Component* 11](#_Toc57212096)

[Gambar 2.9 SDLC model *waterfall* 15](file:///D:\SKRIPSI\skripsi%20lengkap%20(Autosaved).docx#_Toc57212097)

[Gambar 2.10 Notasi Sistem 17](file:///D:\SKRIPSI\skripsi%20lengkap%20(Autosaved).docx#_Toc57212098)

[Gambar 2.11 Notasi *Use Case* 17](file:///D:\SKRIPSI\skripsi%20lengkap%20(Autosaved).docx#_Toc57212099)

[Gambar 2.12 Notasi *Actor* 17](file:///D:\SKRIPSI\skripsi%20lengkap%20(Autosaved).docx#_Toc57212100)

[Gambar 2.13 Notasi *Relationship* 17](file:///D:\SKRIPSI\skripsi%20lengkap%20(Autosaved).docx#_Toc57212101)

[Gambar 2.14 Notasi *Data flow diagram* 19](#_Toc57212102)

[Gambar 2.15 Notasi *Entity Relationship Diagram* 21](#_Toc57212103)

[Gambar 2.16 *Flow Graph* 22](file:///D:\SKRIPSI\skripsi%20lengkap%20(Autosaved).docx#_Toc57212104)

[Gambar 3.1 Metodologi Penelitian 25](file:///D:\SKRIPSI\skripsi%20lengkap%20(Autosaved).docx#_Toc57212105)

[Gambar 4.1 *Use Case Diagram* 32](#_Toc57212106)

[Gambar 5.1 *Data Flow Diagram* Level 0 40](#_Toc57212107)

[Gambar 5.2 Data Flow Diagram Level 1 41](#_Toc57212108)

[Gambar 5.3 *Data Flow Diagram* Level 2 Proses 2 41](#_Toc57212109)

[Gambar 5.4 *Data Flow Diagram* Level 2 Proses 3 42](#_Toc57212110)

[Gambar 5.5 *Data Flow Diagram* Level 2 Proses 5 42](#_Toc57212111)

[Gambar 5.6 *Data Flow Diagram* Level 2 Proses 6 42](#_Toc57212112)

[Gambar 5.7 *Structure Chart* 43](#_Toc57212113)

[Gambar 5.8 *Enitity Relationship Diagram* 44](#_Toc57212114)

[Gambar 5.9 Perancangan Antarmuka Pengguna Halaman Mencatat Transaksi Pemasukan 45](file:///D:\SKRIPSI\skripsi%20lengkap%20(Autosaved).docx#_Toc57212115)

[Gambar 5.10 Perancangan Antarmuka Pengguna Halaman Mencatat Transaksi Pengeluaran 45](#_Toc57212116)

[Gambar 5.11 Perancangan Antarmuka Pengguna Halaman Jurnal Per Hari 48](#_Toc57212117)

[Gambar 5.12 Perancangan Antarmuka Pengguna Halaman Jurnal Per Bulan 48](#_Toc57212118)

[Gambar 5.13 Implementasi Antar Muka Jurnal Per Hari 56](#_Toc57212119)

[Gambar 5.14 Implementasi Antar Muka Jurnal Per Hari Setelah *Review* 56](#_Toc57212120)

[Gambar 6.1 *Flow Graph* fungsi komponen reviewJurnal 58](#_Toc57212121)

[Gambar 6.2 Hasil *Compability Test* Menggunakan *SortSite* 60](#_Toc57212122)

DAFTAR LAMPIRAN

[LAMPIRAN A PANDUAN WAWANCARA 65](#_Toc56514931)

# PENDAHULUAN

## Latar Belakang

Perkembangan dunia industri yang pesat menuntut pelaku industri untuk terus melakukan inovasi pada industrinya terutama dengan hadirnya era industri 4.0. Inovasi dari segi peningkatan layanan pelanggan hingga peningkatan layanan untuk menunjang operasional perusahaan, seperti contoh layanan e*-complain* yang digunakan sebagai media pengaduan kritik dan saran ataupun e-*commerce* sebagai media penjualan secara *online* yang kini tengah marak keberadaannya. Selain itu, terdapat contoh lain berupa layanan dalam bentuk aplikasi diluar sistem utama, seperti *wave accounting* yang berfungsi sebagai pendataan keuangan ataupun google bisnis yang berfungsi sebagai pendataan bisnis yang disediakan oleh google.

Industri 4.0 bukan hanya berpengaruh pada industri besar namun juga pada usaha menengah dan kecil. Seperti di Malang, industri 4.0 terus dikembangkan diberbagai industri salah satunya sektor makanan. Toko Roti Amaya merupakan industri lokal yang bergerak dibidang produksi dan penjualan roti dan kue. Efisiensi menjadi kunci dari industri roti yang merupakan perusahaaan berskala besar dengan keuntungan kecil. Bermula dari efisiensi penggunaan bahan, efektifitas antara manusia dan mesin, serta efisiensi produksi dan distribusi. Roti, pastri dan berbagai kue lainnya diproduksi dan didistribusikan dalam satu siklus sehingga permintaan dan produksi harus sejalan satu sama lain untuk menghindari produksi berlebihan. Jika terjadi kesalahan dalam perhitungan jumlah kebutuhan bahan baku dengan permintaan pasar mengakibatkan kerugian besar bagi industri.

Sebagai langkah efisiensi untuk mengetahui jumlah permintaan dan produksi dalam satu siklus dibutuhkan laporan khusus yang memuat informasi pendapatan dan biaya produksi. Permasalahan muncul ketika laporan tersebut dibuat secara manual oleh akuntan. Toko Roti Amaya menggunakan *Microsoft Excel* untuk membuat laporan keuangan. Proses bermula dari setiap divisi dan toko terkait mengirimkan bukti transaksi melalui *email* kepada akuntan. Akuntan mencatat bukti transaksi untuk selanjutnya data tersebut disamakan dengan mutasi rekening dari masing-masing rekening bank oleh akuntan. Setelah dicocokan oleh akuntan, data tersebut diolah menjadi laporan keuangan secara manual. Untuk melakukan perhitungan setiap transaksi, akuntan Toko Roti Amaya sebagai penanggung jawab dalam pengelolaan dan pelaporan keuangan perusahaan, memiliki rumus sendiri untuk mempermudah perhitungan.

Berdasarkan masalah diatas, diperlukan Sistem Manajemen Akuntansi Keuangan untuk menunjang kebutuhan operasional Toko Roti Amaya dalam pengelolaan keuangan yang lebih mudah, cepat, akurat dan terstruktur sesuai dengan standar akuntansi yang telah disesuaikan dengan kebutuhan Toko Roti Amaya serta mampu meminimalisir kesalahan dalam menyusun laporan keuangan.

Sistem yang dibangun merupakan sistem manajemen akuntansi keuangan berbasis *website* menggunakan *react js*. *Website* dipilih mengingat kebutuhan perusahaan untuk mengakses laporan melalui platform yang dapat diakses oleh seluruh pihak terutama setelah tersedianya perangkat komputer perusahaan. *React JS* dipilih karena mendukung implementasi *reusable* komponen sehingga pengembang tidak lagi terfokus pada manajemen kode melainkan pada implementasi algoritma dan proses bisnis.

## Rumusan Masalah

Berdasarkan latar belakang diatas maka disimpulkan permasalahan sebagai berikut:

1. Bagaimana hasil analisis dan spesifikasi kebutuhan sistem manajemen akuntansi keuangan berbasis website untuk pelaporan data keuangan di Toko Roti Amaya Malang yang sesuai dengan kebutuhan tersebut?
2. Bagaimanakah rancangan sistem perangkat lunak yang sesuai dengan spesifikasi persyaratan sistem tersebut?
3. Bagaimanakah hasil implementasi sistem perangkat lunak yang sesuai dengan rancangan sistem tersebut?
4. Bagaimanakah hasil pengujian sistem perangkat lunak untuk pelaporan data keuangan di toko Roti Amaya Malang tersebut?

## Tujuan

Adapun tujuan yang ingin dicapai melalui skripsi ini adalah:

1. Menganalisis dan menyusun spesifikasi kebutuhan sistem perangkat lunak untuk pelaporan keuangan di Toko Roti Amaya Malang.
2. Merancang sistem perangkat lunak sesuai kebutuhan untuk sistem perangkat lunak.
3. Mengimplementasikan rancangan sistem perangkat lunak.
4. Menguji sistem perangkat lunak tersebut secara fungsional dan non-fungsional sesuai kebutuhan.

## Manfaat

Manfaat yang ingin dicapai melalui skrispsi ini adalah:

Sistem dapat digunakan oleh Perusahaan Toko Roti Amaya dalam membantu membuat laporan keuangan yang valid dan efisien.

Sebagai upaya dalam membantu mewujudkan industri 4.0 bagi usaha kecil dan menengah di Kota Malang.

## Batasan Masalah

Beberapa batasan pembuatan sistem adalah sebagai berikut:

Sistem dibuat untuk menangani laporan keuangan yang bersumber dari transaksi yang berhubungan dengan:

Sumber pemasukan berasal dari penjualan harian serta kerja sama penjualan dengan pihak kedua.

Pengeluaran kas, meliputi: modal bahan baku, perawatan alat mesin, distribusi kebutuhan penjualan, pengeluaran umum yang akan diolah menjadi jurnal, dan laporan keuangan.

Sistem tidak memasukkan buku besar dan neraca saldo kedalam siklus akuntansi karena menyesuaikan kebutuhan perusahaan.

Sistem berfokus pada pengelolaan dan pelaporan keuangan perusahaan.

Sistem hanya memuat tahap pelaporan keuangan dan tidak menyertakan hasil analisis laporan keuangan.

## Sistematika Pembahasan

Untuk mempermudah dalam menyusun dan memahami isi dari Skripsi ini, menggunakan sistematika sebagai berikut:

**BAB I PENDAHULUAN**

Bab ini membahas mengenai latar belakang, rumusan masalah, tujuan masalah, metedologi, dan sistematika penulisan.

**BAB II LANDASAN KEPUSTAKAAN**

Bab ini membahas pengetahuan yang menjadi dasar teori yang berhubungan erat dan diperlukan dalam mendukung pokok pembahasan skripsi, meliputi: siklus akuntansi, *software development life cycle*, react JS, *restful API*.

**BAB III METEDOLOGI**

Bab ini membahas metode yang digunakan dalam membangun perangkat lunak tersebut dari awal hingga sistem terbentuk.

**BAB IV ANALISIS KEBUTUHAN**

Bab ini memuat hasil analisis kebutuhan berupa kebutuhan fungsional dan non-fungsional yang telah didapat kedalam diagram dan skenario.

**BAB V PERANCANGAN DAN IMPLEMENTASI**

Bab ini memuat perancangan utama perangkat lunak dan perancangan komponen sistem serta hasil akhir dari sistem yang telah dibuat.

**BAB VI PENGUJIAN**

Bab ini memuat cara pengujian sistem serta hasil dari pengujian tersebut berdasarkan kebutuhan fungsional dan non-fungsional yang sebelumnya dibuat.

**BAB VII PENUTUP**

Bab ini memuat kesimpulan dari keseluruhan proses pembuatan perangkat lunak beserta saran dan keberlangsungan.

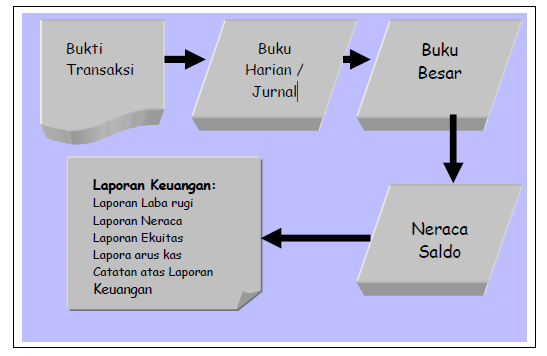
# LANDASAN KEPUSTAKAAN

Bab ini membahas landasan kepustakaan yang digunakan sebagai acuan dalam proses penelitian. Landasan kepustakaan berisi tentang kajian kepustakaan dari penelitian sebelumnya yang memiliki kemiripan dengan topik penelitian ini, selain itu juga berisi teori singkat yang berkaitan dengan permasalahan dan ruang lingkup pembahasan sebagai landasan dalam penelitian ini. Landasan kepustakaan pada penelitian ini diantaranya adalah siklus akuntansi, teknologi pengembangan sistem yang diantaranya siklus akuntansi, *react* *js*, *MySQL*, *HTML*, *restful API* serta *unified modeling language*, *structured analysis and design*, danpengujian perangkat lunak.

## Siklus Akuntansi

### Laporan Keuangan

PSAK No. 1 tentang Penyajian Laporan Keuangan (revisi 2009) menyatakan laporan keuangan adalah suatu penyajian terstruktur dari posisi keuangan dan kinerja keuangan suatu entitas. Dalam pengertian sederhana yang dikutip oleh (Maith, 2013) laporan keuangan adalah laporan yang menunjukkan kondisi keuangan perusahaan pada saat ini atau dalam suatu periode tertentu. (Susilo, 2009) menyatakan bahwa laporan keuangan adalah hasil akhir dari proses akuntansi yang memuat informasi-informasi dan memberikan keterangan-keterangan mengenai data ekonomi perusahaan yang terdiri dari daftar-daftar yang menunjukan posisi keuangan dan hasil kegiatan perusahaan untuk satu periode yang meliputi neraca, laporan laba rugi dan laporan perubahan keuangan. Laporan keuangan memegang peranan penting untuk menilai perkembanganan perusahaan juga prestasi yang dicapai perusahaan pada saat lampau, sekarang dan rencana pada waktu yang akan datang.



Gambar 2.1 Siklus Akuntansi

Sumber: Muawanah & Poernawati (2008)

Proses pelaporan keuangan sering disebut sebagai siklus akuntansi. Dalam gambar 2.1 siklus akuntansi dimulai dari melapirkan bukti transaksi, dicatat kedalam buku harian selanjutnya melakukan *posting* pada buku besar kemudian diringkas dan diakhiri dengan laporan keuangan.

Menurut (Maith, 2013) Laporan keuangan terdiri dari neraca, laporan rugi-laba, laporan perubahan modal dan laporan arus kas. Tetapi, sesuai dengan pernyataan standar akuntansi keuangan No. 1 (revisi 2009) tentang penyajian laporan keuangan terdiri dari beberapa komponen, yaitu: (a) laporan posisi keuangan pada akhir periode; (b) laporan laba rugi komprehensif selama periode; (c) laporan perubahan ekuitas selama periode; (d) laporan arus kas selama periode; (e) catatan atas laporan keuangan.

Laporan Laba Rugi

Laporan laba rugi merupakan laporan yang berisi pendapatan dan beban (pengeluaran) selama periode waktu tertentu. Contoh laporan laba rugi dapat dilihat dalam gambar 2.2. Dalam gambar tersebut dapat dilihat bahwa laporan laba rugi hasil yang didapat pada laporan laba rugi menjadi acuan bagi laporan ekuitas pemilik.

Laporan Ekuitas

Laporan ekuitas yang juga dikenal sebagai laporan penghubung antara laporan laba rugi dengan neraca, merupakan laporan yang berisi perubahan ekuitas (modal) pemilik pada periode waktu tertentu. Contoh laporan ekuitas dapat dilihat dalam gambar 2.3. Dalam gambar tersebut dapat dilihat bahwa laporan ekuitas menjadi acuan bagi neraca.

Neraca

(Muawanah & Poernawati, 2008) menuturkan bahwa neraca merupakan laporan keuangan yang menunjukkan aset, kewajiban, dan ekuitas pemilik per tanggal tertentu. Bentuk neraca ada 2 (dua), yaitu bentuk akun (*account form*) dan bentuk laporan (*report form*). Contoh neraca dapat dilihat dalam gambar 2.4. Dalam gambar tersebut dapat dilihat bahwa neraca menjadi acuan bagi laporan arus kas.

Laporan Arus Kas

Menurut (Muawanah & Poernawati, 2008) Laporan arus kas terdiri dari 3 (tiga) bagian, yaitu: (1) arus kas dari aktivitas operasi, (2) aktivitas investasi, (3) aktivitas pendanaan. Contoh laporan arus kas dapat dilihat dalam gambar 2.5.

Catatan Atas Laporan Keuangan

Catatan atas laporan keuangan adalah pesan khusus yang dituliskan oleh pembuat laporan pada laporan keuangan yang berisi hal-hal yang perlu diketahui oleh pemeriksa laporan.



Gambar 2.2 Contoh Laporan Laba Rugi

Sumber: Muawanah & Poernawati (2008)

Sumber: Muawanah & Poernawati (2008)



Gambar 2.3 Contoh Laporan Ekuitas Pemilik

Sumber: Muawanah & Poernawati (2008)



Gambar 2.4 Contoh Neraca

Sumber: Muawanah & Poernawati (2008)



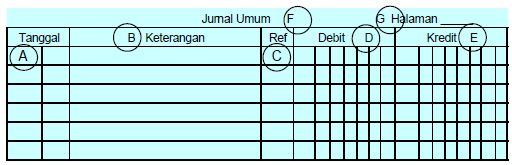
Gambar 2.5 Contoh Laporan Arus Kas

Sumber: Muawanah & Poernawati (2008)

### Jurnal

Umumnya untuk mencatat transaksi yang dilakukan sehari-hari diperlukan sebuah buku. Buku yang digunakan untuk mencatat setiap transaksi tersebut dinamakan buku harian, serta aktivitas pencatatan dinamakan jurnal. Menurut (Muawanah & Poernawati, 2008) Jurnal dilakukan atas dasar dokumen sumber yang disebut dengan bukti transaksi menurut aturan debit kredit.

(Muawanah & Poernawati, 2008) menuturkan bahwa Jurnal merupakan catatan akuntansi pertama setelah bukti transaksi. Fungsi jurnal adalah menyediakan catatan yang lengkap dan permanen dari semua transaksi perusahaan yang disusun dalam urutan kronologis kejadiannya sebagai referensi di masa mendatang. Tujuan mencatat transaksi ke dalam jurnal adalah untuk menunjukkan pengaruh setiap transaksi ke dalam akun perusahaan. Berdasarkan fungsinya jurnal dibagi menjadi 4 bentuk meliputi: jurnal umum, jurnal khusus, jurnal pembalik, dan jurnal penutup.



Gambar 2.6 Contoh Jurnal Umum 2 Kolom

Sumber: Muawanah & Poernawati (2008)

Dalam Gambar 2.6 merupakan ilustrasi jurnal yang dibagi kedalam 7 kolom.

Kolom Tanggal (A)

Kolom ini berfungsi untuk mencatat tanggal setiap transaksi. Pencatatan tanggal harus berurutan sesuai dengan tanggal paling awal hingga paling akhir.

Kolom Keterangan (B)

Kolom ini berfungsi untuk mencatat ayat-ayat jurnal dari setiap transaksi sesuai dengan urutan debet kredit yang dimana ayat jurnal debet harus terlebih dahulu dicatat, untuk setelah itu mencatat ayat jurnal kredit.

Kolom Referensi (C)

Kolom ini berisi ayat jurnal sebagai penanda ayat jurnal yang sudah di-*posting* kedalam buku besar.

Kolom Debet (D)

Kolom ini digunakan untuk mencatat jumlah yang harus didebet dari suatu transaksi.

Kolom Kredit (E)

Kolom ini digunakan untuk mencatat jumlah yang harus didebet dari suatu transaksi.

Judul Jurnal (F)

Halaman Pada Buku Harian (G)

### Harga Pokok Produksi

(Dunia & Abdullah, 2012) mengemukakan bahwa biaya produksi adalah biaya-biaya yang terjadi sehubungan dengan kegiatan manufaktur. Biaya produksi dibagi ke dalam tiga kategori besar, yaitu: bahan langsung (*direct material*), tenaga kerja langsung (*direct labor*), dan biaya overhead pabrik (*manufacturing overhead*). Adapun manfaat dari informasi harga pokok produksi menurut (Mulyadi, 2015) adalah: menentukan harga jual produk, memantau realisasi biaya produksi, menghitung laba rugi periodik, dan menentukan harga pokok persediaan produk jadi dalam proses yang disajikan neraca.

Dalam perhitungan harga pokok produksi yang digunakan toko roti Amaya menggunakan metode *full costing* dimana metode tersebut menghitung seluruh biaya yang dikeluarkan pada proses produksi. Menurut (Yulianti & Saputra, 2017) komponen biaya dalam perhitungan harga pokok produksi diantaranya biaya bahan baku, biaya tenaga kerja langsung, biaya overhead pabrik, serta perlengkapan kantor dan transportasi. Penentuan harga pokok produksi menjadi acuan untuk menentukan harga pokok penjualan suatu produk. Berikut disampaikan rumus perhitungan harga pokok produksi dan harga pokok penjualan:

Perhitungan bahan baku yang digunakan = saldo awal bahan baku + pembelian bahan baku – saldo akhir bahan baku

Perhtiungan biaya produksi = bahan baku yang digunakan + biaya tenaga kerja langsung + biaya overhead produksi + biaya angkut pembelian

Perhitungan harga pokok produksi = total biaya produksi + saldo awal bahan baku dalam proses – saldo akhir bahan baku dalam proses

Perhitungan harga pokok penjualan = harga pokok produksi + persediaan barang awal (persediaan roti diawal bulan x) – persediaan barang akhir (persediaan roti diakhir bulan x)

## Teknologi Pengembangan Sistem

### *Hypertext Markup Language*

Hypertext Markup Language atau HTML adalah bahasa yang digunakan untuk membuat halaman web. Seperti namanya yaiut markup language, html bukanlah bahasa pemrogaman melainkan system untuk mengidentifikasi dan menggambarkan komponen dokumen seperti *headings*, paragraf, dan *list* (Robbins, 2012).

HTML tersusun dari elemen-elemen. Elemen yang dibutuhkan untuk membuat html dinyatakan dengan tag <html>, <head>, dan <body> berikut dengan tag pasangannya. Elemen *head* berisi informasi tentang dokumen terkait sedangkan *body* berisi teks yang tersusun dari link, grafik, paragraf dan elemen lainnya. Pada react js, html sering dituliskan pada jsx. Hal ini dilakukan sebab proses *compile* jsx yang cepat serta penggunaanya yang sederhana. Berikut adalah pola dasar HTML sebagai berikut:

<html>

<head>

Berisi informasi mengenai dokumen HTML

</head>

<body>

Berisi informasi yang ditampilkan pada *web browser*

</body>

</html>

### MySQL

Data merupakan kumpulan informasi yang diperoleh berdasarkan fakta, seperti contoh data angka kelahiran, data nomor telfon warga dikota X. Sedangkan *database* adalah kumpulan besar dari data yang disimpan ke dalam tabel. Tabel ini setidaknya terdiri dari satu kolom atau lebih. Sebuah database juga menghubungkan satu tabel dengan tabel lainnya menggunakan relasi berdasarkan informasi dari data yang disimpan (Suehring, 2002). Untuk membuat suatu database dibutuhkan bahasa khusus yang disebut dengan SQL.

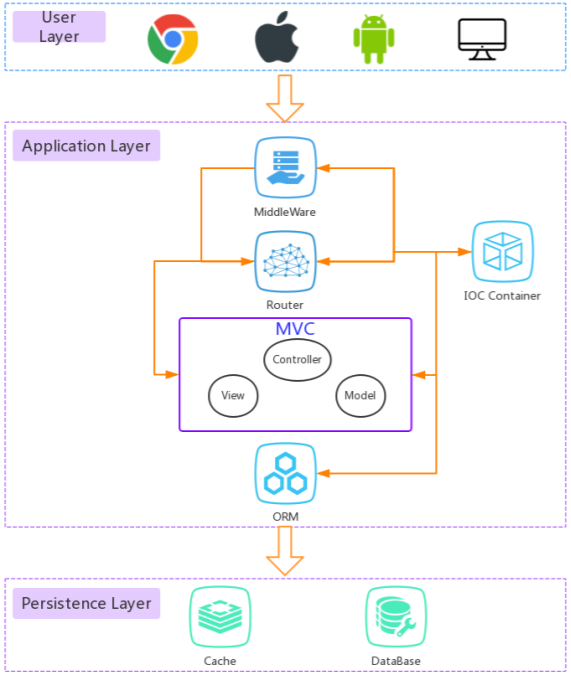
Menurut (Suehring, 2002) SQL atau *Structured Query Language* adalah spesialisasi tipe dari bahasa pemrogaman untuk pengembangan dengan relasi database seperti MySQL, Oracle, PostgresSQL dan lainnya.

Pengertian MySQL Menurut Nugroho (2009 : 1) MySQL adalah sebuah program database server yang mampu menerima dan mengirimkan datanya dengan sangat cepat, multi user, serta menggunakan perintah standar SQL. MySQL merupakan Free Software dibawah lisensi GNU/GPL (General Public License).

MySQL dapat diakses melalui 2 cara yaitu *command prompt* dan *phpmyadmin*. *Phpmyadmin* umum digunakan pada pengembangan website karena merupakan *web-based interface* sehingga mudah untuk digunakan. MySQL memiliki berbagai sintaks untuk menghasilkan satu *database* lengkap. Contoh sintaks dasar seperti create, insert, select, update, from, where. Berikut adalah penggunaan sintaks untuk mengambil data dari tabel bukutamu yang bertempat tinggal diJakarta.

### Restful API

*Application Programming Interface* (API) merupakan perangkat lunak untuk mengintegrasi dua aplikasi yang berbeda secara bersamaan sehingga saling terhubung satu sama lain. Sedangkan *Restful API* atau *Representational state transfer* merupakan arsitektur yang terdapat di dalam API yang memungkinkan sistem komputer untuk mengakses dan memanipulasi representasi dari sumber daya internet menggunakan *uniform* dan *stateless operation*. Setiap sumber daya seperti objek, algoritma, *database* diwakili dengan *identifier* unik yang disebut sebagai *uniform resource identifier* (URI). URI memiliki format sebagai berikut */user/name*. *Stateless Operations* adalah metode HTTP yang digunakan untuk mengkases sumber daya diantaranya: *GET*, *PUT*, *POST*, *DELETE*, *HEADER*, dan *OPTIONS* dan hasilnya dikirim kembali ke pengguna. *Stateless* pada *stateless operation* memiliki arti setiap *server* dalam kluster dapat melayani klien pada setiap permintaan (Chen, Ji, Fan, & Zhan, 2017).



Gambar 2.7 Desain Arsitekur *Restful API*

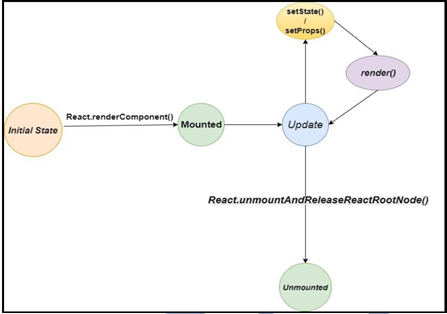
Sumber: (Chen, Ji, Fan, & Zhan, 2017)

(Chen, Ji, Fan, & Zhan, 2017) Menjelaskan desain aristektur *Restful API* dalam gambar 2.7dimulai ketika pengguna mengirimkan permintaan *Restful API* melalui HTTP, *server web* pertama kali menerima *request* untuk selanjutnya dikirimkan melalui PHP. Eksekusi sesungguhnya dimulai ketika PHP *framework* menginisialisasi *routine*. PHP *framework* mengirimkan *request* melalui *middleware engine* untuk menjaring HTTP *request* berdasarkan *base class* sebelum diolah oleh logika. Setelah penjaringan selesai *routing component* bertanggung jawab untuk mengirimkan (*dispatch*) *request* berdasarkan kecocokan antara *request* URI dengan tabel konfigurasi. Selanjutnya, MVC *controller* mengambil alih pengiriman *request*. Secara umum, *controller* bertanggung jawab untuk mengolah logika dan pengambilan data (*data fetch*). Dalam membangun Restful API hanya *model* dan *controller* yang terlibat. Diakhir proses hanya JSON dan data yang terenkripsi (*data encoded*) yang kembali. Dalam gambar 2.7 terdapat ORM atau *Object Relational Mapping* yang merupakan *framework* untuk memanipulasi hubungan antar data yang disediakan oleh *framework* *laravel*.

### React JS

React JS merupakan sebuah *component based* *library* untuk membangun antarmuka pengguna (UI) yang mendukung *reusable* komponen. React JS menggunakan Virtual DOM (*Document Object Model*) untuk melakukan *render* ke sisi server dan hanya menampilkan perubahan yang terjadi tanpa melakukan render ulang seluruh DOM. React JS juga menghadirkan *extensions* bernama JSX dengan fungsi menyertakan *script* HTML kedalam javascript (A M & Sonpatki, 2016).

Menurut (Aggarwal, 2018) React JS menggunakan blok bangunan *user interface* dasar yang sama seperti *website* biasa. Sehingga hanya perlu memasang blok-blok tersebut menggunakan JavaScript. Blok-blok JavaScript yang telah dikompilasi membentuk DOM. DOM dihasilkan melalui metode render() yang ada pada setiap komponen react. Metode React.renderComponent () dipanggil untuk menghasilkan DOM perantara yang nantinya diubah menjadi DOM HTML.



Gambar 2.8 Siklus Hidup *React Component*

Sumber: (Aggarwal, 2018)

Dalam gambar 2.8 digambarkan siklus hidup dari komponen pada react yang terdiri dari 3 tahapan. Pertama adalah *mounting process* dilakukan ketika DOM perantara dibuat. DOM perantara diubah menjadi node kontainer dari DOM asli. *Mounting process* terjadi ketika metode React.renderComponent () dipanggil. Proses kedua adalah *update state*, ketika terjadi perubahan pada komponen dengan menggunakan metode setState () atau perubahan properti menggunakan metode setProps () diikuti dengan penggunaan metode render (), maka DOM dan data disamakan sehingga hanya perubahan terbaru yang ditampilkan. Tahapan terakhir adalah *unmounted* terjadi jika *child* komponen tidak lagi dipanggil pada metode render () (Aggarwal, 2018).

#### React Hooks

Salah satu permasalahan yang ditemukan ketika menggunakan *stateless component* atau fungsi komponen adalah tidak bisa menggunakan *state* atau *lifecyle methods* didalamnya. Pada versi react js sebelumnya ketika menggunakan fungsi komponen sering dijumpai proses *refactor component* menjadi kelas komponen saat *state* atau *lifecycle* dibutuhkan. React JS meluncurkan versi 16.8 dengan menambahkan fungsi terbaru yaitu *react hooks* untuk menjawab permasalahan tersebut. React Hooks diperkenalkan oleh tim React untuk melakukan *state management* dan *side effects* di dalam fungsi komponen. *Hooks* *state* dan *lifecycle methods* dapat digunakan tanpa harus menuliskannya dikelas (Abramov, 2020).

*State management* pada fungsi komponen menggunakan *useState*. Fungsi *useState* digunakan ketika suatu fungsi membutuhkan beberapa *state* sekaligus. Fungsi *useState* digunakan sebagai pengganti *this.setState* pada kelas komponen. Fungsi *useState* hanya memiliki satu argumen yaitu *state* awal. *State* awal tidak harus berbentuk objek namun juga dapat berbentuk angka atau *string*. Fungsi *useState* mengembalikan 2 nilai yaitu *state* saat ini dan fungsi untuk melalukan pembaruan. Seperti contoh kode dibawah terdapat fungsi *useState* dengan argumen angka 0 sebagai nilai *state* awal untuk variable *count*. Fungsi *useState* mengembalikan nilai *count* dan setCount sebagai fungsi untuk melakukan pembaruan (Abramov, 2020).

import React, { useState } from 'react';

function Example() {

const [count, setCount] = useState(0);

Komponen biasa melakukan proses efek samping seperti pengambilan data atau interaksi dengan DOM. Proses efek samping pada fungsi komponen menggunakan *useEffect*. Pada kelas komponen proses tersebut dilakukan di dalam *lifecycle methods* seperti *componentWillMount*, *componentDidMount*, *componentWillUnmount*. Fungsi useEffect dijalankan setelah komponen selesai *render*. *React* mengingat dan memanggil fungsi tersebut setelah DOM melakukan pembaruan. Kode dibawah merupakan contoh penggunaan fungsi *useEffect* untuk melakukan pengambilan data. Fungsi *useEffect* dijalankan ketika terjadi perubahan pada variabel *result*.

import React, { useState, useEffect } from 'react';

const Result = () => {

const [result, setResult] = useState({});

useEffect(() => {

async function fetchData() {

await fetch("https://jsonplaceholder.typicode.com/posts ")

.json()

.then(result => setResult(result)) }

Selain *useEffect* dan *useState*, *react hooks* juga memiliki fungsi bernama *useContext*. Fungsi *useContext* digunakan untuk berbagi data global pada komponen *react js*. Alasan menggunakan *useContext* adalah react js tidak dapat mengakses *state* secara langsung apabila *state* berada di komponen *parent* sedangkan komponen yang akan menggunakan *state* tersebut berada di komponen *child*. Cara untuk mengakses *state* tersebut dapat menggunakan *props*, namun menjadi masalah baru jika komponen yang digunakan semakin bertambah. Solusi lain yaitu dengan menggunakan *useContext* (Abramov, 2020).

### MDB React

*Material Design Bootstrap* atau MDBootstrap adalah *framework* yang digunakan untuk membangun *responsive web applications* serta dapat digunakan untuk membangun *mobile* atau *desktop applications*. MDB React dibangun dengan menggabungkan *react* dan *bootstrap* sehingga dapat digunakan untuk membangun aplikasi berbasis *react* dengan tetap menggunakan *bootstrap* sebagai *framework*.

MDB React juga dapat digunakan bersama dengan material-ui. MDB React digunakan sebagai *framework* utama yang didalamnya juga memuat kode material-ui. Pada MDB React tidak lagi menggunakan fungsi umum pada html melainkan menggunakan kode tersendiri seperti *MDBBtn* berfungsi untuk membuat tombol, *MDBCol* dan *MDBRow* untuk membuat kolom dan baris. Contoh sederhana penggunaan *MDB React* dan *Material-ui* untuk menampilkan tombol posting jurnal seperti sintaks dibawah ini:

import { MDBCard, MDBCardBody, MDBRow, MDBCol, MDBBtn} from 'mdbreact';

return (

        <div>

            <MDBRow>

                <MDBCol lg='9'>

                    <h4>Jurnal Umum | Perhari</h4>

                </MDBCol>

                    <MDBBox display="flex" justifyContent="center">

                        {save && (  <MDBBtn onClick={handlePosting} disabled={posting}>

                                Posting Jurnal

                            </MDBBtn>

                        )}

            </MDBBox>

            </MDBRow>

### *Material-UI*

Material-UI merupakan *framework* khusus react js yang bersifat *open* untuk merancang halaman web berdasarkan prinsip *material design*. Material-UI bersifat mandiri yaitu hanya digunakan ketika dibutuhkan dan tidak bergantung pada *global style sheet* seperti css pada umumnya. Contoh sederhana penggunaan material-ui yaitu dengan menggunakan *import* seperti sintaks dibawah ini:

import Button from ‘@material-ui/core/Button’;

function App () {

return (

<Button variant = “contained” color = “primary”> Hello Word </Button> )}

### Lumen

Lumen merupakan *open-source* PHP *micro-framework* untuk membangun *microservices* dan API. Terdapat 2 hal utama sebagai syarat penggunaan lumen yaitu menggunakan PHP versi 7.3 keatas dan *composer* untuk mengatur *dependencies*. Saat ini lumen dapat digunakan bersama dengan *MySQL*, *Postgres*, *SQLite*, *SQL Server* (Otwel, 2020). Lumen menggunakan beberapa komponen yang berasal dari framework laravel diantaranya:

*Routing* baik *routing* dasar, penamaan *routing*, *routing* berkelompok seperti *middleware*.

*Authentication* menggunakan mekanisme *stateless* seperti *tokens*.

*Cache driver* sebagai media penyimpanan seperti *database*, *memcache* dan *redis*.

## *Software Development Life Cycle*

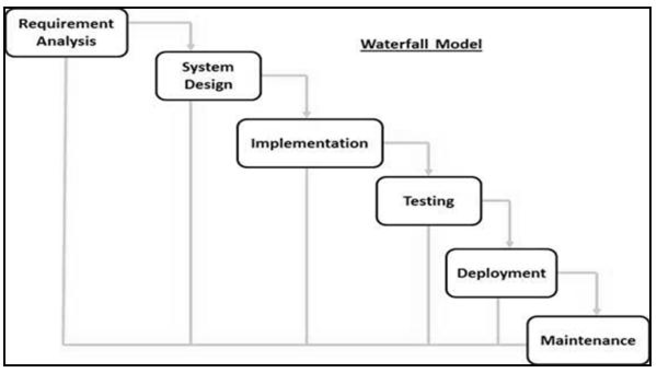
Menurut (Luea, Loo, Tham, & Tan, 2012) *Software Development Life Cycle* merupakan proses pengembangan atau pemeliharaan sistem perangkat lunak. Sumber lain mengatakan bahwa *software development life cycle* merupakan deskripsi prespektif tentang bagaimana sistem perangkat lunak dikembangkan. Proses deskriptif menjelaskan sejarah tertentu tentang bagaimana sistem dikembangkan. Proses ini juga digunakan untuk memahami dasar dan meningkatkan pemahaman dalam pengembangan sistem perangkat lunak serta (Rajmohan & Suburayan, 2010).

SDLC dibagi menjadi 2 pendekatan yaitu pendekatan tradisional dan modern. Pendekatan tradisional seperti model *waterfall*, v-model, RUP serta agile merupakan contoh pendekatan secara modern. Pembeda antara pendekatan tradisional dan modern adalah pendefisian dan dokumentasi kebutuhan. Pendekatan tradisional mengharuskan pendefinisian kebutuhan di awal serta seluruh dokumentasi kebutuhan lengkap sebelum menjalankan proses selanjutnya. Sedangkan pendekatan modern yang berdasar pada pengembangan secara *incremental* dan *iterative* dimana setiap fase dijalankan berulang sesuai dengan respon pengguna, sehingga adanya kemungkinan pendefisian kebutuhan dilakukan ketika proses lain berjalan (Rajmohan & Suburayan, 2010).

Secara garis besar proses pengembangan sistem perangkat lunak memilki 5 tahapan diantaranya persiapan, perolehan, pengujian, pengerjaan, pemeliharaan. Pada umumnya fase SDLC terdiri dari inisialisasi, perencanaan, desain, pengembangan, pegujian, pengerjaan, dan pemeliharaan. Jumlah dari fase yang ada pada setiap SDLC bergantung pada karakteristik projek dan manajemen jumlah pekerja dalam setiap projek. Sehingga jumlah fase antara SDLC pada projek satu dengan lainnya tentu bisa berbeda (Rajmohan & Suburayan, 2010).

### Model *Waterfall*

Model *waterfall* merupakan model SDLC tertua dan menjadi model yang sering digunakan pada projek pemerintah dan perusahaan besar lain. Model ini sering digunakan karena sederhana, selain itu model ini bekerja dengan baik untuk proyek dengan kontrol kualitas tinggi karena dokumentasi dan perencanaan yang sudah matang diawal, sehingga model *waterfall* dianggap cocok untuk projek-projek kecil dengan kebutuhan yang telah terdefinisi di awal. Setiap tahapan pada model *waterfall* tidak tumpah tindik, artinya tahapan pertama sampai dengan akhir dilakukan dalam satu kali proses sebelum memulai proses selanjutnya.



Gambar 2.9 SDLC model *waterfall*

Sumber: Arora, Arora, & India (2016)

Model *waterfall* dilakukan secara *sequential* dari atas menuju kebawah dengan tahapan pertama analisis kebutuhan, desain, koding, pengujian dan pemeliharaan seperti dalam gambar 2.9. Berikut penjelasan singkat dari setiap tahapan model *waterfall*:

Analisis kebutuhan, pada tahap ini pengembang mengumpulkan informasi dari pengguna untuk selanjutnya menganalisis informasi tersebut menjadi dokumentasi kebutuhan. Selain itu, pengembang menetapkan perjanjian kepada pengguna terkait spesifikasi dan fitur perangkat lunak.

Desain, pada tahap ini kebutuhan diformulasi menjadi desain yang mudah dipahami oleh pengembang lain seperti desain algoritma, desain antar muka, desain arsitektur sistem, konsep skema database, struktur data. Pada tahap ini juga dijabarkan masalah beserta penyelesaiannya.

Koding, seluruh kebutuhan diolah menggunakan bahasa tertentu sehingga bisa digunakan pada lingkungan sebenernya.

Pengujian, perangkat lunak diuji sesuai dengan kondisi yang mungkin terjadi serta pengujian setiap kebutuhan sesuai dengan dokumentasi diawal. Pada tahap ini juga *bugs* ditemukan dan diperbaiki.

Pemeliharaan, tahap ini dilakukan setelah sistem rilis, dimana ketika terdapat modifikasi, peningkatan, perbaikan kerusakan.

## UML (*Unified Modeling Language*)

*Unified Modeling Language* adalah teknik pemodelan sistem perangkat lunak sebagai alat pendokumentasian dan menjelaskan spesifikasi sistem (Mulyani, 2016). UML terdiri dari banyak elemen-elemen grafis yang digabungkan dalam bentuk diagram. Tujuan representasi elemen-elemen grafis ke dalam diagram adalah untuk menyajikan beragam sudut pandang dari sebuah sistem berdasarkan fungsi masing-masing diagram tersebut. Kumpulan dari beragam sudut pandang inilah yang disebut sebuah model (Satzinger, 2012).

Analisis kebutuhan sistem merupakan tahapan awal dan utama untuk membuat pondasi dalam langkah pengembangan sistem (Indrajit, 2001). Analisis kebutuhan sangatlah penting bagi sebuah sistem karena pada bagian ini pengembang membutuhkan informasi yang dapat membantu dalam membangun sistem yang akan dibangun.

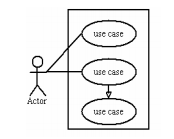
### Use Case Scenario

*Use Case Scenario* merupakan penjelasan secara tekstual dari sekumpulan skenario interaksi. Setiap skenario mendeskripsikan urutan aksi/langkah yang dilakukan aktor ketika berinteraksi dengan sistem, baik yang berhasil maupun gagal. UC scenario dijelaskan secara tekstual dalam beberapa format tergantung kebutuhannya, yaitu singkat (*brief*), informal (*casual*), atau lengkap (*fully dressed*) (Larman, 2005), yang bisa dijelaskan dalam bentuk tabel dengan 1 kolom atau 2 kolom. Pada format singkat, penjelasan diberikan cukup 1 paragraf yang mengacu hanya pada skenario yang berhasil. Pada format informal, penjelasan diberikan dalam beberapa paragraf yang mencakup semua skenario, baik yang berhasil maupun gagal. Sedangkan, pada format lengkap, penjelasan dibuat secara detil disertai dengan bagian bagian pendukung yang penting.

### *Use Case* Diagram

*Use case* diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Sebuah use case merepresentasikan sebuah interaksi antara aktor dengan sistem (Dharwiyanti & Satrio Wahono, 2003). Dalam pembuatan *use case* diagram, terdapat notasi – notasi yang perlu diperhatikan yaitu:

Sistem



Gambar 2.10 Notasi Sistem

Sumber: (Henderi, 2009)

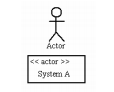
Gambar batasan (*boundries*) sebuah sistem menggunakan persegi panjang yang berisi *use case - use case* (Henderi, 2009). Aktor – aktor yang terlibat pada setiap *use case* ditempatkan pada bagian luar *boundaries* sistem.

*Use Case*

Gambar 2.11 Notasi *Use Case*

Sumber: (Henderi, 2009)

Gambar *use case* menggunakan lingkaran berbentuk bulat telur (*ovals*) Beri nama *ovals* tersebut dengan kata kerja (*verbs*) yang menggambarkan fungsi - fungsi sistem (Henderi, 2009).

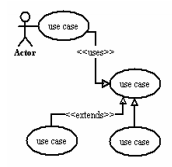
Aktor

Gambar 2.12 Notasi *Actor*

Sumber: (Henderi, 2009)

*Actor* adalah seseorang atau sesuatu yang harus berinteraksi dengan sistem atau sistem yang dibangun/dikembangkan (Henderi, 2009). Biasanya untuk penamaan aktor disesuaikan dengan nama pengguna.

*Relationship*



Gambar 2.13 Notasi *Relationship*

Sumber: (Henderi, 2009)

*Relationships* merupakan gambaran relasi/hubungan antara sebuah *actor* dan *use case* dengan sebuah garis sederhana. Dalam relasi-relasi antara use cases, ada yang menggunakan tanda anak panah-anak panah pada ”*uses*” yang lainnya atau gunakan ”*extends*”. Relasi yang menggunakan ”*uses*” mengartikan bahwa ada *use case* yang dibutuhkan oleh *use case* yang lain untuk bisa melakukan sebuah permintaan sedangkan yang me nggunakan ”*extends*” mengartikan bahwa ada beberapa pilihan tertentu pada tingkatan yang lebih bawah yang ada pada *use case* (Henderi, 2009).

## *Structured Analisys and Design*

Pendekatan perancangan terstruktur atau *structured analisys and design* ditemukan pada tahun 1977, merupakan hasil turunan pemrogaman terstruktur. Fokus dari pendekatan terstruktur adalah memecah masalah dalam proses bisnis menjadi bagian-bagian kecil untuk nantinya disatukan kembali menjadi satu kesatuan utuh.

Tujuan dari penggunakan pendekatan terstruktur adalah memecah permasalahan yang komplek dalam suatu proses bisnis sehingga menghasilkan sistem yang mudah untuk dipelihara, fleksibel dan memiliki dokumentasi yang lengkap sesuai dengan anggaran biaya pengembangan sistem serta meningkatkan produktivitas dan kualitas sistem.

Pendekatan terstruktur terdiri dari 2 aktivitas yaitu struktur analisa dan struktur desain. Struktur analisa bertujuan untuk menangkap struktur detail sistem dapat dipahami oleh pengguna. Pada struktur analisa proses utama sistem dianalisa dan aliran data antar proses direpresentasikan dengan diagram (Mall, 2018).

Struktur desain bertujuan untuk mendefinisikan solusi yang cocok untuk diimplementasikan pada bahasa pemrogaman. Pada struktur desain algoritma dan struktur data dari setiap modul didesain sehingga dapat langsung diimplementasi oleh pengembang sesuai bahasa pemrograman yang digunakan (Mall, 2018). Dalam mengimplementasi pendekatan terstruktur digunakan alat dan teknik tertentu, diantaranya *data flow diagram*, *structural diagram* dan *entity relationship diagram*.

### *Data Flow Diagram*

*Data flow diagram* adalah model grafik hierarki dari suatu sistem yang menunjukkan aktivitas atau fungsi pemrosesan yang berbeda yang dilakukan sistem dan pertukaran data di antara fungsi-fungsi tersebut. DFD bukan hanya merepresentasikan aliran data namun juga digunakan untuk mendefinisikan alur sebuah proses (Mall, 2018).

*Data flow diagram* merupakan diagram hirarki yang menggambarkan sistem berdasarkan level. Level 0 atau *context diagram* merupakan level tertinggi yang terdiri dari satu proses besar. Level 0 menggambarkan keseluruhan sistem dan dinamai sesuai dengan nama sistem. Level 1 terdiri dari 3 – 7 proses utama yang merepresentasikan kebutuhan fungsional sistem. Pada *data flow diagram* juga dikenal istilah dekomposisi. Dekomposisi adalah menguraikan proses pada level sebelumnya menjadi sub-proses pada level selanjutnya. Setiap proses didekomposisi menjadi 3 – 7 sub-proses. Penomoran pada setiap proses mengikuti level dari proses tersebut, sebagai contoh proses A pada level 1 dituliskan sebagai 1.0 dan sub-proses A pada level 2 dituliskan dengan 1.1 begitu terus hingga level selanjutnya. Dalam gambar 2.14 terdapat notasi yang digunakan untuk menggambarkan *data flow diagram* (Mall, 2018).

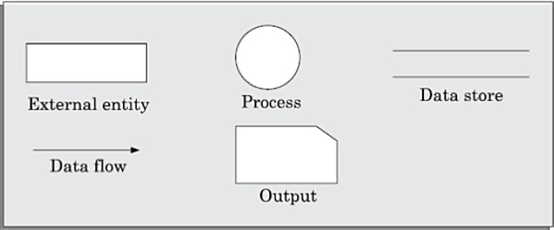
*Function symbol*, sebuah fungsi digambarkan dengan lingkaran. Simbol fungsi diberi anotasi sesuai dengan nama fungsi sistem.

*External entity symbol*, entitas eksternal seperti *library* direpresentasikan dengan kotak. Entitas eksternal adalah entitas fisik di luar sistem yang berinteraksi dengan sistem dengan memasukkan data ke sistem atau dengan menggunakan data yang dihasilkan oleh sistem. Selain pengguna, simbol entitas eksternal dapat digunakan untuk mewakili perangkat keras dan perangkat lunak eksternal seperti perangkat lunak aplikasi lain yang berinteraksi dengan perangkat lunak yang dimodelkan.

*Data flow symbol*, aliran data digambarkan dengan busur terarah. Simbol aliran data mewakili aliran data yang terjadi antara dua proses atau antara entitas eksternal dan proses dalam arah panah aliran data. Simbol aliran data biasanya dianotasikan sesuai dengan nama data.

*Data store symbol*, penyimpanan data direpresentasikan dengan dua garis sejajar. Penyimpanan data mewakili struktur data atau file fisik pada disk. Setiap penyimpanan data terhubung ke proses melalui simbol aliran data. Arah panah aliran data menunjukkan apakah data sedang dibaca dari atau ditulis ke dalam penyimpanan data. Panah yang mengalir masuk atau keluar dari penyimpanan data secara implisit mewakili seluruh data penyimpanan data dan karenanya panah yang menghubungkan ke penyimpanan data tidak perlu dianotasi dengan nama item data yang sesuai.

*Output symbol*, Simbol output digunakan saat hard copy diproduksi.



Gambar 2.14 Notasi *Data flow diagram*

Sumber: (Mall, 2018)

### *Structure Chart*

*Structure chart* atau grafik strukturmerupakan transformasi dari hasil *data flow diagram* yang digunakan untuk merepresentasikan arsitektur sistem. Pada struktur desain menggambarkan modul yang menyusun sistem, ketergantungan atau *dependencies* antar modul dan parameter yang diteruskan di antara modul yang berbeda. Terdapat 2 strategi yang digunakan untuk merubah *data flow diagram* menjadi grafik sturktur yaitu (Mall, 2018):

*Transform analysis* atau analisis transformasi mengidentifikasi fungsi utama modul, masukkan serta keluaran data dari modul. Proses masukkan merubah masukkan berbentuk fisik menjadi masukkan berbentuk logika. Proses keluaran merubah keluaran fisik menjadi keluaran berbentuk logika. Grafik struktur menyertakan sub-proses yang dibutuhkan setiap proses utama atau disebut sebagai *factoring*. *Factoring* menambahkan modul untuk baca dan tulis, modul untuk menangani error, inisialisasi dan proses *terminate*.

*Transaction analysis* atau analisis trasaksi merupakan alternatif dari analisis transformasi yang digunakan untuk mendefinisikan spesifik proses. Untuk setiap transaksi, lacak masukkan dan keluaran. Setiap proses dipetakan sesuai dengan modul pada grafik struktur.

Notasi dari grafik struktur dijelaskan sebagai berikut (Mall, 2018):

*Rectangular boxes*, modul digambarkan sebagai kotak dengan nama sesuai nama modul tersebut.

*Module invocation arrows*, anak panah yang menghubungkan dua modul selama eksekusi program yang diteruskan dari satu modul ke modul lain.

*Data flow arrows*, anak panah yang lebih kecil muncul bersama dengan *module invocation arrows*. Penamaan *data flow arrows* disesuaikan dengan data yang terlibat.

*Library modules*, sebuah modul yang merupakan gabungan beberapa modul digambarkan dengan kotak dua garis.

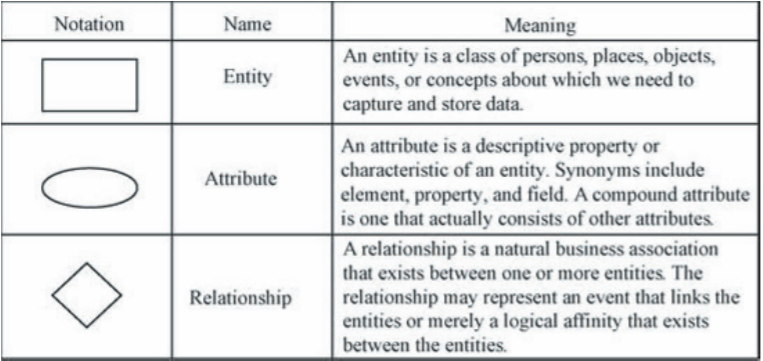
*Selection*, belah ketupat menggambarkan pilihan dari hasil keluaran sebuah modul.

*Repetition*, anak panah setengah lingkaran menggambarkan sebuah modul yang digunakan kembali.

### *Entity Relationship Diagram*

*Data modelling* atau pemodelan data adalah teknik mengatur dan mendokumentasikan data sistem. Pemodelan data disebut juga sebagai pemodelan basis data karena model data pada akhirnya diimplementasikan sebagai basis data (Li & Chen, 2009). *Entity relationship diagram* merupakan diagram yang menggambarkan struktur database secara konseptual. ERD menggunakan sebuah teknik untuk mengatur atribut dengan mengelompokkan data sehingga terbentuk atribut yang tidak berulang, stabil, fleksibel serta entitas adaptif atau sering disebut dengan *normalization*.

ERD dinyatakan sebagai entitas, atribut dan hubungan antar entitas yang dijelaskan dalam gambar 2.15 (Al-Masree, 2015). Entitas menggambarkan data yang disimpan seperti contoh orang, tempat, objek atau konsep. Atribut mendeskripsikan properti atau karakter dari entitas. Atribut diklasfikasi menggunakan *key* atau kunci. *Key* merupakan atribut yang mengasumsikan nilai unik untuk setiap entitas. Key dibagi ke dalam beberapa jenis seperti *primary key* yaitu jenis atribut yang paling sering digunakan untuk mengidentifikasi entitas tunggal secara uni. *Relationship* mewakili peristiwa yang menghubungkan entitas atau hanya afinitas logis yang ada di antara entitas. *Relationship* atau hubungan didefinisikan berdasarkan jumlah kemunculan minimum dan maksimum sebuah entitas dengan entitas lainnya atau disebut dengan kardinalitas. Kardinalitas bersifat dua arah dan harus didefinisikan di kedua arah setiap hubungan (Li & Chen, 2009).



Gambar 2.15 Notasi *Entity Relationship Diagram*

Sumber: (Li & Chen, 2009)

## Pengujian Perangkat Lunak

Pengujian adalah fase terakhir dalam tahap pengembangan sistem sebelum sistem dirilis. Pengujia memiliki serangkain proses uji perangkat lunak untuk memastikan perangkat lunak tidak terdapat kecacatan (*bugs*) dan bekerja sesuai dengan kebutuhan. Pengujian memiliki proses tersendiri yang bersifat *iterative*, proses tersebut terdiri dari: analisa pengujian, desain pengujian, eksekusi pengujian, identifikasi masalah dan menyelesaikan masalah.

Pada pengujian perangkat lunak terdapat beberapa teknik yang digunakan untuk yang disesuaikan dengan fase pengujian. Pada umumnya pengujian dibagi menjadi tiga strategi sesuai dengan lingkungan pengujian yaitu *black box*, *white box* dan *grey box.* *Black box* menguji perilaku *internal* sistem seperti fungsionalitas, performa, kegunaan hingga *acceptance testing* dan *beta testing*. White box menguji perilaku *external* sistem seperti *unit testing*, *code coverage*, *code review*, *error handling testing*, *complexity testing*, *path condition testing*. *Grey box* menguji baik *internal* dan *external* sistem seperti *integration testing* dan *regression testing*.

### *White Box Testing*

Pengujian kotak putih atau *white box testing* atau *code based testing* merupakan pengujian berdasarkan kode program dari sistem yang diuji. Pengujian kotak putih memungkinkan untuk mengidentifikasi kasus uji berdasarkan bagaimana sebuah fungsi diimplementasikan. Teknik yang umum digunakan untuk melakukan pengujian kotak putih adalah *basis path testing* (Jorgensen, 2014). Aspek pengujian pada pengujian kotak putih diantaranya:

Memastikan bahwa semua jalur algoritma telah diuji minimal sekali.

Menguji seluruh logika keputusan (*true* atau *false*).

Mengeksekusi seluruh *loop* dalam batasan yang ditentukan.

Validasi struktur data internal.

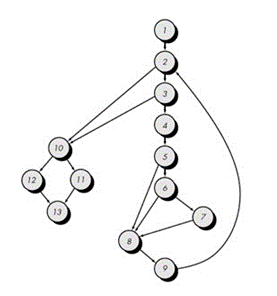
Pengujian jalur dasar merupakan teknik pengujian kotak putih berdasarkan ukuran tingkat kompleksitas dari algoritma hasil perancangan. Langkah-langkah untuk melakukan pengujian jalur dasar diantaranya:

1. Mendefinisikan *flow graph* berdasarkan *mapping* dari *flow chart* atau struktur algoritma.
2. Menentukan ukuran kompleksitas (*cyclomatic complexity*).
3. Mendefinisikan kasus uji.

Menurut (Pressman, 2001) *Cyclomatic complexity* merupakan pengukuran metrik untuk perangkat lunak yang menyediakan ukuran kuantitatif dari kompleksitas logika suatu program. Ketika digunakan dalam konteks metode pengujian jalur dasar, nilai yang dihitung untuk *cyclomatic complexity* menentukan jumlah jalur independen dan memastikan bahwa semua pernyataan dieksekusi setidaknya sekali. Sehingga didapatkan rumus V(G) = E – N + 2, dimana nilai E adalah *edges* dan nilai N adalah *nodes*. Selain dengan menggunakan rumus tersebut*, cyclomatic complexity*dapat ditentukan dengan dua cara lain:

Menentukan jumlah *region*dari *flow graph*.

Menentukan jumlah node simpul/bercabang ditambahkan dengan angka 1.



Gambar 2.16 *Flow Graph*

Sumber: Pressman (2001)

Berdasarkan gambar 2.16 perhitungan *cyclomatic complexity* dengan menggunakan 3 cara sebagai berikut:

1. V(G) = 17 edges –
2. ][ 13 nodes + 2 = 6
3. V(G) = 6 region
4. V(G) = 5 node simpul + 1 = 6

### *Black Box Testing*

Pengujian kotak hitam atau sering disebut *black box testing* merupakan pengujian berdasarkan pada kebutuhan fungsional. Pengujian kotak hitam berfokus mengamati hasil keluaran berdasarkan nilai tertentu tanpa mempertimbangkan detail kode program. Terdapat beberapa teknik yang dapat digunakan untuk melakukan pengujian kotak hitam, diantaranya *Boundary Value Analysis*, *Equivalence Class Partitioning*, *Decision Table Testing*, *Cause Effect* Graph. Pengujian BVA digambarkan dengan sebuah fungsi yang memiliki dua variabel yaitu x1 dan x2 dengan batasan yang tidak mungkin disebutkan, sehingga didefinisikan sebagai a ≤ x1 ≤ b dan c ≤ x2 ≤ d (Jorgensen, 2014).

*Boundary Value Analysis* merupakan teknik pengujian kotak hitam yang berfokus pada batas atas dan batas bawah dari suatu nilai yang diujikan kepada sistem. Menurut (Andriansyah, 2018) adapun hal yang mendasari *Boundary Value Analysis* adalah:

BVA fokus pada suatu batasan nilai dimana kemungkinan terdapat cacat yang tersembunyi.

BVA mengarahkan pada pemilihan kasus uji yang melatih nilai-nilai batas. BVA merupakan desain teknik kasus uji yang melengkapi *Equivalence classtesting*. Dari pada memfokuskan hanya pada kondisi masukkan, BVA juga menghasilkan kasus uji dari domain keluaran.

BVA menguji input disekitar batas atas maupun batas bawah sebuah rentang nilai yang valid.

BVA menguji nilai maksimal dan minimal.

BVA menguji batas struktur data yang digunakan, misal ukuran *array*.

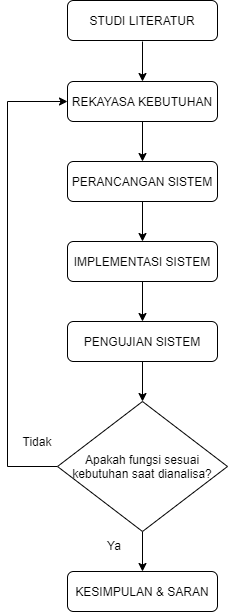
### *System Testing*

Pengujian sistem dilakukan ketika sistem dalam keadaan lengkap. Pengujian sistem tidak membutuhkan pengetahuan mengenai kode program. Tujuan dari pengujian sistem adalah melatih sistem secara keseluruhan. Setiap pengujian harus dilakukan dengan tujuan yang berbeda, hal ini sebagai bentuk verifikasi sistem telah terintegrasi dan berfungsi sesuai fungsionalitas (Jan, Shah, Johar, Shah, & Khan, 2016).

Pengujian kesesuaian atau *compability testing* merupakan pengujian non-fungsional untuk memeriksa perilaku sistem saat berada dilingkungan yang berbeda. *Compability testing* dilakukan pada berbagai jenis lingkungan seperti sistem operasi, jaringan, perangkat keras, perangkat lunak, *browser* dan *database* (Jan, Shah, Johar, Shah, & Khan, 2016). *Compability testing* membutuhkan alat uji khusus seperti *Sortsite*, *lambda test*, *BrowserStack*, *Functionize* dan *Experitest*.

# METODOLOGI PENELITIAN

Langkah-langkah yang dilakukan dalam membuat sistem informasi meliputi: studi literatur, rekayasa kebutuhan, perancangan sistem, implementasi, pengujian, serta kesimpulan dan saran.



Gambar 3.1 Metodologi Penelitian

## Studi Literatur

Studi literatur digunakan untuk mengkaji mengenai pengembangan sistem manajemen akuntansi keuangan dan teori-teori yang digunakan untuk membantu pengembangan perangkat lunak tersebut.

Adapun teori dan pustaka yang berkaitan dengan skripsi ini meliputi:

Laporan keuangan

Jurnal

Harga pokok produksi

React JS

MDB React

Material-UI

MySQL

HTML

Restful API

Analisa Kebutuhan

Perancangan Kebutuhan

## Rekayasa Kebutuhan

Rekayasa kebutuhan merupakan satu proses besar yang melibatkan beberapa proses lain diantaranya elisitasi kebutuhan, spesifikasi kebutuhan serta validasi dan verifikasi kebutuhan. Elisitasi kebutuhan dilakukan untuk mengumpulkan dan mengetahui kebutuhan yang digunakan dalam pengembangan sistem manajemen akuntansi keuangan, dimana elisitasi kebutuhan mengacu pada beberapa perangkat lunak sejenis. Elisitasi kebutuhan dilakukan dengan metode wawancara serta merujuk pada dokumen pendukung seperti buku dan jurnal. Setelah memahami permasalahan, hasil elisitasi kebutuhan dimodelkan kedalam *use diagram* dan *use case scenario*.

## Perancangan

Tahapan perancangan dimulai dari perancangan kebutuhan fungsional dan non-fungsional dari sistem. Perancangan dilakukan dengan pendekatan *structured analysis and design*. Perancangan dimulai dari *use case diagram* dan *use case scenario* dilanjut dengan perancangan struktur analisa menggunakan *data flow diagram*. Perancangan struktur desain atau arsitekutr sistem menggunakan *structure chart*. Perancangan basis data menggunakan *entity relationship diagram*. Perancangan antarmuka untuk merancang tampilan atau antarmuka sistem yang dibangun.

## Implementasi

Implementasi merupakan tahapan pembangunan perangkat lunak dengan acuan dari hasil rekayasa kebutuhan dan perancangan yang telah dilakukan. Tahap implementasi diantaranya:

Implementasi dari sisi akuntansi berupa rumus matematika yang digunakan untuk membuat laporan keuangan.

Implementasi sistem menggunakan HTML, Javascript, React JS, dan Restful API dengan menggunakan *framework material-ui* dan MDBReact pada tampilan dan tahapan pengembangan sistem.

Implemetasi basis data menggunakan DBMS MySQL pada server *locallhost*.

## Pengujian

Pengujian dilakukan dengan tujuan untuk mengetahui seberapa tepat antara kebutuhan fungsional dan non-fungsional yang telah dibuat:

Metode pengujian yang digunakan:

*White Box Testing*

*Black Box Testing*

## Kesimpulan

Pengambilan kesimpulan dilakukan setelah semua tahap perancangan implementasi, dan pengujian telah dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis metode yang diterapkan. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi serta memberikan pertimbangan atas pengembangan metode selanjutnya.

# REKAYASA KEBUTUHAN

Rekayasa kebutuhan menjelaskan deskripsi sistem dan analisis kebutuhan sistem. Deskripsi sistem dan analisis kebutuhan sistem diperlukan untuk memberikan gambaran sistem yang dibuat serta dapat menjadi dokumentasi yang akan membantu pengembangan sistem. Hasil analisis kebutuhan yang akan dijelaskan pada sub-bab selanjutnya merupakan hasil akhir dari implementasi dan verifikasi dengan pihak *end-user*.

## Deskripsi Sistem

Sistem manajemen akuntansi keuangan perusahaan merupakan sistem yang diperuntukkan bagi akuntan serta karyawan terkait seperti marketing serta operator perusahaan dalam melakukan pelaporan keuangan baik pemasukkan serta pengeluaran. Sistem mencakup pada pencatatan transaksi, *review* transaksi, *posting* jurnal, serta *generate* laporan keuangan. Setiap pengguna memiliki hak akses masing-masing, hak akses digunakan sebagai batasan bagi pengguna terkait data serta fungsi yang dapat diakses. Marketing memliki akses sampai pencatatan transaksi. Operator perusahaan memiliki akses untuk pencatatan transaksi, menambahkan produk, rekapitulasi produk dan rekapitulasi bahan baku. Akuntan memiliki akses yang mencakup *review* transaksi, posting jurnal, serta *generate* laporan keuangan.

Alur sistem dimulai dari pencatatan transaksi yang dibagi 2 yaitu pengeluaran dan pemasukkan oleh divisi marketing dan operator keuangan. Selanjutnya transaksi tersebut divalidasi kebenarannya oleh akuntan. Jika hasilnya valid maka transaksi dapat diteruskan ke proses posting jurnal. Jika transaksi tidak valid maka akuntan dapat menolak transaksi yang artinya transaksi harus diperbaiki sebelum dilanjutkan ke proses posting jurnal. Proses posting jurnal dilakukan secara otomatis, artinya pengguna hanya perlu menekan tombol maka otomatis sistem menampilkan hasil jurnal. Tahap terakhir adalah mengubah hasil posting jurnal kedalam laporan keuangan yang prosesnya juga dilakukan secara otomatis. Laporan keuangan berupa dokumen yang sudah disiapkan sesuai standar akuntansi dan kebutuhan dari Toko Roti Amaya.

Sistem ini dibuat menggunakan bahasa pemrograman javascript dengan menggunakan *library* react js dan *extensions* jsx serta *framework* material-ui dan MDB React. Dari sisi *database* menggunakan mySQL serta menggunakan *restfull API* untuk menghubungkan server dengan klien yang bertujuan untuk mengambil API dari sisi *backend* yang berperan sebagai validasi transaksi pada setiap akun. Sistem ini dapat diakses menggunakan komputer yang terhubung dengan internet.

## Identifikasi Aktor

Identifikasi aktor ditujukan untuk mempermudah mengetahui aktor yang terlibat dalam sistem yang hendak melakukan pelaporan keuangan. Sistem manajemen keuangan memiliki 3 pengguna yaitu marketing, operator perusahaan dan akuntan. Deskripsi singkat mengenai masing-masing aktor ditunjukkan dalam tabel 4.1.

Tabel 4.1 Identifikasi Aktor Sistem Manajemen Keuangan

|  |  |  |
| --- | --- | --- |
| NO | Identifikasi Aktor | Deskripsi |
|  | Akuntan | Akuntan bertanggung jawab untuk mengulas dan validasi transaksi, posting jurnal serta menghasilkan laporan keuangan. |
|  | Operator Perusahaan | Operator perusahaan bertanggung jawab pada pencatatan keuangan perusahaan secara umum seperti logistik dan distribusi, serta mengelola produk dan bahan baku. |
|  | Marketing | Marketing bertanggung jawab pada pencatatan keuangan berupa pemasukkan hasil kerja sama penjualan dengan pihak luar. |

## Identifikasi Kebutuhan

Identifikasi kebutuhan sistem menjelaskan tentang kebutuhan fungsional dan non-fungsional sistem. Daftar kebutuhan sistem ditujukkan dalam tabel 4.1 sampai 4.9.

### Aturan Penomoran Kebutuhan Fungsional dan Non Fungsional

Setiap kebutuhan diatur menggunakan aturan penomoran seperti terlihat pada tabel 4.2. Untuk menghubungkan setiap kode penomoran menggunakan tanda *underscore* (\_). Contoh: SMKP\_F\_WEB\_010 artinya sistem manajemen kebutuhan fungsional platform web bernomor 10.

Tabel 4.2 Penomoran Spesifikasi Kebutuhan Fungsional

|  |  |
| --- | --- |
| Kode Penomoran | Keterangan |
| SMKP | Sistem Manajemen Keuangan Perusahaan |
| F | Kode representasi dari kebutuhan fungsional |
| NF | Kode representasi dari kebutuhan non fungsional |
| WEB | Kode representasi dari platform website |
| NOMOR | Angka penomoran kebutuhan |

### Daftar Spesifikasi Kebutuhan Fungsional

Tabel 4.3 Tabel Spefikasi Kebutuhan Fungsional

|  |  |  |
| --- | --- | --- |
| NO SKPL | Kebutuhan | Use Case |
| SMKP\_F\_WEB\_001 | Sistem dapat memberikan layanan untuk mencatat transaksi baik pemasukkkan dan pengeluaran. | Mencatat Transaksi |
| SMKP\_F\_WEB\_002 | Sistem dapat memberikan layanan untuk mengubah transaksi yang telah dicatat sebelumnya. | Mengubah Transaksi |
| SMKP\_F\_WEB\_003 | Sistem dapat memberikan layanan untuk melihat kembali transaksi yang pernah dicatat. | Menampilkan List Transaksi |
| SMKP\_F\_WEB\_004 | Sistem dapat memberikan layanan untuk menghapus transaksi. | Menghapus Transaksi |
| SMKP\_F\_WEB\_005 | Sistem dapat memberikan layanan untuk menambahkan produk terbaru berserta harga dan resep. | Menambahkan Produk |
| SMKP\_F\_WEB\_006 | Sistem dapat memberikan layanan untuk melihat kembali produk yang pernah dimasukkan. | Menampilkan List Produk |
| SMKP\_F\_WEB\_007 | Sistem dapat memberikan layanan untuk menambahkan bahan baku yang digunakan. | Menambahkan Bahan Baku |
| SMKP\_F\_WEB\_008 | Sistem dapat memberikan layanan untuk mendata bahan baku masuk, keluar serta merangkum stok bahan baku yang tersedia. | Rekapitulasi Bahan Baku |
| SMKP\_F\_WEB\_009 | Sistem memberikan layanan untuk mendata produksi yang sedang dilakukan pada hari tersebut. | Rekapitulasi Produksi |
| SMKP\_F\_WEB\_010 | Sistem dapat memberikan layanan untuk mengulas transaksi yang telah dicatat dalam bentuk memberikan pesan kepada transaksi tersebut. | Review Transaksi |
| SMKP\_F\_WEB\_011 | Sistem dapat memberikan layanan untuk menyetujui atau menolak transaksi sesuai syarat pencatatan transaksi. | Validasi Transaksi |
| SMKP\_F\_WEB\_012 | Sistem dapat memberikan layanan untuk melakukan posting jurnal berdasarkan transaksi yang ada. | Posting Jurnal |
| SMKP\_F\_WEB\_013 | Sistem dapat memberikan layanan untuk mengulas jurnal dalam bentuk memberikan pesan pada jurnal bersangkutan. | Review Jurnal |
| SMKP\_F\_WEB\_014 | Sistem dapat memberikan layanan untuk menerbitkan laporan keuangan secara otomatis sesuai dengan standar. | Generate Laporan Keuangan |
| SMKP\_F\_WEB\_015 | Sistem dapat memberikan layanan untuk mencetak laporan keuangan secara langsung. | Mencetak Laporan Keuangan |
| SMKP\_F\_WEB\_016 | Sistem dapat memberikan layanan untuk mengirimkan laporan keuangan melalui email. | Mengirimkan Laporan Keuangan |
| SMKP\_F\_WEB\_017 | Sistem dapat memberikan layanan untuk mengunduh laporan keuangan. | Mengunduh Laporan Keuangan |

### Daftar Spesifikasi Kebutuhan Non Fungsional

Tabel 4.4 Tabel Spesifikasi Kebutuhan Non Fungsional

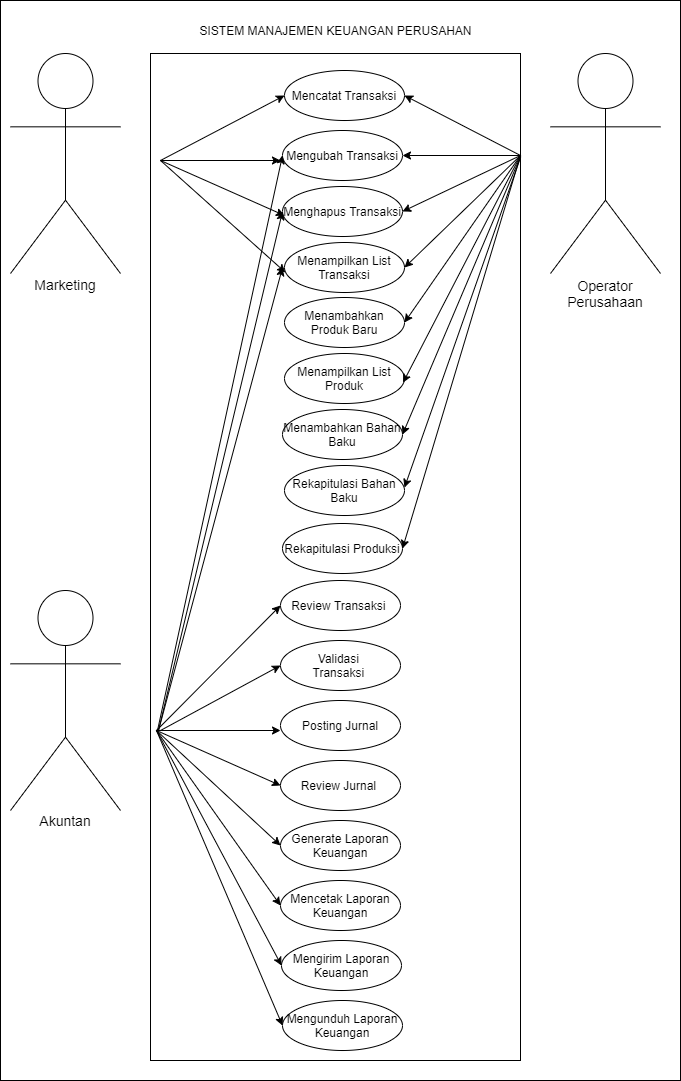
|  |  |  |
| --- | --- | --- |
| NO SKPL | Kebutuhan | Nama Kebutuhan |
| SMKP\_NF\_WEB\_001 | Sistem dapat diakses 24 jam. | *Availability* |
| SMKP\_NF\_WEB\_002 | Sistem dapat dioperasikan menggunakan *browser* seperti google chrome, mozilla firefox, dan safari. | *Compability* |

## Pemodelan Kebutuhan

Pada sub bab ini dibahas mengenai pemodelan berupa *use case* diagram dan *use case* scenario. *Use case* diagram digunakan untuk memodelkan perilaku pengguna yang diimplementasikan ke dalam sistem yang dibuat serta nantinya setiap detail dari *use case* diagram dijelaskan kembali ke dalam *use case scenario*.

### *Use Case* Diagram

*Use case* diagram digunakan untuk memodelkan perilaku aktor yang diimplementasikan ke dalam sistem. Pada diagram terdapat 3 aktor yaitu marketing, operator perusahaan dan akuntan. *Use case* diagram bisa dilihat dalam gambar 4.1.



Gambar 4.1 *Use Case Diagram*

### Use Case Scenario

*Use case scenario* menjelaskan secara terperinci *use case* diagram dalam gambar 4.1. *Use case scenario* menjelaskan langkah pengguna dalam menjalankan setiap kebutuhan, dimulai dari penjelasan singkat mengenai kebutuhan, syarat untuk menjalankan kebutuhan hingga hasil dari kebutuhan tersebut. Use case scenario dijelaskan pada tabel 4.3 – 4.20.

Tabel 4.5 *Use Case Scenario* Mencatat Transaksi

|  |  |
| --- | --- |
| Mencatat Transaksi | |
| Kode Kebutuhan | SMKP\_F\_WEB\_001 |
| Objektive | Sistem harus mampu memberikan layanan untuk melakukan pencatatan transaksi baik berupa teks, tanggal, angka dan unggah dokumen. |
| Aktor | Operator Perusahaan, Marketing |
| Pre\_Conditions | Aktor berada pada halaman pencatatan transaksi |
| Main Flow | 1. Aktor memilih sif kerja 2. Aktor mengisikan tanggal tagihan 3. Aktor mengisikan total tagihan 4. Aktor memilih metode pembayaran 5. Aktor mengunggah foto ataupun dokumen tagihan 6. Aktor memilih produk yang terjual 7. Aktor mengisikan jumlah produk yang terjual |
| Alternative Flow | 1. Aktor hanya bisa memasukkan angka pada semua *field* nominal. 2. Jika aktor tidak mengisikan satu saja *field* maka sistem tidak bisa melanjutkan proses pencatatan. |
| Post\_Conditions | 1. Sistem menyimpan catatan tagihan 2. Sistem menampilkan catatan kedalam list pada halaman list transaksi 3. Sistem menampilkan notifikasi catatan tersimpan |

Tabel 4.6 *Use Case Scenario* Mengubah Transaksi

|  |  |
| --- | --- |
| Mengubah Transaksi | |
| Kode Kebutuhan | SMKP\_F\_WEB\_002 |
| Objektive | Sistem harus mampu memberikan layanan untuk mengubah transaksi yang pernah dicatat sebelumnya. |
| Aktor | Operator Perusahaan, Marketing, Akuntan |
| Pre\_Conditions | 1. Aktor berada pada halaman list transaksi. |
| Main Flow | 1. Aktor memilih transaksi yang diubah. 2. Aktor melihat pesan pada bagian bawah transaksi yang dipilih. 3. Aktor mengubah transaksi sesuai dengan pesan yang ada. |
| Alternative Flow | 1. Aktor dapat menambahkan produk terjual melalui menu ini jika memiliki nomor tagihan yang sama. |
| Post\_Conditions | 1. Sistem menyimpan perubahan data pada transaksi terkait. 2. Sistem menampilkan kembali transaksi pada list transaksi. |

Tabel 4.7 *Use Case Scenario* Menampilkan List Transaksi

|  |  |
| --- | --- |
| Menampilkan List Transaksi | |
| Kode Kebutuhan | SMKP\_F\_WEB\_003 |
| Objektive | Sistem harus mampu memberikan layanan untuk menampilkan list transaksi yang telah dicatat sebelumnya berdasarkan jenis dan tanggal transaksi. |
| Aktor | Operator Keuangan, Marketing. |
| Pre\_Conditions | 1. Aktor berada pada halaman list transaksi. |
| Main Flow | 1. Aktor memilih jenis transaksi. 2. Aktor memilih tanggal transaksi. |
| Alternative Flow | 1. Aktor bisa melihat list transaksi tanpa memilih jenis dan tanggal transaksi. |
| Post\_Conditions | 1. Sistem menampilkan list transaksi sesuai dengan batasan pilihan dari aktor. |

Tabel 4.8 *Use Case Scenario* Menghapus Transaksi

|  |  |
| --- | --- |
| Menghapus Transaksi | |
| Kode Kebutuhan | SMKP\_F\_WEB\_004 |
| Objektive | Sistem harus mampu memberikan layanan untuk menghapus transaksi yang tidak sesuai. |
| Aktor | Operator Keuangan, Marketing, Akuntan. |
| Pre\_Conditions | 1. Aktor berada pada halaman list transaksi. |
| Main Flow | 1. Aktor memilih transaksi. 2. Aktor menghapus transaksi. |
| Alternative Flow | - |
| Post\_Conditions | 1. Sistem menghapus transaksi dari list transaksi. 2. Sistem menampilkan list transaksi. |

Tabel 4.9 *Use Case Scenario* Menambahkan Produk

|  |  |
| --- | --- |
| Menambahkan Produk | |
| Kode Kebutuhan | SMKP\_F\_WEB\_005 |
| Objektive | Sistem harus mampu memberikan layanan untuk menambahkan produk baru beserta harga dan menu. |
| Aktor | Operator Perusahaan |
| Pre\_Conditions | 1. Aktor berada pada halaman produk. |
| Main Flow | 1. Aktor mengisikan nama produk. 2. Aktor mengisikan kode produk. 3. Aktor memilih kategori produk. 4. Aktor memilih satuan produk. 5. Aktor mengunggah foto produk 6. Aktor mengisikan harga satuan jual produk. 7. Aktor memilih nama bahan baku. 8. Aktor mengisikan jumlah bahan baku yang digunakan. 9. Aktor mengisikan harga bahan baku. |
| Alternative Flow | 1. Aktor dapat menambahkan produk baru tanpa mengunggah foto produk. |
| Post\_Conditions | 1. Sistem menambahkan produk baru. 2. Sistem menampilkan list produk. |

Tabel 4.10 *Use Case Scenario* Menampilkan List Produk

|  |  |
| --- | --- |
| Menampilkan List Produk | |
| Kode Kebutuhan | SMKP\_F\_WEB\_006 |
| Objektive | Sistem harus mampu memberikan layanan untuk menampilkan list produk yang telah dicatat sebelumnya berdasarkan nama produk. |
| Aktor | Operator Keuangan. |
| Pre\_Conditions | 1. Aktor berada pada halaman list produk. |
| Main Flow | 1. Aktor mengisikan nama produk. |
| Alternative Flow | 1. Aktor bisa melihat list produk tanpa memilih nama produk. |
| Post\_Conditions | 1. Sistem menampilkan list produk sesuai dengan batasan pilihan dari aktor. |

Tabel 4.11 *Use Case Scenario* Menambahkan Bahan Baku

|  |  |
| --- | --- |
| Menambahkan Bahan Baku | |
| Kode Kebutuhan | SMKP\_F\_WEB\_007 |
| Objektive | Sistem harus mampu memberikan layanan untuk menambahkan bahan baku terbaru. |
| Aktor | Operator Perusahaan |
| Pre\_Conditions | 1. Aktor berada pada halaman list bahan baku. |
| Main Flow | 1. Aktor memilih kategori hpp bahan baku. 2. Aktor mengisikan nama bahan baku. 3. Aktor memilih kategori bahan baku. 4. Aktor mengisikan satuan beli bahan baku. 5. Aktor mengisikan satuan guna bahan baku. 6. Aktor mengisikan unit konversi bahan baku. |
| Alternative Flow | - |
| Post\_Conditions | 1. Sistem menambahkan bahan baku terbaru. 2. Sistem menampilkan list bahan baku. |

Tabel 4.12 *Use Case Scenario* Rekapitulasi Bahan Baku

|  |  |
| --- | --- |
| Rekapitulasi Bahan Baku | |
| Kode Kebutuhan | SMKP\_F\_WEB\_008 |
| Objektive | Sistem dapat memberikan layanan untuk mendata bahan baku masuk, keluar serta merangkum stok bahan baku yang tersedia. |
| Aktor | Operator Perusahaan |
| Pre\_Conditions | 1. Aktor harus sudah melakukan fungsionalitas SMKP\_F\_WEB\_001. 2. Aktor harus sudah melakukan fungsionalitas SMKP\_F\_WEB\_009. 3. Aktor berada pada halaman rekap bahan baku. |
| Main Flow | 1. Aktor memilih kategori hpp bahan baku. |
| Alternative Flow | - |
| Post\_Conditions | 1. Sistem menampilkan rekap bahan baku terbaru sesuai dengan tanggal. |

Tabel 4.13 *Use Case Scenario* Rekapitulasi Produksi

|  |  |
| --- | --- |
| Rekapitulasi Produksi | |
| Kode Kebutuhan | SMKP\_F\_WEB\_009 |
| Objektive | Sistem memberikan layanan untuk mendata produksi yang sedang dilakukan pada hari tersebut. |
| Aktor | Operator Perusahaan |
| Pre\_Conditions | 1. Aktor berada pada halaman list produk. |
| Main Flow | 1. Aktor mengisikan jumlah produk yang diproduksi. 2. Aktor memilih tanggal produksi. 3. Aktor memilih tanggal estimasi produksi selesai. |
| Alternative Flow | - |
| Post\_Conditions | 1. Sistem menampilkan resep sesuai dengan jumlah produksi. 2. Sistem menampilkan list produk dalam proses produksi sesuai dengan tanggal. |

Tabel 4.14 *Use Case Scenario* Review Transaksi

|  |  |
| --- | --- |
| Review Transaksi | |
| Kode Kebutuhan | SMKP\_F\_WEB\_010 |
| Objektive | Sistem harus mampu memberikan layanan untuk mengulas transaksi yang tersedia dengan cara memberikan pesan. |
| Aktor | Akuntan |
| Pre\_Conditions | 1. Aktor berada pada halaman list transaksi. |
| Main Flow | 1. Aktor memilih transaksi yang diulas. 2. Aktor memberikan pesan terkait transaksi yang dipilih. |
| Alternative Flow | 1. Aktor dapat mengulas transaksi tanpa harus memberikan pesan. |
| Post\_Conditions | 1. Sistem menampilkan pesan yang telah dimasukkan. |

Tabel 4.15 *Use Case Scenario* Validasi Transaksi

|  |  |
| --- | --- |
| Validasi Transaksi | |
| Kode Kebutuhan | SMKP\_F\_WEB\_011 |
| Objektive | Sistem harus mampu memberikan layanan untuk validasi transaksi yang tersedia dengan menyetujui atau menolak transaksi. |
| Aktor | Akuntan |
| Pre\_Conditions | 1. Aktor berada pada halaman list transaksi. |
| Main Flow | 1. Aktor memilih transaksi yang divalidasi. 2. Aktor menyetujui atau menolak transaksi. |
| Alternative Flow | 1. Jika transaksi ditolak maka transaksi harus diperbaiki sebelum diajukan kembali untuk proses *review*. |
| Post\_Conditions | 1. Sistem menampilkan validasi transaksi pada halaman list transaksi. |

Tabel 4.16 Use Case Scenario *Review* Jurnal

|  |  |
| --- | --- |
| Review Jurnal | |
| Kode Kebutuhan | SMKP\_F\_WEB\_012 |
| Objektive | Sistem harus mampu memberikan layanan untuk mengulas jurnal dengan cara mengubah jurnal atau memberikan pesan sesuai dengan kebutuhan. |
| Aktor | Akuntan |
| Pre\_Conditions | 1. Aktor harus sudah melakukan fungsionalitas sebelumnya. 2. Aktor harus sudah melalukukan posting jurnal per hari. 3. Aktor berada pada halaman jurnal. |
| Main Flow | 1. Aktor mengubah jurnal sesuai dengan kebutuhan. 2. Aktor memberikan pesan. |
| Alternative Flow | 1. Aktor bisa mengulas jurnal tanpa perlu mengubah jurnal. 2. Aktor bisa mengulas jurnal tanpa perlu memberikan pesan. |
| Post\_Conditions | 1. Sistem menampilkan pesan dari jurnal. 2. Sistem menampilkan jurnal yang telah diperbarui. |

Tabel 4.17 *Use Case Scenario* *Posting* Jurnal

|  |  |
| --- | --- |
| Posting Jurnal | |
| Kode Kebutuhan | SMKP\_F\_WEB\_013 |
| Objektive | Sistem harus mampu memberikan layanan untuk merubah setiap transaksi kedalam jurnal. |
| Aktor | Akuntan |
| Pre\_Conditions | 1. Aktor harus melakukan fungsionalitas sebelumnya. 2. Aktor berada pada halaman jurnal. |
| Main Flow | 1. Aktor melakukan posting jurnal. |
| Alternative Flow | - |
| Post\_Conditions | 1. Sistem menampilkan jurnal secara otomatis. |

Tabel 4.18 *Use Case Scenario* Generate Laporan Keuangan

|  |  |
| --- | --- |
| Generate Laporan Keuangan | |
| Kode Kebutuhan | SMKP\_F\_WEB\_014 |
| Objektive | Sistem harus mampu memberikan layanan untuk menerbitkan laporan keuangan secara otomatis sesuai standar berdasarkan jurnal. |
| Aktor | Akuntan |
| Pre\_Conditions | 1. Aktor harus melakukan posting jurnal. 2. Aktor berada pada halaman laporan keuangan. |
| Main Flow | 1. Aktor melakukan generate laporan keuangan. |
| Alternative Flow | 1. Jika aktor melakukan perubahan pada jurnal maka harus kembali melakukan generate laporan keuangan. |
| Post\_Conditions | 1. Sistem menampilkan laporan keuangan dalam bentuk dokumen. |

Tabel 4.19 *Use Case Scenario Generate* Laporan Keuangan

|  |  |
| --- | --- |
| Generate Laporan Keuangan | |
| Kode Kebutuhan | SMKP\_F\_WEB\_015 |
| Objektive | Sistem harus mampu memberikan layanan untuk mencetak laporan keuangan. |
| Aktor | Akuntan |
| Pre\_Conditions | 1. Aktor harus sudah melakukan fungsionalitas SMKP\_F\_WEB\_014. 2. Aktor berada pada halaman laporan keuangan. |
| Main Flow | 1. Aktor mencetak laporan keuangan. |
| Alternative Flow | - |
| Post\_Conditions | 1. Sistem meyambungkan dengan alat cetak yang tersedia. |

Tabel 4.20 *Use Case Scenario* Mencetak Laporan Keuangan

|  |  |
| --- | --- |
| Mencetak Laporan Keuangan | |
| Kode Kebutuhan | SMKP\_F\_WEB\_016 |
| Objektive | Sistem harus mampu memberikan layanan untuk mengirim laporan keuangan. |
| Aktor | Akuntan |
| Pre\_Conditions | 1. Aktor harus sudah melakukan fungsionalitas SMKP\_F\_WEB\_014.   Aktor berada pada halaman laporan keuangan. |
| Main Flow | Aktor mengirim laporan keuangan. |
| Alternative Flow | - |
| Post\_Conditions | 1. Sistem mengirimkan laporan keuangan melalui email. |

Tabel 4.21 *Use Case Scenario* Mengunduh Laporan Keuangan

|  |  |
| --- | --- |
| Mengunduh Laporan Keuangan | |
| Kode Kebutuhan | SMKP\_F\_WEB\_017 |
| Objektive | Sistem harus mampu memberikan layanan untuk mengunduh laporan keuangan. |
| Aktor | Akuntan |
| Pre\_Conditions | 1. Aktor harus sudah melakukan fungsionalitas SMKP\_F\_WEB\_014. 2. Aktor berada pada halaman laporan keuangan. |
| Main Flow | 1. Aktor mengunduh laporan keuangan. |
| Alternative Flow | - |
| Post\_Conditions | 1. Sistem mengunduh laporan keuangan dalam bentuk pdf. |

## Analisis Data

Pada bagian analisis data menjelaskan data yang digunakan pada sistem untuk diimplementasikan pada bagian perancangan data. Dalam pengembangan sistem tidak menutup kemungkinan untuk melakukan penambahan atau pengurangan data. Analisis data yang digunakan pada sistem manajemen keuangan perusahan diantaranya sebagai berikut:

Data pengguna *username* dan *password*.

Data transaksi baik pemasukkan dan pengeluaran diantaranya nomor transaksi, sif kerja, tanggal transaksi, jumlah transaksi dalam rupiah, metode pembayaran transaksi, total transaksi, jenis transaksi, harga per satuan, jumlah barang, dan deskripsi barang.

Data akuntansi diantaranya nama transaksi, tanggal transaksi, total transaksi dalam rupiah, nomor referensi, total transaksi kredit, dan total transaksi debit

# PERANCANGAN DAN IMPLEMENTASI

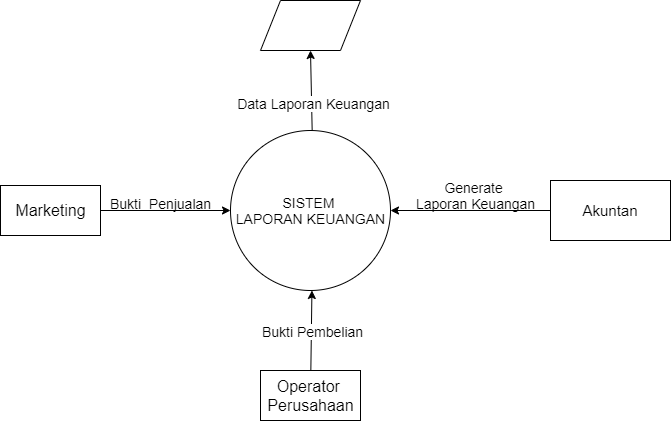
## Perancangan

Perancangan sistem dilakukan dengan menggunakan pendekatan *structured analysis and design* yang berfokus pada memecah proses bisnis menjadi bagian-bagian kecil. Perancangan sistem yang akan dibahas meliputi perancangan struktur analisis, perancangan struktur desain, perancangan basis data, dan perancangan antarmuka.

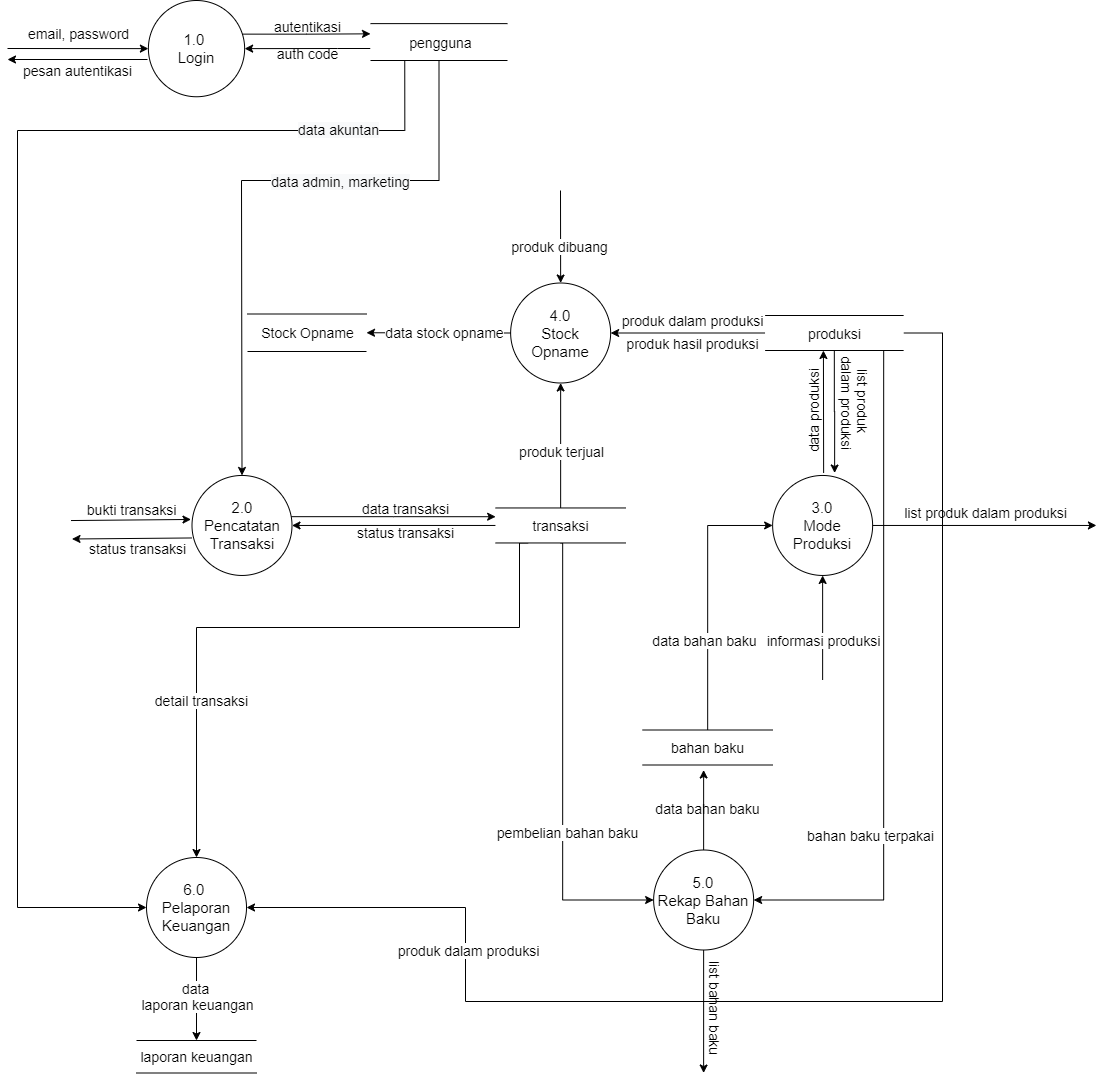
### Perancangan Struktur Analisis

Perancangan struktur analisis digambarkan dengan *data flow diagram* dalam gambar 5.1 – 5.6. Dalam gambar 5.1 digambarkan *data flow diagram* level 0 yang menggambarkan keseluruhan sistem dan entitas eksternal yang terlibat serta hasil dari sistem. Entitas eksternal yang terlibat adalah marketing, operator perusahaan dan akuntan serta hasil dari sistem adalah dokumen laporan keuangan. Dalam gambar 5.2 digambarkan *data flow diagram* level 1 yang menggambarkan proses yang terjadi di dalam sistem yang terdiri dari 6 proses besar.

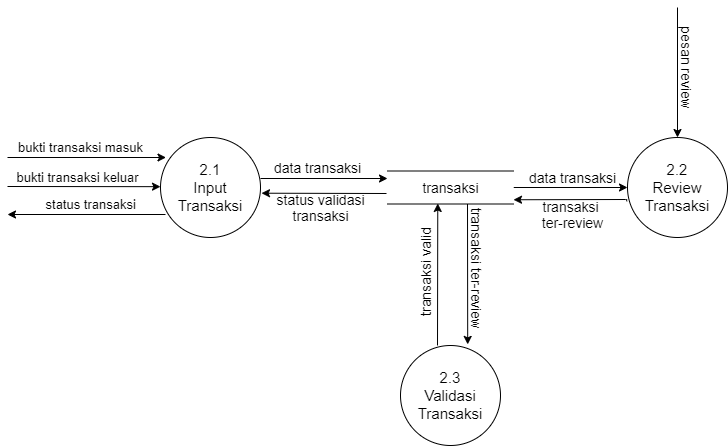
Gambar 5.3 – 5.6 merupakan *data flow diagram* level 2 yang menggambarkan sub-proses dari proses besar pada level sebelumnya. Gambar 5.3 menggambarkan menggambarkan dekomposisi proses 2 yaitu pencatatan transaksi yang terdiri dari sub-proses input transaksi, *review* transaksi dan validasi transaksi. Gambar 5.4 menggambarkan dekomposisi proses 3 yaitu mode produksi dengan sub-proses input produk baru, produksi produk dan rekapitulasi produk dengan hasil akhir keseluruhan proses adalah list rekapitulasi produk. Gambar 5.5 menggambarkan dekomposisi proses 5 yaitu rekapitulasi bahan baku dengan sub proses input bahan baku dan rekapitulasi bahan baku. Gambar 5.6 menggambarkan dekomposisi proses 6 yaitu pelaporan keuangan dengan sub-proses posting jurnal per hari, posting jurnal per bulan, *review* jurnal dan generate laporan keuangan.



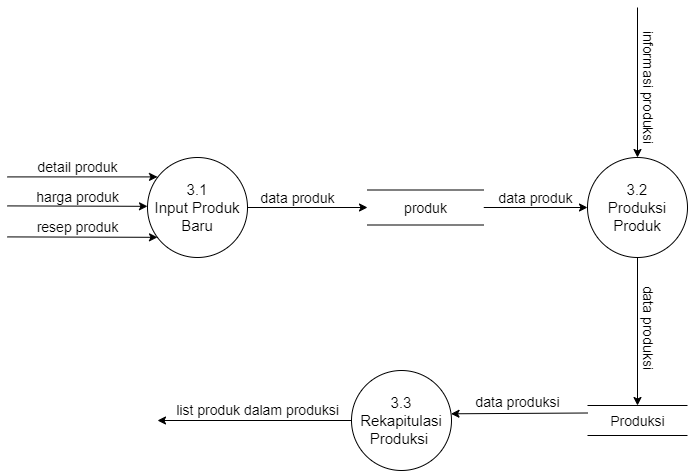
Gambar 5.1 *Data Flow Diagram* Level 0



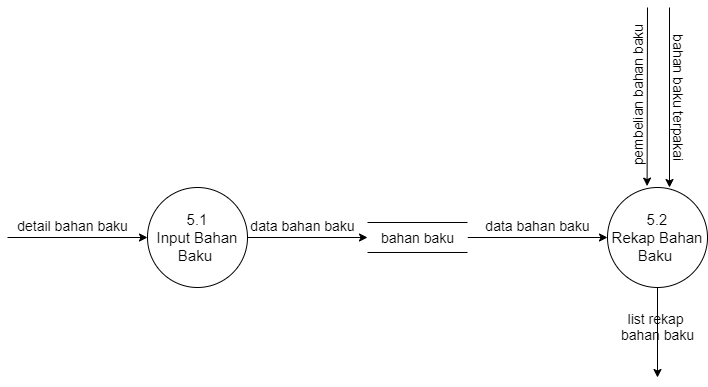
Gambar 5.2 Data Flow Diagram Level 1



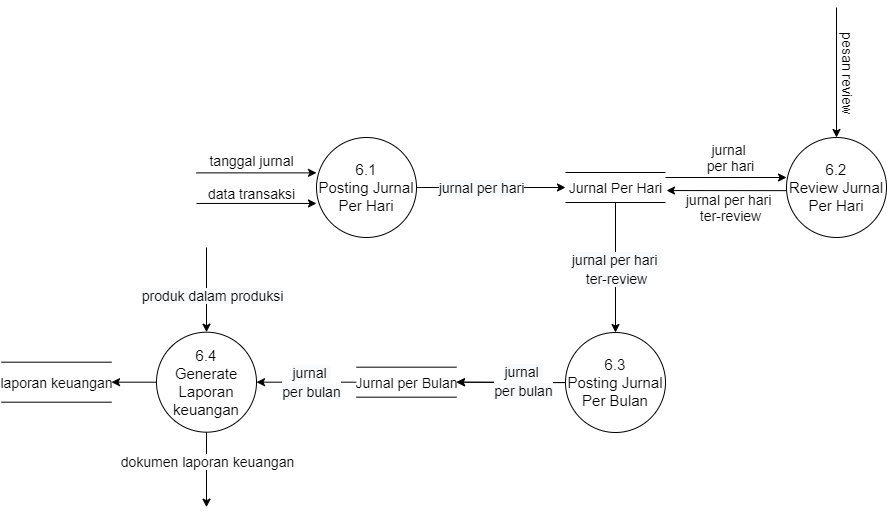
Gambar 5.3 *Data Flow Diagram* Level 2 Proses 2



Gambar 5.4 *Data Flow Diagram* Level 2 Proses 3



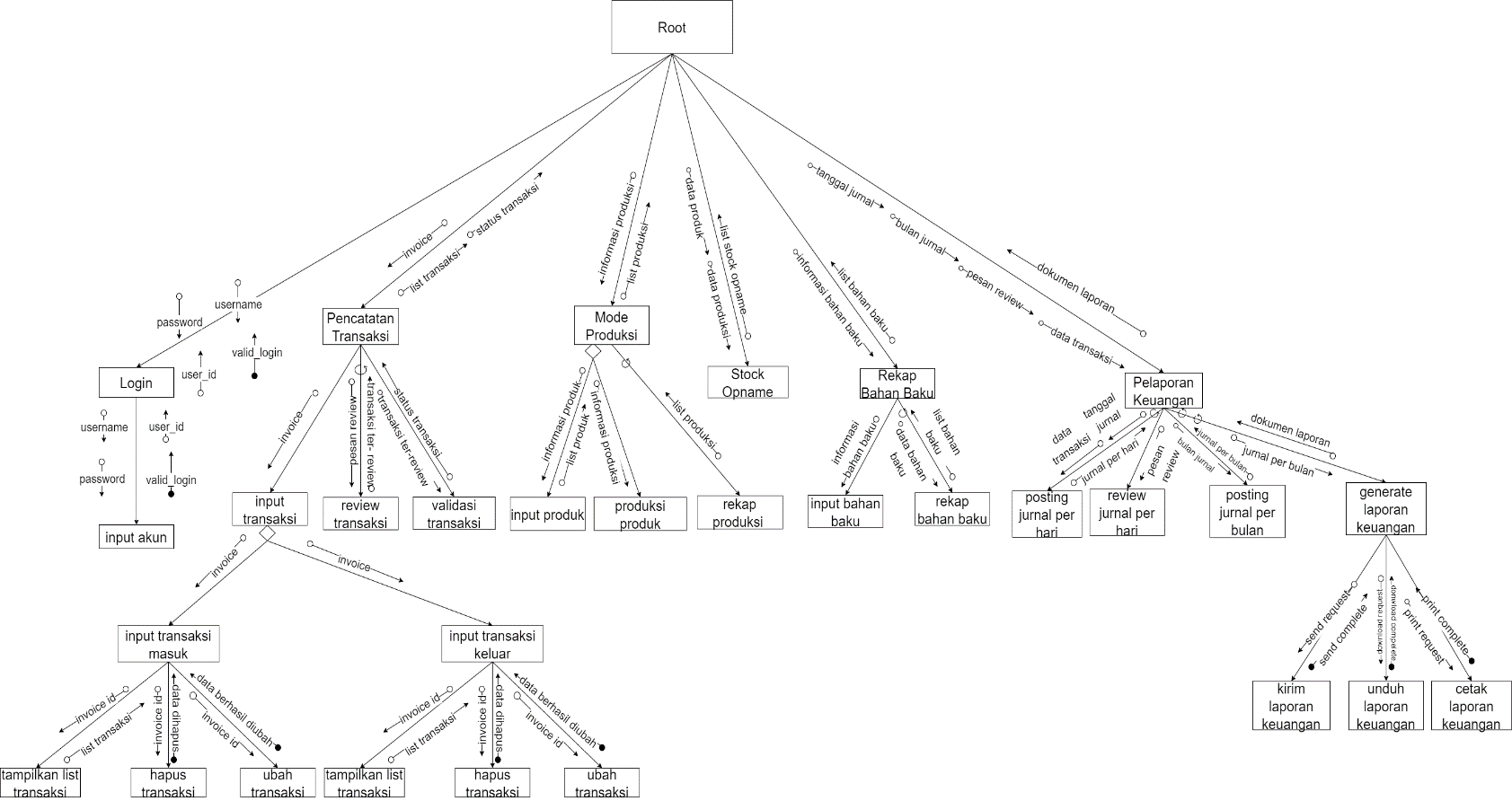
Gambar 5.5 *Data Flow Diagram* Level 2 Proses 5



Gambar 5.6 *Data Flow Diagram* Level 2 Proses 6

### Perancangan Struktur Desain

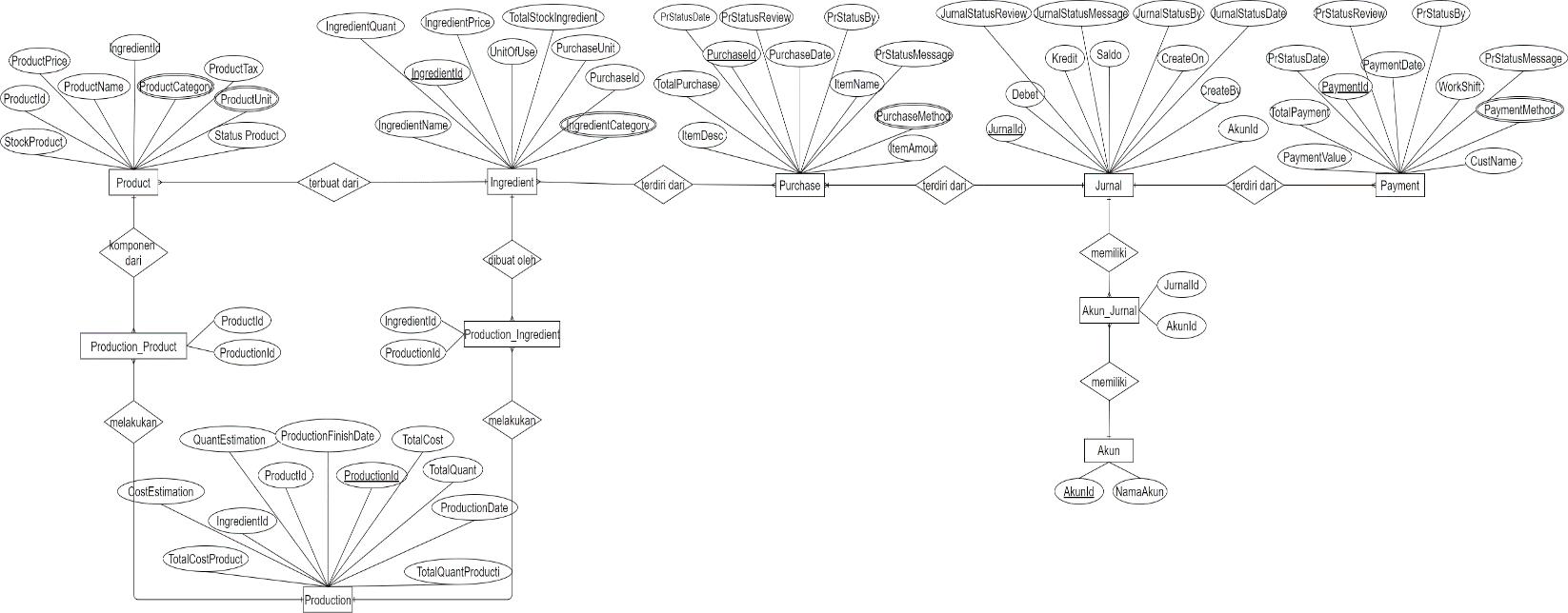
Perancangan struktur analisis digambarkan dengan grafik strukturdalam gambar 5.7. Grafik struktur dibuat berdasarkan *data flow diagram*, *use case scenario* *dan use case diagram*.



Gambar 5.7 *Structure Chart*

### Perancangan Basis Data

Perancangan basis data berdasarkan *entity relationship diagram* yang merupakan gambaran fisik dari tabel *database* serta digambarkan juga relasi antar database. Dalam gambar 5.8 adalah *entity relationship diagram* dari sistem manajemen akuntasi keuangan yang telah dinormalisasi.



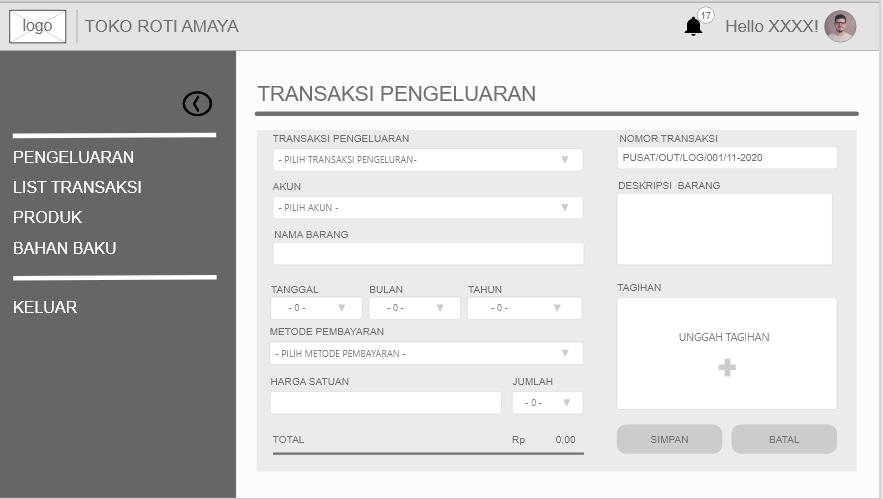
Gambar 5.8 *Enitity Relationship Diagram*

### Perancangan Antarmuka

Perancangan antarmuka menggambarkan tampilan sistem yang diakses oleh pengguna serta menjelaskan alur perpindahan halaman pada antarmuka sistem manajemen akuntansi keuangan. Perancangan antarmuka memperhitungkan kenyamanan serta fungsionalitas setiap komponen sehingga memenuhi kepuasan serta kenyamanan pengguna (*user experience*) terhadap sistem.

Halaman Mencatat Transaksi

Gambar 5.9 Perancangan Antarmuka Pengguna Halaman Mencatat Transaksi Pemasukan



Gambar 5.10 Perancangan Antarmuka Pengguna Halaman Mencatat Transaksi Pengeluaran

Pada fungsionalitas mencatat transaksi terdapat 2 sub pencatatan yaitu pencatatan penjualan oleh marketing dan transaksi kebutuhan perusahaan yang ditunjukkan dalam gambar 5.9 dan 5.10. Dalam tabel 5.1 dijelaskan komponen perancangan antarmuka halaman mencatat transaksi.

Tabel 5.1 Informasi Elemen Halaman Mencatat Transaksi Penjualan

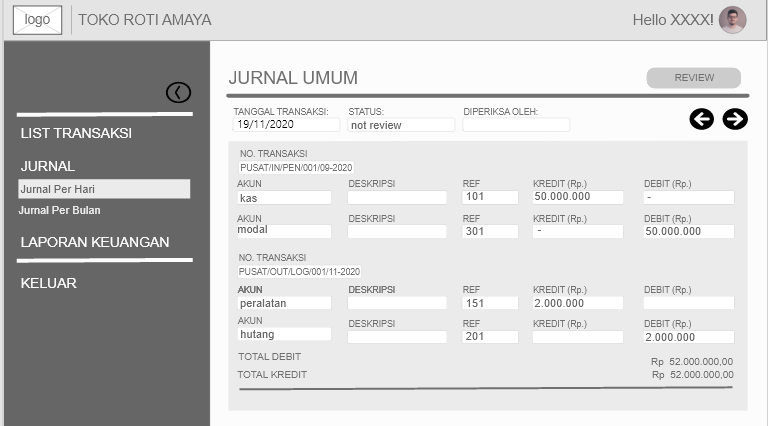
|  |  |  |
| --- | --- | --- |
| No | Komponen | Detail Fungsi |
| 1. | Menu “Pemasukkan” | Menampilkan sub menu pemasukkan |
| 2. | Menu “List Transaksi” | Menampilkan halaman list transaksi yang telah dimasukkan |
| 3. | Kolom “Nomor Transaksi” | Menampilkan secara otomatis nomor transaksi berdasarkan jenis transaksi, tanggal transaksi, serta divisi yang bersangkutan. |
| 4. | Kolom Teks “Sif Kerja” | Menampilkan sif kerja dari serangkaian pilihan di menu *dropdown* yang akan dipilih oleh pengguna |
| 5. | Kolom *Dropdown* “Tanggal” | Menampilkan tanggal dalam bentuk angka di menu *dropdown* yang akan dipilih oleh pengguna |
| 6. | Kolom *Dropdown* “Bulan” | Menampilkan bulan dalam bentuk tulisan di menu *dropdown* yang akan dipilih oleh pengguna |
| 7. | Kolom *Dropdown* “Tahun” | Menampilkan tahun dalam bentuk angka di menu *dropdown* yang akan dipilih oleh pengguna |
| 8. | Ikon “Tambah” | Menampilkan kolom baru untuk menambah produk |
| 9. | Kolom Teks “Nama Pelanggan” | Menambahkan *input* baru berupa teks berisi nama pelanggan pada kolom yang tersedia |
| 10. | Kolom Teks “Total *Invoice*” | Menambahkan *input* baru berupa angka berisi total *invoice* pada kolom yang tersedia |
| 11. | Kolom Dropdown “Metode Pembayaran” | Menampilkan berbagai metode pembayaran dari serangkaian pilihan di menu *dropdown* yang akan dipilih oleh pengguna |
| 12. | Kolom Unggah “Unggah Tagihan” | Menambahkan *input* baru berupa dokumen ataupun foto yang diunggah oleh pengguna |
| 13. | Kolom “Produk Terjual” | Memilih produk yang terjual berdasarkan tagihan pembeli |
| 14. | Kolom “Jumlah Produk Terjual” | Memasukkan jumlah produk yang terjual berdasarkan tagihan pembeli |
| 15. | Kolom Teks “Total” | Menampilkan total invoice yang dimasukkan pengguna secara otomatis oleh sistem |
| 16. | Tombol “Simpan” | Menyimpan transaksi yang dicatat pengguna untuk ditampilkan pada halaman lihat transaksi |
| 17. | Tombol “Batal” | Membatalkan transaksi yang dicatat pengguna |

Tabel 5.2 Informasi Elemen Halaman Mencatat Transaksi Kebutuhan Perusahaan

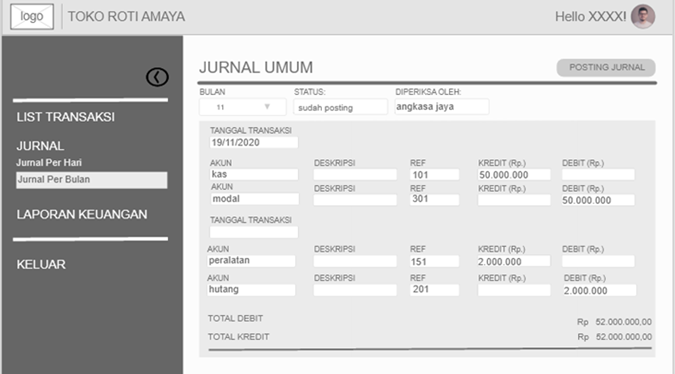
|  |  |  |
| --- | --- | --- |
| No | Komponen | Detail Fungsi |
| 1. | Menu “Pemasukkan” | Menampilkan sub menu pemasukkan |
| 2. | Menu “Lihat Transaksi” | Menampilkan halaman list transaksi yang telah dimasukkan |
| 3. | Menu “Produk” | Menampilkan produk yang terdiri dari beberapa sub menu |
| 4. | Menu “Bahan Baku” | Menampilkan halaman bahan baku yang terdiri dari beberapa sub menu |
| 5. | Kolom “Transaksi Pengeluaran” | Menampilkan pilihan jenis transaksi pengeluaran antara logisitik atau operasional perusahaan |
| 6. | Kolom “Akun” | Menampilkan akun berdasarkan jenis dan transaksi pengeluaran yang dilakukan |
| 7. | Kolom Teks “Nama Barang” | Menambahkan *input* baru berupa teks berisi nama barang pada kolom yang tersedia |
| 7. | Kolom Teks “Deskripsi Barang” | Menambahkan *input* baru berupa teks berisi penjelasan singkat mengenai barang pada kolom yang tersedia |
| 8. | Kolom *Dropdown* “Tanggal” | Menampilkan tanggal dalam bentuk angka di menu *dropdown* yang akan dipilih oleh pengguna |
| 9. | Kolom *Dropdown* “Bulan” | Menampilkan bulan dalam bentuk tulisan di menu *dropdown* yang akan dipilih oleh pengguna |
| 10. | Kolom *Dropdown* “Tahun” | Menampilkan tahun dalam bentuk angka di menu *dropdown* yang akan dipilih oleh pengguna |
| 11. | Kolom Dropdown “Metode Pembayaran” | Menampilkan berbagai metode pembayaran dari serangkaian pilihan di menu *dropdown* yang akan dipilih oleh pengguna |
| 12. | Kolom Teks “Harga Satuan” | Menambahkan *input* baru berupa angka berisi harga satuan barang pada kolom yang tersedia |
| 13. | Kolom *Dropdown* “Jumlah” | Menampilkan angka di menu *dropdown* yang akan dipilih oleh pengguna |
| 14. | Kolom Teks “Nomor Transaksi” | Menampilkan nomor transaksi secara otomatis oleh sistem |
| 15. | Kolom Unggah “Unggah Tagihan” | Menambahkan *input* baru berupa dokumen ataupun foto yang diunggah oleh pengguna |
| 16 | Kolom Teks “Total” | Menampilkan total invoice yang dimasukkan pengguna secara otomatis oleh sistem |
| 17. | Tombol “Simpan” | Menyimpan transaksi yang dicatat pengguna untuk ditampilkan pada halaman lihat transaksi |
| 18. | Tombol “Batal” | Membatalkan transaksi yang dicatat pengguna |

Halaman Posting Jurnal

Dalam gambar 5.11 merupakan halaman posting jurnal per hari dan gambar 5.12 merupakan halaman posting jurnal per bulan. Kedua jurnal tersebut memiliki fungsi dan komponen yang sama yang membedakan jurnal umum per hari harus melalui proses *review* sebelum akhirnya di­-*generate* kedalam jurnal umum per bulan. Detail komponen dan penjelasan halaman antarmuka posting jurnal dijelaskan pada tabel 5.3 dan 5.4.



Gambar 5.11 Perancangan Antarmuka Pengguna Halaman Jurnal Per Hari



Gambar 5.12 Perancangan Antarmuka Pengguna Halaman Jurnal Per Bulan

Tabel 5.3 Informasi Elemen Halaman Posting Jurnal Per Hari

|  |  |  |
| --- | --- | --- |
| No | Komponen | Detail Fungsi |
| 1. | Menu “List Transaksi” | Menampilkan sub menu list transaksi |
| 2. | Menu “Jurnal” | Menampilkan sub menu jurnal yaitu jurnal per hari dan jurnal per bulan |
| 3. | Sub Menu “Jurnal Per Hari” | Menampilkan halaman antarmuka jurnal per hari |
| 4. | Sub Menu “Jurnal Per Bulan” | Menampilkan halaman antarmuka jurnal per bulan |
| 6. | Menu “Laporan Keuangan” | Menampilkan halaman laporan keuangan |
| 7. | Kolom Teks “Nomor Transaksi” | Menampilkan nomor transaksi secara otomatis oleh sistem |
| 8. | Kolom Teks “Status” | Menampilkan status jurnal secara otomatis oleh sistem sesuai dengan keadaan saat itu |
| 9. | Kolom Teks “Diperiksa Oleh” | Menampilkan pihak yang memeriksa jurnal tersebut |
| 10. | Tombol “Generate” | Pengguna menekan tombol tersebut untuk menggabungkan jurnal per hari ke jurnal per bulan |
| 11. | Tombol “Review” | Pengguna menekan tombol tersebut untuk me-*review* jurnal |
| 12. | Tombol “Selanjutnya” | Menampilkan jurnal sesudah dari data yang ditampilkan |
| 13 | Tombol “Sebelum” | Menampilkan jurnal sebelum dari data yang ditampilkan |
| 14. | Kolom Teks “Tanggal Transaksi” | Menampilkan tanggal transaksi |
| 15. | Kolom Teks “Akun” | Menampilkan teks berisi nama transaksi pada kolom yang tersedia |
| 16. | Kolom Teks “Harga Satuan” | Menampilkan angka berisi harga satuan barang pada kolom yang tersedia |
| 17. | Kolom Teks “Deskripsi” | Menampilkan deskripsi singkat mengenai transaksi pada kolom yang tersedia |
| 18. | Kolom Teks “Ref” | Menampilkan angka berisi nomor akun terkait transaksi pada kolom yang tersedia |
| 19 | Kolom Teks “Kredit(Rp.)” | Menampilkan angka berisi nilai kredit transaksi pada kolom yang tersedia |
| 20. | Kolom Teks “Debit(Rp.)” | Menampilkan angka berisi nilai debet transaksi pada kolom yang tersedia |
| 21. | Kolom Teks “Total Kredit(Rp.)” | Menampilkan angka berisi total kredit jurnal pada kolom yang tersedia |
| 22. | Kolom Teks “Total Debit(Rp.)” | Menampilkan angka berisi total debet jurnal pada kolom yang tersedia |

Tabel 5.4 Informasi Elemen Halaman Posting Jurnal Per Bulan

|  |  |  |
| --- | --- | --- |
| No | Komponen | Detail Fungsi |
| 1. | Menu “List Transaksi” | Menampilkan sub menu list transaksi |
| 2. | Menu “Jurnal” | Menampilkan sub menu jurnal yaitu jurnal per hari dan jurnal per bulan |
| 3. | Sub Menu “Jurnal Per Hari” | Menampilkan halaman antarmuka jurnal per hari |
| 4. | Sub Menu “Jurnal Per Bulan” | Menampilkan halaman antarmuka jurnal per bulan |
| 6. | Menu “Laporan Keuangan” | Menampilkan halaman laporan keuangan |
| 7. | Kolom Teks “Nomor Transaksi” | Menampilkan nomor transaksi secara otomatis oleh sistem |
| 8. | Kolom Teks “Status” | Menampilkan status jurnal secara otomatis oleh sistem sesuai dengan keadaan saat itu |
| 9. | Kolom Teks “Diperiksa Oleh” | Menampilkan pihak yang memeriksa jurnal tersebut |
| 10. | Tombol “Posting” | Pengguna menekan tombol tersebut untuk posting jurnal kedalam buku besar |
| 11. | Kolom Teks “Tanggal Transaksi” | Menampilkan tanggal transaksi |
| 12. | Kolom Teks “Akun” | Menampilkan teks berisi nama transaksi pada kolom yang tersedia |
| 13 | Kolom Teks “Harga Satuan” | Menampilkan angka berisi harga satuan barang pada kolom yang tersedia |
| 14. | Kolom Teks “Deskripsi” | Menampilkan deskripsi singkat mengenai transaksi pada kolom yang tersedia |
| 15. | Kolom Teks “Ref” | Menampilkan angka berisi nomor akun terkait transaksi pada kolom yang tersedia |
| 16. | Kolom Teks “Kredit(Rp.)” | Menampilkan angka berisi nilai kredit transaksi pada kolom yang tersedia |
| 17. | Kolom Teks “Debit(Rp.)” | Menampilkan angka berisi nilai debet transaksi pada kolom yang tersedia |
| 18. | Kolom Teks “Total Kredit(Rp.)” | Menampilkan angka berisi total kredit jurnal pada kolom yang tersedia |
| 19 | Kolom Teks “Total Debit(Rp.)” | Menampilkan angka berisi total debet jurnal pada kolom yang tersedia |

## Implementasi

Implementasi merupakan penjelasan singkat mengenai pengembangan sistem dimulai dari perangkat pengembangan sistem, implementasi data, implementasi algoritma, dan implementasi antar muka.

### Spesifikasi Pengembangan Sistem

Pengembangan sistem menggunakan perangkat lunak dan perangkat keras sebagai berikut:

Spesifikasi Perangkat Keras

*Processor* :Intel® Core™ i5-7200

*Hard Disk Drive* : 32 GB

*RAM* : 8 GB

*System Model* : LENOVO MUNPJ43

Spesifikasi Perangkat Lunak

*Operating System* : Windows 10 64-bit Single (10.0, build 18363)

*Programming Language* : Javascript, HTML

*Text Editor* : Visual Studio 1.50.1

*Server* : XAMPP v3.2.2

*Basis Data*  : MariaDB 10.3.24

### Implementasi Basis Data

Implementasi basis data dilakukan dengan menggunakan *Data Definition Language* (DDL). DDL mendefinisikan struktur sebuah tabel pada basis data dengan menyertakan perintah yang akan dieksekusi. Implementasi basis data pada sistem akuntansi manajemen keuangan pada studi kasus Toko Roti Amaya dijelaskan pada tabel berikut.

Tabel 5.3 Implementasi Basis Data *Review* Jurnal

|  |  |
| --- | --- |
| NO | KODE |
| 1 | CREATE TABLE IF NOT EXISTS `journalreviews` ( |
| 2 | `id` bigint(20) unsigned NOT NULL AUTO\_INCREMENT, |
| 3 | `transaction\_date` date NOT NULL, |
| 4 | `memo` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL, |
| 5 | `status` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL, |
| 6 | `reviewer\_id` bigint(20) unsigned DEFAULT NULL, |
| 7 | `review\_date` datetime DEFAULT NULL, |
| 8 | `created\_at` timestamp NULL DEFAULT NULL, |
| 9 | `updated\_at` timestamp NULL DEFAULT NULL, |
| 10 | PRIMARY KEY (`id`), |
| 11 | KEY `journalreviews\_reviewer\_id\_foreign` (`reviewer\_id`), |
| 12 | CONSTRAINT `journalreviews\_reviewer\_id\_foreign` FOREIGN KEY (`reviewer\_id`) REFERENCES `users` (`id`) ON DELETE SET NULL ON UPDATE SET NULL |

Tabel 5.4 Implementasi Basis Data PostingJurnal

|  |  |
| --- | --- |
| NO | KODE |
| 1 | CREATE TABLE IF NOT EXISTS `journals` ( |
| 2 | `id` bigint(20) unsigned NOT NULL AUTO\_INCREMENT, |
| 3 | `account\_id` bigint(20) unsigned NOT NULL, |
| 4 | `jreview\_id` bigint(20) unsigned NOT NULL, |
| 5 | `date` date NOT NULL, |
| 6 | `debit` double(13,2) NOT NULL, |
| 7 | `credit` double(13,2) NOT NULL, |
| 8 | `description` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL DEFAULT '-', |
| 9 | `created\_at` timestamp NULL DEFAULT NULL, |
| 10 | `updated\_at` timestamp NULL DEFAULT NULL, |
| 11 | PRIMARY KEY (`id`), |
| 12 | KEY `journals\_account\_id\_foreign` (`account\_id`), |
| 13 | KEY `journals\_jreview\_id\_foreign` (`jreview\_id`), |
|  | CONSTRAINT `journals\_account\_id\_foreign` FOREIGN KEY (`account\_id`) REFERENCES `accounts` (`id`) ON DELETE CASCADE ON UPDATE CASCADE, |
| 14 | CONSTRAINT `journals\_jreview\_id\_foreign` FOREIGN KEY (`jreview\_id`) REFERENCES `journalreviews` (`id`) ON DELETE CASCADE ON UPDATE CASCADE |

### Implementasi Algoritma

Implementasi algoritma review jurnal per hari

Implementasi algoritma review jurnal berada pada *functional component* handleReview seperti ditunjukkan pada tabel 5.6. Didalam fungsi tersebut terdapat variabel data dan *method* setReview serta fungsi GetDailyJournal berparameter *date*. Fungsi GetDailyJournal berasal dari *file* JournalContext.js yang ditunjukkan oleh tabel 5.7. Fungsi tersebut berfungsi sebagai wadah berbagi data secara global ke *react compenent*, data yang dimaksud adalah *date*. Fungsi GetDailyJournal juga digunakan untuk memberikan *response* dan *error message* melalui API.

Pada file Component.js terdapat *react hooks* berupa *useState* dan *useContext* yang diinisialisasi pada fungsi JurnalPerhariComponent. Method setReview diubah menjadi *true* yang artinya terdapat perubahan *state* yang menyebabkan *rendering component*. Fungsi memoComponent digunakan untuk membuat komponen untuk menampung masukkan pengguna pada method SetMemo berupa teks. Komponen MDBBtn pada fungsi memoComponent digunakan untuk menyimpan hasil *review*. *Return* digunakan sebagai wadah *rendering* semua *component*, *component* untuk *rendering* fungsi handleReview adalah MDBBtn.

Tabel 5.5 Implementasi Algoritma Review Jurnal Perhari (Component.js)

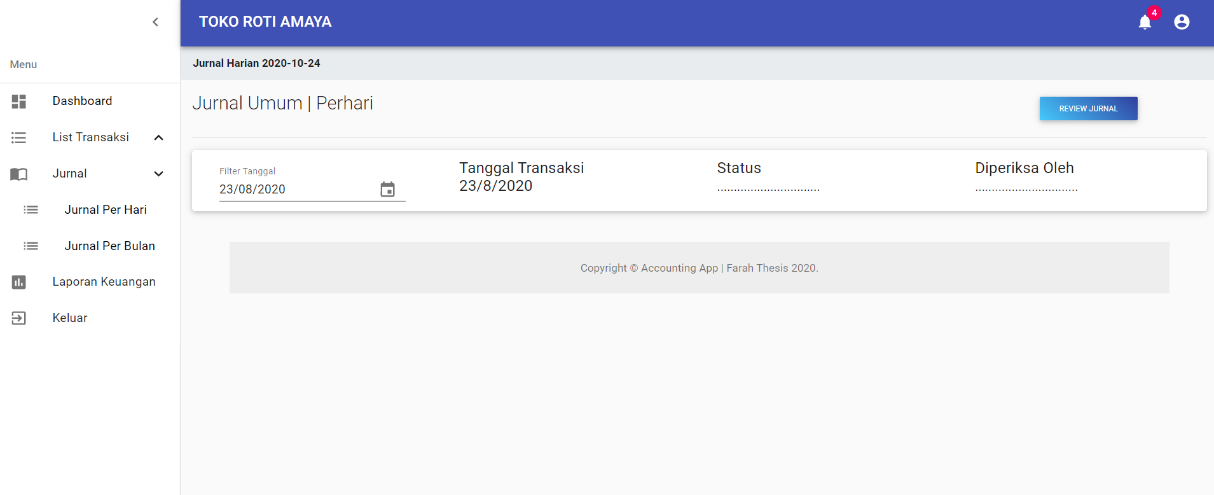
|  |  |
| --- | --- |
| NO | KODE |
| 1 | const JurnalPerhariComponent = ({ props, params }) => { |
| 2 | const { isAuthenticated } = useContext(AuthContext) |
| 3 | const { state, GetDailyJournal, PostingJournal } = useContext(JournalContext) |
| 4 | const [review, setReview] = useState(false) |
| 5 | const [memo, setMemo] = useState('') |
| 6 | const [save, setSave] = useState(false) |
| 7 | const reviewStatus = () => { |
| 8 | if (!review) { |
| 9 | return <small style={{ color: 'black' }}> </small> |
| 10 | } else if (review && !save) { |
| 11 | if (state.additionalData.review\_status === 'Not Reviewed') { |
| 12 | return <small style={{ color: 'orange' }}>In Review</small> |
| 13 | } else { |
| 14 | return <small style={{ color: 'green', fontWeight: 'bold' }}>{state.additionalData.status}</small> } |
| 15 | } else if (review && save && !posting) { |
| 16 | if (state.additionalData.review\_status === 'Not Reviewed') { |
| 17 | return <small style={{ color: 'orange' }}>{state.additionalData.status}</small> |
| 18 | } else { |
| 19 | return <small style={{ color: 'green', fontWeight: 'bold' }}>Siap Posting</small> } |
| 20 | } else { |
| 21 | return <small style={{ color: 'green' }}><b>Direview & Diposting</b></small> } } |
| 22 | const memoComponent = () => { |
| 23 | if (state.additionalData.review\_status === 'Not Reviewed') { |
| 24 | return ( |
| 25 | <Fragment> |
| 26 | <TextField fullWidth label="Memo" variant="outlined" margin="normal" multiline rows={3} rowsMax={4} onChange={(e) => setMemo(e.target.value)} /> |
| 27 | <MDBBtn color="dark-green" gradient="blue" onClick={handleSave} disabled={posting} > |
| 28 | Simpan </MDBBtn> |
| 29 | </Fragment> ) |
| 30 | } else { |
| 31 | return ( |
| 32 | <TextField fullWidth label="Memo" variant="outlined" margin="normal" multiline rows={3} rowsMax={4} defaultValue={state.additionalData.memo} disabled /> |
| 33 | ) } } |
| 34 | const handleReview = () => { |
| 35 | GetDailyJournal(date) |
| 36 | setReview(true) } |
| 37 | return ( |
| 38 | <div> |
| 39 | <MDBRow> |
| 40 | <MDBCol lg='9'> |
| 41 | <h4>Jurnal Umum | Perhari</h4> |
| 42 | </MDBCol> |
| 43 | <MDBBox display="flex" justifyContent="center"> |
| 44 | <MDBBtn color="dark-green" size="sm" gradient="blue" onClick={handleReview} disabled={review}> Review Jurnal |
| 45 | </MDBBtn> }) |
| 46 | export default JurnalPerhariComponent; |

Tabel 5.6 Implementasi Algoritma Review Jurnal (Context.js)

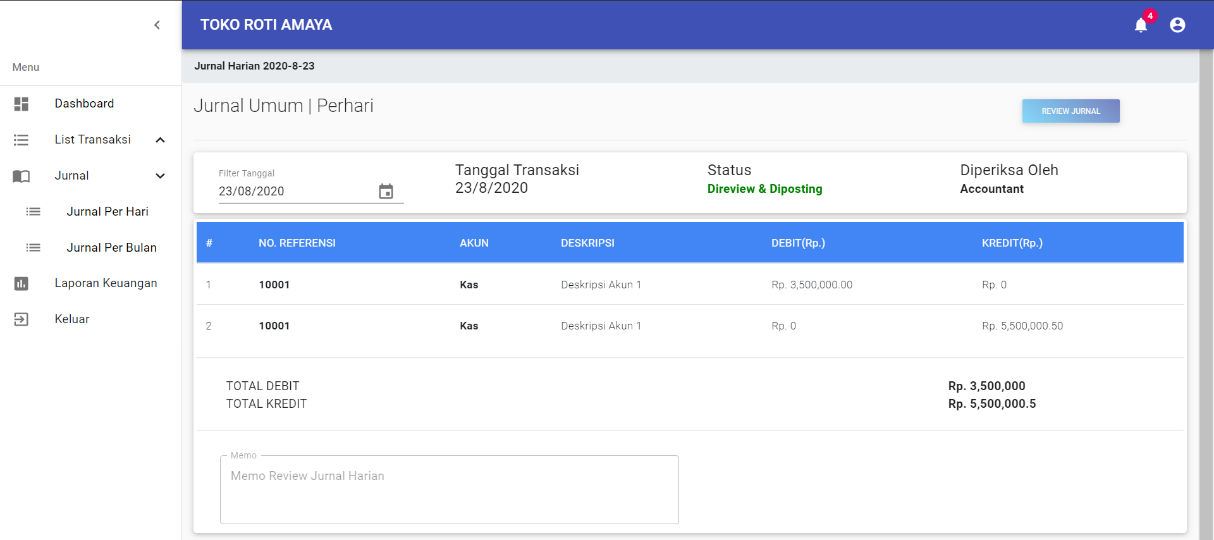
|  |  |
| --- | --- |
| NO | KODE |
| 1 | const JournalReducer = (state, action) => { |
| 2 | switch(action.type){ |
| 3 | case 'LOADING': |
| 4 | return { |
| 5 | ...state, |
| 6 | loading: true, |
| 7 | message: action.payload, } |
| 8 | case 'NO-LOADING': |
| 9 | return { |
| 10 | ...state, |
| 11 | loading: false, |
| 12 | message: '', } |
| 13 | case 'LIST-JOURNAL-DAILY': |
| 14 | return { |
| 15 | ...state, |
| 16 | listJournalDaily:action.payload } |
| 17 | case 'LIST-JOURNAL-DAILY-ADDITIONAL': |
| 18 | return { |
| 19 | ...state, |
| 20 | additionalData:action.payload } |
| 21 | default: |
| 22 | return state } } |
| 23 | const GetDailyJournal = dispatch => (date) => { |
| 24 | dispatch ({type: 'LOADING', payload: 'Menampilkan Data Jurnal...'}) |
| 25 | axios.put(`${API}/journal/${date}/review`, data) |
| 26 | .then(res => { |
| 27 | alert(JSON.stringify(res)) |
| 28 | if(res.data.success){ |
| 29 | dispatch({type: 'NO-LOADING'}) |
| 30 | dispatch({ type: 'LIST-JOURNAL-DAILY', payload: res.data.data }) |
| 31 | dispatch({ type: 'LIST-JOURNAL-DAILY-ADDITIONAL', payload: res.data.additional\_data }) |
| 32 | } else { |
| 33 | alert(res.data.message) |
| 34 | dispatch({ type: 'NO-LOADING' }) |
| 35 | dispatch({ type: 'LIST-JOURNAL-DAILY', payload: [] }) } |
| 36 | }).catch(error => { |
| 37 | dispatch({type: 'NO-LOADING'}) |
| 38 | alert(error) }) } |
| 39 | export const {Provider, Context} = CreateDataContext( |
| 40 | JournalReducer, |
| 41 | {GetDailyJournal, PostingJournal}, |
| 42 | {loading: false, message:'', listJournalDaily:[], additionalData:''} ) |

### Implementasi Antar Muka

Implementasi antar muka review jurnal per hariPada halaman jurnal per hari terdapat *sidebar* menu, tombol “Review Jurnal”, ikon “Kalender” untuk memilih tanggal jurnal, kolom “Tanggal Transaksi”, kolom “Status”, kolom “Diperiksa Oleh” dan tabel Jurnal. Dalam gambar 5.13 menunjukkan halaman jurnal per hari sebelum memilih tanggal transaksi sedangkan Dalam gambar 5.14 menunjukkan halaman jurnal per hari setelah di-*review* sehingga terdapat perubahan status menjadi direview & diposting serta terdapat nama *reviewer*.



Gambar 5.13 Implementasi Antar Muka Jurnal Per Hari



Gambar 5.14 Implementasi Antar Muka Jurnal Per Hari Setelah *Review*

# PENGUJIAN SISTEM

## Pengujian Kondisi Jalur

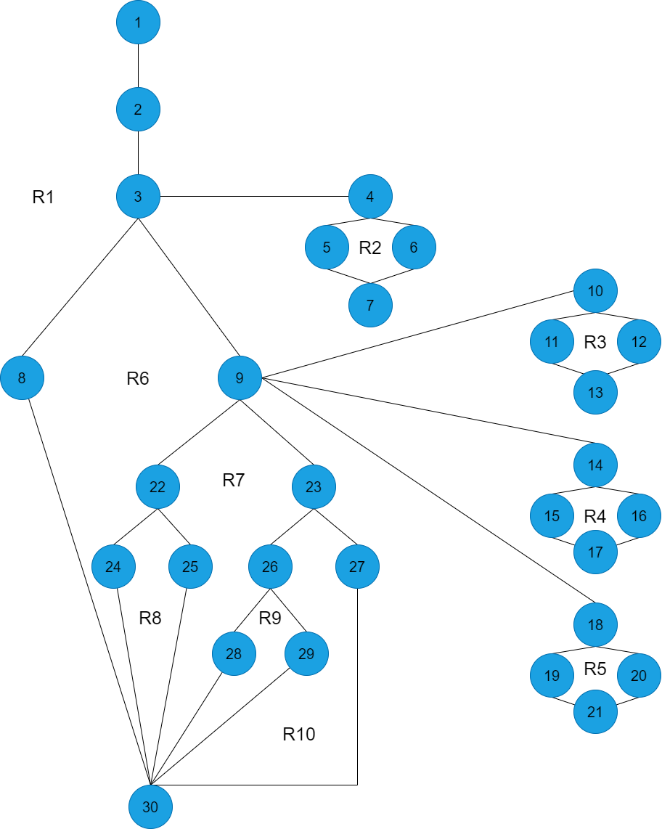
Pengujian kondisi jalur pada *functional component* reviewJurnal dari *file* JurnalPerhariComponent.jsx

*Pseudocode*

Tabel 6.1 *Pseudocode* fungsi komponen reviewJurnal

|  |  |
| --- | --- |
| NO | Pseudocode |
| 1. | Mulai |
| 2. | Inisialisasi fungsi reviewStatus |
| 3. | If tidak review |
| 4. | Nilai colour = black dari atibut *style* padaproperti *small* |
| 5. | Else if review dan tidak save |
| 6. | If reviewStatus dari state additionalData = not reviewed |
| 7. | Nilai colour = orange dari atibut *style* padaproperti *small* dan tampilkan In Review |
| 8. | Else |
| 9. | Nilai colour = green, fontWeight = bolt dari atibut *style* padaproperti *small* dan tampilkan status dari state additionalData |
| 10. | Else if review dan save dan tidak posting |
| 11. | If reviewStatus dari state additionalData = not reviewed |
| 12. | Nilai colour = orange dari atibut *style* padaproperti *small* dan tampilkan status dari state additionalData |
| 13. | Else |
| 14. | Nilai colour = green, fontWeight = bolt dari atibut *style* padaproperti *small* dan tampilkan Siap Posting |
| 15. | Else |
| 16. | Nilai colour = green dari atibut *style* padaproperti *small* dan tampilkan Direview & Diposting |
| 17. | Selesai |

*Flow Graph*

**

Gambar 6.1 *Flow Graph* fungsi komponen reviewJurnal

Perhitungan *Cyclomatic Complexity*

V(G) = 10 R

V(G) = 38 E – 30 N + 2 = 10

V(G) = 9P + 1 = 10

Jalur Independen : 10

**Jalur 1:** 1-2-3-8-30

**Jalur 2:** 1-2-3-9-22-24-30

**Jalur 3:** 1-2-3-9-22-25-30

**Jalur 4:** 1-2-3-9-23-26-28-30

**Jalur 5:** 1-2-3-9-23-26-29-30

**Jalur 6:** 1-2-3-9-23-27-30

**Jalur 7:** 1-2-3-4-5-7

**Jalur 8:** 1-2-3-9-10-11-13

**Jalur 9:** 1-2-3-9-14-16-17

**Jalur 10:** 1-2-3-9-18-19-20

## Pengujian Validasi

Dalam Tabel 6.1 merupakan tabel pengujian validasi yang terdiri dari nama fungsi yang diujikan, kasus uji, hasil yang diharapakan, hasil sesungguhnya dan status. Fungsi yang diujikan pada pengujian validasi adalah posting jurnal per hari dan mencatat transaksi dengan 7 kasus uji seperti yang tertera pada Tabel 6.1. Pengujian validasi menggunakan metode *boundary value analysis* yang menguji nilai batas atas dan batas bawah pada fungsi yang diuji.

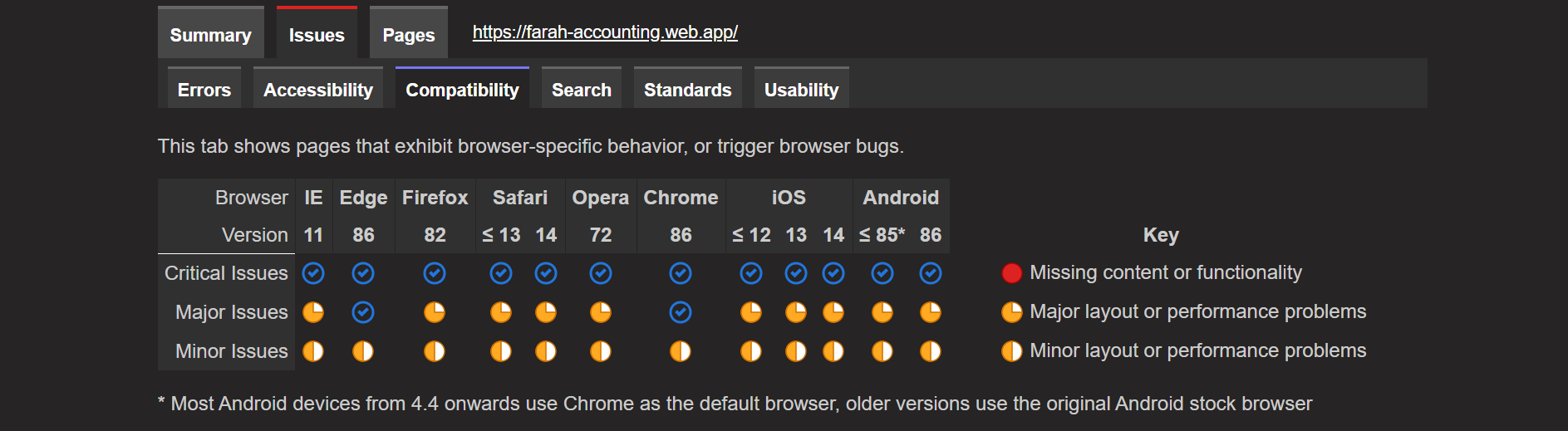
Tabel 6.2 Pengujian Validasi Sistem

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NO | Nama Fungsi | Kasus Uji | Hasil yang Diharapkan | Hasil | Status |
| 1. | Posting Jurnal Per Hari | Tanggal transaksi = null | Muncul pesan peringatan berwarna merah bahwa tidak ada data pada tanggal yang dipilih | Terdapat peringatan pada *front end* dan *back end* bahwa tidak ada data pada tanggal yang dipilih | Pass |
| Pesan review = null | Muncul pesan peringatan untuk melanjutkan posting | Terdapat peringatan untuk melanjutkan posting jurnal | Pass |
| Pesan review = 10.000 karakter | Muncul pesan peringatan untuk melanjutkan posting | Terdapat peringatan untuk melanjutkan posting jurnal | Pass |
| 2. | Mencatat Transaksi | *field* jumlah item = -5 | Tombol *down* *field* jumlah item tidak lagi berubah ketika posisi 0 | Tombol *down* *field* jumlah item minimun sampai pada posisi 0 | Pass |
| Field jumlah item = 200000000000 | *Field* jumlah item tampilkan angka sesuai masukkan dan *field* total tampilkan hasil perkalian dengan harga satuan | *Field* jumlah item tampilkan angka sesuai masukkan dan *field* total tampilkan hasil perkalian dengan harga satuan | Pass |
| Dekripsi produk = 10.000 karakter | Muncul pesan peringatan untuk mengisi kolom yang kosong | Terdapat peringatan dari *backend* untuk periksa masukkan | Pass |
| Dekspsi produk = null | Muncul pesan peringatan untuk mengisi kolom yang kosong | Terdapat peringatan dari *backend* untuk periksa masukkan | Pass |

## Pengujian Kesesuaian

Pengujian kesesuaian atau *compability testing* dilakukan pada dua elemen pengujian *website browser* dan *mobile enviroment*. Pengujian dilakukan pada 6 *website browser* diantaranya: *chrome* versi 86, *internet explorer* versi 11, *safari* versi 13 kebawah, *edge* versi 86, *opera* versi 72 dan *firefox* versi 82 serta 2 *mobile enviroment* yaitu: *ios* versi 12 kebawah dan *android* versi 85 kebawah. Dalam Gambar 6.2 menggambarkan *compability testing* yang dilakukan dengan menggunakan aplikasi *sortsite*.

Hasil pengujian pada *website browser* tidak terdapat masalah serius yang mempengaruhi sistem seperti kehilangan konten ataupun fungsionalitas. Namun ditemukan masalah major dan minor pada beberapa *website browser*. Masalah major terjadi karena *website browser* tidak mendukung properti khusus CSS seperti *grid*, *filter*, *display flex* dan backdrop-*filter*. Masalah minor terjadi karena *website browser* tidak mendukung beberapa *selector* dan *value* CSS seperti *visited* dan *content: none*. Dari semua *website browser* hanya *chrome* dan *egde* yang tidak memiliki masalah major dan semua *website browser* memiliki masalah minor. Hasil pengujian pada *mobile environment* juga tidak terdapat masalah serius terkait fungsionalitas sistem. Namun baik *ios* dan *android* keduanya memiliki masalah major dan minor.



Gambar 6.2 Hasil *Compability Test* Menggunakan *SortSite*

# KESIMPULAN

## Kesimpulan

Berdasarkan hasil pembahasan, disimpulkan bahwa React JS merupakan *javascript* *library* untuk membangun antar muka pengguna berbasis komponen yang membantu pengembang fokus pada penyelesaian logika bisnis bukan mengatur manajemen kode program. Hal ini dapat dilihat dari hasil implementasi kode programmenggunakan *stateless component* dan *react hooks*. Sistem tidak menggunakan kelasmelainkan hanya kumpulan fungsi sehingga properti dapat digunakan berulang kali tanpa perlu didefinisikan pada setiap *children*.

Pengembangan sistem melalui beberapa tahapan diantaranya rekayasa kebutuhan dilakukan untuk mengetahui kebutuhan pengguna secara detail berdasarkan hasil wawancara dengan pengguna dan disesuaikan dengan teknologi yang digunakan. Tahapan perancangan dilakukan untuk memetakan kebutuhan penggunaan sehingga diperoleh dokumen sebagai paduan implementasi. Tahapan implementasi dilakukan untuk menerapkan dokumen perancangan kedalam kode program. Tahapan pengujian dilakukan untuk menguji sistem secara keseluruhan.

Tahapan pengujian dilakukan dengan strategi pengujian kotak hitam dan kotak putih. Metode pengujian yang dilakukan diantaranya pengujian kondisi jalur, kesesuaian dan validasi fungsional. Berdasarkan pengujian kondisi jalur dengan menguji fungsi reviewJurnal diperoleh hasil bahwa semua jalur telah setidaknya sekali dilewati. Pada pengujian validasi terdapat tujuh kasus uji dengan hasil semua kasus uji sesuai dengan ekspetasi hasil pengujian. Pengujian kesesuaian dengan menguji sistem pada enam *website browser* dan dua *mobile environment*, didapatkan hasil sistem berjalan sesuai dengan fungsionalitas meskipun terdapat masalah major dan minor.

## Saran

Adapun saran pengembangan sistem manajemen akuntansi keuangan perusahaan toko roti Amaya berbasis web:

Integrasi antara sistem keuangan perusahaan dengan kasir toko sehingga menjadi kesatuan sistem yang utuh dan menambah nilai komersil pada sistem.

Melengkapi keseluruhan siklus akuntansi dengan menambahkan buku besar, neraca saldo dan laporan kas kedalam sistem.

Menambahkan kebutuhan non-fungsional untuk meningkatkan kualitas sistem.

Melakukan *user acceptance testing* kepada pengguna sebagai pesetujuan akhir kelayakan sistem.

Melakukan *visual GUI testing* dan *API testing* sesuai dengan teknik pengujian terbaru yang digunakan oleh *QA Engineer*.

# DAFTAR REFERENSI

A M, V., & Sonpatki, P. (2016). *REACT JS by example: Building Modern Web Applications With React .* Mumbai: Packt Publishing.

Abramov, D. (2020, January 1). *Usage With React.* Retrieved September 1, 2020, from Redux JS: https://redux.js.org/basics/usage-with-react

Aggarwal, S. (2018, March). Modern Web-Development using ReactJS. *International Journal of Recent Research Aspects, 5*(1), 133 - 137.

Alegroth, E. (2015). *Visual GUI Testing: Automating High-Level Software Testing in Industrial Practice.* Goteborg: Department of Computer Science & Engineering.

Almada-Lobo, F. (2015). The Industry 4.0 revolution and the future of Manufacturing Execution Systems (MES). *Journal of Innovation Management, 3*, 16 - 21.

Al-Masree, H. K. (2015). Extracting Entity Relationship Diagram (ERD) From Relational Database Schema. *International Journal of Database Theory and Application, VIII*(3), 15-26. Retrieved June 18, 2020, from http://dx.doi.org/10.14257/ijdta.2015.8.3.02

Andriansyah, D. (2018). Pengujian Kotak Hitam Boundary Value Analysis Pada Sistem Informasi Manajemen Konseling Tugas Akhir. *Indonesian Journal on Networking and Security, VII*(1), 13-18. Retrieved May 1, 2020, from https://core.ac.uk/reader/228811730

Arora, R., Arora, N., & India, M. (2016). Analysis of SDLC Models.

Chen, X., Ji, Z., Fan, Y., & Zhan, Y. (2017). Restful API Architecture Based on Laravel Framework. *Journal of Physics: Conference Series*, 012016. doi:10.1088/1742-6596/910/1/012016

Dharwiyanti, S., & Satrio Wahono, R. (2003). Pengantar Unified Modeling Language (UML). *Kuliah Umum IlmuKomputer.Com*. Retrieved 10 7, 2019, from http://rosni-gj.staff.gunadarma.ac.id/Downloads/files/14321/10.+Unified+Modeling+Language.pdf

Dunia, F. A., & Abdullah, W. (2012). *Akuntansi Biaya.* Jakarta: Salemba Empat.

Henderi, S. (2009). OBJECT ORIENTED MODELLING WITH UNIFIED MODELING LANGUAGE (UML). *DIKTAT PERANCANGAN SISTEM INFORMASI*, 1-73.

Indonesia, I. A. (2009). *Standar Akuntansi Keuangan: PSAK No. 1 – Penyajian Laporan Keuangan.* Jakarta: Salemba Empat.

Indrajit, R. E. (2001). *Pengantar konsep dasar manajemen sistem informasi dan teknologi informasi.* Jakarta: Elex Media Komputindo.

Jan, S. R., Shah, S. T., Johar, Z. U., Shah, Y., & Khan, F. (2016, March 31). An Innovative Approach to Investigate Various Software Testing Techniques and Strategies. *International Journal of Scientific Research in Science, Engineering and Technology, II*(2), 682-689. Retrieved May 1, 2020, from https://www.researchgate.net/profile/Fazlullah\_Khan2/publication/303280520\_An\_Innovative\_Approach\_to\_Investigate\_Various\_Software\_Testing\_Techniques\_and\_Strategies/links/576e56e008ae842225a849ca.pdf

Jorgensen, P. C. (2014). *Software Testing A Craftsman’s Approach* (4th ed.). Florida: CRC Press.

Larman, C. (2005). *Applying UML and patterns: An Introduction to Object-Oriented Analysis and Design Iterative Development .* Boston: Addison Wesley Professional.

Li, Q., & Chen, Y.-L. (2009). *Modeling and Analysis of Enterprise and Information Systems .* Berlin: Springer.

Luea, Y. B., Loo, W. K., Tham, W. Y., & Tan, S. F. (2012). Software Development Life Cycle AGILE vs Traditional Approaches. *International Conference on Information and Network Technology, 37*, 162-167.

Maith, H. A. (2013). ANALISIS LAPORAN KEUANGAN DALAM MENGUKUR KINERJA KEUANGAN PADA PT. HANJAYA MANDALA SAMPOERNA TBK. *EMBA, I*(3), 619-628.

Mall, R. (2018). *Fundamentals Of Software Engineering* (Vol. V). Delhi: PHI Learning Private Limited.

Muawanah, U., & Poernawati, F. (2008). *KONSEP DASAR AKUNTANSI DAN PELAPORAN KEUANGAN.* Jakarta: Direktorat Pembinaan Sekolah Menengah Kejuruan.

Mulyadi. (2015). *Akuntansi Biaya* (5 ed.). Yogyakarta: Sekolah Tinggi Ilmu Manajemen YKPN.

Mulyani, S. (2016). *Metode Analisis dan Perancangan Sistem.* Bandung: Abdi Sistematika.

Naik, K., & Tripathy, P. (2008). *Software testing and quality assurance theory and practice.* New Jersey: John Wiley & Sons, Inc.

Otwel, T. (2020, September 2020). *Lumen Documentation*. Retrieved October 10, 2020, from Lumen Laravel: https://lumen.laravel.com/docs/8.x/installation

Pressman, R. (2001). *Software Engineering: A Practitioner's Approach.* New York: McGraw Hill Higher Education.

Rajmohan, R., & Suburayan, V. (2010, January). Evolving A New Model (SDLC Model-2010) For Software Development Life Cycle (SDLC). *International Journal of Computer Science and Network Security, 10*, 112-119.

Ranjita, S., Panthi, V., Behera, P. K., & Mohaprata, D. P. (2012, March). Automatic Test case Generation From UML State Chart Diagram. *International Journal of Computer Applications, XLII*, 26 - 36. doi:10.5120/5705-7756

Robbins, J. N. (2012). *Learning Web Design Fourth Edition A Beginner’s Guide to HTML, CSS, JavaScript, and Web Graphics.* Canada: O’Reilly Media, Inc.

Saraswat, S. (2019, November 11). *Unified Modeling Language (UML) | State Diagrams*. Retrieved from GeeksForGeeks: https://www.geeksforgeeks.org/unified-modeling-language-uml-state-diagrams/

Satzinger, J. W. (2012). *Introduction to systems analysis and design : an agile, iterative approach.* Canada: Course Technology.

Shalahudin, M., & Rosa, A. (2011). Modul Pembelajaran Rekayasa (Terstruktur dan Berorientasi Objek). *Modula*.

Suehring, S. (2002). *MySQL Bible.* New York: Wiley Publishing, Inc .

Sugi Ardana, I. M. (2019, October). Pengujian software menggunakan metode boundary value analysis dan decision table testing. *Jurnal Teknologi Informasi ESIT, XIV*(3), 40-47. Retrieved May 1, 2020, from http://www.jurnal-eresha.ac.id/index.php/esit/article/view/152/113

Susilo, B. (2009). *Analisa Laporan Keuangan untuk Menilai Kinerja Keuangan Perusahaan.*

Yulianti, & Saputra, R. S. (2017). Analisis Harga Pokok Produksi Roti Berdasarkan Metode Full Costingdan Variable Costing. *Jurnal Online Insan Akuntan, II*, 229-236. Retrieved July 4, 2020, from http://ejournal-binainsani.ac.id/index.php/JOIA/article/view/642/524

LAMPIRAN A PANDUAN WAWANCARA

## A.1 Daftar Pertanyaan Wawancara

Daftar pertanyaan wawancara ini berfungsi sebagai acuan dalam menjawab rumusan masalah pada penelitian yang berjudul “Pengembangan Sistem Manajemen Akuntansi Keuangan Perusahaan Untuk Toko Roti Amaya Berbasis Website Menggunakan React JS”. Berikut pertanyaan wawancara untuk menjawab rumusan masalah bagaimana hasil analisis dan spesifikasi kebutuhan sistem manajemen akuntansi keuangan pada Toko Roti Amaya:

1. Bagaimana langkah dalam proses pelaporan keuangan?
2. Kapan pelaporan keuangan dilakukan?
3. Bagaimana template dokumen laporan keuangan yang digunakan Amaya bakery?
4. Siapa pihak yang terlibat dalam proses pelaporan?
5. Apa peran masing-masing pihak yang terlibat dalam pembentukan laporan keuangan?
6. Apakah proses pelaporan antara toko dan marketing sama?
7. Bagaimana proses pelaporan keuangan antara pusat dan toko?
8. Apa dokumen yang dihasilkan dari laporan keuangan?
9. Apa dokumen yang dibutuhkan untuk menunjang proses pelaporan keuangan?
10. Apa data yang dibutuhkan untuk menunjang proses pelaporan keuangan?
11. Apakah fungsi dari sistem yang sudah digunakan Amaya bakery saat ini? Apakah sistem tersebut juga dipergunakan untuk fungsi lain selain pelaporan keuangan misalnya stock opname?
12. Apa basis dari sistem yang digunakan saat ini?
13. Bagaimana proses pengiriman dokumen penunjang laporan keuangan?
14. Dari semua siklus akuntansi, apa proses akuntansi yang paling dibutuhkan oleh Amaya bakery?
15. Apa harapan dari sistem semisal dapat digunakan oleh pihak Amaya bakery?

## A.2 Hasil Wawancara

Tanggal : 24 Februari 2020

Waktu : 14.10 – 15.00

Narasumber : Bapak Hera

Jabatan : Marketing Amaya bakery

Jawaban:

1. Dari masing masing divisi atau toko kirim bukti transaksi ke akuntan, dari akuntan akan mencatat dan mencocokan bukti tagihan yang ada dengan mutasi rekening. Kalau sudah cocok baru dibuat laporan keuangan nya manual begitu. Kalau ada yang belum sesuai ditanyakan balik ke kita.
2. Laporan keuangan dibuat setiap bulan, tapi untuk pelaporan bukti transaksi dari marketing, toko atau admin dilakukan setiap hari.
3. Untuk detail template laporan maaf belum bisa ditunjukkan, tapi kurang lebih sama dengan laporan keuangan perusahaan pada umumnya. Ada pengeluaran dan pemasukkan.
4. Akuntan saja.
5. Akuntan yang membuat laporan. Untuk toko dan marketing seperti saya hanya laporan bukti saja ke akuntan.
6. Secara keseluruhan proses kurang lebih sama, kalau dari toko mereka sudah ada laporan keuangan sendiri nanti tinggal kirim bareng bukti tagihan. Kalau seperti saya juga tinggal kirim bukti tagihan manual.
7. Kalau dari toko biasanya sesuai sif, ditoko itu ada dua sif pagi dan siang. Pagi dari jam 06.00 – 13.00 lalu sif siang dari jam 13.00 – 21.00. Setiap transaksi dari sif harus langsung dilaporkan hari itu juga ke akuntan.
8. Sepengetahuan saya laporan keuangan itu isinya laporan laba rugi dan neraca.
9. Kalau dokumen paling hanya butuh struk penjualan dan struk pemesanan.
10. Data yang dibutuhkan seperti detail item yang terjual, detail transaksi seperti jam, jumlah dan pembayaran via apa (debit, ovo, cash)
11. Sistem untuk buat laporan keuangan. Untuk rekap itu ada sistem sendiri di logistik.
12. Sistem menggunakan dekstop, per divisi ada komputernya, untuk toko juga begitu.
13. Pengirimannya bisa via email atau kadang via whatsapp.
14. Yang pasti kita butuh jurnal, lalu laporan laba rugi, neraca disesuaikan dengan standar yang ada dan kebutuhan perusahaan juga.
15. Harapannya sistem bisa membatu Amaya bakery lebih cepat dalam proses pelaporan jadi kita tidak perlu dua kali input transaksi dan kirim bukti tagihan. Laporan keuangan juga akurat perhitungannya.