# KATA PENGANTAR

Puji dan syukur kehadirat Tuhan Yang Maha Esa yang telah melimpahkan rahmat dan berkat-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul “TRANSMISI DATA MENGGUNAKAN SISTEM RFID PADA SMART SHOPPING CART UNTUK EFISIENSI PROSES CHECKOUT PADA KASSA SUPERMARKET DI INDONESIA**”** dengan baik dan tepat waktu. Skripsi ini dibuat dengan tujuan memenuhi salah satu syarat untuk gelar kesarjanaan pada Program Teknik Informatika di Universitas Bina Nusantara.

Dalam kesempatan ini, penulis menyadari bahwa skripsi ini tidak dapat terselesaikan tanpa dukungan dari berbagai pihak. Terima kasih penulis ucapkan terutama kepada:

1. Bapak Prof. Dr. Ir. Harjanto Prabowo, M.M., selaku Rektor Universitas Bina Nusantara yang telah mendukung penulis dalam penulisan skripsi ini.
2. Bapak Fredy Purnomo, S.Kom., M.Kom., selaku *Dean of School of Computer Science* *Program* yang telah mendukung penulis dalam penulisan dan kegiatan skripsi ini.
3. Bapak Dr. Derwin Suhartono, S.Kom., M.T.I., selaku *Head of Computer Science* *Program* yang telah mendukung penulis dalam penulisan dan kegiatan skripsi ini.
4. Bapak Hanry Ham, S.Kom., M.T.I., selaku dosen pembimbing yang telah mendukung penulis dalam penulisan skripsi ini.
5. Seluruh Dosen, Staf, dan Karyawan Universitas Bina Nusantara yang telah turut mendukung penulisan skripsi ini.
6. Tidak lupa pula penulis ucapkan terima kasih yang sebesar-besarnya untuk keluarga, teman- teman, dan semua pihak yang tidak dapat penulis sebutkan satu per satu, yang telah memberikan saran, kritik, semangat, dan bantuan-bantuan lainnya baik secara langsung maupun tidak langsung kepada penulis dalam menyelesaikan penulisan skripsi ini.

Akhir kata penulis berharap semoga skripsi ini dapat memberikan manfaat bagi semua pihak serta menambah wawasan bagi kita semua khususnya dalam bidang teknik informatika. Adapun skripsi ini masih jauh dari sempurna. Oleh karena itu, dengan harapan besar penulis menerima kritik dan saran yang membangun agar penulisan skripsi dapat diperbaiki di kemudian hari. Sekian dan terima kasih atas perhatiannya.

Jakarta, Mei 2020

Penulis

# BAB 1

## PENDAHULUAN

### 1.1 Latar Belakang

Antrean saat melakukan *checkout* dalam sebuah *supermarket* merupakan hal yang krusial dan menjadi salah satu poin penting dalam usaha ritel. Antrean yang panjang ketika melakukan proses *checkout* adalah kondisi *environment* yang tidak ideal bagi pembeli. *Supermarket* dengan antrean yang panjang dapat menyebabkan penurunan jumlah dan loyalitas pelanggan. Sesuai dengan survei yang dilakukan oleh Box Technologies dan Intel di UK pada tahun 2016, didapati bahwa 86% dari responden cenderung menghindari toko/ritel yang memiliki antrean yang panjang. Selain itu, 70% dari responden cenderung tidak ingin kembali berbelanja di toko tersebut apabila mereka pernah merasakan antrean yang panjang pada kunjungan mereka yang sebelumnya, karena membuat mereka frustasi dan tidak dapat mencapai tujuannya (*Jie, 2010; Colin, 2010*). Permasalahan antrean dapat ditangani dengan cara efisiensi proses. Kecepatan dan penghematan waktu dapat menunjang aktivitas lainnya seperti kembali ke tempat kerja, pergi ke tujuan selanjutnya dalam pusat perbelanjaan, dan sebagainya.

Menurut jurnal *Japanese Auto-Checkout Piloting Expands*, konsep yang digunakan oleh *supermarket* yang ada di luar negeri seperti Lawson di Jepang adalah konsep *internet of things* (IoT) untuk mengefisiensikan proses pembelanjaan. Salah satu perangkat yang termasuk dalam IoT dan digunakan oleh *supermarket*-*supermarket* tersebut adalah *Radio Frequency Identification* (RFID). RFID dapat digunakan untuk pertukaran data melalui jaringan. RFID terdiri dari 2 komponen utama yaitu RFID *tag* dan RFID *reader*. RFID *tag* akan mentransmisikan data ke RFID *reader* yang akan diproses selanjutnya.

Pada sistem yang sudah berjalan dan umum digunakan sekarang, sistem *checkout* menggunakan *barcode* sebagai alat bantu untuk mengidentifikasi suatu barang. Penggunaan *barcode* sebagai alat pengenal tersebut memiliki kekurangan yaitu hanya bisa melakukan *scan* satu barang pada satu waktu sehingga ketika kuantitas barang yang dibeli meningkat maka waktu untuk proses *checkout* juga semakin lama. Selain itu, *barcode* memiliki batasan dimana *barcode* harus berada pada jarak baca dari *scanner* sehingga *barcode* harus diarahkan secara langsung ke *scanner*. Maka dapat disimpulkan bahwa pemanfaatan sistem RFID pada proses *checkout* memerlukan waktu yang lebih sedikit dibandingkan dengan *barcode*.

Dengan demikian, sebagai tawaran solusi efisiensi proses *checkout* di kassa *supermarket*, dibuatlah *smart shopping cart* dengan menerapkan sistem RFID pada keranjang belanja masing-masing pembeli sehingga proses pemindaian *barcode* secara manual tidak lagi diperlukan di kassa, melainkan cukup menerima data dari RFID *reader* di keranjang belanja sebelum masuk ke proses pembayaran. Dasar pemilihan RFID *tag* dan RFID *reader* mencakup biaya yang relatif murah dibanding teknologi pengenal lainnya seperti *Ultra Wideband Realtime Location Service* (UWB RTLS) dan *Computer Vision*, kecepatan yang dibutuhkan untuk pengiriman data antara RFID *tag* dengan RFID *reader*, kemudahan pengimplementasian pada proses belanja yang sudah ada, serta kepraktisan pemrograman pada RFID *reader* yang berbasis Arduino dengan kapabilitas dan kapasitas yang cukup tinggi.

### 1.2 Perumusan Masalah

Rumusan masalah pengembangan sistem ini antara lain:

1. Bagaimana cara mengurangi antrean yang terjadi di *supermarket* akibat pemindaian *barcode* secara manual?
2. Bagaimana cara memastikan akurasi pemindaian produk di *smart shopping cart* sehingga diperoleh data yang tepat mengenai setiap produk, kemudian menghitung total harga sesuai total belanja?
3. Bagaimana cara mengurangi waktu yang diperlukan ketika proses *checkout* barang?
4. Bagaimana cara memungkinkan daftar belanja dapat dipantau pada *smartphone* secara *realtime*?

### 1.3 Ruang Lingkup

Ruang lingkup dari pengembangan sistem ini antara lain:

1. Batasan geografis berupa supermarket di Jakarta, Indonesia.
2. Sistem keamanan yang sudah berjalan seperti *security gate* dari *Supermarket* tetap dipasang.
3. Keranjang belanja selalu tersedia.
4. Mesin RFID *reader*, mesin di kassa dan aplikasi di *smartphone* selalu terkoneksi *Internet* yang stabil.
5. Pembayaran dilakukan dengan *Point of Sales* (POS) yang sudah terintegrasi dengan supermarket, tidak termasuk dalam sistem *Smart Shopping Cart*.

Batasan pengembangan RFID *system*:

1. Pengembangan mencakup *database* dan *server* yang digunakan *supermarket* terkait produk, *smart shopping cart,* serta aplikasi *Smart Shopping Assistant*.
2. Pengembangan terfokus pada akurasi deteksi produk pada *smart* *shopping* *cart* menggunakan sistem RFID.
3. Pengembangan mencakup kenyamanan aplikasi untuk me*monitor* *smart shopping cart.*

### 1.4 Tujuan & Manfaat

Tujuan dari pengembangan sistem ini, antara lain:

1. Mengurangi antrean dan berhasil melakukan *checkout*.
2. Memindai produk secara akurat di dalam *smart shopping cart* menggunakan RFID *reader* dan memberikan data pada *server* yang bisa dipantau secara langsung menggunakan aplikasi berbasis *smartphone* yaitu *Smart Shopping Assistant*.
3. Mempercepat proses *checkout* di *supermarket* dengan menjumlahkan total bayar saat proses belanja sebelum di kassa dengan menggantikan *barcode* dengan RFID *tag*.

Manfaat dari pengembangan sistem ini, antara lain:

1. Bagi penulis
   1. Dapat menambah pengalaman dalam pembuatan sistem yang dapat diterapkan pada skala kecil ataupun besar.
   2. Dapat menerapkan materi-materi selama proses pembelajaran di perkuliahan.
2. Bagi pengguna
   1. Pelanggan tidak merasa dirugikan secara waktu.
   2. Loyalitas pelanggan pada *supermarket* meningkat.
   3. Pelanggan dapat memantau daftar belanja secara realtime pada *smartphone*.
   4. Biaya implementasi sistem tidak terlalu mahal dan tidak diperlukan perubahan ruang ataupun alur belanja yang sudah ada, tidak mengurangi lapangan pekerjaan di *supermarket*.
   5. Sistem dapat diimplementasikan dalam skala yang lebih besar untuk menanggapi kebutuhan bisnis ritel lebih lanjut.

### 1.5 Metode yang Digunakan

Dalam penerapan sistem *Internet of Things* pada alur pembelanjaan di *supermarket* yang sudah berjalan, digunakan metode berikut:

1. RFID (*Radio Frequency Identifier*)
   1. *Passive* RFID *Tag* sebagai tanda pengenal produk.
   2. RFID *reader* sebagai pembaca tanda pengenal.
2. Arduino

Arduino berfungsi sebagai *microcontroller* yang akan digunakan untuk memproses data yang telah ditangkap RFID *reader*.

1. *Database*

Tempat penyimpanan data-data transaksi serta produk yang dijual oleh *supermarket* seperti *ID* barang, jenis, harga dan lain sebagainya.

1. WLAN (*Wireless Area Network*)

Jaringan yang menghubungkan perangkat - perangkat didalamnya melalui koneksi nirkabel.

1. *Server*

Entitas dimana data dikumpulkan untuk didistribusikan ke komponen-komponen dalam sistem.

1. *Waterfall* SDLC (*Software Development Life Cycle*)

Metode/acuan yang digunakan dalam pengembangan *software*.

1. *Black Box Testing*

Uji coba kelayakan aplikasi.

1. *Integration Testing*

Uji coba alur kerja sistem *smart shopping cart*.

### 1.6 Sistematika penulisan

Sistematika dalam penyusunan skripsi ini adalah sebagai berikut:

1. BAB 1 PENDAHULUAN

Bab ini membahas mengenai latar belakang, rumusan masalah, ruang lingkup, tujuan dan manfaat, metode penelitian, serta sistematika penulisan.

1. BAB 2 TINJAUAN REFERENSI

Bab ini berisikan teori-teori yang diperoleh dari berbagai literatur dan *tools* yang digunakan dalam pengerjaan project yang dikerjankan.

1. BAB 3 METODOLOGI

Bab ini membahas mengenai kerangka berpikir, analisis kebutuhan, rancangan aplikasi, perancangan sistem, dan perancangan basis data.

1. BAB 4 HASIL DAN PEMBAHASAN

Bab ini berisikan hasil penelitian berupa implementasi aplikasi yang bertujuan untuk menyelesaikan masalah yang telah didefinisikan sebelumnya.

1. BAB 5 KESIMPULAN

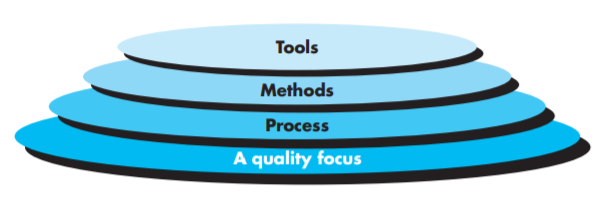
Bab ini berisikan kesimpulan dari penelitian yang telah dilakukan dan saran-saran untuk pengembangan dari penelitian ini kedepannya.

# BAB 2

# TINJAUAN REFERENSI

### 2.1 Software Engineering

*Software Engineering* merupakan sebuah pendekatan pada pengembangan, pengoperasian serta pemeliharaan *software* secara sistematis, teratur dan kuantitatif, ataupun pembelajaran dari pendekatan tersebut. Pressman (2015, p.15-16) menambahkan bahwa selain keteraturan, dibutuhkan kemampuan beradaptasi serta ketangkasan yang signifikan, dan digambarkan dalam sebuah proses yang berlapis-lapis.



**Gambar 2.1 Lapisan-lapisan Software Engineering**

Fondasi dari *software engineering* sendiri merupakan *quality focus* yang kemudian diwujudkan dengan proses, kemudian proses tersebut dijalankan dengan metode-metode tertentu, dan metode-metode tersebut didukung dengan *tools* yang ada. *Quality focus* yang dibuat antara lain *milestone*, standar yang didefinisikan serta perubahan-perubahan diberikan wadah agar dapat dipantau terus-menerus. Kemudian proses didefinisikan sesuai dengan *milestone* untuk mencapai standar yang ada. Metode yang digunakan berupa seperangkat tugas yang mencakup tahap mengkomunikasikan *requirement*, *design* dan *modelling*, membuat *source* dari program itu sendiri, *testing* serta maintenance ataupun support terkait dengan *software* yang diciptakan. *Tools* yang digunakan dapat berupa otomasi ataupun otomasi semu yang memerlukan bantuan manusia untuk merealisasikan metode-metode tersebut, adapun *tools* yang digunakan untuk melakukan integrasi antar satu proses dengan lainnya.

### 2.2 Software Development Life Cycle (SDLC)

Menurut Pressman (2015), sebuah *software* dikembangkan layaknya seperti sebuah *object* yang memiliki siklus dengan 5 tahap secara umum, yaitu tahap *Communication*, *Planning*, *Modelling*, *Construction*, dan *Deployment*.

### 2.3 Object Oriented Programming (OOP)

*Object* *Oriented* *Programming* merupakan sebuah paradigma pemrograman yang menganggap segala sesuatu sebagai *object* yang memiliki data berupa atribut dan aksi, prosedur ataupun kapabilitasnya disebut dengan method. Menurut paradigma ini, berjalannya program merupakan proses berinteraksinya satu *object* dengan lainnya, melakukan fungsi sesuai dengan komponennya masing-masing.

### 2.4 Internet of Things

*Internet of Things* (IoT) adalah sebuah sistem terhubung antara alat komputasi, mekanikal dan mesin *digital*, *object*, hewan, atau orang yang disediakan dengan tanda pengenal unik (UIDs) dan kemampuan untuk mentransfer data melalui network tanpa memerlukan interaksi *human-to-human* atau *human-to-computer*. “*Thing*” dalam *Internet of Things* bisa berupa orang yang *hearth monitoring implant*, sebuah peternakan dengan *biochip transponder*, sebuah mobil yang mempunyai sensor untuk memberitahu pengemudi apabila tekanan udara didalam ban rendah atau *object* apapun yang dapat ditanamkan IP *address* dan bisa mengirimkan data melalui *network*.

### 2.5 Radio Frequency Identification (RFID)

*Radio Frequency Identification* (RFID) adalah teknologi yang menggunakan gelombang radio untuk secara otomatis mengenali orang atau *object* dari jarak beberapa inci atau beberapa kaki. Teknologi RFID menggunakan *automatic data capture system* yang membantu dalam meningkatkan efisiensi sistem. Teknologi ini menggunakan kombinasi dari *tag* dan *reader* untuk mengenali sebuah *object*. RFID *tag* adalah sebuah alat elektronik berukuran kecil yang terdiri dari *microchip* dan *antenna*. In*form*asi yang tersimpan dalam sebuah *tag* dapat ditransmisikan secara nirkabel. RFID *tag* dibagi menjadi 3 tipe, yaitu:

1. RFID *Tag* Pasif

RFID *Tag* Pasif adalah *tag* yang tidak mempunyai *power supply*. *Tag* ini memperoleh daya dari RFID *reader*. RFID *reader* akan mengirimkan gelombang elektromagnetik yang akan menimbulkan arus pada antena yang dimiliki oleh RFID *tag*. RFID *tag* lalu akan merefleksikan sinyal transmisi dan akan menambahkan informasi yang ada dengan mengatur sinyal refleksi tersebut.

1. RFID *Tag* Semi-Pasif

RFID Tag Semi-Pasif adalah RFID *tag* yang menggunakan baterai sebagai *power supply*, sedangkan untuk cara berkomunikasi dengan RFID *reader* adalah sama dengan RFID *tag* pasif.

1. RFID *Tag* Aktif

RFID *Tag* Aktif adalah RFID *tag* yang juga punya *power supply* sendiri. *Tag* aktif memiliki jarak baca yang lebih jauh daripada *tag* pasif. RFID *tag* aktif akan mengirimkan sinyal secara mandiri yang selanjutnya akan ditangkap RFID *reader*.

*Code* yang disimpan dalam RFID *tag* akan dipasangkan ke sebuah *object* sehingga *object* tersebut bisa dikenali. Lalu *reader* akan membaca informasi yang ditransmisikan oleh *tag* tersebut.

### 2.6 Unified Modelling Language (UML)

Menurut Wixom (2015), adalah suatu teknik diagram yang digunakan untuk memberikan gambaran umum mengenai suatu *system* yang dibangun dengan metode *object* oriented. Selama perkembangannya UML telah melewati beberapa versi. Saat ini versi dari UML adalah versi 2.5. UML versi 2.5 memiliki 15 teknik diagram yang digunakan untuk menggambarkan suatu model dari sebuah *system*. UML dipecah menjadi 2 bagian besar yaitu untuk structure diagram dan untuk behavior diagram.

#### 2.6.1 Use Case Diagram

*Use* *case* adalah alat yang digunakan untuk memahami proses bisnis dan fungsionalitas suatu *system*. Sebuah *use*-*case* diagaram digunakan ketika mengumpulkan dan mendefinisikan kebutuhan untuk suatu *system*.  Sebuah *use* *case* diagram mendefinisikan dengan cara yang sangat sederhana fungsionalitas utama dari suatu *system* dan *use*r-*use*r yang akan berinteraksi dengan *system* tersebut.

Elemen-elemen dari *use* *case* diagram terdiri atas:

1. *Actor*s

*Actor* adalah *role* yang yang digunakan seorang *use*r untuk berinteraksi dengan *system*. *Actor* dapat merepresentasikan seorang individu atau sebuah *system* lain yang dapat berinteraksi dengan *system* terkait.

1. *Association*

*Use* *case* terhubung dengan *actor* melalui asosiasi yang dilambangkan dengan sebuah garis.

1. *Use* *case*

*Use* *case* dilambangkan dengan oval pada *use* *case* diagram. *Use* *case* merupakan major process yang memberikan keuntungan kepada *actor*.

1. *Subject boundary*

*Subject boundary* merupakan *scope* dari sebuah *system* yang menggambarkan secara jelas bagian *external* maupun *internal* dari *system*.

|  |  |
| --- | --- |
| Simbol | Arti |
|  | *Actor* |
|  | *Use* *case* |
|  | Subject |
|  | asosiasi |
|  | Include *relationship* |
|  | Exclude *relationship* |
|  | Generalization *Relationship* |

**Tabel 2.1 Simbol dan Arti dalam Pembuatan *Use Case***

#### 2.6.2 Use Case Description

*Use* *case* *description* digunakan untuk memodelkan detail *requirement* untuk sebuah sistem. Mulai dari *importance level, primary* *actor*, *trigger*, deskripsi singkat *use* *case*, releasi *use* *case* tersebut dengan *use* *case* lain, *flow* dari *use* *case* dan alternatif *flow* *use* *case* tersebut. (Wixom).

#### 2.6.3 Activity Diagram

Menurut Wixom (2015), *Activity diagram* digunakan untuk menambah pemahaman kita mengenai proses bisnis dan model *use*-*case* kita. Secara teknis, sebuah *activity diagram* bisa digunakan untuk berbagai aktivitas bisnis proses. *Activity diagram* akan menggambarkan alur bisnis dari *system* yang dibuat.

Elemen-elemen dari sebuah *Activity Diagram* antara lain:

1. *Action* dan aktivitas

*Action*s dan aktivitas merepresentasikan sebuah prilaku. *Action* dan aktivitas digambarkan dengan bentuk persegi panjang bersudut tumpul pada activity diagram. Komponen ini memiliki nama yang dimulai dengan kata kerja dan diakhir dengan kata benda misalnya, “*Get Patient Information*”. Penamaan sebuah komponen *action* dan aktivitas tidak boleh terlalu panjang namun harus memberikan informasi yang cukup sehingga pembaca dapat dengan mudah mengerti apa yang dilakukan komponen *action* dan aktivitas tersebut.

1. *Object* *nodes*

*Action* dan aktivitas secara tipikal memodifikasi atau mentrans*form*asi *object*. *Object* nodes menggambarkan *object* yang ditrans*form*asi tersebut pada activity diagram. *Object* node digambarkan dengan bentuk persegi panjang dan memiliki nama yang dituliskan didalam persegi tersebut. Secara esensial, *object* node merepresentasikan alur informasi dari satu aktivitas ke aktivitas lain.

1. Alur kontrol dan alur *object*

Ada 2 tipe alur didalam *activity diagram*, yaitu alur kontrol dan alur *object*. Alur kontrol memodelkan jalur eksekusi dari sebuah bisnis proses. Alur kontrol dilambangkan dengan garis dengan kepala panah yang menunjukan arah dari alur tersebut. Alur kontrol hanya bisa menghubungkan aktivitas dengan aktivitas lainnya. Alur *object* memodelkan jalur *object* didalam sebuah bisnis proses. Dikarenakan aktivitas dan aksi memodifikasi atau mentrans*form*asi *object*, alur *object* diperlukan untuk menunjukkan *object* yang masuk dan keluar dari sebuah aksi atau aktivitas. Alur *object* digambarkan dengan garis putus-putus dengan kepala panah yang menunjukan arah dari alur *object* tersebut.

4. *Control nodes*

Ada 7 control nodes pada activity diagram yaitu:

* 1. *Initial node*

Menggambarkan awal dari sebuah rangkaian aktivitas

* 1. *Final activity node*

Digunakan untuk menghentikan semua alur kontrol dan alur *object* dalam rangkaian aktivitas.

* 1. *Final flow node*

Digunakan untuk menghentikan *control flow* atau *object* *flow*.

* 1. *Decision node*

Digunakan untuk merepresentasikan sebuah kondisi tes untuk memastikan bahwa *control flow* atau *object* flow hanya memiliki 1 *path*.

* 1. *Merge node*

Digunakan untuk menyatukan kembali path berbeda yang dihasilkan oleh *decision node*.

* 1. *Fork node*

Digunakan untuk memulai aktivitas yang berjalan paralel.

* 1. *Join node*

Digunakan untuk menggabungkan kembali aktivitas yang berjalan paralel.

1. *Swimlanes*

*Activity diagram* dapat memodelkan sebuah bisnis proses independen dari beberapa *object*. *Swimlane* digunakan untuk memisahkan aktivitas berdasarkan tanggung jawab dari *object* mana yang harus menjalankan aktivitas tersebut.

|  |  |
| --- | --- |
| Simbol | Arti |
|  | *action* |
|  | activity |
|  | *class* |
|  | control flow |
|  | *object* flow |
|  | initial node |
|  | *final* activity node |
|  | *final* flow node |
|  | Decision node |
|  | Fork node |
|  | Join node |
|  | *Swimlane* |

**Tabel 2.2 Simbol dan Arti dalam Pembuatan *Activity Diagram***

#### 2.6.4 Class Diagram

*Class* *diagram* adalah sebuah static model yang menunjukkan relasi antar *class*-*class* di dalam suatu sistem.

Elemen-elemen dari sebuah *class* diagram meliputi:

1. *Class*

Komponen utama suatu *class* diagram adalah *class*, *class* menyimpan informasi dari sebuah *system*. Sebuah *class* merupakan representasi dari orang, tempat, atau *object* apapun yang akan ditangkap *system*. Sebuah *class* digambar dengan sebuah persegi panjang yang terbagi menjadi 3 bagian, bagian atas merupakan nama dari *class* tersebut, bagian tengah merupakan atribut dari *class* tersebut, dan bagian bawah adalah operasi-operasi apa saja yang dapat dilakukan *class* tersebut.

1. *Relationship*

Tujuan utama dari suatu *class* diagram adalah untuk menunjukan hubungan, atau asosiasi yang dimiliki suatu *class* dengan *class* lainnya. Dalam *class* diagram, sebuah relasi digambarkan dengan sebuah garis diantara 2 *class*. Relasi tersebut diberikan sebuah label berupa nama dari relasi tersebut atau peran yang dimiliki oleh *class* tersebut dalam relasi itu. Label tersebut juga bisa diberikan tanda panah berbentuk segitiga agar memudahkan cara pembacaan relasi.

*Relationship* juga memiliki *multiplicity*, yang menyatakan bagaimana sebuah instance dari suatu *object* dapat diasosiasikan dengan *instance* lainya.

1. *Generalization and aggregation associations*

Sebuah generalisasi menunjukkan bahwa sebuah *class* (subclass) merupakan turunan dari *class* lain (superclass), yang berarti bahwa atribut-atribut dan operasi-operasi dari *superclass* juga valid bagi *object* dari *subclass*. Generalisasi digambarkan dengan sebuah garis dari *subclass* ke *superclass* dengan kepala panah yang menunjuk *superclass*.

Sebuah agregasi digunakan ketika *class* meliputi *class* lain, misalnya *class* *department* yang meliputi *class* *employee*, *class* kendaraan yang meliputi *class* roda, dan *class* kantor yang meliputi *class* meja.

#### 2.6.5 Entity Relationship Diagram

*Entity* *Relationship Diagram* merupakan suatu *diagram* yang digunakan untuk menggambarkan model suatu *database* dengan pendekatan *top-down* untuk mendesain suatu sistem *database* (Connolly & Begg, 2014). Dalam membuat suatu ERD, data paling penting harus ditentukan terlebih dahulu (*entities*) dan hubungan diantaranya (*relationship*s). *Entities* merupakan suatu kumpulan *object* yang memiliki ciri yang sama, yang biasanya menggambarkan *object* pada dunia nyata. Setelah itu, ditambahkan informasi yang lebih lengkap pada entitas tersebut, berupa atribut, dan *constraints* (batasan-batasan pada masing masing data).

Hubungan antar entitas disebut dengan hubungan asosiasi. Hubungan - hubungan ini biasa diberi tanda panah dan nama berupa kata kerja yang mendeskripsikan fungsionalitas dari hubungan tersebut.

Suatu entitas dapat memiliki suatu hubungan rekursif, dimana entitas tersebut dapat berpartisipasi dalam beberapa role yang berbeda. Misalnya suatu entitas staf, dapat berpartisipasi dalam 2 *role* yaitu sebagai staf, dan *supervisor*.

Kadangkala dalam suatu relasi terdapat atribut-atribut yang saling melekat di dalamnya. Atribut-atribut tersebut membentuk suatu entitas baru yang dapat digambarkan dengan menghubungkan garis putus-putus dari entitas tersebut menuju relasi yang menghasilkan entitas tersebut.

Pada suatu entitas, terdapat suatu batasan (*constraints*) yang berlaku, sesuai dengan batasan yang ada dalam dunia nyata. Salah satu batasan yang paling utama adalah *multiplicity*, yang merupakan jumlah partisipasi yang memungkinan dalam hubungan suatu entitas dengan entitas lainnya. *Multiplicity* pada ERD digambarkan menggunakan jangkauan domain nilai pada masing masing ujung suatu relasi.

Dalam merepresentasikan domain nilai *multiplicity* terdapat beberapa cara, misalnya:

|  |  |
| --- | --- |
| Batasan Multiplicity | Arti |
| 0..1 | 0 atau 1 presensi entitas |
| 1..1 (1) | Tepat 1 presensi entitas |
| 0..\* (\*) | 0 atau banyak presensi entitas |
| 1..\* | 1 atau banyak presensi entitas |
| 5..10 | Minimum 5 sampai maksimum 10 presensi entitas |
| 0,3,6-8 | 0 atau 3, 6, 7 atau 8 presensi entitas |

**Tabel 2.3 Tabel *Domain* Nilai *Multiplicity* pada ERD**

### 2.7 *Database*

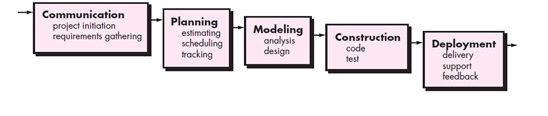
*Database* adalah suatu tempat yang didesainuntuk menjadi tempat penyimpanan data dalam jumlah yang banyak dan data - data tersebut dapat dimanipulasi sesuai dengan kebutuhan dari suatu organisasi. Dengan adanya *database*, data - data yang disimpan menjadi lebih terorganisir serta manajemen data menjadi lebih mudah.

Connolly & Begg (Connolly & Begg, 2015) menjelaskan bahwa *Database* adalah suatu kumpulan data yang saling terhubung atau berkaitan secara logis satu dengan yang lainnya yang bertujuan untuk memenuhi kebutuhan suatu organisasi. Dalam mengatur dan mengontrol data data tersebut, diperlukan suatu *software* *Database Management System* (DBMS). DBMS memungkinkan *use*r untuk mendefinisikan, membuat, memelihara, dan mengontrol akses pada suatu *database*. Dalam pengoperasiannya DBMS menyediakan fasilitas DDL (*Data Definition Language*) dan DML (*Data Manipulation Language*) yang membuat *user* bisa membuat dan memanipulasi *database* tersebut. Kedua fasilitas tersebut dijalankan menggunakan *Query Language.* SQL (*Structured Query Language*) adalah contoh *Query Language* yang paling banyak digunakan. Terdapat berbagai macam jenis *database*, namun *database* dapat dibagi menjadi dua kelompok besar yaitu SQL *database* (*relational database)* dan NoSQL *database* (*non-relational/distributed database*).

### 2.8 Waterfall Model

Menurut Pressman (2015), model *waterfall* atau yang biasa disebut dengan *classic life cycle*, menggunakan pendekatan-pendekatan sistematis dan sequential untuk proses *development* yang dimulai dari *communication* (*customer* *specification*), *planning*, *construction*, dan *deployment*.  Model *waterfall* biasa digunakan ketika *requirement* dari proses *development* telah didefinisikan dengan jelas pada awalnya.

Kesederhanaan metode pengembangan *waterfall* mengakibatkan diperlukannya fokus pada masing-masing tahap pengembangan dan tidak melanjutkan ke tahap berikutnya sama sekali. Metode ini sesuai untuk proyek yang tidak terdapat banyak revisi di setiap tahapan yang ada ataupun perubahan pada kebutuhan awal ataupun tujuan, serta untuk pengerjaan dalam waktu yang singkat. Tahap-tahap metode *waterfall*, antara lain:



**Gambar 2.2 Skema *Waterfall* SDLC.**

Tahapan SDLC *waterfall* digambarkan sebagai berikut:

1. *Communication* atau komunikasi, dimana terjadi pengumpulan data, analisis data hingga penyajian data secara *final*. Data yang dikumpulkan terdiri atas data primer dan data sekunder. Data primer merupakan data yang diperoleh langsung dari sumber dan dikumpulkan sambil memperhatikan etika-etika profesional yang ada, berupa observasi terhadap pihak dan lingkungan terkait, yaitu kartu *Supermarket* atau toko swalayan di Indonesia dimana letak geografisnya berada di sekitar penulis, termasuk wawancara pada pihak pengelola dan karyawan yang terlibat pada alur perbelanjaan, serta survei pada konsumen yang berbelanja.

2. *Planning* atau perencanaan, mencakup melakukan definisi terhadap sistem, metode teknis yang dilakukan, sumber daya yang diperlukan untuk berjalannya sistem, alokasi waktu hingga selesainya pengembangan, serta pemantauan berupa metode dokumentasi.

3. *Modelling* atau perancangan, dimana rancangan sistem baik secara alur, interaksi antar sistem, interaksi antar komponen dalam sistem, struktur *database*, dirancang sesuai dengan definisi terhadap sistem yang sebelumnya sudah terbentuk.

4. *Construction* atau konstruksi, tahap dimana pengembangan code dilakukan, mulai dari fitur-fitur kecil hingga menjadi sistem secara keseluruhan. Sembari pengembangan dilakukan, *testing* dilakukan pada setiap unit secara *Black Box Testing* ataupun pada setiap fitur untuk menemukan *error*.

5. *Deployment* atau penerapan, dimana akan dilakukan penggabungan modul-modul yang sudah dibuat dan dilakukan pengujian untuk mengetahui apakah sistem yang dibuat telah sesuai dengan rancangan atau adanya penemuan *defect*. Disertakan juga pemantauan terhadap sistem yang diimplementasi pada lingkungan tujuan yaitu *supermarket*, kemudian respon dari penerapan tersebut akan dibuat sebagai kebutuhan baru dalam kegiatan evaluasi terhadap produk yang sudah selesai.

### 2.9 Arduino

Arduino adalah sebuah *platform* elektronik *open-source* yang memiliki *hardware* dan *software* yang mudah untuk digunakan. Kita bisa memberikan perintah-perintah ke *hardware* arduino kita melalui inputan / instruksi yang kita masukkan kedalam mesin arduino kita.

Untuk membuat perintah / instruksi ke dalam mesin Arduino, diperlukan pembuatan *script* perintah dengan bahasa pemrograman Arduino yang bisa dibuat di IDE (*Integrated Development Environment*) milik arduino yaitu *the Arduino Software.* Dengan perintah yang sudah dibuat dan dimasukkan kedalam mesin dalam bentuk program, maka mesin Arduino akan bekerja sesuai dengan perintah yang sudah dibuat sebelumnya.

Mesin / *hardware* dari Arduino memiliki kemampuan untuk di kostumisasi. Dengan kemampuan ini, perintah yang bisa kita masukkan kedalam mesin Arduino akan semakin beragam. Selain itu, Arduino juga menawarkan beberapa keunggulan lain yaitu terjangkau, bisa terhubung dengan berbagai *platform*, *hardware* dan *software* yang bebas untuk di kostumisasi, selain itu, bahasa pemrogramannya mudah dipahami bahkan oleh seorang pemula sekalipun.

Arduino memiliki berbagai jenis mesin yang diproduksi. Mesin-mesin yang ada memiliki perbedaan pada jumlah input dan output yang dimiliki. Selain itu, ada beberapa mesin yang sudah memiliki modul tertentu yang sudah terpasang di dalamnya. Misalnya Wemos D1 Mini yang sudah memiliki modul / perangkat kecil yang bertujuan untuk membuat hubungan dengan jaringan nirkabel (*wireless*).

#### 2.9.1 Arduino Uno

Arduino Uno adalah salah satu tipe mikrokontroler dari arduino. Ariduino Uno menggunakan ATmega328P sebagai mikrokontrollernya. Arduino Uno memiliki 14 digitan I/O pin, 6 pin analog, 16 MHz quartz *crystal*, USB *connection*, *power* *jack*, ICSP *header* dan *reset* *button*. Untuk menggunakan USB *cable* yang bisa dikoneksikan ke laptop atau ke baterai sebagai sumber tenaga.

### 2.10 Android

Android adalah sebuah *platform* *mobile* *open* *source* yang dimiliki oleh *Open* *Handset* *Alliance*, yang digunakan untuk perangkat *smart*phone. Pada hakikatnya, Android merupakan Linux yang dimodifikasi menggunakan *integrated* *programming* *interface* menggunakan bahasa pemrograman Java.

### 2.11 Visual Studio

Visual Studio merupakan sebuah *integrated* *development environment* (IDE) yang dimiliki oleh Microsoft dan digunakan untuk membangun program mulai dari skala kecil hingga besar.

#### 2.11.1 Visual Studio Code

Visual Studio Code merupakan sebuah *source* *code* *editor* *open* *source* yang ringan namun sangat *powerful* yang bisa digunakan dalam berbagai platform *desktop* misalnya Windows, macOS and Linux. Visual Studio Code hadir dengan *built-in support* JavaScript, TypeScript, dan Node.js serta memiliki kemampuan untuk diintegrasikan dengan *extention* bahsa pemrograman lain.

### 2.12 Flutter

Flutter adalah sebuah *open* *source* *software* *development* kit (SDK) yang terfokus pada pembuatan komponen *use*r *interface* (UI) secara lintas *platform*. Flutter menggunakan bahasa C, C++ dan Dart dalam pengembangannya.

### 2.13 PostgreSQL

PostgreSQL merupakan suatu *open source relational database management system.* Selain itu, PostgreSQL juga berperan sebagai *database.* Dengan banyak bahasa pemrograman yang bisa terhubung dengan PostgreSQL, maka penulisan *code* untuk memanipulasi data dalam *database* menjadi lebih mudah.

Dalam mengatur dan mengelola PostgreSQL, ada berbagai *tools* yang bisa digunakan, antara lain:

1. **psql**

psql merupakan *command line interface* yang dapat digunakan untuk menuliskan *query* dan mengoperasikan PostgreSQL. *Tools* ini merupakan *tools* yang muncul paling awal, dan dilengkapi dengan berbagai *command.*

1. **pgAdmin**

pgAdmin adalah *tools* yang digunakan untuk melakukan berbagai operasi dalam PostgreSQL dalam bentuk aplikasi grafis pada *desktop.* Dengan adanya *user interface*, pgAdmin sangat cocok digunakan untuk orang yang tidak familiar atau kurang familiar dengan PostgreSQL.

### 2.14 REST

*Representational State Transfer* (REST), menurut Fielding (2000) seperti yang dikutip oleh Masse (2011, h. 5) pada disertasinya, merupakan suatu abstraksi dari elemen - elemen arsitektur dalam suatu *distributed* *hypermedia* *system*. Disimpulkan bahwa REST adalah sekumpulan batasan batasan arsitektur yang bertujuan untuk meminimalisir latensi dan komunikasi antar jaringan sekaligus memaksimalkan independensi dan skalabilitas.

Dalam konsep REST, suatu *server* akan menyediakan akses berbentuk URI (*Uniform Resource Identifier*) terhadap suatu *resource* pada *client* sehingga *client* dapat mengakses dan memodifikasi *resource* tersebut. REST *server* juga tidak menyimpan *state* dari *client session* (*stateless*) sehingga apabila *client* membutuhkan informasi *state* aplikasi, *client* harus mengirimkan data *state* tersebut ke *server*.

#### 2.14.1 HTTP Method

Protokol yang digunakan dalam REST dalam berkomunikasi adalah protokol HTTP. Fungsi HTTP yang digunakan antara lain:

1. GET merupakan fungsi yang menyediakan akses pada suatu data namun tidak diperbolehkan melakukan modifikasi.
2. POST merupakan fungsi yang bertujuan untuk membuat data baru.
3. DELETE merupakan fungsi yang digunakan untuk menghapus suatu data.
4. PUT merupakan fungsi yang digunakan untuk memperbarui suatu data.

### 2.15 Node.js

Node.js adalah suatu *platform* *server side* yang ditujukan untuk membuat suatu *scalable* *network application* secara *open source. Framework* adalah suatu kumpulan kerangka kerja yang digunakan untuk mempermudah proses pembuatan aplikasi. Dalam pemanfaatannya pada Node.js, terdapat berbagai macam *framework* yang bisa digunakan misalnya express.js, dan pg.

### 2.16 User Interface

*Use*r yang dapat berupa manusia ataupun proses lainnya dapat berinteraksi dengan suatu sistem, terutama memberikan input yang kemudian akan diproses dengan logika sistemnya, dengan menggunakan *use*r *interface*. *Use*r *Interface* mementingkan nilai *usability*, yaitu sebuah penilaian yang bersifat kualitatif terhadap kemudahan dan efisiensi *use*r dalam menggunakan fungsi utama dari suatu sistem, karena sifatnya yang berinteraksi secara langsung dengan *use*r. Dalam mencapai nilai-nilai *usability* yang maksimal, ilmu-ilmu seperti psikologi dan ergonomi digabungkan dalam *use*r *interface*.

### 2.17 Eight Golden Rules

*Eight* *Golden* *Rules* merupakan pedoman dalam membuat sebuah *use*r *interface* yang nyaman digunakan oleh *use*r yang mencakup 8 poin penting antara lain:

* 1. *Strive for Consistency*

Konsistensi dibutuhkan antar halaman dalam satu aplikasi ataupun antara aplikasi yang masih berhubungan. Dengan demikian, akan membuat *use*r nyaman dalam mengeksplorasi aplikasi tanpa takut berpindah aplikasi.

2. *Cater to Universal Usability*

Sebuah *use*r *interface*, haruslah bisa digunakan oleh berbagai macam *use*r dengan latar belakang yang berbeda, tingkat pemahaman yang berbeda, atau perbedaan lainnya. *Use*r baru maupun *use*r lama tentu juga ada perbedaan, maka aplikasi harus bisa menyediakan tampilan yang bisa dimengerti keduanya.

3. *Offer Informative Feedback*

*Feedback* tidak harus selalu dengan jawaban dari aplikasi ke *use*r, tetapi dapat berupa perubahan antar muka setiap *use*r melakukan aksi, dengan demikian *use*r paham bahwa aksinya sudah direspon oleh aplikasi.

4. *Design Dialog to Yield Closure*

Desain suatu layar harus jelas informasinya. Apakah suatu proses sudah selesai dilakukan, atau sedang dilakukan sehingga *use*r tahu apakah dia harus menunggu atau tidak.

5. *Prevent Error*

Suatu aplikasi yang baik, harus bisa mengarahkan *use*r untuk suatu event tertentu sehingga tidak terjadi kesalahan yang dilakukan *use*r akibat ketidaktahuan *use*r. Misalnya dengan memberikan contoh *form*at isian dalam sebuah *form* dalam aplikasi.

6. *Permit Easy Reversal of Action*

Aplikasi sebaiknya menyediakan metode untuk membatalkan aksi yang telah dilakukan *use*r sehingga *use*r dapat melakukan eksplorasi terhadap aplikasi tanpa rasa khawatir.

7. *Support Internal Locus of Control*

Aplikasi sebaiknya memiliki tampilan yang bisa diatur oleh *use*r sendiri sesuai preferensi dari *use*r. Hal ini dapat meningkatkan kepuasan *use*r terhadap aplikasi yang sangat mempengaruhi UX terhadap aplikasi tersebut.

8. *Reduce Short-term Memory Load*

Dalam merancang suatu *use*r *interface*, perlu diperhatikan juga kemudahan penggunaannya, bukan hanya tata letak semua komponen yang ada, namun bisa juga mengurangi *effort* yang dilakukan *use*r dalam memberikan inputan ke dalam aplikasi.

### 2.18 Lima Faktor Terukur Manusia

Sesuai dengan Shneiderman dan Plaisant (2013, h.32-33), lima faktor terukur manusia mengatur aspek-aspek yang perlu dievaluasi dalam *User* *Interface* yang meliputi:

* 1. Waktu untuk belajar

Ukuran berdasarkan jangka waktu yang dibutuhkan pengguna untuk mempelajari cara kerja dari suatu sistem secara relevan dan seefektif ataupun senyamannya.

* 1. Kecepatan performa

Waktu yang dibutuhkan aplikasi untuk melakukan fungsinya

* 1. Tingkat *error* yang dilakukan pengguna

Jumlah *error* dan jenis *error* yang mungkin ditemukan pengguna saat menggunakan aplikasi.

* 1. Kepuasan subjektif

Tingkat kepuasan pengguna terhadap beberapa aspek tertentu.

* 1. Daya ingat pengguna

Ukuran seberapa lama pengguna dapat mempertahankan ingatan atau pengetahuannya terhadap aplikasi selama rentang waktu tertentu.

### 2.19 Material Design

Material *design* adalah sebuah prinsip desain *Mobile* *App* yang dikembangkan oleh Google pada tahun 2014 dengan tujuan membuat bahasa visual yang menyampaikan maksud secara seragam antar *platform*, *device* dan input sementara masih mengakomodasi kreativitas inovasi dan teknologi yang ada. Material *design* terfokus pada penggunaan *“cards”* sebagai komponen utama, didukung dengan adanya *grid*, animasi, transisi, *padding* dan *depth*.

Beberapa prinsip material *design* sesuai dengan dokumentasi resmi yang dibuat oleh Google (diakses 18 Januari 2020) adalah:

1. Material sebagai metaphor, yaitu benda yang ditampilkan mengikuti sifat benda aslinya.
2. Tegas dan grafis, yaitu menekankan tipografi yang tepat untuk menyampaikan suatu makna.
3. Gerakan yang bermakna, dimana animasi digunakan untuk menyatakan suatu umpan balik tanpa kata-kata.
4. Wadah yang fleksibel, menyatakan bahwa wadah dari sebuah *container* dalam material *design* bersifat dapat menyesuaikan isi yang diinginkan *use*r.
5. *Cross-platform,* dimana prinsipnya dapat diterapkan dan tidak memiliki hasil yang berbeda saat diterapkan di perangkat-perangkat yang berbeda.

#### 2.19.1 Flutter Material Design

Flutter Material Design diperkenalkan sebagai prinsip utama yang diterapkan pada *widget*-*widget* di dalam *class* yang sudah didefinisikan oleh Flutter *framework* sendiri sehingga memastikan setiap *widget* dalam flutter menganut Material *Design*. Lingkup dari *widget*-*widget* mencakup struktur aplikasi dan navigasinya, tombol, input, dialog, pengingat, panel, tampilan informasi dan layout dalam Flutter sendiri.

### 2.20 Software Testing

*Software* *Testing* merupakan sebuah proses menjalankan sistem dengan maksud untuk mencari kesalahan (Myers, 2011, dalam Pressman, 2015). *Software* *Testing* memanfaatkan test *case* yang berisi alur kerja pada setiap fungsi yang akan diuji. Test *case* dapat dilakukan secara manual dengan manusia dan menjalankan fungsi-fungsi yang ada di dalam sistem, ataupun dengan bantuan komputer (CAST, *Computer-Aided Software Testing*). *Software* *testing* dikatakan berhasil jika dapat menemukan kesalahan yang sebelumnya tidak ditemukan.

#### 2.20.1 Black Box Testing

*Black Box* *Testing*, disebut juga *Specification Testing* (Pressman, 2015), merupakan sebuah *testing* yang dilakukan terhadap setiap modul yang ada di dalam suatu sistem. *Black Box* sendiri dapat diartikan sebagai satu bagian dari sistem yang spesifik yang merespon kepada suatu stimuli dan menghasilkan sebuah respon/*output*, dimana prosesnya dapat dilihat secara transparan oleh *use*r.

#### 2.20.2 Integration Testing

*Integration* *Testing* adalah sebuah teknik sistematik untuk mengkonstruksi sebuah arsitektur *software* dimana pada saat yang sama melakukan test terhadap *error*-*error* yang belum ditemukan terasosiasi dengan *interface*. Objektifnya adalah untuk mengambil *component* yang telah dites dan membangun sebuah struktur program yang telah disesuaikan oleh *design*.

#### 2.20.3 System Testing

*System* *testing* merupakan sebuah jenis *test* yang menggabungkan *software* dengan elemen-elemen lainnya seperti *hardware*, manusia dan informasi serta variabel bebas lainnya. Kesalahan dapat sangat memungkinkan ketika dalam pengembangan *software* tidak ada tahapan integrasi dengan sistem secara keseluruhan. Sementara *software* *testing* melibatkan pengembang dari aplikasi, *system* *testing* melibatkan orang lain yang berkaitan dengan tempat dimana *software* diimplementasikan, seperti contohnya, penggunanya secara langsung.

### 2.21 Software Evaluation

*Software Evaluation* merupakan tahap yang mengikuti *Deployment* dalam *Waterfall Model* ataupun tahapan terakhir dari SDLC lainnya. Menurut Pressman (2015, h.342-343), evaluasi dilakukan terhadap setiap tahap yang terjadi pada SDLC, dan menghasilkan umpan balik yang akan digunakan dalam produk selanjutnya ataupun *increment* jika model dilanjutkan dalam bentuk *Incremental Waterfall*. Evaluasi dilakukan dengan cara membandingkan ekspektasi ataupun rumusan masalah dengan output yang ada, walaupun cara-caranya sebenarnya bervariasi dari sekedar *“test drive”* hingga serangkaian studi banding yang menggunakan metode-metode statistik ataupun umpan balik dari sekumpulan *use*r.

#### 2.21.1 Mobile App Evaluation

*Mobile* *App* dievaluasi dengan cara menilai *usability* dan *accessibility*, oleh karena itu disebut juga dengan *Use*r *Evaluation* karena hubungannya erat dengan nilai kegunaannya pada *user*. Hasil penilaian terhadap kedua nilai tersebut akan digunakan untuk merancang desain pada tahap selanjutnya. Aspek-aspek yang dievaluasi (Andreou, 2011, seperti yang dikutip oleh Pressman, 2015 h. 396) antara lain:

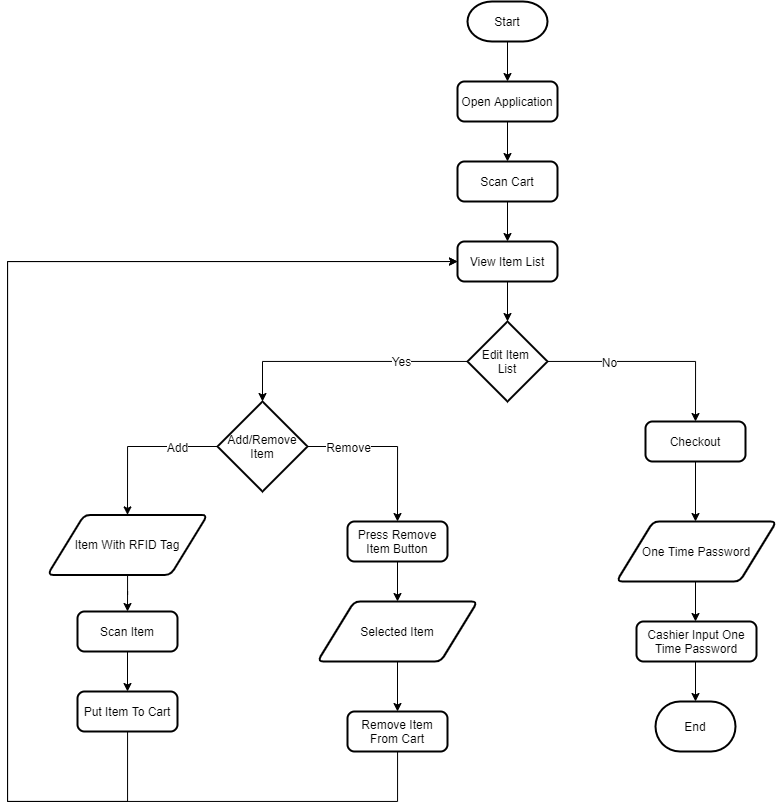
1. *Ubiquity* (keumuman); setiap *use*r tidak merasa asing dengan sistem walaupun menawarkan sebuah solusi yang baru dan tidak memerlukan pengalaman yang benar-benar baru untuk dapat mempergunakan sistem sebagaimana semestinya.
2. *Personalization* (personalisasi); setiap *use*r dapat memperoleh hasil yang dapat ia sesuaikan dengan kondisinya sendiri ataupun memberikan rasa kepemilikan terhadap sistem.
3. *Flexibility* (fleksibilitas); setiap *use*r dapat menggunakan sistem sebagaimana seharusnya tanpa terbatas *platform*, OS, dan kendala-kendala perbedaan *mobile* *device* lainnya.
4. *Localization* (lokalisasi); setiap *use*r dapat melakukan proses yang diinginkan secara lokal tanpa memiliki ketergantungan pada kontrol dari pihak lain.

# BAB 3

## METODE PENELITIAN

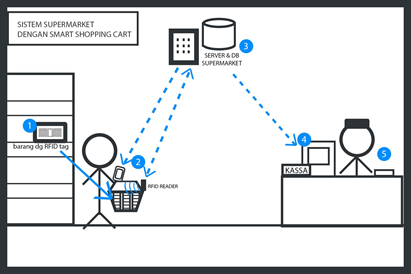
### 3.1 Kerangka Berpikir

Berikut ilustrasi alur sistem *Smart Shopping Cart* yang dikembangkan:



**Gambar 3.1 *Flowchart* *Smart* *Shopping* *Cart* *System*.**

Rancangan sistem akan dibuat untuk memenuhi skema transmisi data pada alur belanja seperti berikut:



**Gambar 3.2 Skema Belanja Menggunakan *Smart* *Shopping* *Cart*.**

Keterangan:

1. Barang di *supermarket* diberikan label berupa RFID *tag*, kemudian dimasukkan ke dalam *smart* *shopping* *cart* oleh pembeli.
2. Setelah *smart* *shopping* *cart* diidentifikasikan oleh *server* melalui pemindaian QR oleh pelanggan, RFID *reader* pada *smart shopping cart* memindai RFID *tag* yang ada pada barang-barang, kemudian mengirim informasi barang yang ada di keranjang pada *server*. Di saat yang bersamaan pembeli dapat menghubungkan aplikasinya dengan *server* untuk mengambil informasi daftar belanja miliknya.
3. *Database* menampung informasi setiap *smart* *shopping* *cart* serta transaksi di dalamnya, sementara *server* melakukan perhitungan transaksi dan menyediakan informasi bagi pembeli.
4. Mesin pada kassa menerima informasi terkait *smart* *shopping* *cart* termasuk total belanjanya dan dapat mengidentifikasi setiap RFID reader pada *smart* *shopping* *cart* pada proses *checkout*.
5. Pembeli melanjutkan proses pembayaran dengan kasir dengan metode pembayaran yang diinginkan.

### 3.2 Analisis Kebutuhan

#### 3.2.1 Metode Pengumpulan dan Analisis Data

Pengumpulan data primer dilakukan dengan metode wawancara, observasi, survei, sementara data sekunder dikumpulkan dengan cara meneliti riset-riset terdahulu serta dokumentasi resmi dari teknologi ataupun pihak yang terlibat.

Data primer yang akan dicari antara lain seputar informasi mengenai kebutuhan pengelola toko dan *feasibility* penerapan teknologi pada *Supermarket*, hambatan yang mungkin terjadi, alur perbelanjaan, permasalahan aktual serta kebiasaan-kebiasaan konsumen yang terjadi di lapangan. Sedangkan data sekunder yang kami kumpulkan antara lain informasi mengenai teknologi RFID, Arduino, penerapan *Internet of Things* secara umum, sistem serupa, serta permasalahan-permasalahan *Supermarket* yang pernah digarap sebelumnya dan solusi sebagai penyelesaiannya.

Analisis data dilakukan secara bertahap dimulai dari pembersihan data dan mengisi data yang sebelumnya nampak tidak diperlukan namun diperlukan, memroses data kuantitatif dengan cara menentukan skala pengukuran setiap data, verifikasi data, membuat kategori setiap data, juga memroses data kualitatif dengan cara membuat tema dari setiap data, mengelompokkannya, kemudian menyajikannya dalam bentuk laporan ataupun statistik.

#### 3.2.2 Hasil Observasi

Menurut observasi yang dilakukan pada Ranch Market, Grand Indonesia, Jakarta Pusat. Ada pula observasi dilakukan pada Foodhall, Grand Indonesia, Jakarta Pusat, dengan hasil sebagai berikut:

##### 3.2.2.1 Observasi di Ranch Market Grand Indonesia

Observasi dilakukan di Ranch Market, Grand Indonesia, Jakarta Pusat, pada beberapa waktu sepanjang bulan November 2019 hingga Januari 2020.

Hasil observasi menunjukkan bahwa jumlah antrean di kassa selalu berkisar antara 3-10 orang, dimana waktu antrean untuk mencapai kassa dan melakukan pembayaran paling cepat 10 menit dan paling lama 45 menit. Hal ini dikarenakan mesin *scan barcode* yang butuh waktu 3-10 detik, kecepatan dari kasir sendiri, jumlah barang yang ada di dalam keranjang belanja setiap pelanggan, dan waktu yang digunakan untuk menyelesaikan pembayaran sendiri. Berdasarkan pengamatan, pembayaran biasanya dilakukan dengan metode *cashless*, baik secara kartu debit, kartu kredit ataupun *e-wallet*. Selama observasi dilakukan, 9 dari 10 pelanggan menggunakan keranjang belanja dibandingkan troli. Jumlah keranjang yang dibawa pelanggan umumnya tidak melebihi 1 keranjang. Jumlah pelanggan yang berbelanja di satu waktu yang bersamaan mencapai rata-rata 50 orang. Di Ranch Market, semua barang belanja memiliki *barcode* (tidak ada barang yang dituliskan harganya secara manual).

Menurut hasil observasi di atas, maka implementasi pemindaian secara mandiri sebelum sampai di kassa dapat mengurangi antrean karena pekerjaan pemindaian *barcode* dapat dilakukan sendiri. Dengan adanya barcode untuk semua barang, implementasi RFID tag pada setiap barang dianggap memungkinkan. Banyaknya pelanggan yang membayar secara *cashless* menunjukkan bahwa pelanggan di *supermarket* kebanyakan sudah memiliki telepon genggam untuk mengunduh aplikasi guna mengoperasikan sistem belanja menggunakan *Smart Shopping Cart*.

##### 3.2.2.2 Observasi di Foodhall Grand Indonesia

Ada pula observasi dilakukan di Foodhall, Grand Indonesia, Jakarta Pusat selama bulan Desember 2019 hingga Januari 2020.

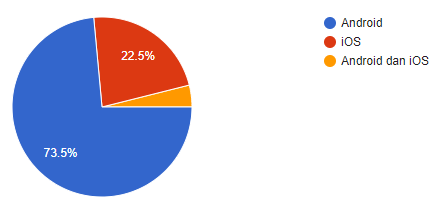
Hasil observasi menunjukkan bahwa pelanggan di *supermarket* umumnya menggunakan keranjang belanja dibandingkan dengan troli. Metode pembayaran yang digunakan umumnya berupa kartu debit, kartu kredit dan *cashless*. Pintu masuk dan pintu keluar *supermarket* terbuka, sehingga mengandalkan integritas dan tenaga kerja security untuk menjaga keranjang yang keluar dan masuk. Waktu yang dibutuhkan untuk antre sebelum sampai di kassa untuk melanjutkan ke pembayaran sekitar 5-15 menit, sedangkan volume pelanggan yang ada di toko di satu waktu sekitar 30 hingga 50 orang.

Menurut hasil observasi, kendala implementasi sistem *Smart Shopping Cart* terletak di pintu masuk *supermarket* yang tidak terlalu terjaga dan relatif terbuka. Hal ini akan menjadi pertimbangan spesifik bagi toko. Adapun produk-produk berupa masakan yang dibuat langsung di tempat yang tidak memiliki barcode, sehingga harga harus diinput secara manual.

#### 3.2.3 Hasil Survei

Survei dilakukan dengan menyebarkan Google *Form*s kepada teman-teman, keluarga dan teman kantor. Total responden mencapai 104 orang. Berikut pertanyaan serta grafik hasil dari kuesioner tersebut:

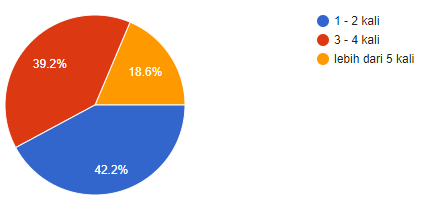
##### Apa jenis *platform* dari *smartphone* yang anda gunakan?



Gambar 3.2 Diagram Survei *Platform* *Smartphone* yang Digunakan

Berdasarkan gambar 3.2, didapati bahwa jenis *platform* paling banyak yang digunakan oleh responden adalah ponsel dengan *platform* Android dengan presentasi sebanyak 72,8%. Oleh karena itu, fokus utama pengimplementasian project *Smart* *Shopping* *Cart* akan dirancang untuk perangkat berbasis Android.

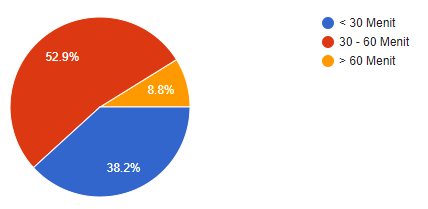
##### Seberapa sering anda berbelanja di *supermarket* dalam satu bulan?



Gambar 3.3 Diagram Survei Frekuensi Belanja di *Supermarket* dalam Satu Bulan

Dari 104 orang responden, 41.7% diantaranya menyatakan bahwa mereka melakukan kegiatan berbelanja di *supermarket* sebanyak 1 sampai 2 kali dalam sebulan. Selain itu, 39.8% diantaranya menyatakan bahwa mereka melakukan kegiatan berbelanja di *supermarket* sebanyak 3 sampai 4 kali dalam sebulan. Dari data tersebut didapati tidak banyak selisih antara 2 kelompok responden tertinggi yang mengindikasikan bahwa kegiatan berbelanja cukup sering dilakukan oleh responden.

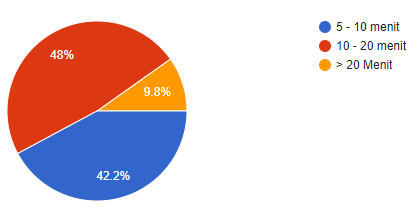
##### Berapa lama waktu yang anda habiskan dalam sekali berbelanja?



Gambar 3.4 Diagram Survei Waktu yang Dihabiskan dalam Sekali Berbelanja

Dari gambar 3.4 didapati bahwa lebih dari 50% responden menghabiskan waktu 30 sampai 60 menit dalam satu kali berbelanja. Jumlah waktu tersebut bukanlah durasi yang singkat. *Smart* *Shopping* *System* tercipta didasari keinginan untuk mengurangi durasi berbelanja sehingga waktu bisa dimanfaatkan dengan lebih bijak dan efisien.

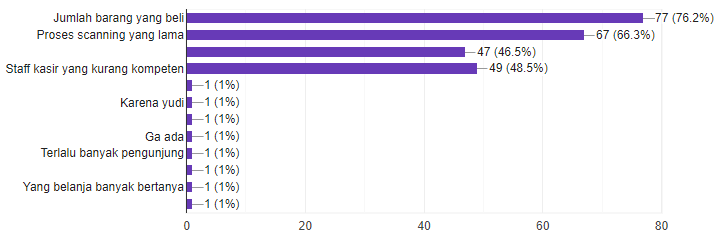
##### Berapa lama waktu yang anda gunakan untuk mengantre dalam berbelanja?



Gambar 3.5 Diagram Survei Waktu yang Digunakan Untuk Mengantre dalam Berbelanja

Berdasarkan hasil survei yang ditunjukkan dalam gambar 3.5, 47,6% responden setuju bahwa mereka menghabiskan 10 sampai 20 menit dari waktu belanja mereka hanya untuk mengantre. Antrean ini yang menjadi salah satu faktor yang membuat durasi berbelanja menjadi lama. Dengan berkurangnya antrean, maka akan berkurang juga durasi yang diperlukan untuk berbelanja yang berakibat meningkatnya kepuasan pelanggan.

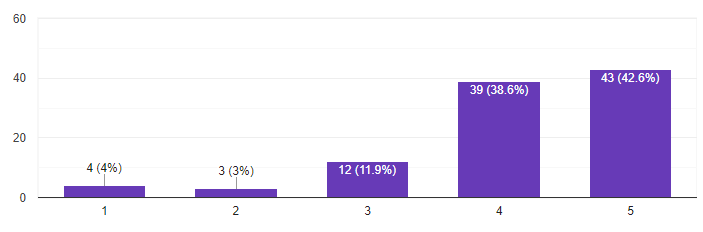
##### Menurut anda apa yang membuat proses pembayaran menjadi lama sehingga menciptakan antrean?



Gambar 3.6 Diagram Survei Hal-Hal yang Menciptakan Antrean

Dari gambar 3.6, dapat disimpulkan bahwa proses *scan* atau input barang di kasir merupakan suatu faktor besar yang memungkinkan antrean untuk terjadi. Dengan dasar tersebut, *Smart* *Shopping* *System* menawarkan suatu solusi untuk menghindari terjadinya antrean dengan memindahkan proses *scan* atau input barang. Proses *scan* atau input barang dilakukan pada saat *customer* sedang berbelanja. Hal ini diharapkan dapat mengurangi penumpukan pada kasa saat proses pembayaran.

##### Apakan antrean yang terjadi ketika proses pembayaran menjadi suatu kendala/kesulitan untuk anda?



Gambar 3.7 Diagram Survei Skala Antrean Sebagai Suatu Kesulitan dalam Proses Pembelajaran

Dari gambar 3.7, 42,2% responden setuju bahwa antrean merupakan suatu kendala pada proses perbelanjaan mereka. Antrean yang menjadi suatu permasalahan ini diharapkan dapat berkurang atau bahkan dihilangkan dengan hadirnya *Smart* *Shopping* *System* ini.

##### 3.2.3.7 Jika ada suatu sistem *smart* *shopping* dimana proses *scan* produk dan perhitungan dilakukan saat anda sedang berbelanja (memasukkan barang kedalam Keranjang) sehingga saat dikasir hanya perlu melakukan pembayaran, apakah anda akan menggunakannya?

Gambar 3.8 Diagram Survei Penggunaan *System* *Smart* *Shopping* Jika Tersedia

Berdasarkan gambar 3.9, mayoritas responden setuju untuk menggunakan aplikasi *Smart* *Shopping* *System*. Dengan respon yang didapatkan, merupakan suatu pertanda baik bahwa sistem ini cukup dinantikan untuk dicoba dan digunakan oleh responden.

##### 3.2.3.8 Apa harapan / fitur yang anda harapkan ada dalam suatu sistem *smart* *shopping*?

Dari hasil analisis yang sudah dilakukan terhadap respon yang diberikan responden, didapati bahwa para responden berharap agar sistem *smart* *shopping* ini bisa terintegrasi juga dengan pembayaran. Selain itu, efisiensi, dan kemudahan pemakaian aplikasi juga menjadi fokus utama yang diharapkan oleh responden. Dengan faktor-faktor tersebut *development* akan dilakukan dengan memperhatikan kemudahan pemakaian aplikasi dan efisiensi aplikasi.

### 3.3 Rancangan Sistem

#### 3.3.1 Rancangan UML

##### 3.3.1.1 Use Casehttps://lh6.googleusercontent.com/A0O56zhmtDCg525vmYnwOc6J8cMcKv-sAKzllpQuLK8uiQGARkAhKZ-e0XGXLPtNifE5ASrqIEiWszbRu45NDpXRLmvX-BRd94JRPRLYaIFxIa3uqXFrpN2Zs1QIrVMv0DrwJ2eg

Gambar 3.9 *Use* *case* diagram *smart* *shopping* *system*

##### 3.3.1.2 Use Case Description

###### 3.3.1.2.1 Scan Cart

*Use Case Name* : *Scan Cart*

*ID* : 1

*Importance Level* : *High*

*Primary Actor* : *Customer*

*Use Case Type* : *Detail*

*Stakeholder and interest* :

* + - * 1. *Customer*

Ingin mulai berbelanja menggunakan smart shopping cart

* + - * 1. *Supermarket*

Memerlukan smart shopping cart untuk mempercepat proses checkout.

*Brief Description* : *Use case* ini mendeskripsikan bagaimana *customer* *supermarket* memasang identitasnya pada keranjang yang dipilih.

*Trigger* : Pelanggan *supermarket* ingin mulai berbelanja

*Trigger Type* : Eksternal

*Relationships* :

* 1. *Association* : *Customer*
  2. *Include* :
  3. *Extend* : *Add Item to The List, View Item List*
  4. *Generalization* :

*Normal Flow of Events*:

* + 1. Pelanggan mengambil kerangjang yang telah terimplementasi smart shopping cart system.
    2. Pelanggan melakukan scanning barcode dari keranjang yang telah dipilih menggunakan smart shopping assistant app.
    3. Jika *customer* baru tiba di *supermarket* dan ingin mulai berbelanja: lakukan S-1.
    4. Jika *customer* mengganti keranjang yang digunakan dengan keranjang lain: lakukan S-2.

*SubFlows*:

* + - 1. S-1: Keranjang baru
         1. Pelanggan mulai berbelanja dan melakukan scanning keranjang.
      2. S-2 Mengganti keranjang

1. Pelanggan memilih menu ‘Change Cart’ dan melakukan scanning keranjang baru.
2. Pelanggan meletakkan keranjang lama.

*Alternate/Exceptional Flows* :

* + - 1. S1—1a: Keranjang memiliki transaksi yang menggantung, oleh karena itu server akan melakukan refresh terhadap transaksi untuk keranjang tersebut dan mengganti dengan transaksi yang baru.

###### 3.3.1.2.2 *Add Item to The List*

*Use Case Name* : *Add Item to The List*

*ID* : 2

*Importance Level* : *High*

*Primary Actor* : *Customer*, *Arduino System*

*Use Case Type* : *Detail*, Esensial

*Stakeholder and Interest* :

1. Pelanggan – ingin menambahkan barang belanja ke dalam daftar belanja.
2. *Arduino System* – melakukan tanggung jawab pengiriman informasi dari RFID *Tag* ke *server*.
3. *Supermarket* – ingin memastikan keakuratan barang dalam daftar belanja.

*Brief Description* : *Use case* ini menjelaskan bagaimana *customer* menambahkan barang ke dalam daftar belanjanya.

*Trigger* : Pelanggan *supermarket* ingin menambahkan barang ke dalam daftar belanjanya.

*Trigger Type* : Eksternal

*Relationships* :

* 1. *Association* : *Arduino System*
  2. *Include* :
  3. *Extend* :
  4. *Generalization* :

*Normal Flow of Events*:

1. Pindai RFID *Sticker* pada produk menggunakan *Arduino* yang sudah terpasang pada *Smart Shopping Cart*.
2. *Arduino System* mengirimkan informasi barang ke *server*.
3. Aplikasi menerima informasi terkait daftar belanja yang sudah diperbaharui dan melakukan *refresh* terhadap daftar belanja pengguna dengan barang yang baru ditambahkan serta total belanja yang baru.
4. *Customer* menambahkan produk ke dalam keranjang belanjanya.

*Subflows*: -

*Alternate/Exceptional Flows*:

* + - 1. 1a. RFID *Sticker* mengalami kerusakan sehingga barang dibawa kepada *supermarket* dan diberikan RFID *sticker* yang baru.

###### 3.3.1.2.3 *View Item List*

*Use Case Name* : *View Item List*

*ID* : 3

*Importance Level* : *High*

*Primary Actor* : *Customer*, *App System*

*Use Case Type* : *Detail*

*Stakeholder and Interest* :

1. *Customer* – ingin melihat daftar barang yang ada di dalam keranjangnya serta total belanja.
2. *Arduino System* – ingin menampilkan barang yang ada di dalam transaksi sesuai yang diterima dari *server*.
3. *Supermarket* – ingin membantu *customer* memantau daftar belanjanya kapanpun.

*Brief Description* : *Use case* ini menjelaskan bagaimana *customer* melihat daftar barang yang ada di keranjang belanjanya.

*Trigger*  : Pelanggan *supermarket* membuka aplikasi Smart *Shopping Assistant* setelah mendaftarkan keranjangnya.

*Trigger Type* : Eksternal

*Relationships* :

* + 1. *Association* : *App System*
    2. *Include* :
    3. *Extend* :
    4. *Generalization* :

*Normal Flow of Events* :

1. *Customer* membuka aplikasi *Smart Shopping Assistant*
2. Aplikasi meminta *detail* transaksi yang ada pada keranjang belanja kepada *server*
3. Aplikasi menampilkan daftar belanja serta jumlah total belanja
   1. Jika *customer* belum memindai suatu barang: lakukan S-1.
   2. Jika *customer* sudah memindai suatu barang: lakukan S-2.
4. Aplikasi menunggu *update* selanjutnya dari *server*

*Subflows* :

S-1: Belum Memindai Barang Belanja

1. Aplikasi menunjukkan bahwa keranjang belanja masih kosong
2. Total belanja menunjukkan angka 0

S-2: Sudah Pernah Memindai Barang Belanja

1. Aplikasi menunjukkan daftar belanja yang sudah dipindai
2. Aplikasi menunjukkan jumlah belanja sesuai dengan barang yang sudah dipindai
3. Jika *customer* menekan tombol “*Edit List*”, pergi ke *Use Case* 4

*Alternate/Exceptional Flows*:

* + - 1. 1a. Pelanggan menekan tombol “*Change Cart*” dan pergi ke *Use Case* 1, S-2.

###### *Remove Item from The List*

*Use Case Name* : *Remove Item from The List*

*ID* : 4

*Importance Level* : *High*

*Primary Actor* : *Customer, App System*

*Use Case Type* : *Detail*

*Stakeholder and Interest* :

1. Pelanggan – ingin menghapus barang belanja yang tidak ada di keranjang belanja lagi.
2. App System – ingin memastikan keakuratan barang di dalam keranjang belanja dengan barang di dalam daftar belanja.
3. *Supermarket* – membutuhkan mekanisme untuk menjamin keakuratan barang dalam daftar belanja.

*Brief Description* : *Use case* ini menjelaskan bagaimana *customer* menghapus barang dari daftar belanjanya.

*Trigger* : *Customer* *supermarket* ingin menghapus barang dari daftar belanjanya.

*Trigger Type* : Eksternal

*Relationships* :

* 1. *Association* : *App System*
  2. *Include* :
  3. *Extend* :
  4. *Generalization* :

*Normal Flow of Events*:

1. *Customer* mengeluarkan barang dari keranjang belanjanya
2. *Customer* menekan tombol “*Edit List*” pada aplikasi, kemudian memilih barang pada daftar belanja yang ingin dihapus
3. Jika jumlah barang yang dihapus hanya ada 1: lakukan S-1
4. Jika jumlah barang yang dihapus lebih dari 1: lakukan S-2
5. Jumlah belanja dikurangi sejumlah harga barang yang dihapus

*Subflows*:

1. S-1: Jumlah Barang Hanya 1
   * + - 1. Barang dihapus
2. S-2: Jumlah Barang Lebih dari 1
   * + - 1. Barang yang dihapus berkurang jumlahnya

*Alternate/Exceptional Flows*: -

###### *Checkout*

*Use Case Name* : *Checkout*

*ID* : 5

*Importance Level* : *High*

*Primary Actor* : *Customer*, *Cashier*

*Use Case Type* : *Detail*

Stakeholder and Interest :

1. *Customer* – ingin melakukan checkout terhadap semua barang di keranjang belanjanya dan melakukan pembayaran.
2. *Cashier* – ingin memastikan barang yang dibeli oleh *customer* dan menyelesaikan pemabayaran.
3. *Supermarket* – ingin melakukan integrasi pembayaran dengan point of sales dan memastikan integritas dari transaksi.
4. *Point of Sales* – ingin menerima pembayaran dari transaksi.

*Brief Description* : *Use case* ini menjelaskan bagaimana *customer* selesai berbelanja dan ingin membayarkan barang yang ada di daftar belanjanya.

*Trigger* : *Customer* *supermarket* ingin melakukan pembayaran terhadap daftar belanjanya.

*Trigger Type* : Eksternal

*Relationships*:

* + 1. *Association* : *Customer*, *Cashier*
    2. *Include* : *Get One-Time Password, Input* *One-Time Password*
    3. *Extend* :
    4. *Generalization* :

*Normal Flow of Events*:

1. Pelanggan menekan tombol “*Checkout*” pada aplikasi *Smart Shopping Assistant*
2. Aplikasi menyajikan *one-time password*
3. Pelanggan menunjukkan *one-time password* kepada *Cashier*
4. *Cashier* memasukkan *one-time password* pada halaman *admin* lalu mengirimnya ke *server*
5. *Server* melakukan validasi dan mengirim informasi transaksi ke halaman *admin*
6. *Cashier* mengarahkan pembayaran dengan *Point of Sales*

*Subflows*: -

*Alternate/Exceptional Flows*:

* + - 1. 3a: *Customer* membatalkan transaksinya dan tidak jadi membayar ataupun kembali berbelanja.

###### Get One Time Password

*Use Case Name* : *Get OTP*

*ID* : 4

*Importance Level* : *High*

Actor : *Customer*, *Server*

*Use Case Type* : Esensial

*Stakeholders and Interests* :

1. *Customer* – ingin mendapatkan OTP untuk melakukan *checkout* transaksi yang ada di dalam keranjang dan melakukan pembayaran.
2. *Server –* ingin mengirimkan OTP dari transaksi yang dikirimkan oleh *Customer.*

*Brief Description* : *Use case* ini mendeskripsikan bagaimana *customer* *supermarket* selesai berbelanja dan ingin mendapatkan OTP dari transaksi mereka.

*Trigge*r : *Customer* *Supermarket* menekan tombol “*Checkout*” pada layar “Daftar Belanja”.

*Trigger Type* : Ekternal

*Relationships*:

1. *Associations* : *Customer*
2. *Include* : -
3. *Extend* : -
4. *Generalization* : -

*Normal Flow of Events*:

* + - 1. *Customer* menekan tombol “*Checkout*” pada layar “Daftar Belanja”
      2. Aplikasi akan mengirimkan *request* kepada *server* untuk mendapatkan OTP transaksi tersebut.
      3. *Server* melakukan pencarian data dari transaksi tersebut kedalam *database.*
      4. *Server* mengembalikan OTP dari transaksi tersebut.
      5. *Response* OTP dari *Server* ditampilkan di layar “Checkout”.

Subflows: *–*

Alternate/Exceptional Flows: *–*

###### Input One Time Password

*Use Case Name* : *Input OTP*

*ID* : 5

*Importance Level* : *High*

Actor : *Cashier*, *Server*

*Use Case Type* : Esensial

*Stakeholders and Interests* :

* 1. *Cashier* – ingin mengirimkan OTP untuk memvalidasi transaksi *customer* *supermarket* dan melanjutkan proses pembayaran.
  2. *Server –* ingin memvalidasi OTP dari transaksi yang dikirimkan oleh kasir*.*

*Brief Description* : *Use case* ini mendeskripsikan bagaimana *cashier* mengirimkan OTP yang diberikan oleh *customer* *supermarket* dari transaksi mereka untuk menutup transaksi tersebut.

*Trigger* : Kasir menekan tombol “*Enter*” pada layar “*Admin Input* OTP”.

*Trigger Type* : *External*

*Relationships* :

1. *Associations* : *Cashier*
2. *Include* : -
3. *Extend* : -
4. *Generalization* : -

Normal Flow of Events:

* + 1. *Cashier* menekan tombol “*Enter*” pada layar “*Admin Input* OTP”.
    2. Aplikasi *admin* akan mengirimkan *request* kepada *server* untuk memvalidasi OTP transaksi tersebut.
    3. *Server* melakukan pencarian data dari transaksi tersebut kedalam *database* berdasarkan OTP yang diinput.
    4. *Server* mengembalikan hasil validasi dari OTP tersebut beserta detail transaksinya.
    5. *Response* dari *Server* ditampilkan di layar “Admin POS”.

*Subflows*: *–*

*Alternate/Exceptional Flows*:

1. 4a: Jika OTP tidak valid, *server* akan mengirimkan respon bahwa OTP yang diinput tidak valid.

##### 3.3.1.3 Activity Diagram

###### 3.3.1.3.1 Scan Carthttps://lh4.googleusercontent.com/lgW3dp3iBLl0qfwWvwJAsFybb2clxZFZKqALCQQFwMbEC0frOR591wq4Q_GDhqx8Hp1nPmpVImNza4ljwwiF7yOmmnLS3Bq6w3KhSmh69imnx5DAJxC2bMp-V7QvftPzrqRl9oEv

Gambar 3.10 *Activity* diagram *scan* *cart*

###### 3.3.1.3.2 Add Item to The Listhttps://lh5.googleusercontent.com/fFFFwXIW_E6MdT2DpOqOXx_RJ5luLkIYyJe9JxZJrmarfERzn3CvTARf9Kct4hyA5b74kBHjaezG0EirAcoqPy50d6Pmtu3ivSwhOp-OtYtV8tvzDjkSeDYxFAp7BEjZqFbmDZK2

Gambar 3.11 Activity diagram add item to the list

###### 3.3.1.3.3 View Listhttps://lh5.googleusercontent.com/bnvkL145udin0yOXrCYor7VrB1y5f68rxJow_4yrt284QmkKWwfeYBxn58oX1YkanacGlsflKhTRuEX0hIs0VhkS8N0kSKfyOmuvaEv06hY0RD1Z1Jz4v8MJg9BM4wiECttLjSG1

Gambar 3.12 *Activity* diagram *view list*

###### 3.3.1.3.4 Remove Item from The Listhttps://lh5.googleusercontent.com/JpbdL51w5aQwIVS_VVs-6rbQFNh70BqA3CF_nY3LYTmdqj-hIt4q5nI0MVjsj4QlaOk4jbOI7Z3qPxu30tZ42oQBuUbLToI1Q268P-L0U_yuGFe4oPAc5_pGzq8JwlBrkWLCufGi

Gambar 3.13 Activity diagram remove item from the list

###### 3.3.1.3.5 Checkouthttps://lh3.googleusercontent.com/-NIlLUgQV_HPXd7RfZGcLsID_iJdQzJU2r96nroCYmfIqfdFBEBVFojPQa0PYt5ZL8UXkSDdVdqhtnnJyzVhah5ktBH2Pm00YCk7rur-frGLClT_byHG-_igb8fKO33oc2BrW6b-

Gambar 3.14 Activity diagram checkout

#### 3.3.2 Mockup Aplikasi

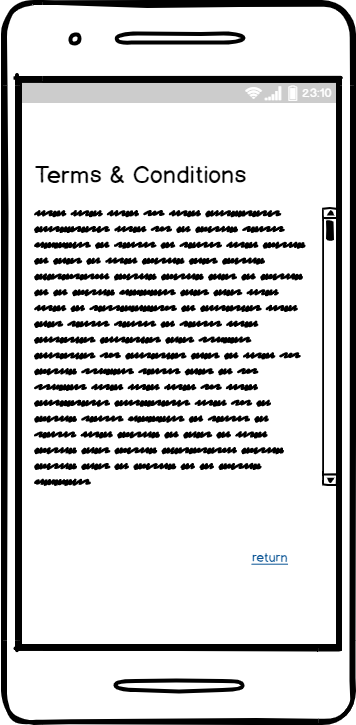
##### 3.3.2.1 Halaman *Boarding*



Gambar 3.15 Halaman *Boarding*

Halaman ini berisi logo dari aplikasi Smart Shopping Assistant serta tombol yang besar di tengah untuk mulai berbelanja, yaitu dengan melakukan pemindaian keranjang. Di ujung kanan bawah terdapat tombol tanda tanya yang mengarah ke halaman “Syarat dan Ketentuan”.

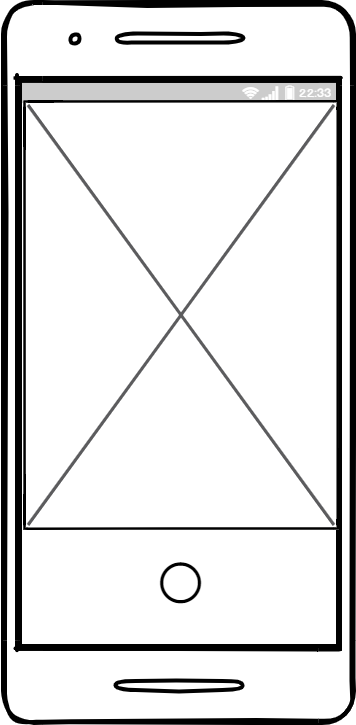
##### 3.3.2.2 Halaman Syarat dan Ketentuan



Gambar 3.16 Halaman Syarat dan Ketentuan

Syarat dan Ketentuan ditampilkan dalam satu halaman dengan *scroll view* agar dapat dibaca dengan ukuran *font* yang optimal.

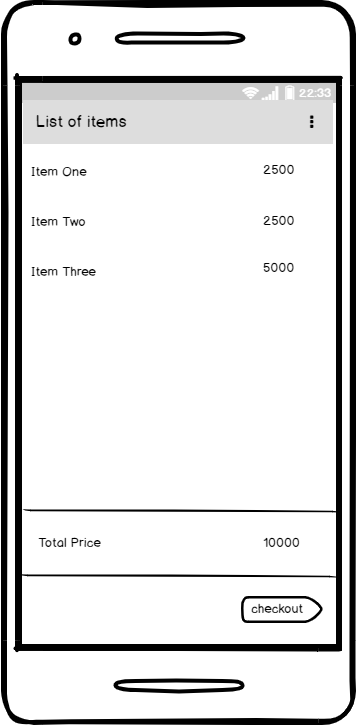
##### 3.3.2.3 Halaman *Scan* Keranjang



Gambar 3.17 Halaman *Scan* Keranjang

Setelah menekan tombol “Scan Keranjang” di halaman *boarding*, aplikasi akan mengarahkan ke kamera agar pengguna dapat memindai QR code yang ada pada keranjang. Setelah QR code berhasil dipindai oleh kamera, aplikasi akan segera mengarahkan pengguna ke halaman “Daftar Belanja”.

##### 3.3.2.4 Halaman Daftar Belanja

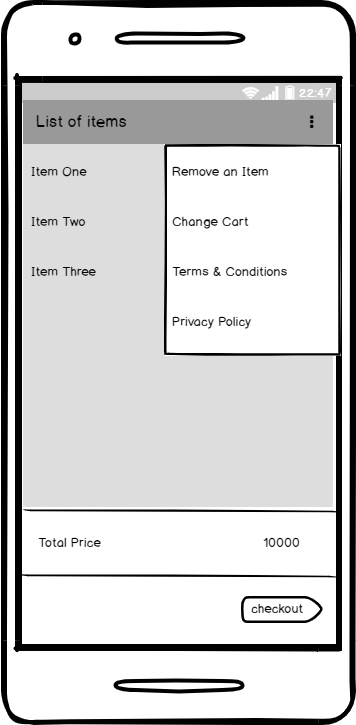


Gambar 3.18 Halaman Daftar Belanja

Barang yang sudah dibelanjakan dengan cara memindai RFID *Tag* barang ke Arduino yang terpasang pada Smart Shopping Cart akan ditampilkan pada halaman ini. Beberapa informasi yang terdapat pada halaman ini mencakup informasi nama barang yang dipindai, jumlah barang tersebut di dalam keranjang, harga satuannya serta harga total belanja pengguna. Tombol checkout tersedia di ujung kanan bawah halaman sehingga dapat di klik kapan saja pengguna sudah siap untuk melakukan pembayaran.

Adapula tombol kebab (tiga titik vertikal) yang berarti menu lainnya. Ketika ditekan, pilihan menu lainnya akan muncul dapat bentuk *pop up*.

##### 3.3.2.5 Halaman Menu Lainnya



Gambar 3.19 Halaman Menu Lainnya

Masih dari halaman daftar belanja, ketika tombol kebab ditekan, akan muncul *pop up* seperti *drop down* dari ujung bawah tombol kebab tersebut yang menampilkan menu lainnya seperti “menghapus barang”, “ganti keranjang”, “syarat dan ketentuan” dan sebagainya.

##### 3.3.2.6 Halaman Ganti Keranjang



Gambar 3.20 Halaman Ganti Keranjang

Ketika menu “ganti keranjang” ditekan, maka pengguna akan diarahkan ke layar kamera untuk memindai kembali QR keranjang lainnya dimana barang belanjaannya akan dipindahkan. Setelah selesai memindai, muncul pesan *pop up* yang mengembalikan pengguna ke daftar belanjanya.

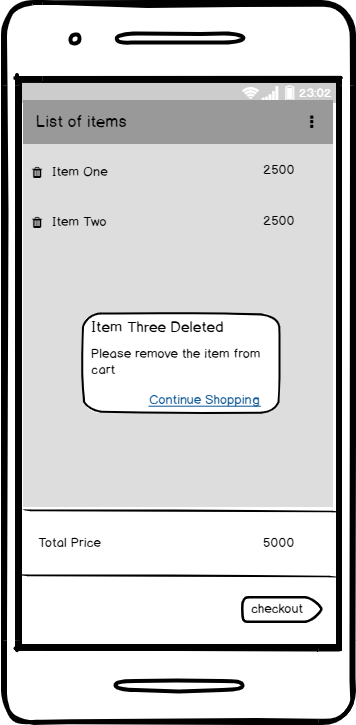
##### 3.3.2.7 Halaman Hapus Barang



Gambar 3.21 Halaman Hapus Barang

Ketika menu “hapus barang” ditekan, maka pengguna akan diarahkan kembali ke layar daftar belanja, namun dengan ikon tong sampah di sebelah setiap barang belanja sehingga memungkinkan pengguna untuk memilih barang belanja manapun yang ingin dihapus.

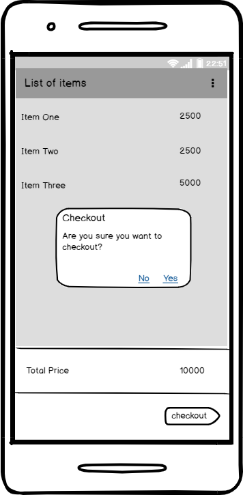
##### 3.3.2.8 Halaman Peringatan Barang Telah Dihapus



Gambar 3.22 Halaman Peringatan Barang Telah Dihapus

Setelah barang dihapus, pengguna akan menemukan pesan *pop up* yang memperingati pengguna untuk mengeluarkan barang yang dihapus dari keranjang untuk memastikan akurasi barang belanjaan yang ada di dalam keranjang.

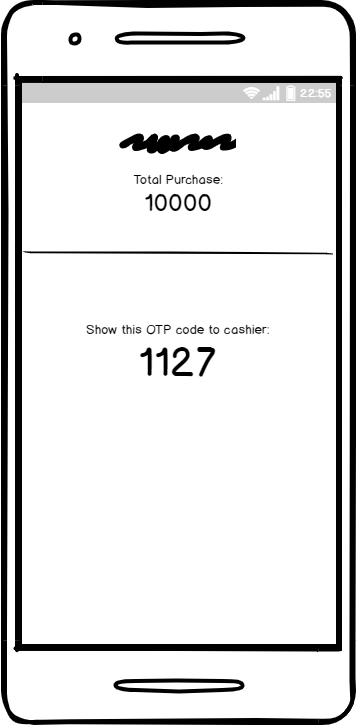
##### 3.3.2.9 Halaman Peringatan *Checkout*

****

Gambar 3.23 Halaman Peringatan *Checkout*

Ketika pengguna akan melakukan *checkout* yaitu menyelesaikan belanjanya, maka pengguna akan ditanyakan apakah yakin untuk menyelesaikan kegiatan belanjanya atau tidak. Hal ini dibuat sedemikian rupa agar tidak terjadi kebingungan jika tombol “*checkout*” secara tidak sengaja tertekan.

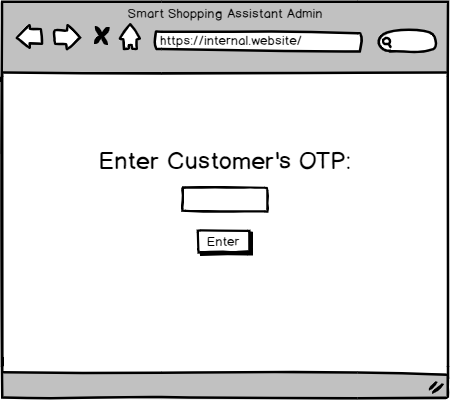
##### 3.3.2.10 Halaman *Checkout*



Gambar 2.24 Halaman *Checkout*

Pengguna disajikan total bayar sesuai dengan semua barang yang sudah dipindai oleh Smart Shopping Cart, serta kode OTP untuk dicocokkan dengan kasir sehingga dapat melanjutkan pembayaran. Pesan yang mengarahkan pengguna untuk menunjukkan kode OTP diharapkan membantu pengguna mengetahui langkah berikut dari prosedur belanja.

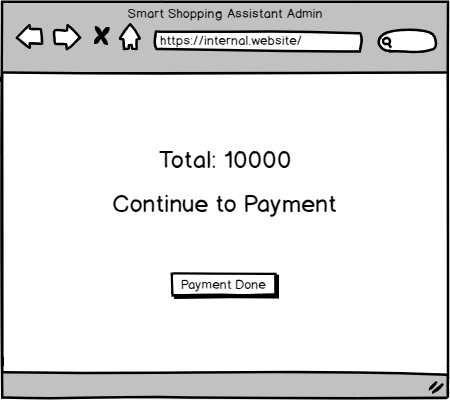
##### 3.3.2.11 Halaman Input OTP oleh Kasir



Gambar 2.25 Halaman Input OTP oleh Kasir

Kasir diberikan satu *field* input untuk memasukkan OTP dari pengguna. Kasir diharapkan menekan tombol *enter* untuk melanjutkan ke pembayaran dan dapat menerima respon dari server berupa informasi belanja dari pengguna dengan OTP tersebut.

##### 3.3.2.12 Halaman Lanjutkan ke Pembayaran



Gambar 2.26 Halaman Lanjutkan ke Pembayaran

Setelah menerima informasi transaksi, total bayar akan disajikan pada layar dan dapat dilanjutkan ke pembayaran. Ketika pembayaran sudah selesai, kasir dapat menekan tombol “pembayaran selesai” agar transaksi dapat ditandai selesai di server.

##### 3.3.2.13 Halaman Selesai Pembayaran



Gambar 2.27 Halaman Selesai Pembayaran

Dengan ditandainya transaksi sudah selesai, layar pada pengguna akan secara otomatis terganti ke tampilan yang menunjukkan bahwa transaksi sudah berhasil, adapula pesan yang berlaku sebagai tanda terima kasih atas pembelanjaan yang dilakukan.

### 3.4 Implementasi

Implementasi sistem dilakukan dengan cara-cara berikut:

#### 3.4.1 Hardware

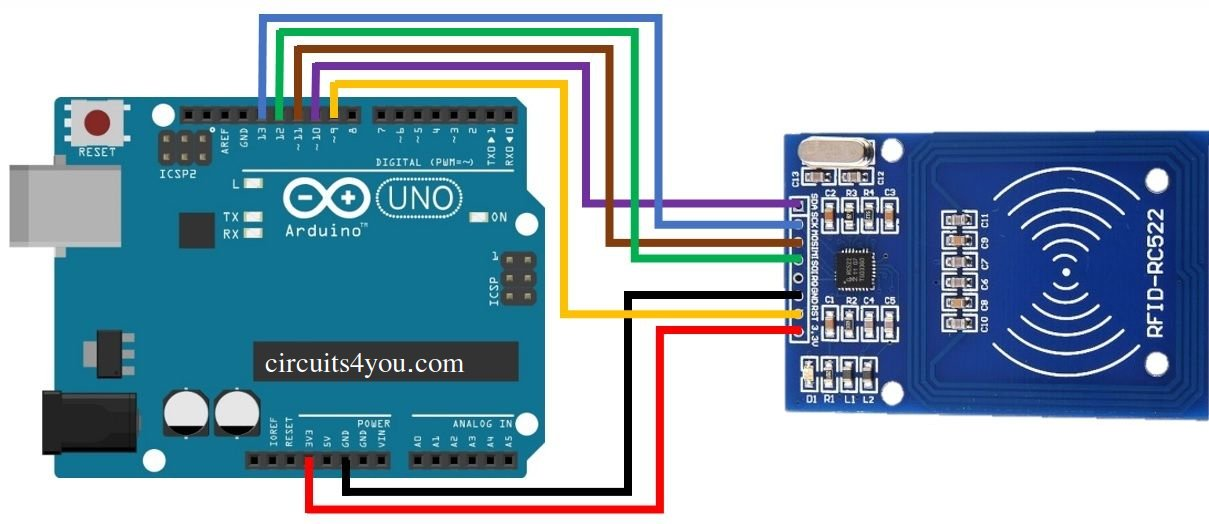
Arduino sebagai alat pengenal pada keranjang dipasang pada pinggir keranjang dan digunakan untuk melakukan transmisi data ataupun mengenalkan produk belanja kepada *customer* sebagai pemegang keranjang selama masa belanja.

Arduino Uno yang digunakan dalam project menggunakan ATmega328P sebagai mikrokontrollernya. Arduino Uno memiliki 14 *digital* I/O pin, 6 pin analog, 16 MHz quartz crystal, USB *connection*, *power* *jack*, ICSP *header* dan *reset* *button*.

Module yang digunakan pada Arduino berupa RC255 selaku sensor terhadap sinyal RFID Tag pasif. Lalu module ESP8266 yang berguna sebagai module wifi agar arduino bisa melakukan hit *service* saat mengirimkan data transaksi seperti informasi keranjang, informasi pengguna dan informasi total belanja.

Keranjang belanja yang digunakan sebagai prototype merupakan keranjang belanja dengan merk Mirani yang dijual secara bebas di pasaran, memiliki bahan plastik berwarna merah dengan jumlah 2 gagang yang bisa dikesampingkan. Arduino yang sudah dilengkapi module-module diatas dipasangkan pada keranjang menggunakan *cable* ties di pinggir keranjang.

1. Arduino RC522 Schema



1. Arduino ESP8266Mod Schema

#### 3.4.2 Mobile App

Visual Studio Code digunakan sebagai *editor* untuk membangun aplikasi, didukung dengan *framework* Flutter sebagai template UI yang berstandar 8 golden rules dan Material UI, dengan bahasa pemrograman Dart.

Pengembangan *Mobile App* dilakukan pada laptop dengan teknologi Virtualization sehingga pengujian setiap unit logika pada *Mobile* *App* dilakukan pada Virtual Device yang diinstall. Virtual Device yang digunakan merupakan Google Nexus 5 dengan operating *system* *Lollipop*, yaitu *Virtual Device* yang standar pada Visual Studio Code untuk pengembangan aplikasi Android.

#### 3.4.3 Software

*Software* terdiri atas *service*, *admin page* dan *database*. *Service* untuk transmisi data dijalankan menggunakan REST. Sementara implementasi *database* berisi data transaksi dan pengguna dibuat menggunakan PostGre. *Admin page* merupakan tampilan *pass-through* untuk menampilkan rincian transaksi sesuai yang ada di *database* dan merupakan *interface* untuk menampilkan kode OTP guna menyelesaikan transaksi serta mengarahkan ke pembayaran.

# BAB 4

## HASIL DAN PEMBAHASAN

* 1. Implementasi Aplikasi
     1. Tampilan Aplikasi Smart Shopping Assistant
     2. Smart Shopping Cart
     3. Testing Sistem Smart Shopping Cart
        1. Black Box Testing
        2. Integration Testing
  2. Penerapan dan Evaluasi
     1. Penerapan Sistem
        1. System Testing
     2. Evaluasi Aplikasi Smart Shopping Assistant
     3. Evaluasi Sistem

# DAFTAR PUSTAKA

Arduino. (n.d.-a). *Arduino - Introduction*. Diakses September 5, 2019, dari https://www.arduino.cc/en/Guide/Introduction

Arduino. (n.d.-b). Arduino Uno Rev3 | Arduino *Official Store*. Diakses September 10, 2019, dari <https://store.arduino.cc/arduino-uno-rev3>

Berdaliev, Y., & James, A. P. (2016). RFID-*Cloud Smart Cart System*. Diambil dari <https://www.researchgate.net/publication/306186989_RFID-Cloud_Smart_Cart_System>

BoxTechnologies, & Intel. (2015). *How long does it take to lose a customer*? (4). Diambil dari https://boxtechnologies.com/wp- content/uploads/2018/03/box\_how\_long\_does\_it\_take\_to\_lose\_a\_customer.pdf

Coskun, V., Ok, K., & Ozdenizci, B. (2013). *Professional* NFC *Application Development for* Android (1st ed.). Hoboken, New Jersey, United States: Wiley.

Dennis, A., Wixom, B. H., & Tegarden, D. (2015). *Systems Analysis and Design: An Object-Oriented Approach with* UML (5th ed.). Hoboken, New Jersey, United States: Wiley.

flutter-dev. (2019, Desember 11). Flutter *Documentation*. Diakses Januari 7, 2020, dari https://flutter.dev/docs#new-to-flutter

Google. (n.d.). *Introduction*. Diakses January 18, 2020, dari https://material.io/design/introduction/

GSM *Association*. (2014, Agustus). *Understanding the Internet of Things* (IoT). Diakses September 5, 2019, dari <https://www.gsma.com/iot/wp-content/uploads/2014/08/cl_iot_wp_07_14.pdf>

Jie, Y. (2010). *The Optimal Supermarket Service. International Journal of Business and Management*, 5(2). https://doi.org/10.5539/ijbm.v5n2p128

Lardinois, F. (2015, April 29). TechCrunch *is now a part of* Verizon Media. Diakses September 15, 2019, dari https://techcrunch.com/2015/04/29/microsoft-shocks-the-world-with-visual-studio-code-a-free-code-editor-for-os-x-linux-and-windows/

Masse, M. (2011). REST API *Design Rulebook*. Culemborg, Netherlands: Van Duuren Media.

OpenJS *Foundation*. (n.d.). *About* Node.Js. Diakses Januari 17, 2020, dari https://nodejs.org/en/about/

PostgreSQL. (n.d.). PostgreSQL: *About*. Diakses Januari 18, 2020, dari https://www.postgresql.org/about/

Roger S. Pressman, & Maxim, B. (2014). *Software Engineering: A Practitioner’s Approach* (8th ed.). New York, United States: McGraw-Hill *Education*.

Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., & Diakopoulos, N. (2016). *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (6th Edition) (6th ed.). London, England, United Kingdom: Pearson.

Yen, L. P. (2016, Desember 22). 8 *Golden Rules Interface Design*. Diakses Januari 17, 2020, from https://socs.binus.ac.id/2016/12/22/8-golden-rules-interface-design/