UNIT TESTING PADA APLIKASI *WEB MOBILE*
(STUDI KASUS BISNIS JASA LAUNDRY)

Disusun Oleh:

N a m a	: Dwi Kusumastuti Puji Rahayu
NIM	: 13523202

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA

2020

HALAMAN PENGESAHAN DOSEN PEMBIMBING

UNIT TESTING PADA APLIKASI *WEB MOBILE* (STUDI KASUS BISNIS JASA LAUNDRY)

TUGAS AKHIR

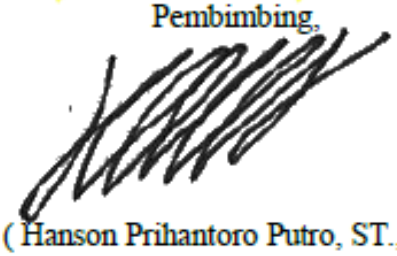
Disusun Oleh:

N a m a : Dwi Kusumastuti Puji Rahayu

NIM : 13523202

Yogyakarta,28 Juli 2020

Pembimbing,



(Hanson Prihantoro Putro, ST.,MT)

HALAMAN PENGESAHAN DOSEN PENGUJI

UNIT TESTING PADA APLIKASI *WEB MOBILE* (STUDI KASUS BISNIS JASA LAUNDRY)

TUGAS AKHIR

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

Yogyakarta, 12 Agustus 2020

Tim Penguji

Hanson Prihantoro Putro, S.T., M.T.

Anggota 1

Ari Sujarwo S.Kom., MIT (Hons).

Anggota 2

Irving Vitra Paputungan S.T., M.Sc.

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Dwi Kusumastuti Puji Rahayu

NIM : 13523202

Tugas akhir dengan judul:

UNIT TESTING PADA APLIKASI *WEB MOBILE* (STUDI KASUS BISNIS JASA LAUNDRY)

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

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

Yogyakarta, 12 Agustus 2020

(Dwi Kusumastuti Puji Rahayu)

HALAMAN PERSEMBAHAN

Alhamdulillahirabbil'alamin, puji syukur kusembahkan kepadaMu ya Allah, Tuhan yang Maha Pengasih dan Penyayang. BerkatMu Ya Rahman Ya Rahim hamba bisa menjadi pribadi yang berilmu dan bersabar. Segala syukur kuucapkan kepadaMu Ya Rabb, karena sudah menghadirkan orang-orang berarti di sekelilingku.

Skripsi ini kupersembahkan untuk:

- ♥ Ibundaku tercinta Baiq Nurhidayati dan Ayahandaku tersayang Basuki, terimakasih yang tak terhingga atas segala pengorbanan, kasih sayang serta doa yang senantiasa mengalir demi keberhasilan anaknya dan semua itu takkan pernah dapat dibalas dengan budi apa pun.
- ♥ Kakakku Warih Suci Perwitosari yang selalu memberikan dukungan dan doa tanpa henti.
- ♥ Adikku Tri Wahyu Mulya Saputra yang selalu memberikan semangat dan dukungan setiap saat.
- ♥ Keponakan kecilku Farhan yang selalu berbagi keceriaan dan kebahagiaan.
- ♥ Sepupuku Karina dan Mas Emi yang selalu membantu menyemangati dan memberi banyak masukan dalam mengerjakan tugas akhir ini
- ♥ Sahabat dan teman seperjuanganku Endar Listianingrum, terima kasih telah menjadi sahabat terbaik selama berada di Jogja.
- ♥ Teman kelas sekaligus teman kos, Sri Yunianita, terima kasih telah banyak membantu menyemangati dan mengajari dalam menerjakan tugas akhir ini.
- ♥ Seluruh keluarga besarku yang ada di Lombok dan Yogyakarta, terima kasih selalu memberi dukungan selama ini.
- ♥ Rekan-rekan seperjuanganku D'family dan teman-teman ETERNITY terima kasih atas dukungan dan bantuannya selama ini. Terima kasih juga untuk kenangan yang kita rajut setiap harinya di Yogyakarta kota penuh memori, atas tawa yang setiap hari kita miliki, dan atas solidaritas yang luar biasa.

.

HALAMAN MOTO

Man Jadda Wajada,

Man Shabara Zhafira,

Man Saara Ala Darbi Washala.

"*Karena sesungguhnya sesudah kesulitan itu ada kemudahan*."
(QS. Alam Nasyroh: 5)

"Disiplin adalah jembatan antara cita-cita dan pencapaiannya."
(Jim Rohn)

KATA PENGANTAR

Puji syukur atas karunia yang Allah SWT berikan, atas limpahan rahmat, dan kasih sayang-Nya, atas petunjuk dan bimbingan yang telah diberikan, sehingga penulis dapat menyelesaikan skripsi yang berjudul "*Unit testing* Pada Aplikasi *Web mobile* (Studi Kasus Bisnis Jasa Laundry)". Tujuan penulisan laporan dan pelaksanaan penelitian tugas akhir ini adalah untuk memenuhi syarat kelulusan guna memperoleh gelar sarjana Teknik Informatika Universitas Islam Indonesia.

Dalam pembuatan dan penyusunan Laporan Tugas Akhir ini, penulis selalu mendapatkan bimbingan, bantuan, dukungan, doa dan motivasi dari banyak pihak. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih sedalam-dalamnya kepada:

1. Bapak Prof. Fathul Wahid, S.T., M.Sc., Ph.D selaku Rektor Universitas Islam Indonesia
2. Bapak Hendrik, S.T., M.Eng selaku Ketua Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
3. Bapak Hanson Prihantoro Putro, ST.,MT selaku dosen pembimbing yang telah banyak memberikan bimbingan, waktu dan ilmunya hingga Laporan Tugas Akhir ini selesai.
4. Bapak dan Ibu Dosen Jurusan Teknik Informatika yang telah memberikan banyak ilmu yang bermanfaat kepada penulis.
5. Kedua Orang Tua tercinta, Ibu Baiq Nurhidayati dan Bapak Basuki yang selalu memberikan dukungan, motivasi, semangat dan doa yang mengalir tanpa henti selama penulis menempuh pendidikan Informatika dan penyusunan Laporan Tugas Akhir ini.
6. Kedua saudara penulis Warih Suci Perwitosari dan Tri Wahyu Mulya Saputra yang selalu memberi semangat dan dukungan
7. Sepupuku Karina dan Mas Emi yang selalu menyemangati dan memberikan masukan dan saran selama penulis mengerjakan tugas akhir
8. Sahabat dan teman seperjuanganku, Endar Listuaningrum terima kasih atas segala bantuannya dari awal kuliah hingga penulis mengerjakan tugas akhir.
9. Sahabatku dan teman curhatku Bunga, April, Chevi, Lian, Listi, Dian, Dewi, Mba Nung, Balqis dan Nita yang selalu dengan sabar mengajariku.
10. Seluruh keluarga besarku yang ada di Lombok dan Yogyakarta, terima kasih atas doa dan dukungannya.

11. Teman-teman mahasiswa D'family dan Informatika UII yang telah memberikan motivasi sehingga penulis dapat menyelesaikan studi.
12. Pihak-pihak lain yang telah membantu penyelesaian skripsi ini yang tidak dapat dituliskan satu persatu.

Penulis mengucapkan banyak terima kasih, semoga Allah SWT selalu melimpahkan karunia, hidayah dan ilmu yang bermanfaat bagi kita semua. Penulis berharap semoga skripsi ini dapat bermanfaat bagi penulis dan pembaca. *Aamiin*.

Yogyakarta, Agustus 2020

(Dwi Kusumastuti Puji Rahayu)

SARI

Unit testing merupakan salah satu metode pengujian perangkat lunak yang digunakan dengan cara menguji unit terkecil dari sebuah kode. Kegunaan *unit testing* bagi pengembang perangkat lunak adalah dapat memperkecil kesalahan atau *bug* yang ada pada sistem. Pengujian *unit testing* dilakukan hingga sistem sudah memenuhi syarat sesuai rancangan. Untuk membuktikan bahwa pengujian *unit testing* dapat efektif digunakan dalam pengembangan sistem, maka dalam penelitian ini dilakukan pengujian aplikasi *web mobile* bisnis jasa laundry dengan *unit testing* yaitu menggunakan PHPUnit .

PHPUnit merupakan salah satu *framework* pengujian yang digunakan pada aplikasi yang menggunakan bahasa pemrograman PHP. Pengujian ini dilakukan dengan PHPUnit dikarenakan PHPUnit dianggap lebih unggul dari hal fungsionalitas, efisiensi, kehandalan dan probabilitas. Metode *white box testing* digunakan karena metode ini merupakan pengujian terhadap struktural dan alur logika kode sehingga dapat digunakan dalam pengembangan pengujian aplikasi laundry ini.

Pada penelitian ini aplikasi dikembangkan menggunakan *framework* Codeigniter yang memiliki konsep MVC (Model, View dan Controllers). Pengujian dilakukan pada bagian kelas Model, di mana dalam bagian tersebut ditemukan beberapa fungsi, namun hanya dipilih 4 fungsi yang dapat diuji karena memiliki beberapa jumlah percabangan. Beberapa tahapan dalam pengujian *unit testing* ini antara lain, memilih beberapa fungsi yang memiliki kondisi percabangan, mengkonversikan *source code* ke dalam bentuk *flowgraph* kemudian menghitung nilai *cyclomatic complexity* untuk menentukan jumlah jalur yang terlewati pada *source code*. Setelah semua terlewati selanjutnya mengukur tingkat keberhasilan pengujian dengan teknik *statement coverage*. Pengujian *white box* ini menggunakan teknik *statement coverage* sebagai pengukur tingkat keberhasilan pengujian. Teknik ini dilakukan dengan menjalankan data uji yang mencakup semua *statement* (*source code*) yang dijalankan agar mencapai nilai 100% pada *statement*. Berdasarkan pengujian yang telah dilakukan maka diperoleh kesimpulan yaitu, pengujian dilakukan dengan *unit testing* menggunakan framework PHPUnit dengan menguji aplikasi berdasarkan unit terkecil, dan menguji setiap fungsi yang telah dipilih. Semua fungsi terpilih tersebut, telah dieksekusi minimal satu kali dan melewati seluruh *statement* sehingga berhasil dan memperoleh nilai *coverage* sebesar 100%.

Kata kunci: *unit testing*, PHPUnit, *web mobile, statement coverage, source code, white box testing*.

GLOSARIUM

White box testing	Metode pengujian yang didasarkan pada pengecekan terhadap detail perancangan.
Statement coverage	Teknik pengujian *white box testing* yang digunakan untuk mengeksekusi *test case*(kasus uji) minimal satu kali dari setiap *statement* (baris kode). Sehingga dapat memperoleh persentase (nilai *coverage*)
Flowgraph	Alur logika program yang direpresentasikan dengan sekumpulan *nodes* (titik) dan *edge* (panah).
Cyclomatic complexity	Metode pengukuran perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logika sebuah program.
Framework	Kerangka kerja yang digunakan untuk mengembangkan aplikasi berbasis desktop atau aplikasi berbasis website

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam pembuatan sistem, diperlukan pengujian untuk memastikan kinerja dari sistem tersebut apakah sudah berjalan dengan benar dan sesuai dengan yang diharapkan atau tidak. Proses pengembangan sistem akan lebih cepat dilakukan karena dengan adanya pengujian dapat lebih mudah mengetahui letak kesalahan jika terdapat *bug* atau *error* dari sistem tersebut. Boris Beizer telah mendefinisikan tingkat kecanggihan pengembangan dalam pengujian perangkat lunak. Pada tingkat terendah, pengujian dianggap tidak berbeda dengan *debugging*. Namun pada tingkat yang lebih tinggi, pengujian menjadi pola pikir yang bertujuan untuk memaksimalkan sistem (Parkin, 1997).

Web mobile merupakan kumpulan halaman HTML berbasis *browser* yang dapat diakses dengan menggunakan perangkat yang adaptif atau dapat menyesuaikan dengan perangkat yang digunakan seperti *smartphone* dan *gadget* atau tablet tanpa harus menginstal aplikasi terlebih dahulu. *Mobile web* memiliki prinsip ”*develop once run everywhere*” yang berati dapat beroperasi pada *platform* yang berbeda dalam sekali pengembangan (Solechah, 2016). Pengembangan aplikasi *web mobile* biasanya menggunakan teknologi web terbuka seperti HTML, CSS dan JavaScript. Aplikasi ini kemudian dapat di *host* di server web yang sudah ada dan diakses di URL standar melalui *web browser*.

Unit testing adalah salah satu metode pengujian perangkat lunak, di mana bagian terkecil/unit dari sebuah kode akan diujikan. Pengujian *unit testing* merupakan keterampilan bagi pengembang perangkat lunak karena dapat memperkecil *bug* pada kode (Kua, 2019). Pengujian *unit testing* dilakukan hingga pembuatan kode sistem telah memenuhi persyaratan seperti yang telah dirancang sebelumnya. Prosedur menggunakan *unit testing* adalah menuliskan *test case* terlebih dahulu untuk semua fungsi dan metode sehingga setiap kali terjadi kesalahan dapat diidentifikasi dan diperbaiki dengan cepat. Jika masing-masing unit telah diuji dan dites semua berhasil, pengujian perangkat lunak dapat dikatakan memenuhi persyaratan. Pada metode ini akan banyak menggunakan teknik *white box testing*, memeriksa jalur tertentu untuk memastikan cakupan jalur yang komplit dan deteksi *error* yang maksimum (Jatnika dan Irwan, 2019). *Unit testing* merupakan tahap mewujudkan perancangan sistem ke

dalam kumpulan program atau unit program. Setiap unit program akan dites untuk memastikan bahwa fungsi unit tersebut sesuai dengan spesifikasinya (Sommerville, 2011).

Tugas akhir ini merupakan lanjutan dari penelitian sebelumnya yaitu Perancangan *Web mobile* dengan Metode *User Centered Design* (UCD) untuk pengelolaan bisnis jasa usaha laundry. Dalam penelitian tersebut baru dilakukan sebatas perancangan, belum mencapai implementasi sistem (Wulandari, 2017). Agar sistem yang dirancang dapat berjalan sesuai tujuannya, maka perlu diimplementasikan kedalam sebuah program. Sistem akan dibuat berbasis *web mobile* dengan pengujian menggunakan metode *unit testing*.

Tujuan dibuatnya sistem dengan menggunakan pengujian *unit testing* adalah untuk memvalidasi bahwa setiap perangkat lunak telah berfungsi sebagaimana mestinya dan mempersingkat waktu *debugging* karena cakupan pengujian yang lebih kecil. Sistem ini akan dibuat berbasis *web mobile* sehingga pengguna dapat mengakses di mana dan kapan saja. Diharapkan dengan adanya sistem ini dapat meminimalisir terjadinya kesalahan dan dapat meningkatkan kepuasan pelanggan laundry.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, maka dirumuskan masalah dalam penelitian ini yaitu bagaimana menguji aplikasi *web mobile* dengan menggunakan metode *unit testing*.

1.3 Batasan Masalah

Penelitian ini memiliki beberapa batasan masalah yaitu:

a. Aplikasi ini digunakan untuk pengelolaan bisnis jasa laundry.
b. Aplikasi ini diuji dengan framework PHPUnit
c. Pengujian dilakukan hanya pada kelas model dari arsitektur MVC

1.4 Tujuan Penelitian

Tujuan dibuatnya sistem dengan melakukan pengujian *unit testing* adalah untuk menguji aplikasi *web mobile* dengan metode *unit testing*.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini antara lain:

a. Mengetahui sejauh mana peran pengujian menggunakan *unit testing* pada aplikasi bisnis jasa laundry

b. Mengetahui sejauh mana aplikasi telah berfungsi dengan baik
c. Meminimalisir kesalahan pada aplikasi
d. Menjadi dasar untuk penelitian terkait pengujian dengan *unit testing*

1.6 Metodologi Penelitian

Metodologi penelitian dilakukan agar dalam pembuatan sistem dapat terarah sesuai rencana dan mendapatkan hasil yang diharapkan. Metodologi yang diterapkan dalam penyusunan penelitian ini antara lain:

1. Studi Literatur

   Studi literatur dilakukan melalui jurnal, buku, internet dan sumber lain yang terkait dengan penelitian ini. Metode ini juga digunakan sebagai referensi landasan teori.

2. Analisi Kebutuhan

   Analisis kebutuhan mendefinisikan dan menentukan kebutuhan spesifik dari aplikasi yang akan diuji.

3. Pengembangan Perangkat Lunak

   Pengembangan dilakukan dengan tahapan yang dibutuhkan untuk membangun dan mengembangkan sebuah sistem dan memilih bagian mana saja yang akan dijadikan bahan pengembagan perangkat lunak.

4. Pengujian *Unit testing* dengan PHPUnit

   Produk yang dihasilkan kemudian diuji menggunakan metode *unit testing* dengan menggunakan *framework* PHPUnit . Dari tahapan ini dapat diketahui apakah aplikasi sudah sesuai dengan rancangan yang telah ada sebelumnya atau belum

5. Evaluasi

   Evaluasi secara keseluruhan terhadap semua kegiatan dalam penelitian agar dapat diambil kesimpulan sesuai dari hasil pengujian *unit testing*.

1.7 Sistematika Penulisan

Penulisan laporan dibuat dengan secara terstruktur untuk mengetahui apa saja yang terlibat dalam penelitian.

a. BAB I PENDAHULUAN

   Memuat tentang latar belakang penulisan, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian dan sistematika penulisan laporan.

b. BAB II LANDASAN TEORI

Memuat teori-teori yang ada dan digunakan sebagai landasan untuk menyeesaikan permasalahan yang ada dalam penelitian ini. Bahasan dalam bagian ini meliputi pengertian dan tahapan dari pengujian *Unit testing*.

c. BAB III ANALISIS KEBUTUHAN

Memuat definisi dan menentukan kebutuhan untuk pengujian aplikasi yang akan diuji. Bahasan dalam bagian ini meliputi apa saja yang diperlukan untuk pengujian.

d. BAB IV IMPLEMENTASI DAN PENGUJIAN

Memuat uraian tentang pengembangan aplikasi yang telah diimplemntasikan dan melakukan pengujian sesuai dengan rancangan sebelumnya. Pada bagian ini dijelaskan uraian hasil dan evaluasi dari pengujian *unit testing* menggunakan PHPUnit.

e. BAB V KESIMPULAN DAN SARAN

Memuat kesimpulan yang merupakan rangkuman dari hasil penelitian serta saran- saran yang perlu diperhatian dalam pengujian *unit testing*.

BAB II
LANDASAN TEORI

2.1 Web Mobile

Pada saat ini, aplikasi *web mobile* sudah tidak asing lagi untuk digunakan baik sebagai media informasi ataupun untuk membantu proses bisnis. Pembuatan aplikasi *web mobile* hampir sama dengan pembuatan aplikasi *website* namun untuk *web mobile* harus ditambahkan fitur *responsive* agar dapat diakses melalui *browser* pada perangkat mobile.

Web mobile menurut Devi (2015) merupakan satu-satunya *platfrom* yang tersedia dan mampu berjalan pada semua perangkat *mobile*, dan perancangan menggunakan standar dan protokol yang sama dengan *desktop web*. Untuk dapat mendesain aplikasi *web* untuk *mobile* harus diperhatikan betul bahwa karakteristik *web* untuk *mobile* berbeda dengan desktop (Indriasari, Adi dan Sidhi 2011). Dalam penelitian yang dilakukan oleh Budi dkk. (2018), dapat disimpulkan *website mobile* adalah situs yang dirancang khusus untuk perangkat *mobile* yang dirancang menggunakan standar dan protokol yang sama dengan *desktop web*.

Terdapat beberapa aspek yang harus diperhatikan dalam perancangan *web mobile*, yaitu keterbatasan fisik, meliputi bentuk fisik yang kecil dan *input* terbatas. Keterbatasan teknis, meliputi tingkat keamanan yang terbatas, faktor fisik yang bervariasi, *input* yang bervariasi seperti *touchscreen*, *numeric keypad*, qwerty *keypad* dan akses data yang bervariasi.

2.2 *Quality Assurance*

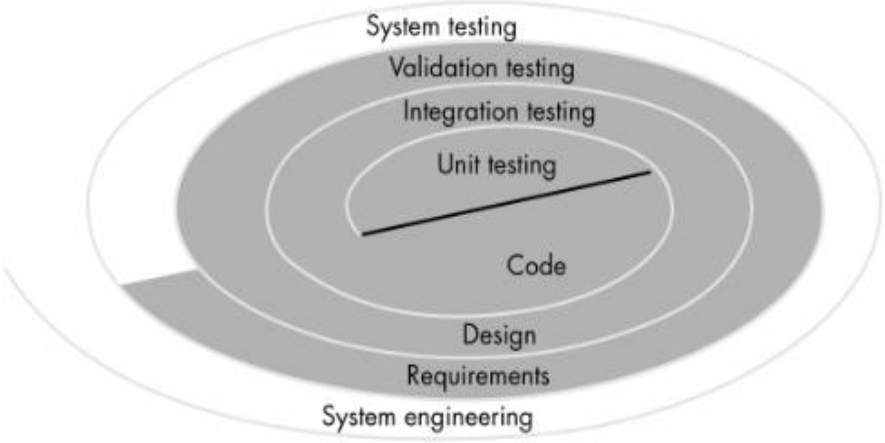
Dalam pengembangan sebuah aplikasi, *quality assurance* atau penjamin mutu sangat dibutuhkan untuk memberikan bukti-bukti untuk membangun kepercayaan bahwa aplikasi tersebut telah berfungsi dengan baik. Menurut (Bani, 2015) *Quality Assurance* (QA) merupakan seluruh kegiatan terencana dan juga sistematis yang diterapkan dalam sistem manajemen mutu untuk membuktikan bahwa suatu produk telah memenuhi syarat. QA sendiri digunakan untuk mencegah produk gagal sehingga dilakukan mulai dari proses perencanaan hingga aplikasi sudah selesai. Untuk mencapai pengembangan aplikasi yang efektif, pemeriksaan aplikasi tidak hanya dilakukan saat tahap akhir saja namun tindakan dilakukan sepanjang siklus proyek dari penyusunan program, perencanaan, pengawasan pemeriksaan dan pengendalian mutu (Soeharto, 1995).

Pekerjaan QA berperan besar untuk pengembangan dan pengujian aplikasi karena QA tidak hanya sebatas menguji, mencoba dan memeriksa namun juga bertanggung jawab untuk memastikan aplikasi yang dikembangakan bekerja dengan baik. Untuk itu, ada beberapa hal yang juga perlu dilakukan seperti mencatat hasil audit, mengevaluasi dan memberikan rekomendasi atau saran untuk menghilangkan penyebab masalah yang ada sebelum sebuah produk siap digunakan. Pengujian sebaiknya dilakukan sedini mungkin dan harus berfokus pada tujuan yang telah ditetapkan sehingga *bug* atau *error* dengan cepat dapat ditemukan dan diatasi sesuai dengan prinsip pengujian. Pengujian dapat dilakukan dengan memulai dari unit terkecil sebuah aplikasi yaitu dengan menggunakan metode unit testing.

2.3 *Unit Testing*

Pengujian *software* diperlukan untuk memastikan *software* yang dibuat dapat berjalan sesuai dengan fungsionalitas yang diharapkan. Sehingga pengembang *software* harus menyiapkan waktu untuk menguji program yang sudah dibuat agar kesalahan ataupun kekurangan dapat dideteksi sejak awal dan dikoreksi secepatnya. Pengujian atau *testing* sendiri merupakan elemen kritis dari jaminan kualitas perangkat lunak dan merupakan bagian yang tidak terpisah dari siklus hidup pengembangan *software* seperti halnya analisis, desain, dan pengkodean (Shi, 2010).

Menurut Pressman (2010: 453), strategi pengujian perangkat lunak terdiri atas 4 (empat) jenis pengujian, yaitu pengujian unit, pengujian integrasi, pengujian validasi, dan pengujian sistem. Strategi pengujian perangkat lunak digambarkan dalam bentuk spiral seperti pada Gambar 2.1 dibawah ini:



Gambar 2.1 Strategi Pengujian (Pressman, 2010)

Penjelasan dari masing-masing tahapan pengujian perangkat lunak akan dijabarkan sebagai berikut (Kartanti, 2015):

a. Pengujian Unit

   Pengujian Unit berfokus pada upaya verifikasi terkecil dari perancangan perangkat lunak, komponen atau modul perangkat lunak. Menurut Williams (2006: 39), pengujian unit dilakukan oleh pengembang. Pengujian unit dilakukan menggunakan pengujian *white box*. Salah satu metode pengujian *white box* adalah pengujian jalur dasar (*basis path*).

b. Pengujian Intregrasi

   Pengujian intregrasi adalah pengujian terhadap unit-unit program yang saling berhubungan atau berintregasi untuk membangun arsitek perangkat lunak.

c. Pengujian Validasi

   Pengujian Validasi Pengujian validasi berfokus pada output sistem berdasarkan tindakan yang dilakukan pengguna. Validasi dikatakan berhasil jika perangkat lunak berfungsi sesuai harapan pengguna.

d. Pengujian Sistem

   Pengujian sistem adalah pengujian yang tujuan utamanya adalah untuk sepenuhnya mewujudkan sistem berbasis komputer. Menurut Pressman (2010: 571), pengujian keamanan, pengujian stress, dan pengujian kinerja merupakan pengujian sistem.

Unit testing merupakan salah satu teknik yang dapat digunakan untuk melakukan pengujian perangka lunak dan berfokus pada bagian terkecil dari sebuah aplikasi. *Unit testing* memungkinkan kita untuk menguji perangkat lunak secara terpisah. Biasanya dengan menguji bagian unit terkecil dan beberapa potongan kode seperti fungsi. Cara yang dilakukan untuk *unit testing* adalah menulis kode *Unit testing* dan *Test case* secara manual ketika akan melakukan pengujian, memerlukan waktu dan sangat rentan terhadap kesalahan.

Menurut Rosa dan Shalahuddin (2013: 277), “pengujian unit fokus pada usaha verifikasi pada unit yang terkecil pada desain perangkat lunak(komponen atau modul perangkat lunak). Setiap unit perangkat lunak diuji agar dapat diperiksa apakah aliran masukan (*input*) dan keluaran (*output*) dari unit sudah sesuai dengan yang diinginkan. Pengujian unit biasanya dilakukan saat kode program dibuat”.

Menurut (Kua, 2019) dalam tulisannya yang berjudul *Unit testing*, terdapat beberapa hal pentingnya menggunakan *unit testing* yaitu:

1. Menggunakan *unit testing* akan menghabiskan sedikit waktu ketika *debugging*, dan sedikit lebih yakin bahwa perubahan kode yang dibuat sudah benar.
2. Cakupan *test* disimpan seminimal mungkin dan ketika terjadi kegagalan maka akan lebih mudah ditemukan letak kegagalan tersebut.
3. Desain yang lebih baik karena pengembang aplikasi berfikir bagaimana kode mereka dikembangkan.
4. Alat regresi yang baik. Selama *unit testing* berlangsung, pengembang dapat yakin bahwa perubahan mereka terhadap kode tidak memiliki dampak negatif pada fungsi aplikasi secara keseluruhan.
5. Mengurangi biaya masa depan. Banyak penelitian telah membuktikan bahwa biaya secara signifikan lebih banyak untuk memperbaiki bug.

Menurut (Khan, 2011) terdapat dua teknik penting dalam pengujian perangkat lunak yaitu *black box* dan *white box*. Pengujian *white box*, merupakan pengujian yang didasarkan pada pengecekan terhadap detail perancangan, menggunakan struktur kontrol dari desain program secara prosedural untuk membagi pengujian ke dalam beberapa kasus pengujian. Secara sekilas dapat diambil kesimpulan pengujian *white box* merupakan petunjuk untuk mendapatkan program yang benar secara keseluruhan. Sedangkan *Black box Testing* merupakan pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak, *test*er dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program.

Unit testing umumnya menggunakan pengujian kotak putih atau *white box*. Pengujian *white box* adalah pengujian yang memperhitungkan mekanisme internal suatu sistem atau komponen (IEEE, 1990). Pengujian *white box* juga dikenal sebagai pengujian struktural, pengujian kotak kosong dan pengujian kotak kaca. Pengujian *white box* dapat mengungkap kesalahan implementasi seperti manajemen yang buruk dengan menganalisis cara kerja internal dan struktur perangkat lunak. Pengujian *white box* berlaku di tingkat integrasi, unit dan sistem dari proses pengujian perangkat lunak. Dalam pengujian ini, *test*er perlu melihat ke dalam kode dan mencari tahu unit kode mana yang berperilaku tidak tepat (Khan, 2012).

Pengujian *White-box* merupakan filosofi perancangan *test case* yang menggunakan struktur control untuk menghasilkan *test case* (Presssman, 2010).

Tujuan dari *white-box testing* menurut Pressman antara lain:

1. Memastikan semua jalur independen telah dieksekusi sedikitnya satu kali.
2. Melaksanakan semua keputusan logis pada sisi benar dan salah.
3. Melaksanakan semua *loop* pada batas operasional.
4. Melakukan struktur data internal untuk memastikan kesahihannya.

Salah satu teknik pengujian pada *white box* adalah *statement coverage*. *Statement coverage* merupakan pengujian yang memastikan setiap *statement* yang dieksekusi dan dijalankan minimal satu kali, satu *statement* yang dijalankan merupakan bagian dari satu *test case* sehingga tidak ada *test case* yang dijalankan dengan *statement* yang sama (Kusumaningtiyas, Fitria dan Diputra, 2016). *Statement coverage* digunakan untuk menguji setiap *Statement* (baris kode) dengan menjalankan pengujian berdasarkan data uji yang sudah ditetapkan. (Yunisa, 2018)

Karakteristik pengujian yang baik menurut Kaner, Falk dan Nguyen (Presman, 2010) yaitu:

a. Pengujian memiliki probabilitas yang tinggi untuk menemukan kesalahan.
b. Pengujian tidak berulang-ulang atau setiap pengujian memiliki tujuan yang berbeda.
c. Pengujian "jenis terbaik", yaitu ketika melakukan pengujian serupa maka pengujian yang memiliki kemungkinan yang paling besar mengungkapkan kesalahan yang digunakan.
d. Pengujian yang tidak terlalu sederhana dan tidak terlalu kompleks.

Ada berbagai *framework* yang dapat digunakan untuk melakukan pengujian unit *test*ing pada aplikasi yang dikembangkan menggunakan bahasa pemrograman PHP, diantaranya adalah behat, selenium, codeception, atoum, PHPSpec, PHPUnit dan masih banyak *framework* lainnya. Dalam pengujian ini penulis memilih PHPUnit sebagai *framework* yang akan digunakan karena selain sebagai *framework* yang berorientasi *programer* untuk PHP, PHPUnit juga memiliki gagasan bahwa pengembang aplikasi harus menemukan kesalahan dalm proyek baru dengan cepat tanpa regresi kode pada bagian lain aplikasi.

2.4 PHPUnit

PHPUnit adalah *framework* yang dibuat oleh Sebastian Bergramn dan merupakan bagian dari xUnit. PHPUnit digunakan pada pengujian yang menggunakan bahasa pemrograman PHP. PHPUnit memiliki gagasan bahwa pengembang aplikasi menemukan kesalahan dalam kode yang baru dilakukan dengan menyatakan bahwa tidak ada regresi kode yang terjadi di basis

kode. PHPUnit dapat memberikan *output* dalam format yang berbeda termasuk Junit XML dan *Test*Dox. PHPUnit merupakan *tools unit testing* yang unggul dalam hal fungsionalitas, efisiensi, kehandalan dan portabilitas dibandingkan dengan *tools* lain seperti *Codeception* dan *SimpleTest* (Sandin, Yassin dan Mohamad 2016).

Pada umumnya seorang programer melakukan pengujian dengan cara membuat *class* terlebih dahulu kemudian melakukan *testing*, setelah itu baru digunakan. Namun *testing* menggunakan PHPUnit memiliki cara yang sedikit berbeda yaitu dengan menuliskan beberapa fungsi untuk melihat hasil sudah sesuai harapan atau tidak(Garry, 2010). Berikut langkah-langkah pengujian menggunakan PHPUnit:

1. Mendesain kelas/fungsi yang ingin diuji. Gambar 2.2 adalah contoh membuat kelas dengan nama Data*Test*

```php
<?php
use PHPUnit\Framework\TestCase;

class DataTest extends TestCase
{
```

Gambar 2.2 Membuat Kelas

2. Membuat rangkaian tes yang digunakan untuk mengecek semua fungsi di kelas yang telah dibuat sebelumnya. Dalam fungsi tes nilai yang diharapkan akan dibandingkan dengan hasil dari fungsi. Perintah *aseertSame* digunakan untuk mengisi parameter ($expected, $a +$b). Gambar 2.3 adalah contoh membuat tes fungsi.

```php
public function testAdd($a, $b, $expected)
{
    $this->assertSame($expected, $a + $b);
}
```

Gambar 2.3 Membuat Tes Fungsi

3. Mengimplementasikan kelas/fungsi dan mengembalikan objek yang telah diimplementasikan. Gambar 2.4 adalah contoh mengimplemantasikan kelas dan mengembalikan objek yang berbentuk array.

```php
<?php
use PHPUnit\Framework\TestCase;

class DataTest extends TestCase
{
    /**
     * @dataProvider additionProvider
     */
    public function testAdd($a, $b, $expected)
    {
        $this->assertSame($expected, $a + $b);
    }

    public function additionProvider()
    {
        return [
            [0, 0, 0],
            [0, 1, 1],
            [1, 0, 1],
            [1, 1, 3]
        ];
    }
}
```

Gambar 2.4 Mengimplemantasikan Kelas

4. Mengetes rangkaian tes apakah lulus atau gagal. Gambar 2.5 adalah contoh mengetes rangkaian tes pada Command Prompt. Terdapat empat tes yang diuji dan ditemukan satu tes gagal

```
$ phpunit DataTest
PHPUnit 7.0.0 by Sebastian Bergmann and contributors.

...F

Time: 0 seconds, Memory: 5.75Mb

There was 1 failure:

1) DataTest::testAdd with data set #3 (1, 1, 3)
Failed asserting that 2 is identical to 3.

/home/sb/DataTest.php:9

FAILURES!
Tests: 4, Assertions: 4, Failures: 1.
```

Gambar 2.5 Mengetes Rangkaian Tes

5. Memperbaiki kesalahan atau error dan kembali ke langkah 4 jika ditemukan kesalahan pada saat pengujian.

2.5 *Test case*

Test case atau kasus uji adalah satu komponen dokumentasi pengujian yang digunakan oleh penguji sebagai panduan untuk melakukan pengujian. Tetapi *test case* secara manual tidak dapat dijadikan jaminan apakah semua kebutuhan sistem telah terpenuhi dalam *test case* yang dibuat (Novelia, 2008). Adapun contoh pembuatan *test case* dapat yang umumnya digunakan dalam suatu pengujian dilihat pada Tabel 2.1

Tabel 2.1 Contoh Model *Test Case*

No	*Test case*	Input	Expected Output	Actual Output	Remarks

2.6 Penelitian Terdahulu

Penelitian ini tentu saja dilakukan tidak lepas dari hasil penelitian-penelitian sebelumnya yang pernah dilakukan sebagai bahan perbandingan dan kajian. Adapun penelitan yang dijadikan perbandingan tidak lepas dari topik penelitian yaitu pengujian perangkat lunak dengan *Unit testing* dengan PHPUnit.

Berikut beberapa contoh peneltian sejenis yang menggunakan pengujian dengan PHPUnit:

a. Pada penelitian sebelumnya (Laksito, 2019) terdapat permasalahan yaitu ditemukannya kendala dalam mengimplementasikan aplikasi pemesanan online kantin, dimana diperlukan koneksi internet untuk menjangkau seluruh pengguna sedangkan sistem sebelumnya berada di lingkungan lokal. Untuk mengatasi masalah tersebut, dalam penelitian ini dilakukan pengembangan aplikasi API Gateway menggunakan *framework* SlimPHP karena dianggap mempunyai kenggulan dalam kecepatan dan ringan saat dijalankan. Sedangkan pengujian unit menggunakan *framework* PHPUnit pada pengujian *white box* untuk melakukan uji kesesuaian algoritma dalam aplikasi API Gateway. Pemilihan *Framework* PHPUnit dilakukan karena dianggap sebagai tools *unit testing* yang unggul dalam hal fungsionalitas, efisiensi kehandalan dan portabilitas. Hasil yang didapatkan dari pengujian tersebut adalah API Gateway menggunakan *framework* slimPHP berhasil dimplementasikan sebagai penghubung antara API yang ada pada lokasi kantin dengan aplikasi kantin online dan setelah

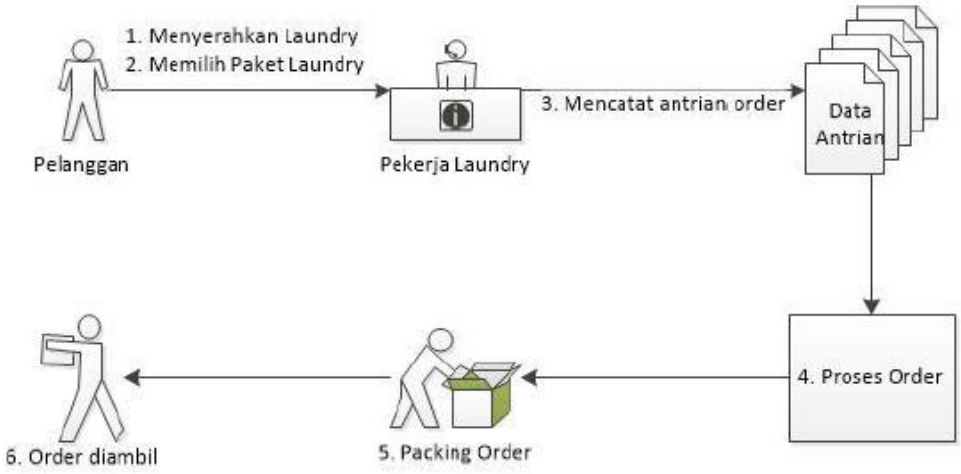
dilakukan pengujian tidak ditemukanya kesalahan atau *error* pada setiap pernyataan di unit tersebut.

b. Dalam sebuah penelitian yang dilakukan oleh Wijanarko dan Mulya (2018) terdapat masalah yaitu penyajian informasi sebuah sistem untuk mendukung pengolahan data. Sistem tersebut memiliki dua versi aplikasi yaitu web dan mobile sehingga harus dapat mentransfer data tanpa harus menunggu proses kerja dari masing-masing sistem. Masalah tersebut diselesaikan dengan membuat antarmuka pemrograman aplikasi sebagai layanan web untuk menjembatani komunikasi antar sistem tersebut. Dibangun dengan metode RESTful untuk berkomunikasi dengan sistem lain. Kemudian aplikasi tersebut diujikan dalam pengujian *white box* menggunakan PHPUnit. Hasil yang dihasilkan adalah sebuah aplikasi yang baik tanpa harus mengganggu dari program lama.

BAB III
ANALISIS KEBUTUHAN

3.1 Alur Proses

Pada penelitian sebelumnya telah dilakukan beberapa alur proses perancangan aplikasi *web mobile* untuk bisnis jasa laundry. Perancangan yang dilakukan berdasarkan metode User Centered Design (UDC) melalui tahapan yaitu melakukan *questioner* dan wawancara kepada pemilik/karyawan dan pelanggan laundry untuk dimintai pendapat mengenai aplikasi laundry dan perancangan sistem aplikasi laundry berbasis *web mobile* (Wulandari, 2017). Alur proses laundry secara umum dapat dilihat pada gambar 3.1.



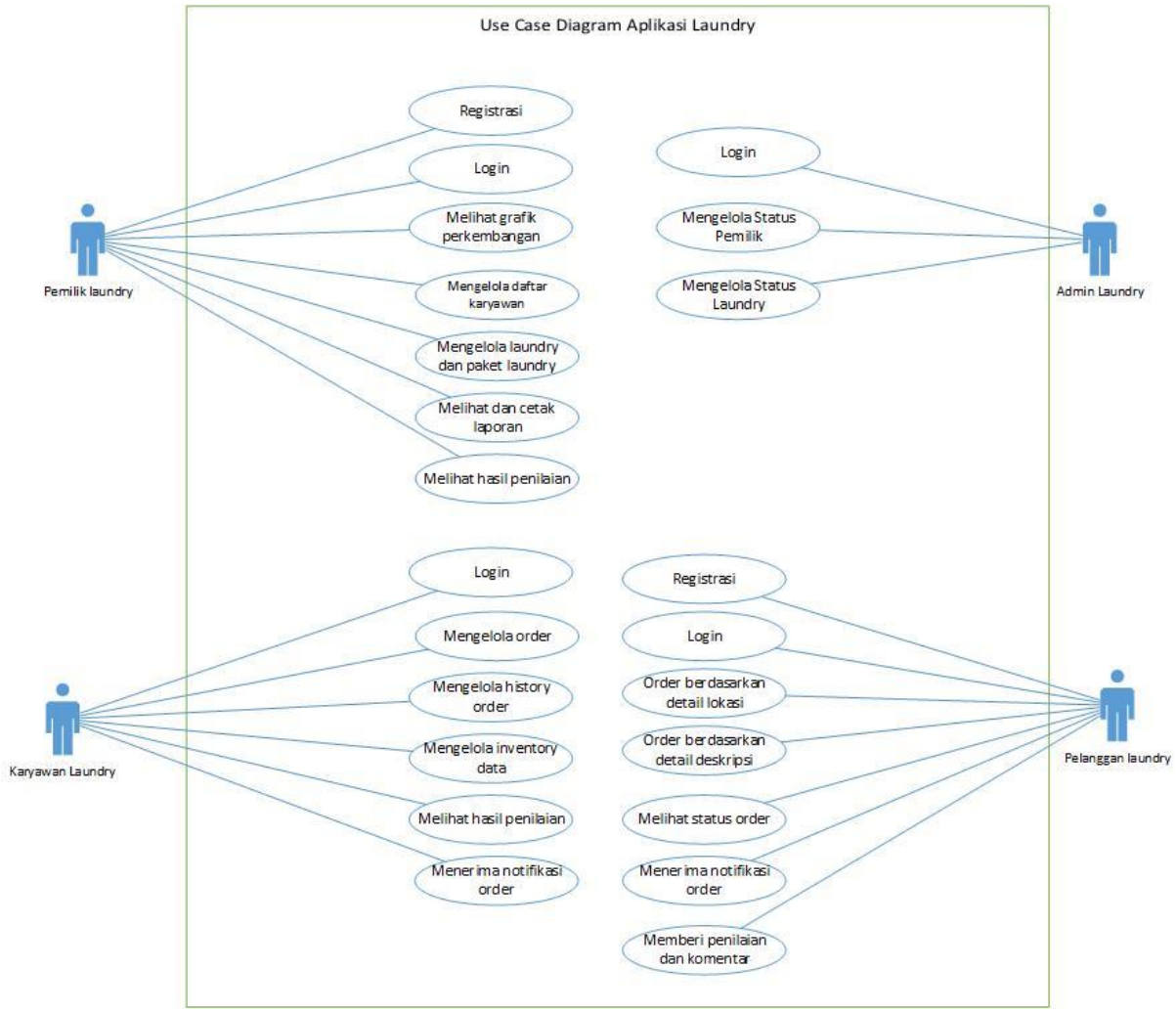
Gambar 3.1 Alur Proses Laundry (Wulandari, 2017)

Gambar 3.1 merupakan gambaran dari alur bisnis usaha laundry, berikut penjelasannya:

a. Pelanggan datang untuk mencucikan pakaian dan diterima oleh bagian penerimaan laundry.
b. Pelanggan memilih paket yang disediakan oleh jasa laundry.
c. Karyawan laundry mencatat order pelanggan dan memasukkan ke dalam antrian.
d. Karyawan laundry memproses order sesuai paket dan antrian.
e. Karyawan menyiapkan hasil order sampai selesai diproses.
f. Pelanggan mengambil order sesuai dengan nomor order.

3.2 *Use Case Diagram*

Use case diagram merupakan gambaran singkat hubungan antara satu atau lebih aktor dengan sistem yang akan dibuat. Dalam *use case* ini akan diketahui fungsi-fungsi apa saja yang ada pada sistem dan siapa saja yang dapat menggunakan fungsi tersebut. Gambar 3.2 merupakan gambar dari *use case diagram* dari aplikasi *web mobile* bisnis jasa laundry yang telah dilengkapi (Wulandari, 2017).



Gambar 3.2 *Use case* Diagram Aplikasi Laundry

3.3 *Questioner* dan Wawancara

Pada peneltian sebelumnya (Wulandari, 2017) telah dilakukan pengumpulan data dengan cara *questioner* dan wawancara yang dilakukan kepada beberapa pemilik/karyawan laundry dan pelanggan laundry untuk dimintai pendapat mengenai hal yang berkaitan dengan perancangan aplikasi *web mobile* untuk pengelolaan bisnis jasa laundry. Jumlah responden

adalah 40 orang yang terdiri dari 10 orang pemilik/karyawan laundry dan 30 orang pengguna laundry.

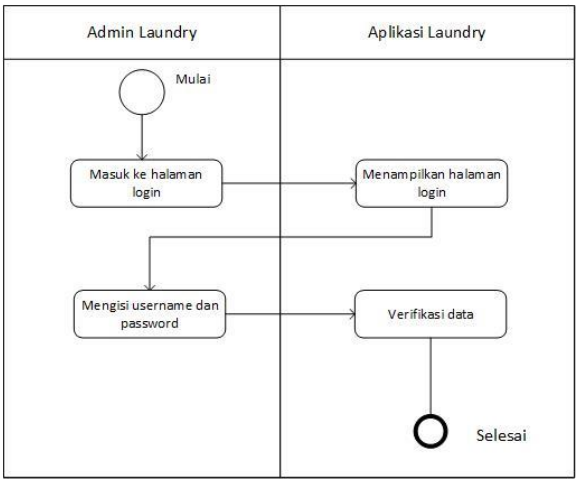
Berdasarkan hasil wawancara dan *questioner* maka dapat disimpulkan bahwa pemilik/karyawan laundry membutuhkan beberapa fitur seperti fitur pencatatan rincian order, laporan order, inventory data yang berisi data harga dan pewangi dan dilengkapi dengan fitur melihat penilaian dan saran/kritik dari para pelanggan laundry. Sedangkan untuk pelanggan laundry, fitur yang dibutuhkan adalah fitur rekomendasi pencarian laundry berdasarkan fasilitas tertentu, deskripsi laundry yang bersangkutan, melihat progress pesanan laundry, notifikasi jika laundry telah selesai dan dilengkapi dengan fitur penilaian dan saran/kritik.

Sedangkan untuk tampilan antarmuka, secara umum pemilik/karyawan laundry lebih memilih warna cerah yang digunakan pada aplikasi dan navigasi berada dibawah *header*. Berbeda dengan pemilik/karyawan laundry, secara umum pelanggan lebih memilih warna yang tidak terlalu cerah dan navigasi berada di paling atas. Untuk intruksi, desain, format, *feedback,* terminologi dan *learnability* menurut pemilik/karyawan dan pelanggan laundry dirasa sudah sesuai dengan rancangan karena sudah memenuhi kebutuhan aplikasi jasa laundry untuk pemilik/karyawan dan pelanggan laundry. Namun untuk ukuran font agar lebih besar sehingga dapat terbaca oleh pengguna yang berumur. Juga tampilan yang menarik dan *user friendly* agar mudah digunakan.

3.4 Desain Proses Bisnis

Analisis proses bisnis adalah kumpulan aktivitas yang saling terkait untuk menyelesaikan suatu masalah. Proses bisnis aplikasi *web mobile* akan dijelaskan dalam bentuk *Activity Diagram*. Pada penelitian sebelumnya, telah dilakukan penggambaran proses bisnis dengan membuat beberapa *Activity Diagram,* namun belum dilakukan penggambaran *diagram activity* untuk mencari laundry berdasarkan detail lokasi dan deskripsi laundry. Selain itu juga terdapat penggambaran *Activity Diagram* inventory data yang terpisah antara harga dan pewangi dan belum terdapat *use case diagram* login. Oleh karena itu, pada tahap ini penulis akan melengkapi *use case diagram* dan *Activity Diagram* dengan beberapa tambahan akivitas untuk melengkapi penelitian sebelumnya (Wulandari, 2017).

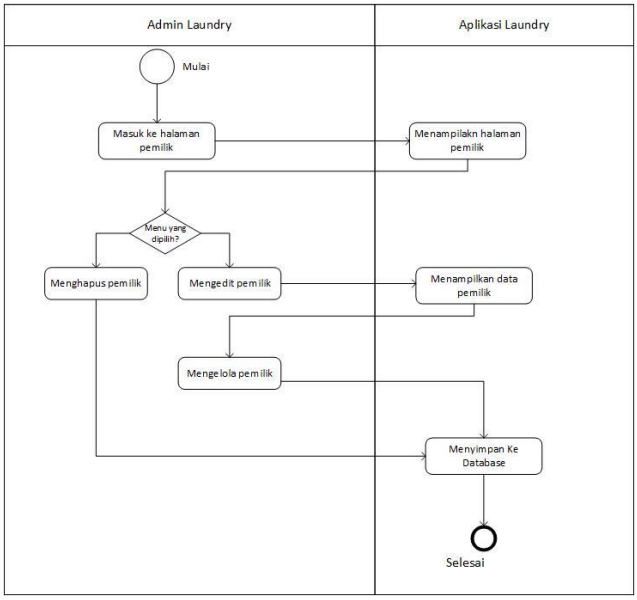
a. *Activity Diagram* Login Admin Laundry



Gambar 3.3 *Activity Diagram* Login Admin

Gambar 3.3 merupakan penjelasan menu login admin, di mana admin terlebih dahulu masuk ke halaman login kemudian mengisi username dan password yang telah terdaftar sebelumnya.

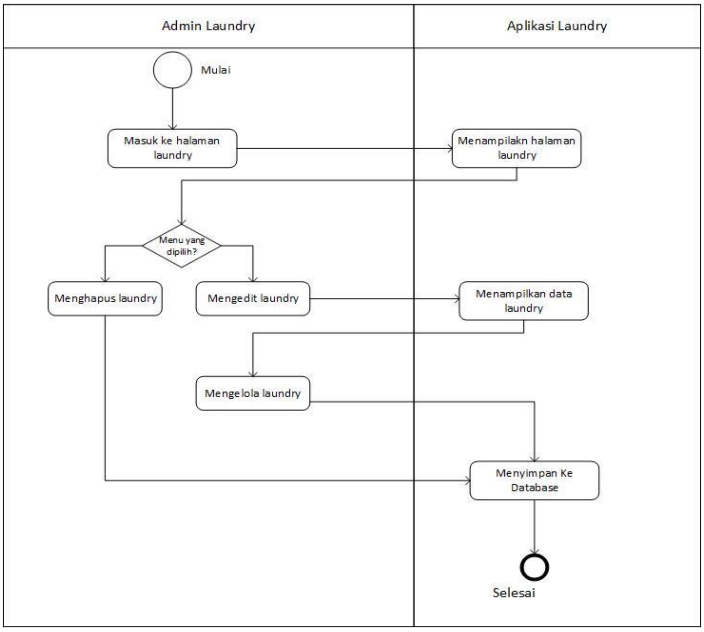
b. *Activity Diagram* Admin Mengelola Status Pemilik



Gambar 3.4 *Activity Diagram* Admin Mengelola Pemilik

Gambar 3.4 merupakan penjelasan menu mengelola pemilik, di mana admin dapat menghapus pemilik dan mengelola status pemilik dari aktif menjadi tidak aktif ataupun sebaliknya.

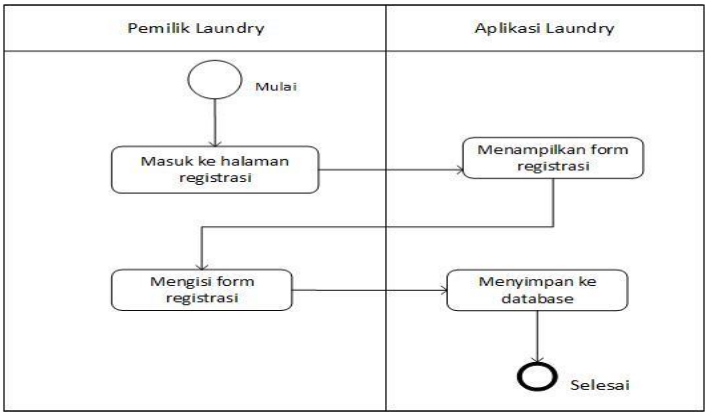
c. *Activity Diagram* Admin Mengelola Status Laundry



Gambar 3.5 *Activity Diagram* Admin Mengelola Laundry

Gambar 3.5 merupakan penjelasan menu mengelola pemilik, di mana admin dapat menghapus pemilik dan mengelola status pemilik dari aktif menjadi tidak aktif ataupun sebaliknya.

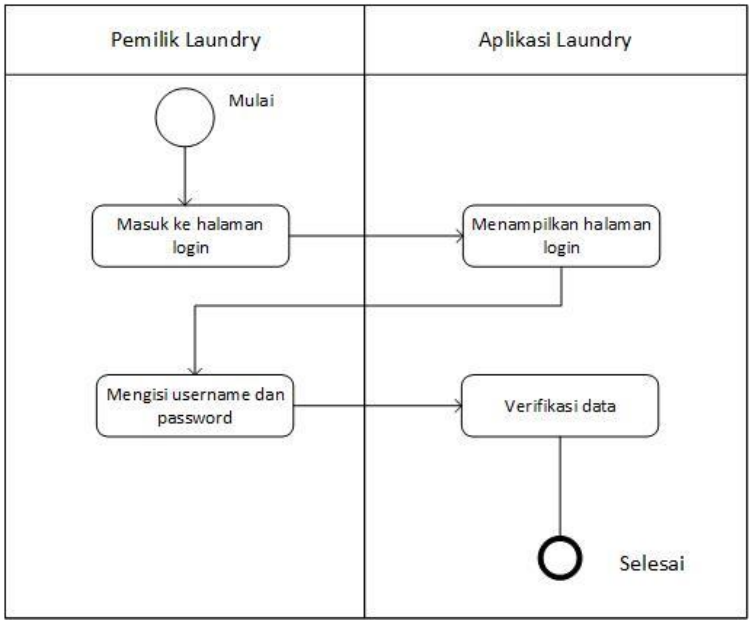
d. *Activity Diagram* Registrasi Pemilik dan Laundry



Gambar 3.6 *Activity Diagram* Registrasi Pemilik Laundry

Gambar 3.6 merupakan penjelasan menu registrasi pemilik. Pada tahap ini pemilik harus mendaftarkan diri dengan cara masuk ke halaman registrasi akun terlebih dahulu, kemudian mengisi form yang sudah ada dan data yang masuk akan disimpan ke *database*.

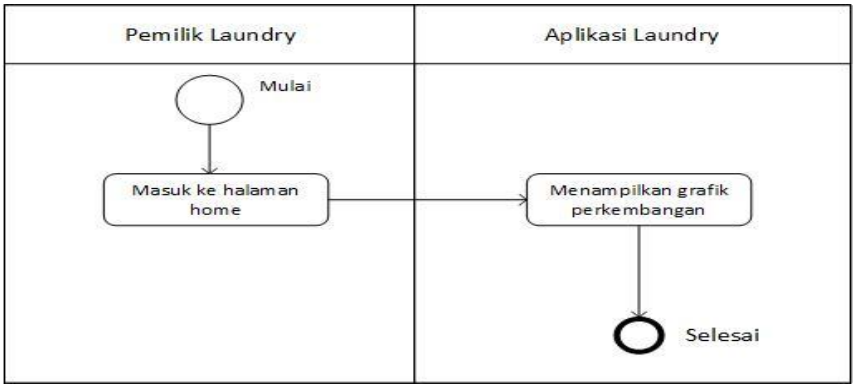
e. *Activity Diagram* Login Pemilik



Gambar 3.7 *Activity Diagram* Login Pemilik

Gambar 3.7 merupakan penjelasan menu login pemilik, di mana pemilik terlebih dahulu mengisi data email dan password yang telah terdaftar sebelumnya untuk dapat mengakses aplikasi laundry.

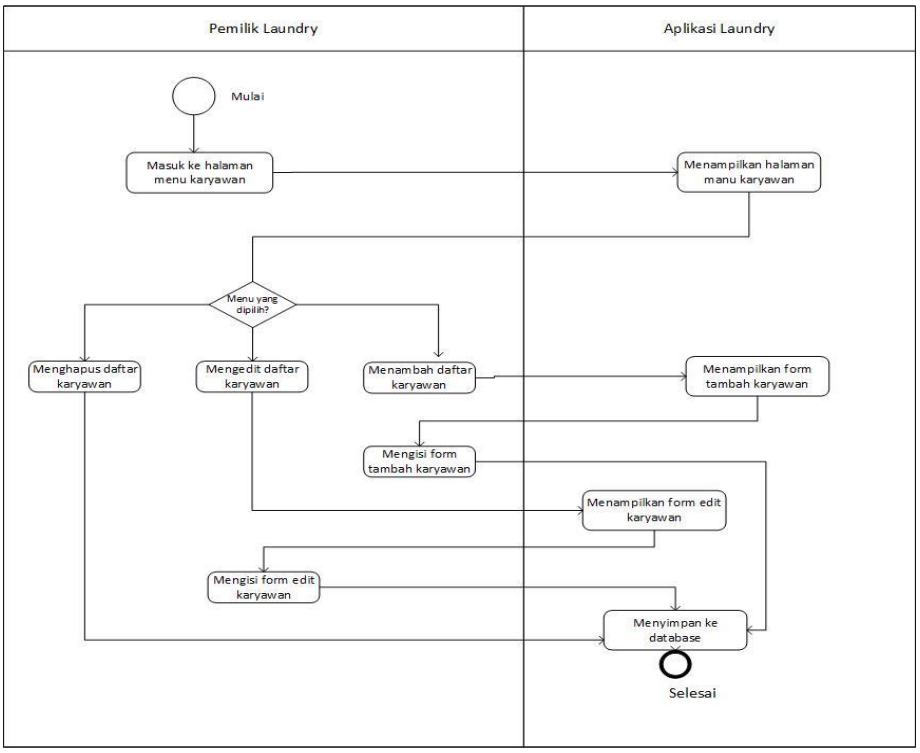
f. *Activity Diagram* Melihat Grafik Laundry



Gambar 3.8 *Activity Diagram* Melihat Grafik Laundry

Gambar 3.8 merupakan penjelasan menu pemilik, di mana pemilik dapat melihat grafik perkembangan laundry yang dimiliki dalam setahun dengan cara masuk ke halaman home pemilik.

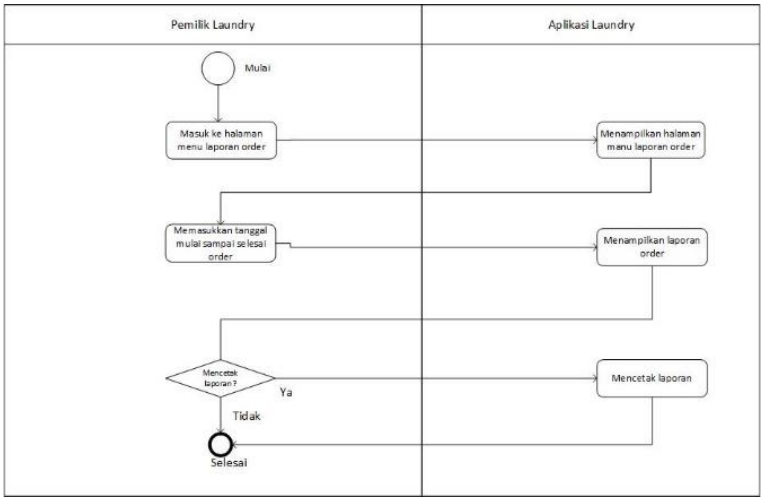
g. *Activity Diagram* Mengelola Daftar Karyawan



Gambar 3.9 *Activity Diagram* Mengelola Daftar Karyawan

Gambar 3.9 merupakan penjelasan menu pemilik, di mana pemilik dapat mengelola data karyawan seperti misalnya menghapus, edit dan menambahkan karyawan. Untuk menambahkan karyawan, pemilik harus masuk ke halaman menu karyawan, setelah ditampilakan halaman menu karyawan, pemilik memilih tambah daftar karyawan, kemudian akan muncul form dan pemilik harus mengisi form tersebut terlebih dahulu. Kemudian data form akan disimpan di *database*.

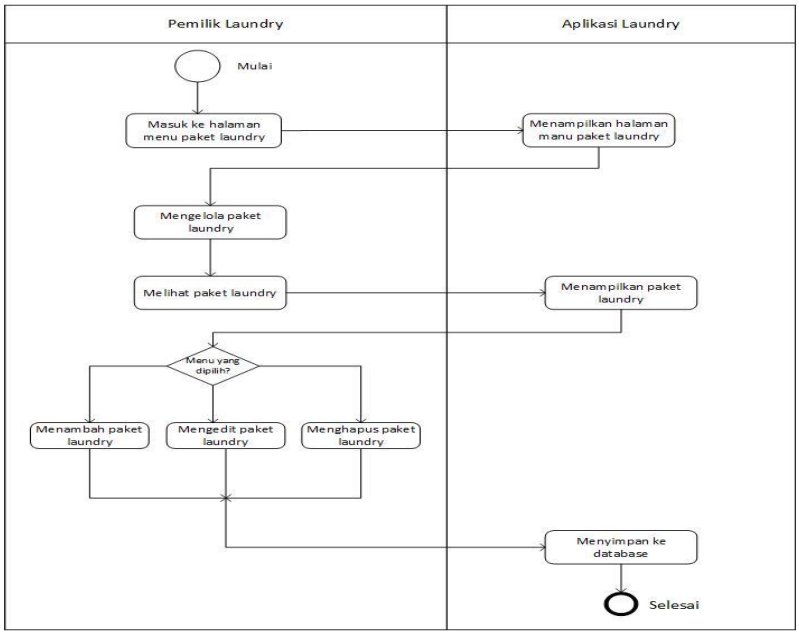
h. *Activity Diagram* Melihat dan Mencetak Laporan



Gambar 3.10 *Activity Diagram* Melihat dan Mencetak Laporan

Gambar 3.10 merupakan penjelasan menu pemilik, di mana pemilik dapat melihat dan mencetak laporan order. Pemilik terlebih dahulu masuk ke halaman menu laporan order kemudian mengisi tanggal mulai sampai selesai order. Setelah mengisi data, maka aplikasi laundry akan menampilkan laporan dan pemilik dapat mencetaknya.

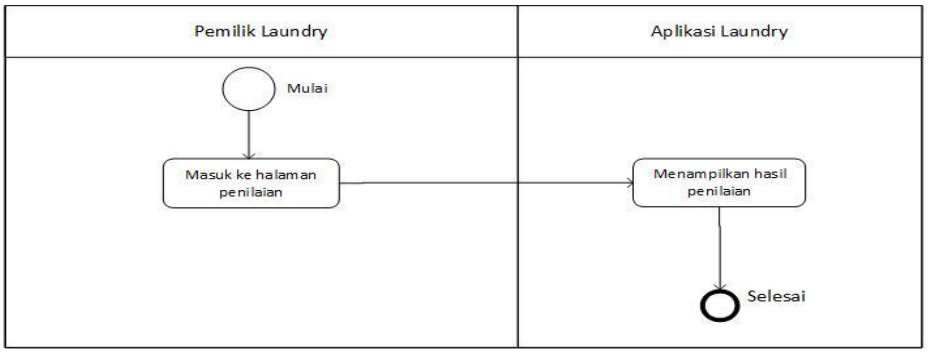
i. *Activity Diagram* Mengelola Paket Laundry



Gambar 3.11 *Activity Diagram* Mengelola Paket Laundry

Gambar 3.11 merupakan penjelasan menu pemilik, di mana pemilik menu mengelola paket laundry. Pemilik masuk ke halaman menu paket laundry, kemudian aplikasi menampilkan menu paket laundry. Pemilik dapat memilih menu menambahkan paket, mengedit paket atau menghapus paket laundy. Kemudian data akan disimpan ke *database*.

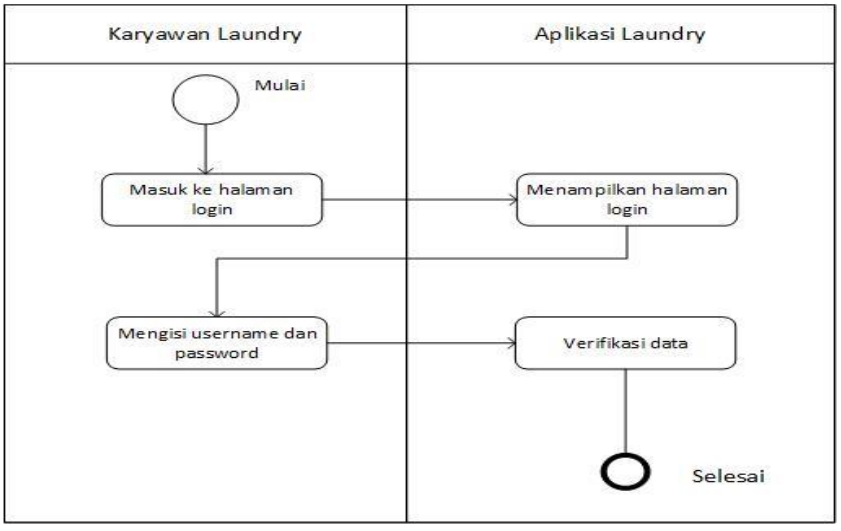
j. *Activity Diagram* Melihat Penilaian



Gambar 3.12 *Activity Diagram* Melihat Penilaian

Gambar 3.12 merupakan penjelasan menu pemilik, di mana pemilik dapat melihat penilaian dari pelanggan dengan cara masuk ke halaman penilaian, kemudian aplikasi laundry akan menampilan penilaian rata-rata dari pelanggan.

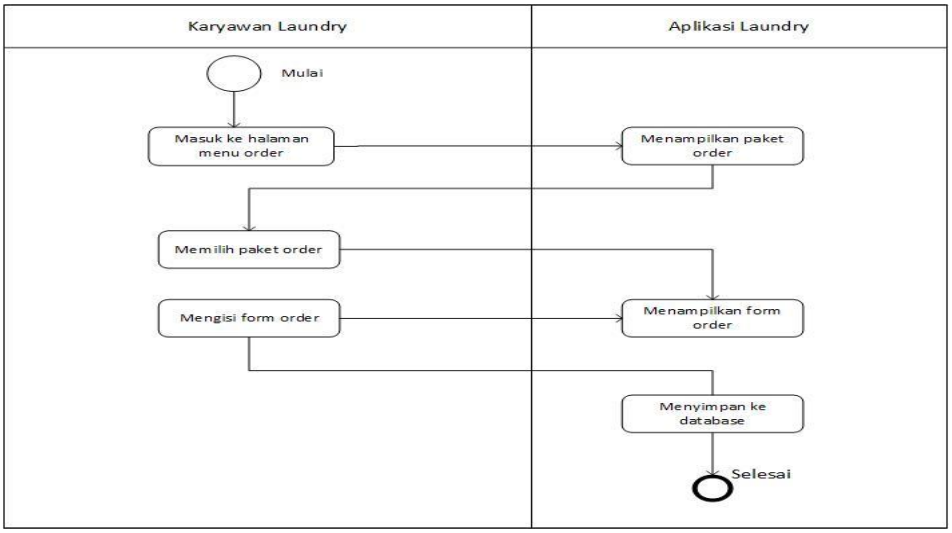
k. *Activity Diagram* Login Karyawan



Gambar 3.13 *Activity Diagram* Login Karyawan

Gambar 3.13 merupakan penejlasan menu login karyawan, di mana karyawan terlebih dahulu mengisi data email dan password yang telah terdaftar untuk dapat mengakses aplikasi laundry.

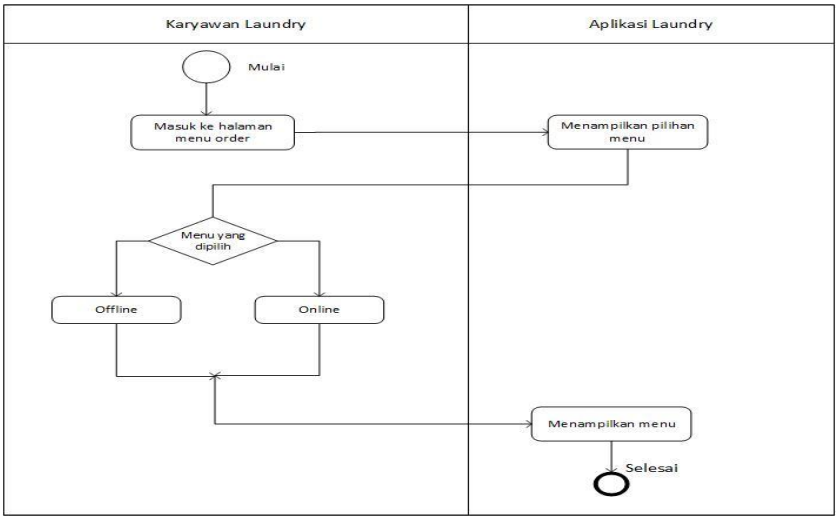
l. *Activity Diagram* Mengelola Order



Gambar 3.14 *Activity Diagram* Mengelola Order

Gambar 3.14 merupakan penjelasan menu karyawan, di mana karyawan dapat mengelola order. Pertama karyawan harus masuk ke halaman menu order kemudian memilih jenis paket dan mengisi form order. Kemudian data akan disimpan ke *database*.

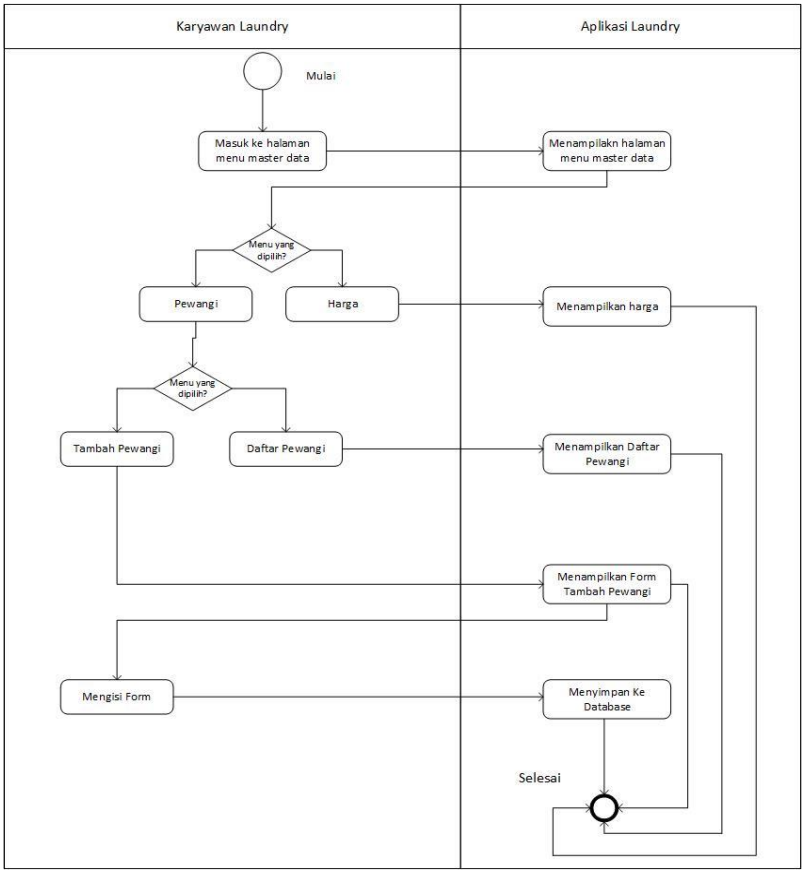
m. *Activity Diagram* History Order



Gambar 3.15 *Activity Diagram* History Order

Gambar 3.15 merupakan penjelasan menu history order. Karyawan dapat memilih jenis order online atau offline untuk melihat daftar order yang ada.

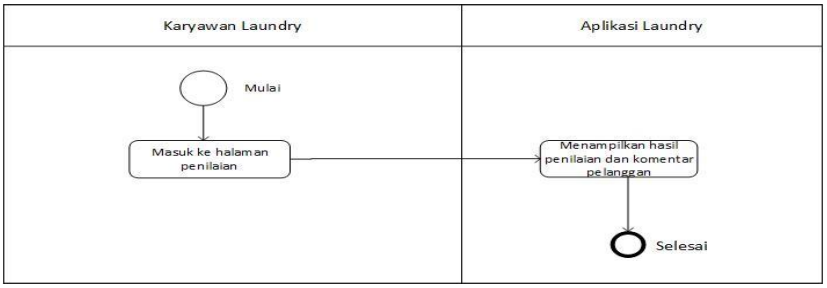
n. *Activity Diagram* Mengelola Inventory Data



Gambar 3.16 *Activity Diagram* Mengunduh Laporan

Gambar 3.16 merupakan penjelasan menu karyawan, di mana karyawan dapat mengelola inventory data dengan cara masuk ke halaman master data terlebih dahulu. Kemudian memilih menu pewangi atau menu harga. Pada menu harga, karyawan dapat menampilkan daftar harga. Sedangkan pada menu pewangi, karyawan dapat melihat daftar pewangi dan juga menambahkan daftar pewangi. Kemudian semua akan disimpan ke *database*.

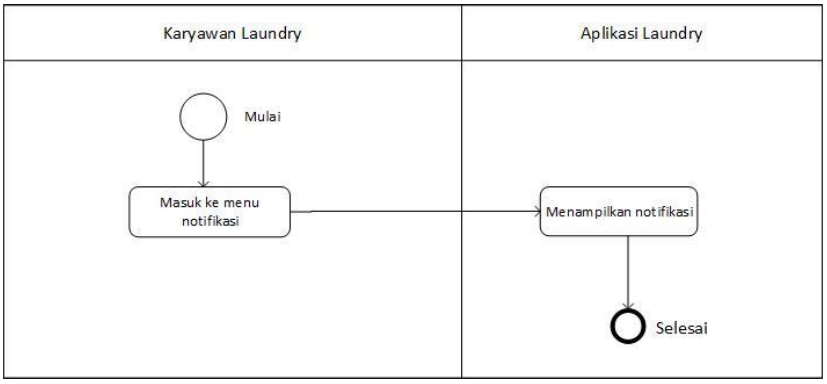
o. *Activity Diagram* Melihat Penilaian dan Komentar



Gambar 3.17 *Activity Diagram* Melihat Penilaian

Gambar 3.17 merupakan penjelasan menu penilaian dan komentar dari pelanggan. Pertama karyawan masuk ke menu penilaian kemudian aplikasi akan menampilkan penilaian dan komentar dari pelanggan.

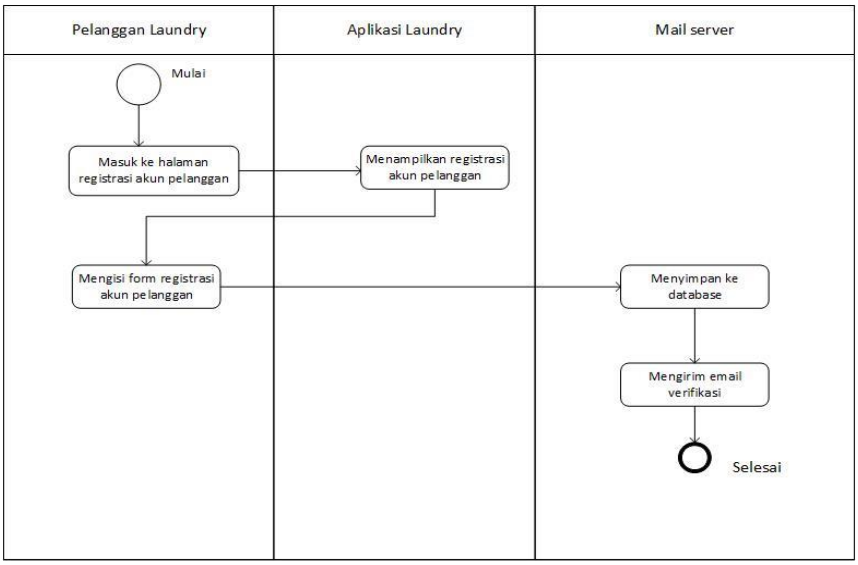
p. *Activity Diagram* Menerima Notifikasi



Gambar 3.18 *Activity Diagram* Melihat Notifikasi

Gambar 3.18 merupakan penjelasan pada menu karyawan, di mana karyawan dapat menerima notifikasi dari pelanggan jika terdapat orderan masuk. Karyawan masuk ke menu notifikasi, kemudian aplikasi akan menampilkan notifikasi yang masuk.

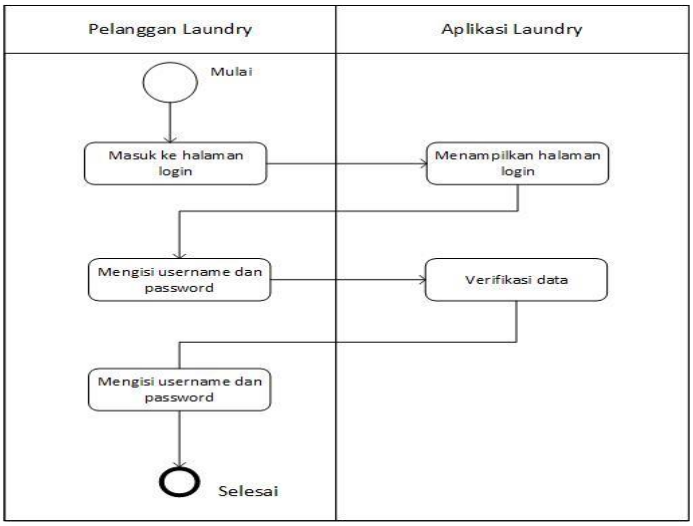
q. *Activity Diagram* Registrasi Pelanggan



Gambar 3.19 *Activity Diagram* Registrasi Pelanggan

Gambar 3.19 merupakan penjelasan menu registrasi pelanggan. Pelanggan terlebih dahulu mendaftarkan diri dengan cara masuk ke halaman registrasi pelanggan. Kemudian pelanggan mengisi form registrasi dan data akan disimpan di *database*.

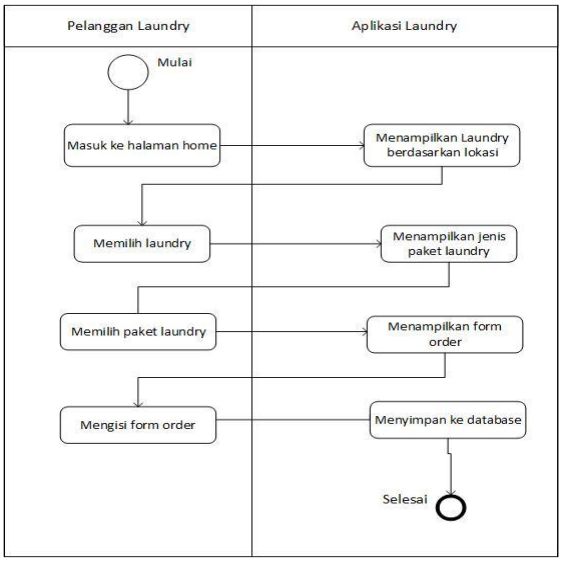
r. *Activity Diagram* Login



Gambar 3.20 *Activity Diagram* Login Pelanggan

Gambar 3.20 merupakan penjelasan menu login pada pelanggan. Untuk masuk ke aplikasi laundry, pelanggan terlebih dahulu login dengan mengisi data email dan password yang telah terdaftar sebelumnya.

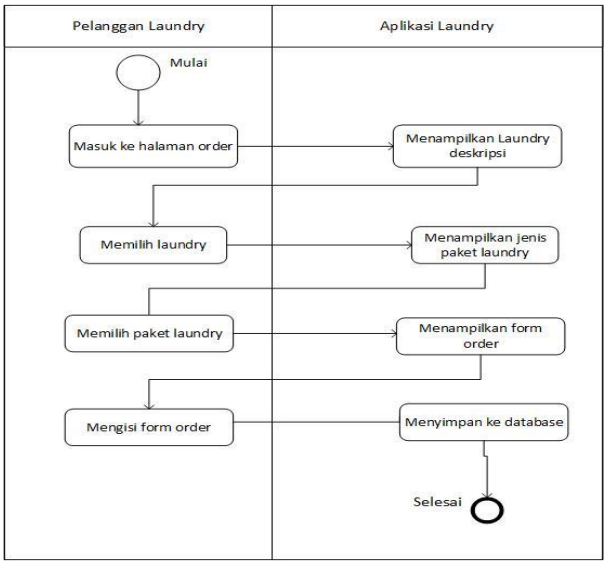
s. *Activity Diagram* Order Laundry Berdasarkan Detail Lokasi



Gambar 3.21 *Activity Diagram* Order Laundry Berdasarkan Detail Lokasi

Gambar 3.21 merupakan penjelasan menu order laundry bagian pelanggan berdasarkan detail jarak. Pelanggan masuk ke halaman home dan aplikasi akan menampilkan daftar laundry dengan detail lokasi yang didukung dengan tambahan peta lokasi. Pelanggan memilih laundry dan paket laundry yang diinginkan. Kemudian pelanggan mengisi data order dan data akan terkirim ke karyawan.

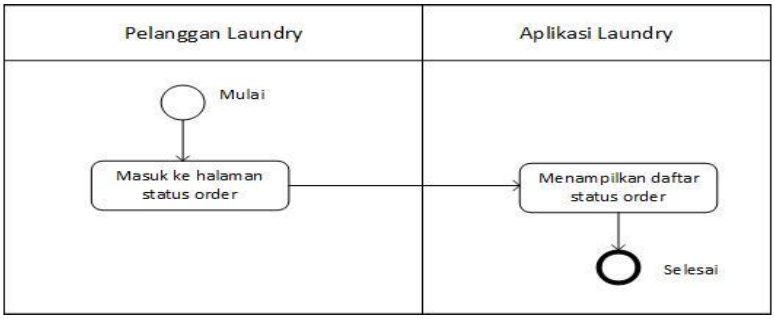
t. *Activity Diagram* Order Luandry Berdasarkan Detail Deskripsi



Gambar 3.22 *Activity Diagram* Mencari Laundry Berdasarkan Detail Deskripsi

Gambar 3.22 merupakan penjelasan menu order laundry bagian pelanggan berdasarkan detail deskripsi. Pelanggan masuk ke halaman order dan aplikasi akan menampilkan daftar laundry beserta diskripsinya. Pelanggan memilih laundry dan jenis paket kemudian mengisi form dan data akan dikirim ke karyawan

u. *Activity Diagram* Melihat Status Order



Gambar 3.23 *Activity Diagram* Melihat Status Order

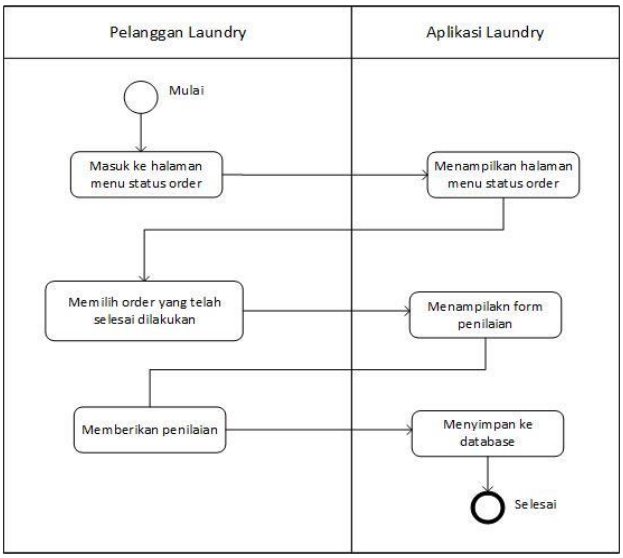
Gambar 3.23 merupakan penjelasan menu pelanggan untuk melihat status order. Pelanggan dapat melihat status order dengan cara masuk ke halaman status oder, kemudian apliaksi akan menampilkan status dan sejarah order.

v. *Activity Diagram* Menerima Notifikasi

Gambar 3.24 *Activity Diagram* Menerima Notifikasi

Gambar 3.24 merupakan penjelasan menu menerima notifikasi bagian pelanggan. Pelanggan dapat menerima notifikasi dari laundry jika status order telah selesai dengan cara melihat menu pemberitahuan.

w. *Activity Diagram* Memberi Penilaian terhadap Lauundry



Gambar 3.25 *Activity Diagram* Memberi Penilaian terhadap Lauundry

Gambar 3.25 merupakan penjelasan menu memberi penilaian. Pelanggan dapat memberi peneliaian terhadap laundry yang telah selesai diproses. Pelanggan terlebih dahulu masuk ke halaman menu status order dan aplikasi akan menampilkan menu status order. Kemudian pelanggan memilih order yang telah dilakukan sebelumnya untuk memberikan penilaian dan melakukan penilaian. Penilaian tersebut akan disimpan ke *database* dan dapat dilihat oleh karyawan.

3.5 Skenario Test

Skenario merupakan cerita pendek yang akan menggambarkan kemungkinan yang akan terjadi pada saat menjalankan aplikasi laundry. Melalui skenario pengembang aplikasi, kita dapat mengetahui bagaimana suatu aplikasi yang telah dibuat dapat membantu para pengguna. Skenario ini dibuat berdasarkan *use case* yang sudah ada pada penelitian sebelumnya (Wulandari, 2017) yang kemudian telah dilengkapi oleh penulis. Terdapat beberapa kemungkinan yang akan terjadi ketika menggunakan aplikasi laundry ini. Skenario ini terjadi pada beberapa aktor yaitu admin, pemilik laundry, karyawan laundry dan pengguna laundry.

a. Skenario pada admin laundry:

   Admin terlebih dahulu login dengan username dan password. Kemudian admin dapat mengelola pemilik laundry dan juga laundry yang terdaftar seperti mengubah status laundry yang aktif menjadi tidak aktif ataupun sebaliknya.

b. Skenario pada user pemilik laundry:

Pemilik terlebih dahulu mendaftarkan akun dengan cara mengisi form regitsrasi laundry. Pemilik dapat mendaftarkan laundrynya setelah mengisi form registrasi. Pemilik dapat mengelola daftar karyawan, melihat dan mencetak laporan order, mengelola paket laundry ataupun menambah daftar laundry dan melihat saran dan kritik dari pelanggan.

c. Skenario pada user karyawan laundry:

Karyawan laundry dapat mengelola inventory data seperti data harga dan data pewangi. Untuk data pewangi, karyawan dapat menambah, mengedit, menghapus dan melihat data pewangi. Karyawan laundry dapat melakukan pengelolaan rincian order, seperti menambahkan rincian order, mengedit status order dan melihat status order (masih dalam proses atau sudah selesai). Selain itu juga dapat melihat saran /kritik dari pelanggan.

d. Skenario pada user pelanggan laundry:

Pelanggan yang ingin menggunakan jasa laundry terlebih dahulu registrasi dan login sebagai akun pelanggan. Pelanggan dapat melihat deskripsi laundry meliputi kelebihan laundry atau promo yang ada, mencari daftar laundry yang diinginkan berdasarkan kategori tertentu seperti fasilitas, lokasi terdekat, harga, melihat status order, menerima notifikasi, melihat status order apakah sudah selesai atau masih dalam proses dan. Selain itu pelanggan dapat memberikan penilaian/rating juga dapat memberikan saran /kritik.

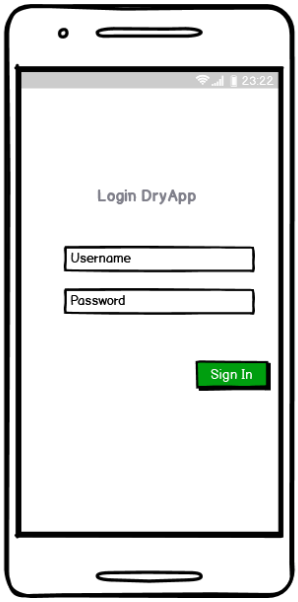
3.6 Desain Implementasi

Desain implementasi digunakan untuk menggambarkan desain tampilan aplikasi yang akan dibuat. Pada penelitian sebelumnya telah dilakukan desain implemantasi berdasarkan hasil dari wawancara dan *questioner* yang telah dilakukan pada pemilik/karyawan laundry dan pelanggan laundry. Pada tahap ini penulis akan melengkapi desain implementasi berdasarkan UCD yang telah dilengkapi.

3.6.1 Desain pada Admin Laundry

a. Tampilan Login Admin

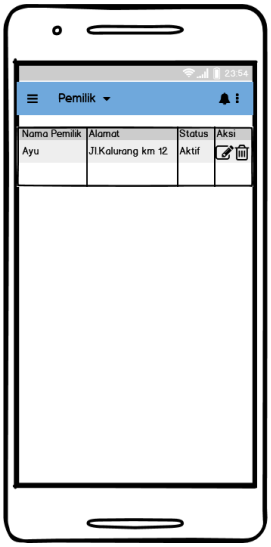
Menu login digunakan ketika admin ingin masuk ke aplikasi laundry dengan memasukkan username dan password yang telah terdaftar. Gambar 3.26 adalah tampilan login pemilik laundry.



Gambar 3.26 Tampilan Login Admin

b. Tampilan Mengelola Status Pemilik

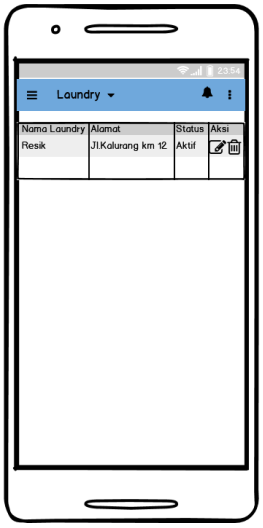
Menu edit status pemilik adalah menu yang digunakan admin ketika akan mengubah status pemilik yang aktif menjadi tidak aktif ataupun sebaliknya. Gambar 3.27 merupakan tampilan edit status pemilik.



Gambar 3.27 Tampilan Edit Status Pemilik

c. Tampilan Edit Status Laundry

Menu edit status laundry adalah menu yang digunakan admin ketika akan mengubah status laundry yang aktif menjadi tidak aktif ataupun sebaliknya. Gambar 3.28 merupakan tampilan edit status laundry.

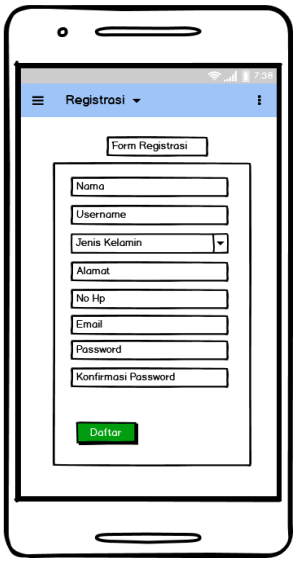


Gambar 3.28 Tampilan Edit Status Laundry

3.6.2 Desain pada Pemilik Laundry

a. Tampilan Registrasi Pemilik dan Laundry

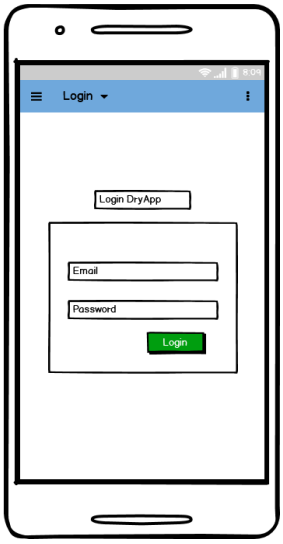
Registrasi pemilik digunakan untuk mendaftrakan akun pemilik sekaligus pemilik dapat mendaftarkan akun laundry dengan mengisi beberapa data. Gambar 3. 29 merupakan tampilan registrasi pemilik dan laundry.



Gambar 3.29 Tampilan Registrasi Pemilik dan Laundry

b. Tampilan Login

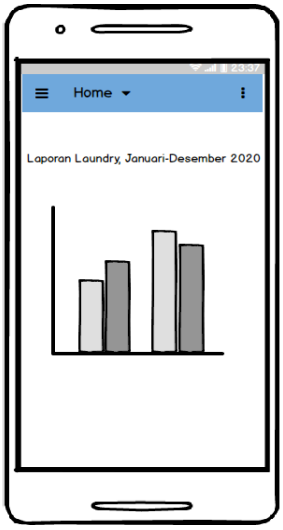
Menu login digunakan ketika user ingin masuk ke aplikasi laundry dengan memasukkan email dan password yang telah terdaftar sebelumnya. Gambar 3. 30 merupakan tampilan login pemilik laundry.



Gambar 3.30 Tampilan Login

c. Tampilan Melihat Grafik Perkembangan Laundry

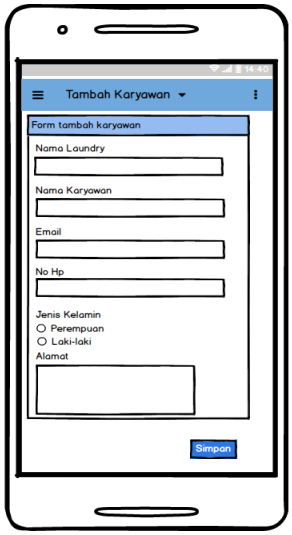
Grafik perkembangan laundry digunakan untuk melihat bagaimana perkembangan laundry selama setahun. adalah tampilan melihat grafik perkembangan laundry. Gambar 3.31 merupakan tampilan grafik perkembangan laundry



Gambar 3.31 Tampilan Melihat Grafik Perkembangan Laundry

d. Tampilan Mengelola Daftar Karyawan

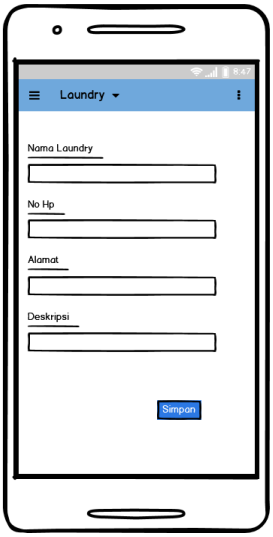
Mengelola daftar karyawan adalah menu yang digunakan pemilik untuk menghapus, edit dan menambah daftar karyawan. Gambar 3. 32 merupakan tampilan mengelola daftar karyawan.



Gambar 3.32 Tampilan Mengelola Daftar Karyawan

e. Tampilan Mengelola Laundry dan Paket Laundry

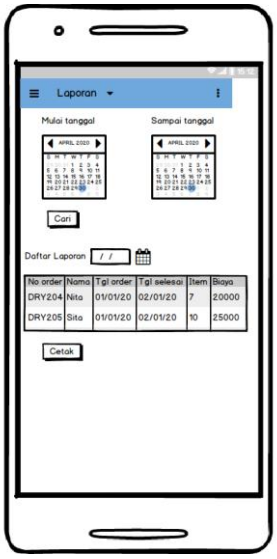
Pada menu ini, pemilik dapat menambah, mengedit dan menghapus daftar laundry dan paket laundry. Gambar 3. 33 merupakan tampilan mengelola laundry



Gambar 3.33 Tampilan Mengelola Laundry

f. Tampilan Melihat dan Cetak Laporan

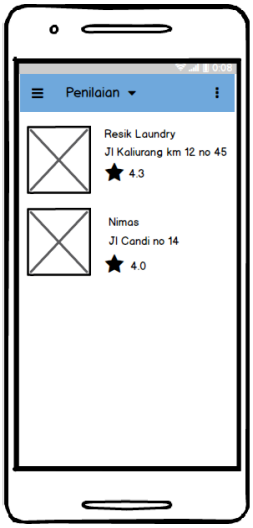
Pada menu ini, pemilik dapat melihat dan mencetak laporan order berdasarkan rentang waktu yang dapat ditentukan. Kemudian ketika hasil laporan muncul maka pemilik dapat mencetaknya. Gambar 3.34 merupakan tampilan melihat dan cetak laporan order.



Gambar 3.34 Tampilan Melihat dan Cetak Laporan

g. Tampilan Melihat Penilaian Pelanggan

Pada menu ini, pemilik dapat melihat hasil kalkulasi penilaian yang telah diberikan oleh pelanggan terhadap laundry yang dimiliki. Gambar 3.35 merupakan tampilan penilaian pelanggan

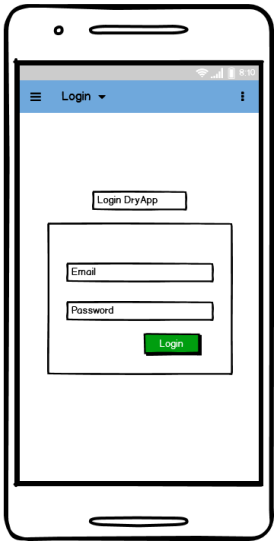


Gambar 3.35 Tampilan Melihat Penilaian Pelanggan

3.6.3 Desain pada Karyawan Laundry

a. Tampilan Login

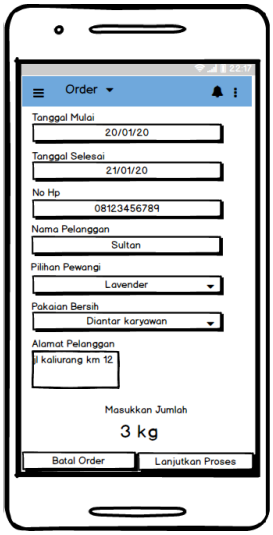
Menu login digunakan ketika karyawan ingin masuk ke aplikasi laundry dengan cara memasukkan email dan password yan telah terdaftar. Gambar 3.36 merupakan tampilan login karyawan laundry.



Gambar 3.36 Tampilan Login

b. Tampilan Mengelola Order

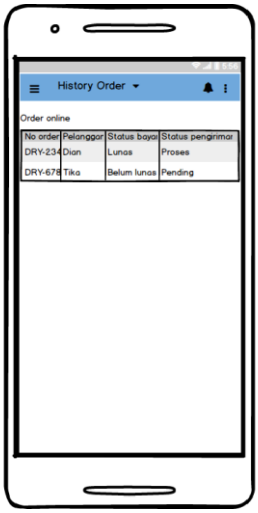
Pada menu ini, karyawan dapat mengelola order seperti memilih paket dan memasukkan data pelanggan yang akan melakukan order. Gambar 3.37 merupakan tampilan mengelola order.



Gambar 3.37 Mengelola Order

c. Tampilan History Order

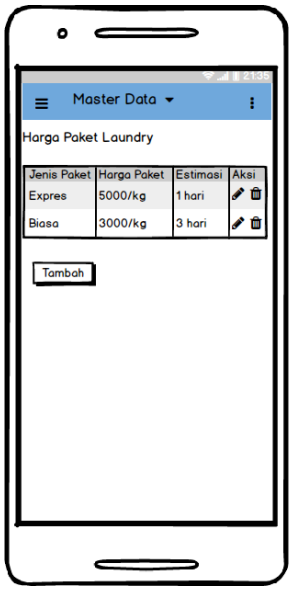
History order adalah menu yang digunakan ketika karyawan akan melihat atau mengubah status order pelanggan seperti status laundry pending, proses atau selesai. Gambar 3.38 merupakan tampilan history order.



Gambar 3.38 Tampilan History Order

d. Tampilan Mengelola Inventory Data

Pada menu ini, karyawan dapat menambah, mengedit dan menghapus paket laundry dan juga pewangi laundry. Gambar 3. 39 merupakan tampilan mengelola inventory data.



Gambar 3.39 Mengelola Inventory Data

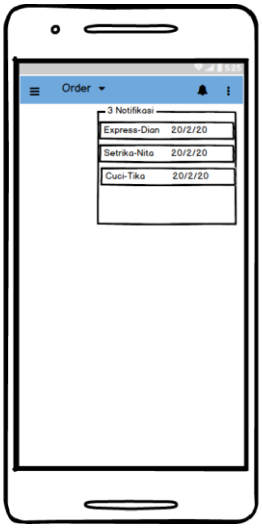
e. Tampilan Melihat Penilaian

Pada menu ini, karyawan dapat melihat penilaian dan kritik dari pelanggan yang telah selesai melakukan order dan memberikan penilaian. Gambar 3.40 merupakan tampilan melihat penilaian.

Gambar 3.40 Tampilan Mengelola Inventory Data

f. Tampilan Menerima Notifikasi Order

Pada menu ini, karyawan dapat melihat notifikasi yang masuk ketika ada pelanggan yang order laundry. Notifikasi akan hilang ketika laundry sudah mulai diproses. Gambar 3.41 merupakan tampilan menerima notifikasi order.

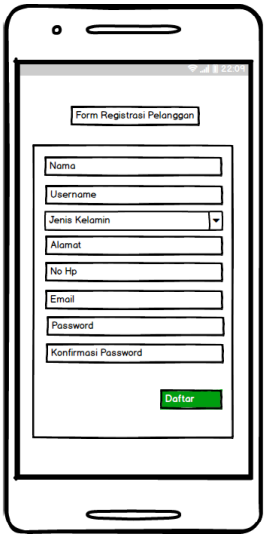


Gambar 3.41 Tampilan Menerima Notifikasi Order

3.6.4 Desain pada Pelanggan Laundry

a. Tampilan Registrasi

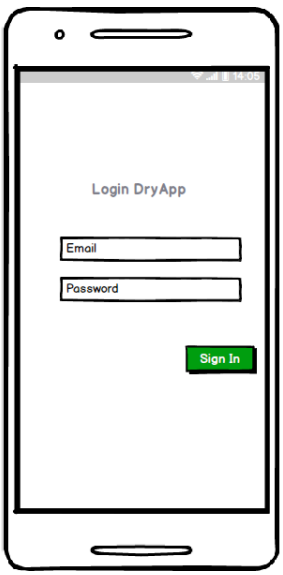
Menu registrasi digunakan ketika pelanggan akan mendaftarkan diri sebagai pelanggan laundry dengan mengisi beberapa data. Gambar 3.42 merupakan tampilan menerima notifikasi odrer.



Gambar 3.42 Tampilan Registrasi

b. Tampilan Login

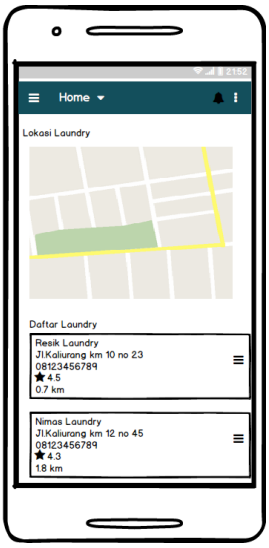
Menu login digunakan ketika pelanggan ingin masuk ke aplikasi laundry dengan cara mengisi email dan password yang telah terdaftar sebelumnya. Gambar 3.43 merupakan tampilan login pelanggan laundry.



Gambar 3.43 Tampilan Login

c. Tampilan Order Berdasarkan Detail Lokasi

Pada menu ini, pelanggan dapat melakukan order dengan melihat letak lokasi laundry yang dilengkapi dengan maps. Gambar 3.44 merupakan tampilan order berdasarkan detail lokasi.



Gambar 3.44 Tampilan Order Berdasarlan Detail Lokasi

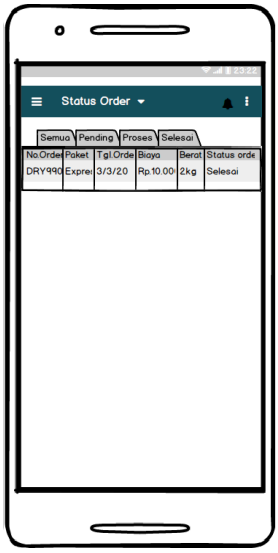
d. Tampilan Order Berdasarkan Detail Deskripsi

Pada menu ini, pelanggan dpaat melaukan order dengan melihat detail diskripsi dari setiap laundry. Selain itu pelanggan juga dapat langsung order melalui halaman ini. Gambar 3.45 merupakan tampilan order berdasarkan detail deskripsi.

Gambar 3.45 Tampilan Order Berdasarkan Detail Deskripsi

e. Tampilan Melihat Status Order

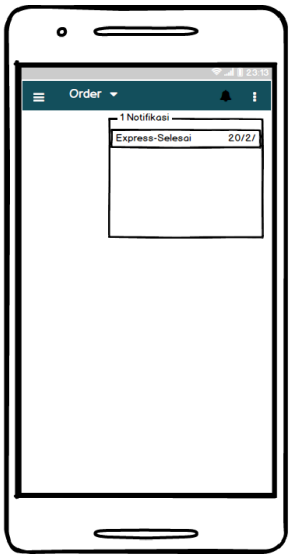
Pada menu ini, pelanggan dapat melihat status order sudah sampai mana proses laundry yang diorder atau melihat berdasarkan status proses. Selain itu pelanggan juga dapat melihat detail order. Gambar 3.46 merupakan tampilan melihat status order.



Gambar 3.46 Tampilan Melihat Status

f. Tampilan Melihat Notifikasi Order

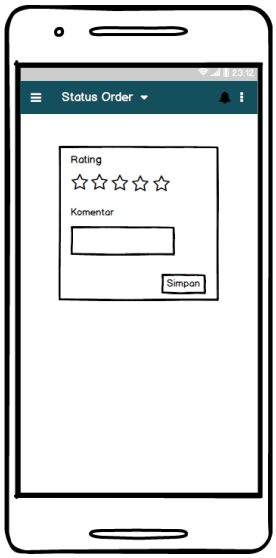
Pada menu ini pelanggan dapat menerima dan melihat notifiksi jika laundry yang diorder telah selesai. Gambar 3.47 merupakan tampilan melihat notifikasi order.



Gambar 3.47 Tampilan Notifikasi order

g. Tampilan Memberi Penilaian dan Saran/Kritik

Pada menu ini pelanggan dapat memberikan penilaian dan komentar jika order telah selesai. Gambar 3.48 merupakan tampilan memberi penilaian dan saran/kritik.



Gambar 3.48 Tampilan Memberi Penilaian dan Saran/Kritik

3.7 Rencana Pengujian

Rencana pengujian dalam penelitian ini adalah akan dilakukan pengujian unit menggunakan PHPUnit. PHPUnit dipilih karena aplikasi ini dikembangkan menggunakan bahasa pemrograman PHP. Kemudian pengujian dilakukan menggunakan metode *white box testing* karena pengujian dilakukan bertahap dan melihat alur logika dengan menghitung jumlah *statement* dengan cara melihat *source code* pada sistem yang dibuat. Teknik *statement coverage* dipilih sebagai pengukur tingkat keberhasilan pengujian. Teknik ini dilakukan dengan menjalankan data uji yang mencakup semua *statement* (*source code*) yang dijalankan agar mencapai nilai 100% pada *statement*.

3.8 *Unit Testing* dengan *White Box Testing*

Pada tahap ini pengujian dilakukan berdasarkan dari segi logika ataupun fungsi-fungsi yang ada pada sistem. Teknik *white box* dengan cara melihat *source code* yang ada pada sistem dan menentukan fungsi pada *use case* yang memiliki fungsi if untuk memeriksa sebuah kondisi telah dieksekusi dengan tepat sehingga menghasilkan keluaran yang valid. *Use case* dan fungsi

yang terdapat dalam dalam setiap kelas kemudian dikonversikan dalam bentuk *flowgraph.* Teknik yang akan digunakan dalam pengujian *white box* ini adalah *statement coverage*.

Tahapan yang dilakukan dalam pengujian ini adalah (Yunisa, 2018):

a. Menentukan atau memilih fungsi dalam kelas-kelas yang tersedia
b. Menghitung jumlah *statement* (baris kode) dalam setiap fungsi
c. Menghitung jumlah kondisi dalam setiap fungsi
d. Menentukan kode program yang akan diuji
e. Mengkonversikan *source code* ke dalam bentuk *flowgraph* (notasi lingkaran) untuk menggambarkan *statement*
f. Menghitung nilai *cyclomatic complexity* untuk menentukan jumlah jalur yang terlewati pada *statement*
g. Membuat rancangan data uji dengan mengidentifikasi setiap jalur

Setelah semua tahap terlewati selanjutnya adalah mengukur tingkat keberhasilan pengujian dengan teknik *statement coverage*. Teknik ini dilakukan dengan menjalankan data uji yang mencakup semua *statement* yang dijalankan. Untuk mencari nilai 100% pada *statement coverage* maka dilakukan perhitungan dengan rumus sebagai berikut:

$$Statement\ coverage = \frac{Number\ Of\ Statement\ Exercised}{Total\ Number\ of\ Statement}$$

3.9 Pemilihan Pengujian Fungsi

Tahapan pengujian dilakukan dengan memilih fungsi yang memiliki percabangan, kemudian menghitung jumlah *statement* dan kondisi yang sudah ditentukan pada setiap fungsi. Karena keterbatasan dalam penulisan di dalam laporan Tugas Akhir ini maka hanya beberapa fungsi pada bagian Model yang akan disampaikan.

Pada bagian model terdapat beberapa tes yang akan diuji. Adapun beberapa daftar *use case* (UC) dan fungsi yang terdapat dalam setiap kelas dapat dilihat pada Tabel 4.1

Tabel 4.1 Daftar Fungsi Model

No *Use case*	Nama File Model	Nama Fungsi	Jumlah Statment	Jumlah Kondisi
UC-01	Madmin	+login()	15	1
UC-02	Makaryawan	+login()	15	1

		+tampil_karyawan()	6	0
		+tambah_karyawan()	5	0
		+hapus_karyawan()	5	0
		+ambil_karyawan()	6	0
		+edit_karyawan()	5	0
		+tampil_karyawan_pemilik()	5	0
		+edit_profil_karyawan()	12	1
UC-03	Mlaundry	+tampil_laundry_pemilik()	6	0
		+tampil_laundry()	5	0
		+tambah_laundry()	3	0
		+ambil_laundry()	6	0
		+ambil_laundry_paket()	5	0
		+tampil_laundry_status()	6	0
		+tampil_paket_karyawan()	5	0
		+tampil_paket_laundry_karyawan()	6	0
		+pemilik_tampil_laundry()	6	0
		+laundry_terdekat()	7	0
		+edit_laundry()	19	3
		+hapus_laundry()	3	0
		+tampil_semua_laundry()	6	0
		+edit_status_laundry()	4	0
UC-04	Morder	+tampil_order()	7	0
		+tampil_order_laundry_online()	10	0
		+tampil_order_laundry_ofline()	10	0
		+tampil_order_laundry_status_ofline()	11	0
		+tambah_order()	3	0
		+ambil_order()	10	0
		+simpan_order()	32	1
		+simpan_item()	15	0
		+item_order()	5	0
		+notifikasi_hari()	10	0
		+setting_pembayaran()	6	0

		+setting_status_pengiriman()	5	0
		+laporan_laundry()	9	0
		+laporan_laundry_tanggal()	14	0
		+tampilan_order_pelanggan()	9	0
		+tampilan_status_order_pelanggan()	10	0
		+tampilan_order_laundry_status_onine()	12	0
		+simpan_berat_order_online()	3	0
		+notifikasi_pelanggan()	12	0
		+laporan_laundry_pertahun()	16	0
UC-05	Mpaket	+tampil_paket()	6	0
		+ambil_paket()	5	0
		+tambah_paket()	3	0
		+hapus()	4	0
		+edit_paket()	4	0
		+tampil_pewangi()	6	0
		+tampil_paket_laundry()	6	0
		+edit_paket_laundry()	4	0
		+tambah_laundry()	3	0
UC-06	Mpelanggan	+login()	19	1
		+tambah_pelanggan()	3	0
		+pelanggan_by_nohp()	5	0
		+tampil_pelanggan()	4	0
		+ambil_pelanggan()	4	0
		+edit_profil_pelanggan()	11	1
UC-07	Mpemilik	+login()	19	1
		+ambil_laundry()	5	0
		+ambil_pemilik()	5	0
		+tambah_pemilik()	3	0
		+edit_profil_pemilik()	11	1
		+tampilan_pemilik_laundry()	4	0
		+hapus_pemilik()	4	0
		+edit_status_pemilik()	4	0

UC-08	Mpewangi	+tampil_pewangi()	6	0
		+tampil_pewangi_laundry()	6	0
		+ambil_pewangi()	6	0
		+ambil_pewangi_by_laundry_id()	6	0
		+ambil_karyawan()	6	0
		+edit_pewangi()	4	0
		+tambah_pewangi()	3	0
		+hapus_pewangi()	4	0
UC-09	Mrating	+simpan_rating_laundry()	3	0
		+cek_rating_laundry()	6	0
		+rating_laundry()	4	0
		+total_rating_laundry()	4	0

BAB IV
IMPLEMENTASI DAN PENGUJIAN

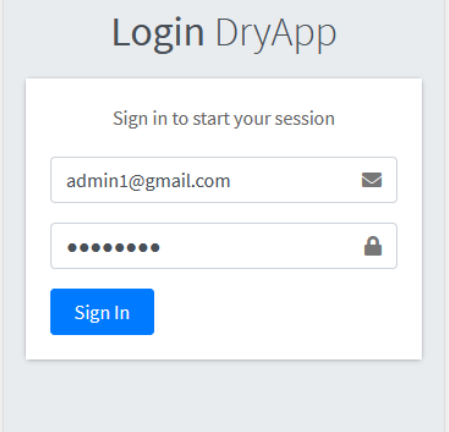
4.1 Implemantasi Program

Implementasi sistem adalah tahapan penerapan sistem yang telah dibuat berdasarkan hasil analisa dan rancangan yang telah dikerjakan sebelumnya. Dalam implementasi aplikasi laundry berasis *web mobile* ini, terdapat empat aktor yaitu admin, pemilik, karyawan dan pelanggan laundry. Aplikasi laundry ini dibangun menggunakan *framework* Codeigniter versi 3.1.11, *bootstrap* versi 4.1.1 dan basis data MySQL. Adapun implementasi dari aplikasi laundry berbasis *web mobile* ini adalah sebagai berikut.

4.1.1 Implementasi pada Akun Admin

a. Login

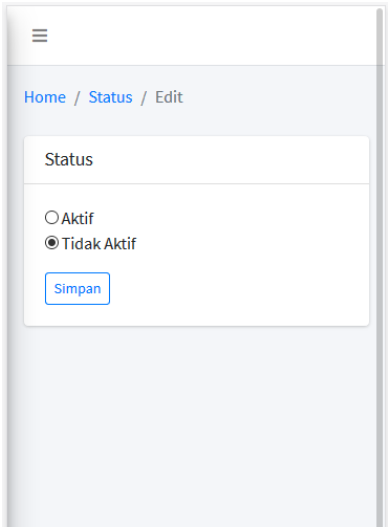
Form login admin adalah form yang digunakan ketika admin akan masuk ke aplikasi laundry dengan mengisi email dan password yang sudah ada di *database*. Gambar 4.1 adalah tampilan login pada admin



Gambar 4.1 Login Admin

b. Mengelola Status Pemilik

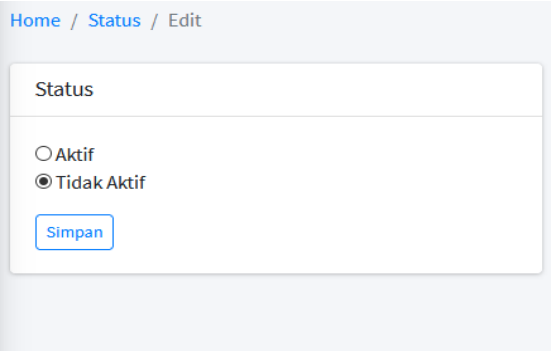
Pada halaman mengelola status pemilik, admin dapat mengedit status pemilik dari aktif, menjadi tidak aktif ataupun sebaliknya dan juga dapat menghapus daftar pemilik jika pemilik tersebut sudah tidak terdaftar Gambar 4.2 adalah tampilan admin mengelola status pemilik.



Gambar 4.2 Mengedit Status Pemilik

c. Mengelola Status Laundry

Pada halaman mengelola status laundry, admin dapat mengedit status laundry dari aktif, menjadi tidak aktif ataupun sebaliknya dan juga dapat menghapus daftar laundry jika laundry tersebut sudah tidak terdaftar. Gambar 4.3 adalah tampilan mengelola status laundry.

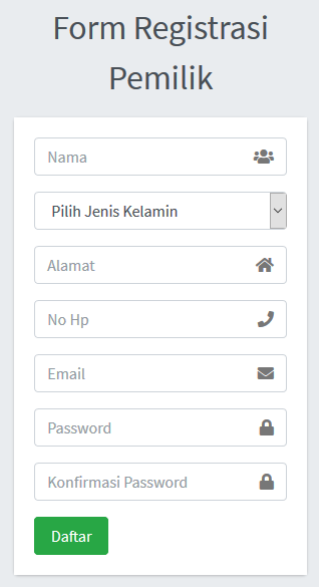


Gambar 4.3 Mengedit Status Laundry

4.1.2 Implementasi pada Akun Pemilik

a. Registrasi Akun Pemilik

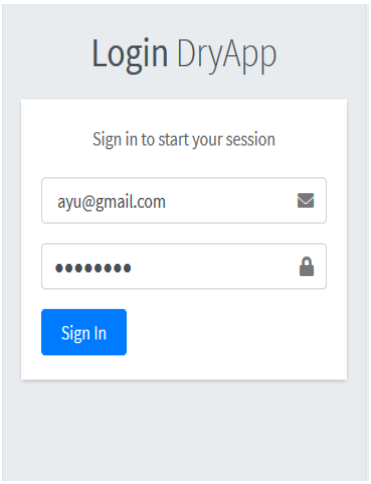
Form registrasi pemilik adalah form yang digunakan untuk pendaftaran pemilik dan laundry dengan mengisi beberapa data seperti pada gambar. Gambar 4.4 adalah hasil implementasi registrasi.



Gambar 4.4 Implementasi Registrasi

b. Login

Form login pemilik adalah form yang digunakan ketika pemilik akan masuk ke aplikasi laundry dengan mengisi email dan password yang sudah terdaftar sebelumnya saat resgistrasi. Gambar 4.5 adalah hasil implementasi form login pemilik.



Gambar 4.5 Implementasi Login

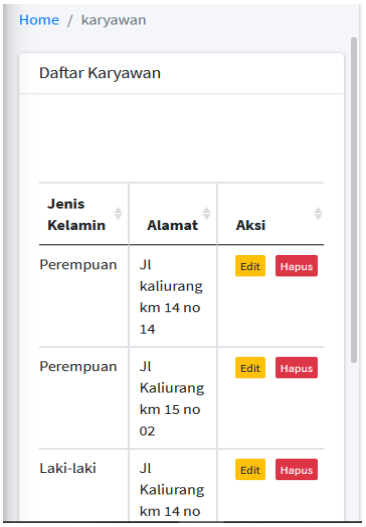
c. Melihat grafik laundry

Pada halaman ini pemilik dapat melihat perkembangan laundrynya dalam setahun yang ditampilkan dalam bentuk grafik. Gambar 4.6 adalah hasil implementasi melihat grafik laundry.

Gambar 4.6 Implementasi Melihat Grafik Laundry

d. Mengelola Daftar Karyawan

Pada halaman ini pemilik dapat mengelola daftar karyawan seperti misalnya menambhkan karyawan dengan cara mengsi beberapa data, mengedit atau menghapus daftar karyawan. Gambar 4.7 adalah hasil implementasi mengelola daftar karyawan.

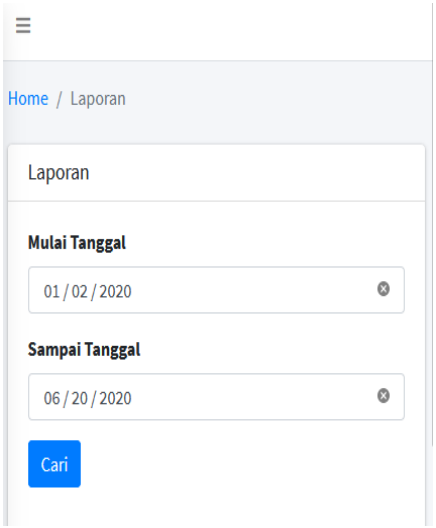


Gambar 4.7 Mengelola Daftar Karyawan

e. Melihat dan Mencetak Laporan

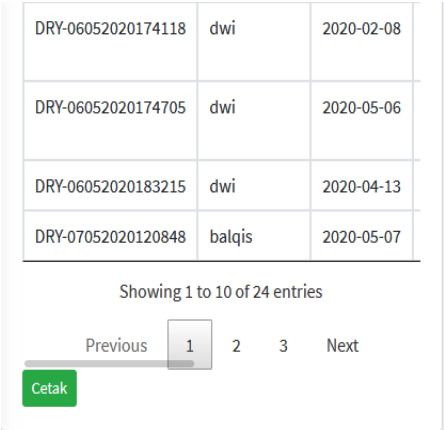
Pada halaman ini pemilik dapat melihat dan mencetak laporan laundry. Untuk melihatnya pemilik harus memasukkan tanggal mulai dan tanggal selesai data yang ingin dilihat. Setelah data laporan muncul sesuai dengan tanggal yang dipilih maka pemilik dapat mencetak laporan tersebut.

Pemilik terlebih dahulu memasukkan rentang waktu laporan yang ingin dilihat pada laundry. Gambar 4.8 adalah hasil implementasi melihat dan mencetak laporan.



Gambar 4.8 Melihat Laporan

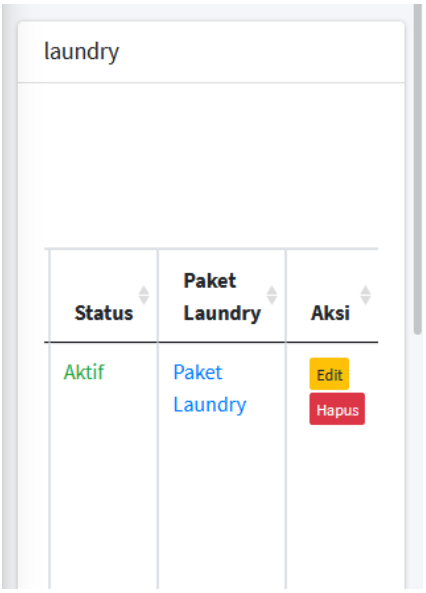
Setelah memilih rentang waktu laporan, maka akan muncul hasil dari laporan. Kemudian pemilik juga dapat mencetak hasil laporan tersebut. Gambar 4.9 adalah implementasi dari halaman cetak laporan yang telah ditentukan rentang waktunya.



Gambar 4.9 Mencetak Laporan

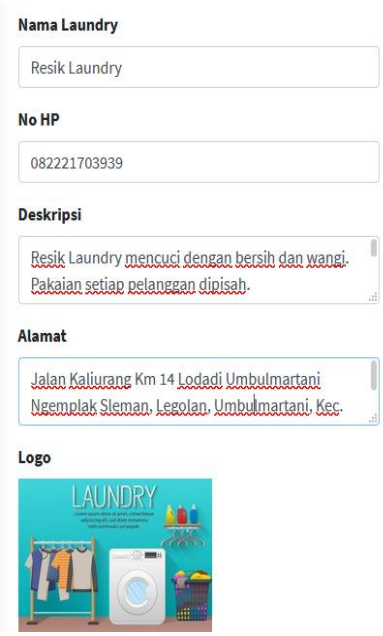
f. Mengelola Laundry dan Paket Laundry

Pada menu ini pemilik dapat mengelola laundrynya, misalnya menambahkan dan menghapus laundry. Selain itu juga pemilik dapat menambahkan daftar paket. Gambar 4. 10 adalah hasil implementasi mengelola laundry dan paket laundry.



Gambar 4.10 Mengelola Laundry

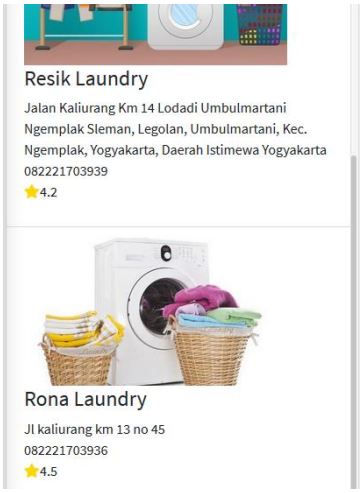
Pemilik dapat mengedit dan menambahkan daftar yang ada pada laundry seperti deskripsi laundry ataupun daftar pada paket laundry seperti jenis laundry. Gambar 4.11 adalah tampilan pemilik mengedit daftar laundry.



Gambar 4.11 Mengedit Laundry

g. Melihat Penilaian

Pada halaman ini pemilik dapat melihat hasil dari penilaian laundry yang telah dilakukan oleh pelanggan. Gambar 4.12 adalah hasil implementasi melihat penilaian.

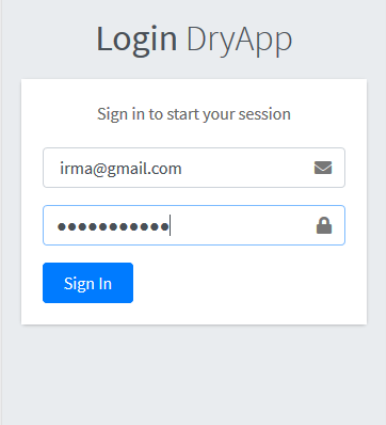


Gambar 4.12 Melihat Penilaian

4.1.3 Implementasi pada Akun Karyawan

a. Login

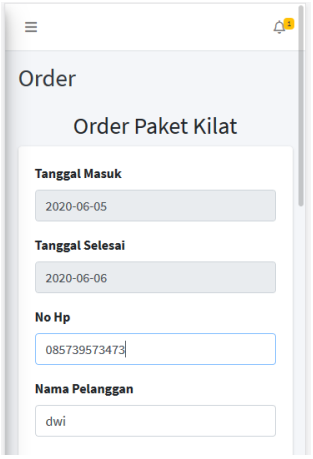
Form login digunakan pelanggan untuk masuk ke aplikasi laundry dengan mengsi email dan password yang sebelumnya telah didaftarkan oleh pemilik laundry. Gambar 4.13 adalah hasil implementasi login karyawan.



Gambar 4.13 Login Karyawan

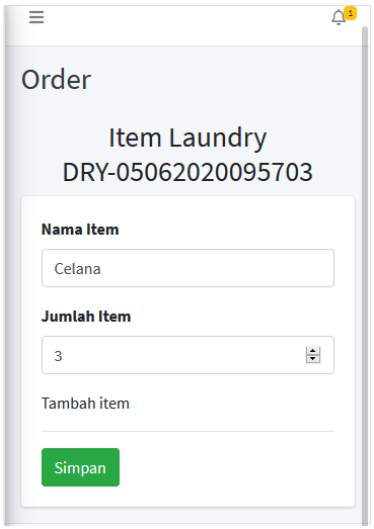
b. Mengelola Order

Pada halaman ini karyawan dapat mengelola order dengan cara memilih jenis paket terlebih dahulu kemudian memasukkan data pelanggan. Selain itu karyawan juga memasukkan nama item dan berat total pakaian yang akan dilaundry. Gambar 4.14 adalah hasil implementasi mengelola order.



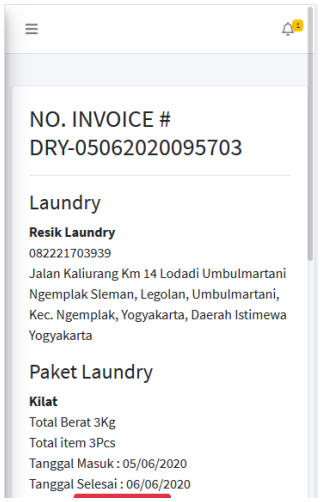
Gambar 4.14 Mengelola Order

Setelah karyawan memasukkan data pelanggan, selanjutnya karyawan memasukkan jenis item dan juga jumlah yang akan dilaundry. Gambar 4.15 adalah tampilan karyawan memasukkan item dan jumlah laundry.



Gambar 4.15 Memasukkan Item Order

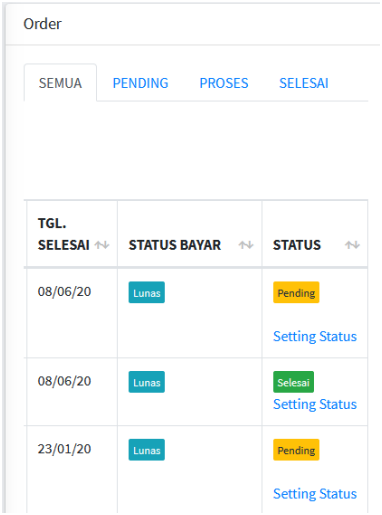
Setelah memasukkan data pelanggan, karyawan laundry dapat melihat nota yang berisi data laundry pelanggan seperti total item, jumlah order dan total bayar. Gambar 4.16 Tampilan Nota



Gambar 4.16 Tampilan Nota

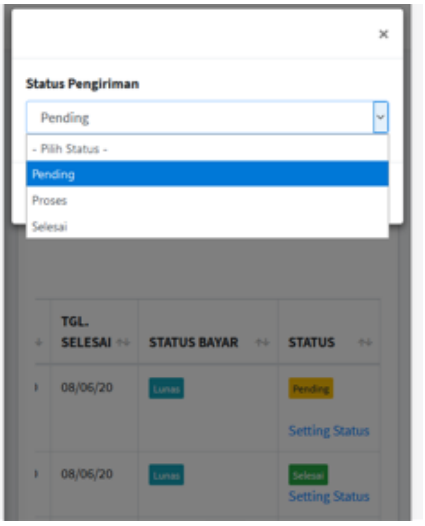
c. Mengelola History Order

Pada menu ini karyawan dapat mengelola history order misalnya megubah status bayar dan status laundry agar pelanggan dapat mengetahui sampai mana proses laundry. Gambar 4.17 adalah hasil impelmentasi mengelola history order



Gambar 4.17 Mengelola History Order

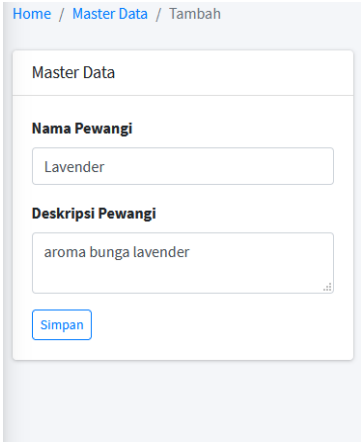
Karyawan juga dapat melihat semua riwayat laundry, baik yang masih bersatus pending, proses ataupun telah selesai. Gambar 4.18 adalah tampilan mengelola status order.



Gambar 4.18 Mengelola Status Order

d. Mengelola Inventory Data

Pada menu ini karyawan dapat mengelola inventory data yang berisi data paket laundry. Karyawan dapat menambahkan, mengedit atau menghapus jenis laundry dan pewangi. Gambar 4.19 adalah hasil implementasi mengelola inventory data.

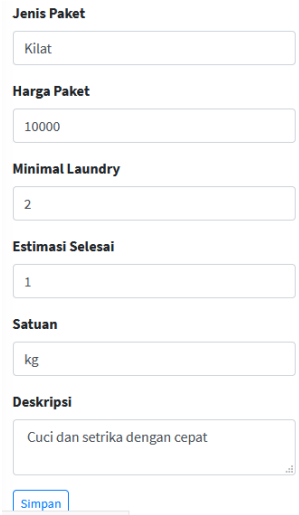


Gambar 4.19 Mengedit Daftar Pewangi

Selain itu karyawan juga dapat mengelola daftar paket pewangi jika karyawan memilih menu pewangi. Karyawan dapat mengubah nama dan deskripsi pewangi. Gambar 4.20 adalah tampilan mengelola daftar laundry.

Gambar 4.20 Mengedit Daftar Pewangi

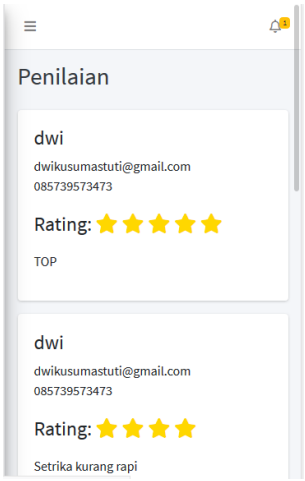
Karyawan dapat mengedit daftar laundry dengan cara mengisi data seperti jenis paket dan harga paket. Gambar 4.21 adalah tampilan karyawan mengedit daftar laundry.



Gambar 4.21 Mengedit Daftar Pewangi

e. Melihat Hasil Penilaian

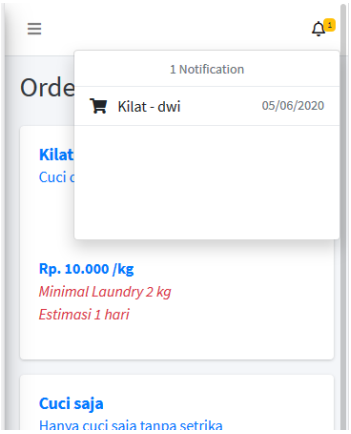
Pada halaman ini karyawan dapat melihat penilaian dan komentar dari para pelanggan yang telah melakuakn laundry dan memberikan penilaian sebelumya. Gambar 4.22 adalah hasil implementasi melihat hasil penilaian.



Gambar 4.22 Melihat Hasil Laundry

f. Menerima Notifikasi Order

Pada menu ini, karyawan dapat melihat notifikasi yang masuk jika terdapat order online dari para pelanggan. Gambar adalah hasil implementasi meneriman notifikasi order. Gambar 4.23 adalah implementasi karyawan menerima notifikasi dari pelanggan.

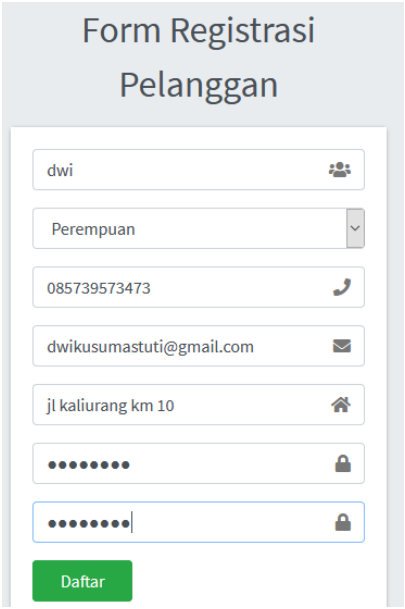


Gambar 4.23 Menerima Notifikasi Order

4.1.4 Implementasi pada Akun Pelanggan

a. Registrasi

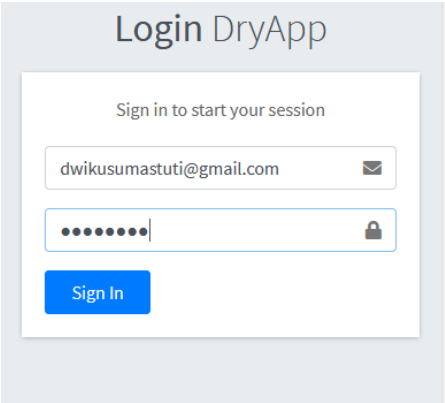
Form registrasi pelanggan adalah form yang digunakan untuk mendaftarkan diri sebagai pelanggan laundry. Data yang masuk kemudian akan disiman ke *database*. Gambar 4.24 adalah hasil implementasi registrasi pelanggan laundry.



Gambar 4.24 Registrasi Pelanggan

b. Login

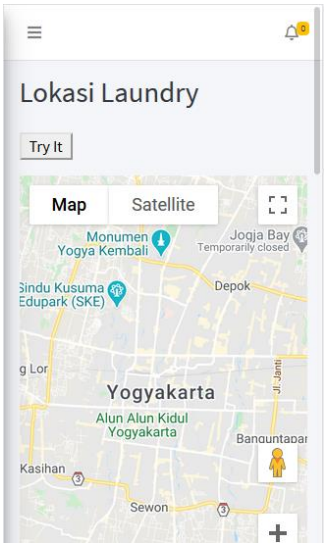
Form login pelanggan adalah form yang digunakan untuk masuk ke aplikasi laundry sebagai pelanggan dengan mengisi email dan password yang telah terdaftar sebelumnya. Gambar 4.25 adalah hasil implementasi login pada pelanggan.



Gambar 4.25 Login Pelanggan

c. Order Berdasarkan Detail Lokasi

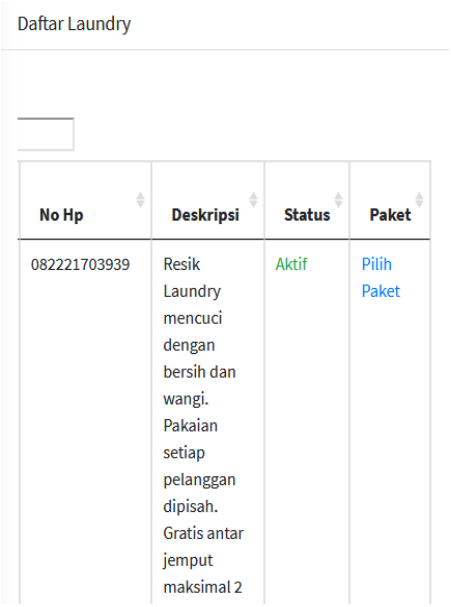
Pada halamana ini, pelanggan dapat melakukan order dengan melihat detail lokasi laundry. Pelanggan dapat mengetahui berapa jarak laundry ke lokasi mereka. Selain itu pelanggan juga dapat mengetahui alamat lengkap dan no hp yang dimilik laundry. Gambar 4.26 adalah hasil implementasi order berdasarkan detail lokasi.



Gambar 4.26 Order Berdasarkan Detail Lokasi

d. Order Berdasarkan Detail Deskripsi

Pada halamana ini, pelanggan dapat melakukan order dengan melihat detail deskripsi laundry. Selain itu pelanggan juga dapat mengetahui alamat lengkap dan no hp yang dimilik laundry. Gambar 4.27 adalah hasil implementasi order berdasarkan detail lokasi.



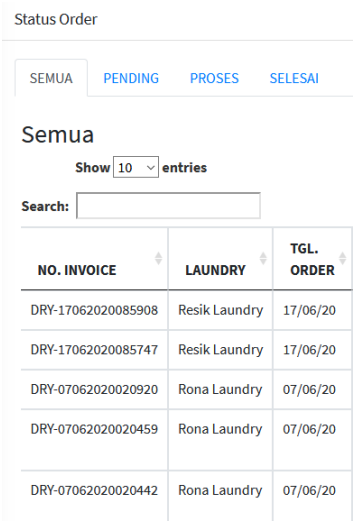
Gambar 4.27 Order Berdasarkan Detail Deskripsi

Setelah melihat deskripsi laundry, kemudian pelanggan dapat langsung mengorder laundry di halaman tersebut dengan cara memilih paket. Gambar 4.28 adalah memilih jenis paket

Gambar 4.28 Memilih Jenis Paket

e. Melihat Status Order

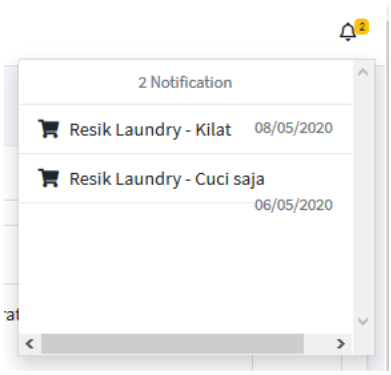
Pada halaman ini pelanggan dapat mengetahui proses dari order laundry mereka. Selain dapat melihat status order, pelanggan juga dapat melihat status bayar. Gambar 4.29 adalah hasil implementasi melihat status order.



Gambar 4.29 Melihat Status Order

f. Menerima Notifikasi Order

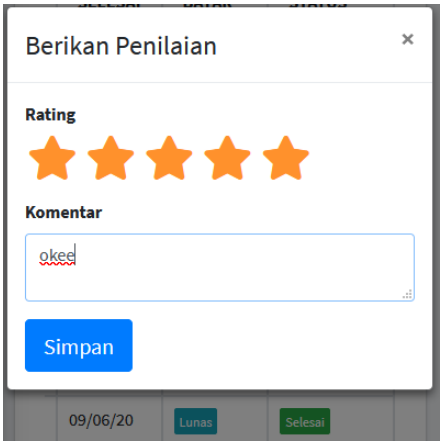
Pada menu ini, pelanggan dapat menerima notifikasi dari laundry yang sedang diorder jika pakaian telah selesai dengan menampilkan nama laundry dan jenis paket yang dipilih. Gambar 4.30 adalah hasil implementasi menerima notifikasi order.



Gambar 4.30 Menerima Notifikasi Order

g. Memberi Penilaian dan Komentar

Pada menu ini, pelanggam dapat memberi penilaian terhadap laundry yang telah selesai diorder dan juga dapat memberikan kometar. Gambar 4.31 adalah hasil implementasi memberi penilaian dan komentar.



Gambar 4.31 Memberi Penilaian dan Kometar

4.2 Pengujian Sistem

Pengujian sistem adalah tahapan terakhir yang dilakukan dalam pengembangan perangkat lunak untuk memeriksa apakah sistem yang dihasilkan sudah berjalan dengan baik dan sesuai dengan tujuan perancangan atau tidak. Selain itu juga untuk mendeteksi jika terdapat kesalahan yang mungkin ditemukan dalam implemetasi kode program. Sistem yang diuji kemudian dihitung tingkat keberhasilannya untuk mengetahui sudah sejauh mana sistem dapat digunakan.

Dalam penelitian ini dilakukan pengujian unit dengan menggunakan *framework* PHPUnit dan *white box testi*ng sebagai metodenya. Kemudian hasil dari pengujian dihitung tingkat keberhasilannya dengan menggunakan teknik *statement coverage* dimana pada teknik ini

dilakukan dengan menjalankan data uji yang mecakup semua *statement*. Kemudian dibentuk *flowgraph* yang terdiri dari *node* dan *edge* dan terisi oleh nomor *statement* untuk menentukan jalur yang akan diuji. Pengujian ini dilakukan untuk memastika semua *statement* pada *source code* yang dilaukan pengujian terbebas dari kesalahan logika program, kemudian menghitung persentase (nilai *coverage*) hasil pengujian dari jumlah pernyataan *statement* yang telah dieksekusi.

4.2.1 Perhitungan Hasil Pengujian dengan *Statement Coverage*

Statement coverage merupakan teknik perhitungan persentase tingkat keberhasilan pengujian. Teknik ini digunakan untuk memastikan bahwa seberapa besar persentase baris *Statement* yang telah berhasil dieksekusi pada suatu fungsi yang memiliki percabangan dengan jalur yang telah ditentukan. Menurut (Buchner, 2012) bahwa untuk menerapkan *statement coverage* setidaknya harus mencapai nilai keberhasilan 100% pada *statement* yang dieksekusi dengan menjalankan *statement* minimal satu kali.

Pada pengujian fungsi percabangan, ada beberapa kondisi yang tidak dilalui sehingga *statement* belum mencapai angka 100%. *Statement coverage* mencapai nilai 100% jika seluruh *statement* dan jalur telah dilewati atau dijalankan. Maka dari itu, untuk mencapai 100% kondisi percabangan lain perlu dieksekusi minimal satu kali sehingga semua *statement* terlewati dan berhasil mencapai nilai 100%

4.2.2 Pengujian pada Kelas Model

Pengujian sistem dilakukan dengan menggunakan *unit testing* yaitu PHPUnit dengan metode *white box testing* dan teknik *statement coverage*. Berdasarkan rencana pengujian telah dipilih beberapa fungsi dari Model yang memiliki jumlah percabangan terbanyak untuk dilakukan pengujian. Daftar kelas Pengujian Model (PM) dijelaskan sebagai berikut:

a. Pengujian PM-01

Pada kelas Madmin terdapat fungsi login() yang digunakan admin untuk login dengan cara memasukkan email dan password yang sudah terdaftar pada *database*. Data yang dimasukkan akan dicek di *database* dan harus terdapat satu data. Jika data yang dimasukkan benar maka proses login berhasil dan akan diproses. Namun, jika data yang dimasukkan salah maka proses login gagal. Gambar 4. 32 adalah *source code* fungsi login().

```
6       function login($email,$password){
7           $password = sha1($password);
8           $this->db->where('email', $email);
9           $this->db->where('password', $password);
10          $ambil=$this->db->get('admin');
11
12          if ($ambil->num_rows()==1){
13
14              $data = $ambil->row_array();
15              $this->session->set_userdata('admin', $data);
16              return 'berhasil';
17          }else{
18              return 'gagal';
19          }
20      }
```

Gambar 4.32 *Source code* Fungsi login()

Pengubahan *source code* menjadi *flowgraph* pada fungsi login() untuk mencari jalur yang dapat dilewati dapat dilihat pada Gambar 4.33

Gambar 4.33 *Flowgraph* login()

Berdasarkan *flowgraph* pada fungsi login() terdapat 14 *statement*, 6 *node*(N) dan 6 *edge* (E). Maka dapat dihitung nilai *cyclomatic complexity* sebagai berikut:

V(G) = E – N + 2 = 6 – 6 + 2 = 2

Hasil dari *cyclomatic complexity* berdasarkan *flowgraph* adalah 2, sehingga ditentukan terdapat 2 jalur untuk fungsi login(). Adapun identifikasi jalur yang mungkin untuk dilakukan uji coba antara lain:

1) Jalur PM-011 = 6-11,12,13-16,19,20

   Admin login dengan data yang sesuai di *database* sehingga berhasil login

2) Jalur PM-012 = 6-11,12,17-18,19,20

   Admin login dengan data yang tidak sesuai di *database* sehingga gagal login

Data uji yang mungkin dilakukan dalam pengujian berdasarkan dari jalur yang ditentukan dan diukur menggunakan *cyclomatic complexity* dapat dilihat pada Tabel 4.2. Di mana

terdapat 2 jalur yang dapat dilewati kemudian dapat diberi masukan berupa data input dan keluaran yang diharapkan.

Tabel 4.2 Rancangan Pengujian Fungsi login()

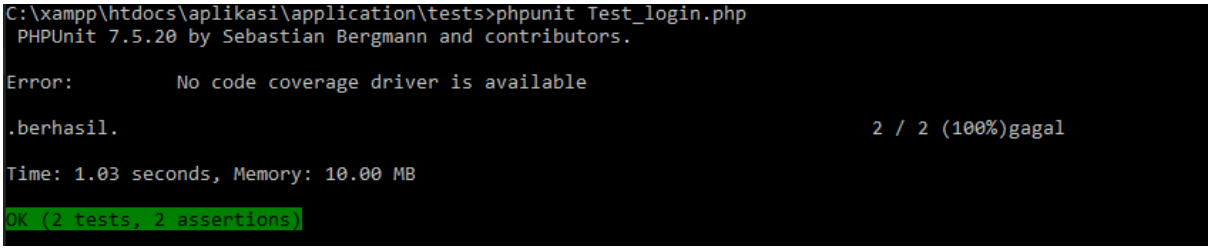
No Jalur	Masukan	Keluaran yang diharapkan
PM-011	["email"]=> "admin1@gmail.com" ["password"]=>"12345678" ["lat"]=>"-7.722037" ["lng"]=>"110.400002"	Berhasil login berdasarkan email dan password yang benar
PM-012	["email"]=> "admin100@gmail.com" ["password"]=>"admin" ["lat"]=>"-7.722037" ["lng"]=>"110.400002"	Gagal login karena email dan password tidak terdaftar di *database*

Pengujian PHPUnit untuk fungsi login dilakukan pada akun admin. Sebelum login admin terlebih dahulu mengisi email dan password yang telah terdaftar sehingga dalam pengujian juga dilakukan pengisian email dan password yang terdaftar. Fungsi pertama menjelaskan admin login sesuai dengan data yang teridentifikasi di *database*, sedangkan yang kedua admin login tidak berdasarkan data yang ada di *database*. Gambar 4.34 adalah pengujian PHPUnit pada fungsi login().

```php
<?php

class Test_login extends TestCase {

   public function testLoginAdminBerhasil(){

   $insert= new Madmin();

   $email ="admin1@gmail.com";
   $password="12345678";
   $hasil=$insert->login($email,$password);
   $this->assertEquals( $hasil, 'berhasil');
   echo $hasil;
}

public function testLoginAdminGagal(){

   $insert= new Madmin();

   $email ="admin2@gmail.com";
   $password="1234567890";
   $hasil=$insert->login($email,$password);
   $this->assertEquals( $hasil, 'gagal');
   echo $hasil;
}
}

```

Gambar 4.34 PHPUnit untuk tes fungsi login().

Pengujian dilakukan dengan memanggil nama kelas. Ketika hasil dari pengujian OK maka pengujian dianggap berhasil. Gambar 4.35 adalah hasil pengujian dari fungsi login(), dimana dijalankan 2 *test* sekaligus yaitu tes berhasil login dan gagal login.



Gambar 4.35 Hasil Pengujian Tes Fungsi login()

Berdasarkan tahapan yang telah dilakukan, diperoleh 2 jalur dan pengujian berhasil. Untuk mengetahui tingkatan keberhasilan dari program yang telah dibuat, maka dibuat sebuah kasus uji (*test case*). Tabel 4.3 Pengujian *statement coverage* UC-01 menunjukan minimal *test* yang diperlukan untuk mencakup nilai 100% menggunakan teknik *statement coverage*.

Tabel 4.3 Pengujian *Statement coverage* UC-01

Test case Id	Jalur	Keluaran Sebenarnya	Keterangan	Tambahan *Statement* Tereksekusi	Nilai *Coverage*
TC01-1	PM-011	Berhasil login sehingga muncul keterangan berhasil karena dimasukkan data yang benar atau sesuai dengan yang di *database*	Berhasil	12	12/14 x 100% =85,7%
TC01-2	PM-012	Gagal login sehingga muncul keterangan gagal karena dimasukkan data yang salah atau tidak ditemukan di *database*	Berhasil	2	14/14 x 100%= 100%

Berdasarkan pengujian yang telah dilakukan, diperoleh nilai *coverage* 100% dengan melakukan *test* pada dua jalur yang sudah ditentukan. Dimana pada jalur PM-011 memperoleh nilai sebesar 85,7%, maka pada pengujian tersebut terdapat *statement* yang belum dieksekusi sehingga dilakukan pengujian pada jalur berikutnya yaitu jalur PM-012 agar mencakup semua

statement pada program sehingga diperoleh nilai 100%. Oleh karena itu, pengujian dengan TC01-1 dan TC01-2, bahwa setiap *statement* telah dieksekusi dengan minimal satu kali *test* pada setiap jalur. Pengujian pada fungsi login() telah berhasil dilakukan dan semua *statement* pada fungsi tersebut telah dieksekusi dengan menggunakan *test* case berdasarkan jalur yang diperoleh dari perhitungan *cyclomatic complexity*.

b. Pengujian PM-02

Pada kelas Mpelanggan terdapat fungsi edit_profil_pelanggan() yang digunakan karyawan untuk mengedit profil. Password dikosongkan ketika karyawan tidak ingin mengedit password. Selanjutnya pelanggan mengedit beberapa data yang ingin diedit, kemudian diproses dan data akan diperbarui. Gambar 4.36 adalah *source code* fungsi edit_profil_pelanggan()

```
49      function edit_profil_pelanggan($inputan, $id_pelanggan){
50          if(!empty($inputan['password'])){
51              $inputan['password']=sha1($inputan['password']);
52          }else{
53              unset($inputan['password']);
54          }
55          $this->db->where('id_pelanggan', $id_pelanggan);
56          $a = $this->db->update('pelanggan', $inputan);
57          $data = $this->ambil_pelanggan($id_pelanggan);
58          $this->session->set_userdata('pelanggan', $data);
59          return $a;
60      }
```

Gambar 4.36 *Source code* Fungsi edit_profil_pelanggan()

Pengubahan *source code* menjadi *flowgraph* pada fungsi edit_profil_pelanggan() untuk mencari jalur yang dapat dilewati dapat dilihat pada Gambar 4.37

Gambar 4.37 *Flowgraph* Fungsi edit_profil_pelanggan()

Berdasarkan *flowgraph* pada fungsi edit_profil_pelanggan(), terdapat 12 *Statement*, 6 *node* (N) dan 6 *edge* (E). Maka dapat dihitung nilai *cyclomatic complexity* sebagai berikut:

V(G) = E – N + 2 = 6 – 6 + 2 = 2

Hasil dari *cyclomatic complexity* berdasarkan *flowgraph* adalah 2, sehingga ditentukan fungsi edit_profil_pelanggan() terdapat 2 jalur. Adapun identifikasi jalur yang mungkin untuk dilakukan uji coba antara lain:

1) Jalur PM-021 = 49,50,51,54,55-60

   Karyawan mengedit profil tanpa mengedit password

2) Jalur PM-022 = 49,50,52-53,54,55-60

   Karyawan mengedit profil dengan mengganti password yang ada

Data uji yang mungkin dilakukan dalam pengujian berdasarkan dari jalur yang ditentukan dan diukur menggunakan *cyclomatic complexity* dapat dilihat pada Tabel 4.4 Rancangan pengujian fungsi edit_profil_pelanggan()

Tabel 4.4 Rancangan pengujian fungsi edit_profil_pelanggan()

No Jalur	Masukkan	Keluaran yang diharapkan
PM-021	["nama_pelanggan"]=>"balqis" ["email_pelanggan"]=>"balqisa@gmail.com" ["password"]=>"" ["nohp_pelanggan"]=>"089647263455" ["jk_pelanggan"]=>"Perempuan" ["alamat_pelanggan"]=> "jl kaliurang km 10 no 55"	Berhasil mengedit profil tanpa mengubah password
PM-022	["nama_pelanggan"]=>"balqis" ["email_pelanggan"]=>"balqisa@gmail.com" ["password"]=>"balqis" ["nohp_pelanggan"]=>"089647263455" ["jk_pelanggan"]=>"Perempuan" ["alamat_pelanggan"]=> "jl kaliurang km 10 no 55"	Berhasil mengedit profil dengan mengubah password

Pengujian PHPUnit untuk fungsi edit profil dilakukan pada akun pelanggan. Pengujian dilakukan dengan cara terlebih dahulu mendefinisikan parameter pelanggan dan mengisi parameter dengan perubahan data. Gambar 4.38 adalah pengujian PHPUnit pada fungsi edit_profil_pelanggan().

```php
<?php

class Test_edit_profil_pelanggan extends TestCase {

   public function testEditPofilPelangganGantiPassword()
{
    $insert = new Mpelanggan();
    $id= 8;
    $parameter= array(
        'nama_pelanggan' => 'balqis',
        'email_pelanggan' => 'balqisa@gmail.com',
        'password_pelanggan' => '12345678',
        'nohp_pelanggan' => '089647263455',
        'jk_pelanggan' => 'Perempuan',
        'alamat_pelanggan' => 'jl kaliurang km 10 no 55'
    );

    $hasil=$insert->edit_profil_pelanggan($parameter,$id);
    $this->assertEquals( $hasil, TRUE);
    echo $hasil;
}

 public function testEditPofilPelangganTanpaGantiPassword()
{
    $insert = new Mpelanggan();
    $id= 8;
    $parameter= array(
        'nama_pelanggan' => 'balqis',
        'email_pelanggan' => 'balqisa@gmail.com',
        'password_pelanggan' => '',
        'nohp_pelanggan' => '089647263455',
        'jk_pelanggan' => 'Perempuan',
        'alamat_pelanggan' => 'jl kaliurang km 10 no 55'
    );

    $hasil=$insert->edit_profil_pelanggan($parameter,$id);
    $this->assertEquals( $hasil, TRUE);
    echo $hasil;
}
}
```

Gambar 4.38 PHPUnit untuk menguji fungsi edit_profil_pelanggan()

Pengujian dilakukan dengan memanggil nama kelas. Ketika hasil dari pengujian OK maka pengujian dianggap berhasil. Gambar 4.39 adalah hasil pengujian dari fungsi edit_profil_pelanggan(), dimana dijalankan 2 *test* sekaligus yaitu berhasil edit profil pelanggan dengan mengganti password dan berhasil edit profil pelanggan tanpa mengganti password.

```
C:\xampp\htdocs\aplikasi\application\tests>phpunit Test_edit_profil_pelanggan.php
 PHPUnit 7.5.20 by Sebastian Bergmann and contributors.

Error:         No code coverage driver is available

.1.                                                                 2 / 2 (100%)1

Time: 1.42 seconds, Memory: 10.00 MB

OK (2 tests, 2 assertions)
```

Gambar 4.39 Hasil Pengujian Fungsi edit_profil_pelanggan()

Berdasarkan tahapan yang telah dilakukan maka diperoleh 2 jalur yang dapat dilewati. Untuk mengetahui tingkatan keberhasilan dari program yang telah dibuat, maka dibuat sebuah kasus uji (*test case*). Tabel 4.5 menunjukan minimal *test* yang diperlukan untuk mencakup nilai 100% menggunakan teknik *statement coverage.*

Tabel 4.5 Pengujian *Statement coverage* UC-03

Test case Id	Jalur	Keluaran Sebenarnya	Keterangan	Tambahan *Statement* Tereksekusi	Nilai *Coverage*
TC02-1	PM-021	Profil berhasil diedit dan password pada profil berhasil diganti karena pada saat memasukkan data password juga diisi	Berhasil	10	10/12 x 100 %= 83,3%
TC02-2	PM-022	Profil berhasil diedit namun password tidak diedit kerena saat memasukan data password dikosongkan.	Berhasil	2	12/12 x 100 % = 100%

Berdasarkan pengujian yang telah dilakukan, diperoleh nilai *coverage* 100% dengan melakukan *test* pada dua jalur yang sudah ditentukan. Dimana pada jalur PM-021 memperoleh nilai sebesar 83,3%, maka pada pengujian tersebut terdapat *statement* yang belum dieksekusi sehingga dilakukan pengujian pada jalur berikutnya yaitu jalur PM-022 agar mencakup semua *statement* pada program sehingga diperoleh nilai 100%. Oleh karena itu, pengujian dengan TC02-1 dan TC02-2, bahwa setiap *statement* telah dieksekusi dengan minimal satu kali *test* pada setiap jalur. Pengujian pada fungsi edit_profil_pelanggan() telah berhasil dilakukan dan semua *statement* pada fungsi tersebut telah dieksekusi dengan menggunakan *test* case berdasarkan jalur yang diperoleh dari perhitungan *cyclomatic complexity*.

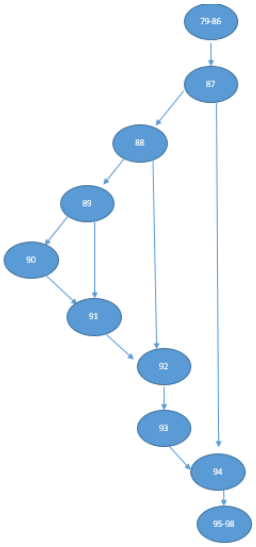
c. Pengujian PM-03

Pada kelas Mlaundry terdapat fungsi edit_laundry(). Fungsi ini berguna untuk pemilik mengedit data laundry yang ada. Di dalam mengedit data laundry terdapat pilihan apakah akan mengedit logo dari laundry atau tidak. Gambar 4.40 adalah *source code* dari fungsi edit_laundry().

```
79      function edit_laundry ($id_laundry,$inputan){
80          $datalama= $this->ambil_laundry($id_laundry);
81
82          $config['upload_path'] = './assets/img/logo/';
83          $config['allowed_types'] ='gif|jpg|png|jpeg';
84
85          $this->load->library('upload', $config);
86
87          if($this->upload->do_upload('logo')){
88              if(!empty($datalama['logo'])){
89                  if(file_exists("../assets/img/logo/".$datalama['logo'])){
90                      unlink("./assets/img/logo/".$datalama['logo']);
91                  }
92              }
93          $inputan['logo'] = $this->upload->data('file_name');
94          }
95          $this->db->where('id_laundry', $id_laundry);
96          $a = $this->db->update('laundry', $inputan);
97          return $a;
98      }
```

Gambar 4.40 *Source code* Fungsi edit_laundry()

Pengubahan *source code* menjadi *flowgraph* pada fungsi edit_laundry() untuk mencari jalur yang dapat dilewati dapat dilihat pada Gambar 4.41.



Gambar 4.41 *Flowgraph* Fungsi edit_laundry()

Berdasarkan *flowgraph* pada fungsi edit_laundry(), terdapat 18 *statement*, 9 *node* (N) dan 10 *edge* (E). Maka dapat dihitung nilai *cyclomatic complexity* sebagai berikut:

V(G) = E – N + 2 = 10 – 9 + 2 = 3

Hasil dari *cyclomatic complexity* berdasarkan *flowgraph* adalah 3, sehingga ditentukan fungsi edit_laundry() terdapat 3 jalur. Adapun identifikasi jalur yang mungkin untuk dilakukan uji coba antara lain:

1) Jalur PM-031 = 87,94,95-97

   Pemilik mengedit laundry dengan menambahkan foto pada laundry.

2) Jalur PM-032 = 87,88,92,93,94,95-97

   Pemilik mengedit laundry dengan dengan mengecek foto laundry dan menambahkan foto pada laundry.

3) Jalur PM-033 = 87,88,89,91,92,93,94,95-97

Pemilik mengedit laundry dengan mengganti foto laundry yang sudah ada sebelumnya.

4) Jalur PM-034 = 87,88,89,90,91,92,93,94,95-97

Pemilik mengedit laundry dengan menghapus dan mengganti foto laundry yang sudah ada sebelumnya.

Data uji yang mungkin dilakukan dalam pengujian berdasarkan dari jalur yang ditentukan dan diukur menggunakan metriks *cyclomatic complexity* dapat dilihat pada Tabel 4.6

Tabel 4.6 Rancangan pengujian fungsi edit_laundry()

No Jalur	Masukan	Keluaran yang diharapkan
PM-031	["id_laundry"]=>"2" ["id_pemilik"]=>"4" ["nama_laundry"]=>"Resik" ["alamat_laundry"]=>"jakal km 12" ["nohp_laundry"]=>"082221703939" ["logo"]=>"laundry2.jpg" ["lat"]=>"-7.685889511298944" ["lng"]=>"110.41883075187843" ["deskripsi_laundry"]=> "cucian pasti bersih." ["status_laundry"]=>"Aktif"	Mengedit dan menghapus logo lama kemudian tersimpan di *database*.
PM-032	["id_laundry"]=>"2" ["id_pemilik"]=>"4" ["nama_laundry"]=>"Resik" ["alamat_laundry"]=>"jakal km 12" ["nohp_laundry"]=>"082221703939" ["logo"]=>"laundry2.jpg" ["lat"]=>"-7.685889511298944" ["lng"]=>"110.41883075187843" ["deskripsi_laundry"]=> "cucian pasti bersih." ["status_laundry"]=>"Aktif"	Mengedit logo dengan mengecek data di library.
PM-033	["id_laundry"]=>"2" ["id_pemilik"]=>"4" ["nama_laundry"]=>"Resik" ["alamat_laundry"]=>"jakal km 12" ["nohp_laundry"]=>"082221703939" ["logo"]=>"laundry2.jpg" ["lat"]=>"-7.685889511298944" ["lng"]=>"110.41883075187843" ["deskripsi_laundry"]=> "cucian pasti bersih." ["status_laundry"]=>"Aktif"	Mengedit dan mengupload foto dengan mengecek data lama.
PM-034	["id_laundry"]=>"2" ["id_pemilik"]=>"4" ["nama_laundry"]=>"Resik" ["alamat_laundry"]=>"jakal km 12" ["nohp_laundry"]=>"082221703939" ["logo"]=>"" ["lat"]=>"-7.685889511298944" ["lng"]=>"110.41883075187843" ["deskripsi_laundry"]=> "cucian pasti bersih." ["status_laundry"]=>"Aktif"	Mengupload dan mengedit logo dengan menggantikan data sebelumnya dan menyimpannya di *database*, namun sebelumnya dilakukan pengecekan dan penghapusan

Pengujian PHPUnit dilakukan pada fungsi edit_laundry(). Pengujian dilakukan dengan cara terlebih dahulu mendefinisikan parameter pelanggan dan mengisi parameter dengan perubahan data. Gambar 4.42 adalah pengujian pada fungsi edit_laundry().

```php
<?php

class Test_edit_laundry extends TestCase {

    public function testEditLaundry(){

    $update = new MLaundry();
    $id= 3;

    $parameter = array(

        'id_laundry' => '3',
        'nama_laundry' => 'Ronai Laundry',
        'alamat_laundry' => 'Jl kaliurang km 13 no 45',
        'nohp_laundry' => '082221703936',
        'logo' => 'laundry1.jpg',
        'lat' => '-7.696125779068155',
        'lng' => '110.41755938480537',
        'deskripsi_laundry' => 'Rona Laundry mencuci dengan bersih dan wangi.
                                Gratis antar jemput maksimal 2 km.
                                Kami bertanggung jawab atas pakaian yang anda laundry',
        'status_laundry' => 'Aktif',
    );

      $hasil=$update->edit_laundry($id, $parameter);

   $this->assertEquals( $hasil, TRUE);
   echo $hasil;
}

}
```

Gambar 4.42 PHPUnit untuk menguji fungsi edit_laundry()

Pengujian dilakukan dengan memanggil nama kelas. Ketika hasil dari pengujian OK maka pengujian dianggap berhasil. Terdapat 4 jalur yang dapat dilalui, dimana pada jalur-jalur tersebut melakukan proses edit logo dengan cara mengecek logo lama kemudian menggantikannya dengan logo baru. Gambar 4.43 adalah hasil pengujian dari fungsi edit_laundry(),

```
C:\xampp\htdocs\aplikasi\application\tests>phpunit Test_edit_laundry.php
 PHPUnit 7.5.20 by Sebastian Bergmann and contributors.

Error:         No code coverage driver is available

.                                                                   1 / 1 (100%)1

Time: 672 ms, Memory: 10.00 MB

OK (1 test, 1 assertion)
```

Gambar 4.43 Hasil Pengujian Fungsi edit_laundry

Berdasarkan tahapan yang telah dilakukan diperolah 4 jalur dan pengujian berhasil. Untuk mengetahui tingkatan keberhasilan dari program yang telah dibuat, maka dibuat sebuah kasus uji (*test case*). Tabel 4.7 menunjukan minimal *test* yang diperlukan untuk mencakup nilai 100% menggunakan teknik *statement coverage*.

Tabel 4.7 Pengujian *Statement coverage* UC-04

Test Case Id	Jalur	Keluaran Sebenarnya	Keterangan	Tambahan *Statement* Tereksekusi	Nilai *Coverage*
TC03-1	PM-031	Berhasil mengedit dan menghapus logo lama kemudian menyimpan di *database*.	Berhasil	14	14/20 x 100% = 70 %
TC03-2	PM-032	Berhasil mengedit logo dengan mengecek data di library.	Berhasil	3	17/20 x 100% = 85%
TC03-3	PM-033	Berhasil mengedit dan mengupload logo dengan mengecek data lama.	Berhasil	2	19/20 x 100% = 95%
TC03-4	PM-034	Berhasil mengupload dan mengedit logo dengan menggantikan data sebelumnya dan menyimpannya di *database*, namun sebelumnya dilakukan pengecekan dan penghapusan	Berhasil	1	20/20 x 100% = 100%

Berdasarkan pengujian yang telah dilakukan, diperoleh nilai *coverage* 100% dengan melakukan *test* pada empat jalur yang sudah ditentukan. Dimana pada jalur PM-031 memperoleh nilai sebesar 70%, maka pada pengujian tersebut terdapat *statement* yang belum dieksekusi sehingga dilakukan pengujian pada jalur berikutnya yaitu jalur PM-032, namun jalur tersebut memperoleh nilai sebesar 85% sehingga dilakukan pengujian pada jalur berikutnya yaitu jalur PM-033, jalur tersebut memperoleh nilai 95% sehingga dilanjutkan ke pengujian PM-034 hingga diperoleh nilai sebesar 100% karena semua *statement* pada program tersebut telah dijalankan. Oleh karena itu, pengujian dengan TC03-1, TC01-2, TC03-3 dan TC03-4, bahwa setiap *statement* telah dieksekusi dengan minimal satu kali *test* pada setiap jalur. Pengujian pada fungsi edit_laundry() telah berhasil dilakukan dan semua *statement* pada fungsi tersebut telah dieksekusi dengan menggunakan *test case* berdasarkan jalur yang diperoleh dari perhitungan *cyclomatic complexity*.

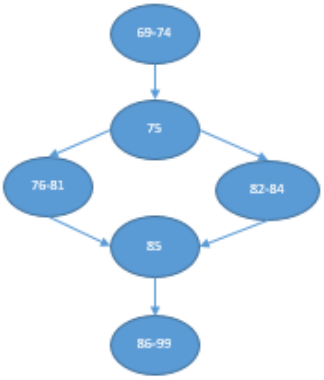
d. Pengujian PM-04

Pada kelas Morder terdapat fungsi simpan_order(). Fungsi ini digunakan ketika karyawan akan melakukan penyimpanan order laundry pelanggan. Karyawan mencari data pelanggan berdasarkan nomor hp, jika data tidak ditemukan maka karyawan harus memasukkan beberapa data pelanggan. Karyawan memasukkan pilihan pewangi laundry jika dalam laundry tersebut memilik pilihan pewangi. Gambar 4.44 adalah *source code* padafungsi simpan_order().

```
67      function simpan_order($inputan, $id_paket){
68      //mengambil data order
69          $this->db->where('id_paket', $id_paket);
70          $paket = $this->db->get('paket')->row_array();
71
72          $this->db->where('nohp_pelanggan', $inputan['nohp']);
73          $ambil=$this->db->get('pelanggan');
74
75          if($ambil->num_rows() == 0){
76              $pelanggan['nama_pelanggan']= $inputan['nama'];
77              $pelanggan['nohp_pelanggan']= $inputan['nohp'];
78              $pelanggan['alamat_pelanggan']= $inputan['alamat'];
79
80              $this->db->insert('pelanggan', $pelanggan);
81              $inputan['id_pelanggan']= $this->db->insert_id('pelanggan');
82          }else{
83              $pelanggan =$ambil->row_array();
84              $inputan['id_pelanggan']= $pelanggan['id_pelanggan'];
85          }
86
87          $inputan['id_paket']=$id_paket;
88          $inputan['id_pewangi']=isset($inputan['id_pewangi'])==false?null:$inputan['id_pewangi'];
89          $inputan['tgl_order']=date("Y-m-d H:i:s");
90          $inputan['tgl_selesai_order']=date('Y-m-d H:i:s', strtotime(date("Y-m-d H:i:s"). '+'.$paket['lama_waktu'].'days'));;
91          $inputan['no_invoice'] = "DRY-".date("dmYHis");
92          unset($inputan['nama']);
93          unset($inputan['nohp']);
94          unset($inputan['alamat']);
95
96          $this->db->insert('order', $inputan);
97          $id_order =$this->db->insert_id('order');
98          return $id_order;
99      }
```

Gambar 4.44 *Source code* Fungsi simpan_order()

Pengubahan *source code* menjadi *flowgraph* pada fungsi simpan_order() untuk mencari jalur yang dapat dilewati dapat dilihat pada Gambar 4.45



Gambar 4.45 *Flowgraph* Fungsi simpan_order()

Berdasarkan *flowgraph* pada fungsi simpan_order(), terdapat 32 *statement*, 6 *node*(N) dan 6 *edge* (E). Maka dapat dihitung nilai *cyclomatic complexity* sebagai berikut:

V(G) = E – N + 2 = 6 – 6 + 2 = 2

Hasil dari *cyclomatic complexity* berdasarkan *flowgraph* adalah 2, sehingga ditentukan independent path(jalur) yang mana jalaur independent dari fungsi simpan_order() terdapat 2 jalur.

Adapun identifikasi jalur yang mungkin untuk dilakukan uji coba antara lain:

1) Jalur PM-041 = 69-74,75,76-81,85,86-99

Menyimpan order dengan mengisi data pelanggan secara manual karena belum tersimpan di *database*.

2) Jalur PM-042 = 69-74,75,82-84,85,86-99

Menyimpan order tanpa mengisi data pelanggan karena sudah tersimpan di *database*

Data uji yang mungkin dilakukan dalam pengujian berdasarkan dari jalur yang ditentukan dan diukur menggunakan metriks *cyclomatic complexity* dapat dilihat pada Tabel 4.8

Tabel 4.8 Rancangan Pengujian fungsi simpan order()

No Jalur	Masukkan	Keluaran yang diharapkan

PM-041	Memasukkan data pelanggan secara manual karean pelanggan belum terdaftar
	`["nama"]=>"Ririn"`
	`["nohp"]=>"081789453456"`
	`["alamat"]=>"jl kaliurang km 10 no 78"`
	`["total_berat"]=>"5"`
	`["total_biaya"]=>"15000"`
	`["pakaian_bersih"]=>"Diambil Pelanggan"`
	`["id_pewangi"]=>"2"`
	`["status_order"]=>"Lunas"`
		Berhasil simpan order berdasarkan nama, no hp dan alamat pelanggan yang dimasukkan secara manual

PM-042	Memasukkan no hp yang terdaftar di *database*, maka akan otomatis menampilkan nama dan alamat pelanggan kerena pelanggan sudah terdaftar
	`["nama"]=>"dwi"`
	`["nohp"]=>"085739573473"`
	`["alamat"]=>"jl kaliurang km 10 no 45"`
	`["total_berat"]=>"5"`
	`["total_biaya"]=>"15000"`
	`["pakaian_bersih"]=>"Diambil Pelanggan"`
	`["id_pewangi"]=>"1"`
	`["status_order"]=>"Lunas"`
		Berhasil simpan order berdasarkan id pelanggan yang diambil dari nomor hp

Pengujian PHPUnit dilakukan pada fungsi simpan_order(). Pengujian dilakukan dengan cara terlebih dahulu mendefinisikan parameter pelanggan dan mengisi parameter sesuai dengan data order. Pada fungsi pertama menjelaskan cara menyimpan order dengan memasukan data pelanggan yang tersimpan di *database*, sedngkan yang kedua adalah menympan order pelanggan dengan memasukkan data secara manual karena pelanggan belum terdaftar. Gambar 4.46 adalah pengujian pada fungsi simpan_order().

```php
<?php

class Test_simpan_order extends TestCase {

  public function testSimpanOrderIdPelanggan(){
 $insert= new Morder();
 $id = 2;

    $inputan = array(
    'nama'=> 'dwi',
    'nohp'=> '085739573473',
    'alamat'=> 'jakal',
    'total_berat'=> '4',
    'total_biaya'=>'40000',
    'pakaian_bersih'=> 'Diantar Karyawan',
    'id_pewangi'=> '1',
    'status_order'=>'Lunas'
    );

   $hasil=$insert->simpan_order($inputan, $id);

   $this->assertEquals( $hasil, TRUE);
   echo $hasil;
}

 public function testSimpanOrderPelanggan(){
 $insert= new Morder();
 $id = 3;
    $inputan = array(
    'nama'=> 'luna',
    'nohp'=> '081757654327',
    'alamat'=> 'jakal',
    'total_berat'=> '5',
    'total_biaya'=>'50000',
    'pakaian_bersih'=> 'Diantar Karyawan',
    'id_pewangi'=> '1',
    'status_order'=>'Lunas'
    );

   $hasil=$insert->simpan_order($inputan, $id);

   $this->assertEquals( $hasil, TRUE);
   echo $hasil;
}
}
```

Gambar 4.46 PHPUnit untuk menguji fungsi simpan_order()

Pengujian dilakukan dengan memanggil nama kelas. Ketika hasil dari pengujian OK maka pengujian dianggap berhasil. Gambar 4.47 adalah hasil pengujian dari fungsi simpan_order(), dimana dijalankan 2 *test* sekaligus yaitu berhasil simpan order berdasarkan masukan data secara otomatis yaitu dengan memasukkan no hp dan berhasil memasukan data secara manual seperti memasukkan nama, alamat dan no hp secara manual.

```
C:\xampp\htdocs\aplikasi\application\tests>phpunit Test_simpan_order.php
 PHPUnit 7.5.20 by Sebastian Bergmann and contributors.

Error:         No code coverage driver is available

.128.                                                               2 / 2 (100%)129

Time: 1.55 seconds, Memory: 10.00 MB

OK (2 tests, 2 assertions)
```

Gambar 4.47 Hasil Pengujian Fungsi simpan_order()

Berdasarkan tahapan yang telah dilakukan maka diperoleh 2 jalur. Untuk mengetahui tingkatan keberhasilan dari program yang telah dibuat, maka dibuat sebuah kasus uji (*test case*). Tabel 4.9 menunjukan minimal *test* yang diperlukan untuk mencakup nilai 100% menggunakan teknik *statement coverage.*

Tabel 4.9 Pengujian *Statement coverage* UC-06

Test case Id	Jalur	Keluaran Sebenarnya	Keterangan	Tambahan *Statement* Tereksekusi	Nilai *Coverage*
TC04-1	PM-041	Berhasil menyimpan data berdasarkan datapelanggan yang dimasukkan secara manual seperti nama, no hp dan alamat	Berhasil	30	30/33 x 100 % =90,9%
TC04-2	PM-042	Berhasil menyimpan data yang dimasukkan secara otomatis karena data teridentifikasi di *database*	Berhasil	3	33/33 x 100 % = 100%

Berdasarkan pengujian yang telah dilakukan, diperoleh nilai *coverage* 100% dengan melakukan *test* pada dua jalur yang sudah ditentukan. Dimana pada jalur PM-041 memperoleh nilai sebesar 90,9%, maka pada pengujian tersebut terdapat *statement* yang belum dieksekusi sehingga dilakukan pengujian pada jalur berikutnya yaitu jalur PM-042 agar mencakup semua *statement* pada program sehingga diperoleh nilai 100%. Oleh karena itu, pengujian dengan TC04-1 dan TC04-2, bahwa setiap *statement* telah dieksekusi dengan minimal satu kali *test* pada setiap jalur. Pengujian pada fungsi simpan_order() telah berhasil dilakukan dan semua *statement* pada fungsi tersebut telah dieksekusi dengan menggunakan *test case* berdasarkan jalur yang diperoleh dari perhitungan *cyclomatic complexity*.

4.2.3 Hasil Pengujian

Pengujian sistem menggunakan *Unit testing* yaitu PHPUnit dilakukan untuk mengetahui seberapa baik program dapat terhindar dari kesalahan seperti kesalahan logika dan asumsi pada eksekusi jalur yang tidak seharusnya.

Berdasarkan pengujian yang telah dilakukan, diperoleh bahwa fungsi bagian Model pada aplikasi bisnis jasa laundry telah berhasil dilakukan dan aplikasi tersebut berjalan dengan baik sesuai dengan proses bisnis yang telah dirancang, sehingga sistem siap digunakan oleh pengguna. Terdapat 4 fungsi yang diuji dimana fungsi tersebut dipilih karena memiliki

percabangan. Pengujian dilakukan dengan mengubah *source code* menjadi *flowgraph* untuk menentukan jalur-jalur yang dapat dilewati sebagai alur pengujian. Alur tersebut digunakan untuk mendapatkan *output* sesuai dengan yang diharapkan dan akan digunakan sebagai dasar pengujian. Setelah itu, dihitung tingkat keberhasilannya dengan menggunakan teknik *statement coverage* untuk melihat persentasi keberhasilan sehingga diperoleh hasil pengujian adalah 100%.

Hasil tingkat keberhasilan pengujian dapat dilihat pada Tabel 4.10 dimana telah dilakukan pengujian untuk empat fungsi dan setiap pengujian pada akhirnya memperoleh nilai 100%.

Tabel 4.10 Tingkat Akurasi Pengujian

UC	PM	TC	Jumlah PM Berhasil
UC-01	PM-011	TC01-1	12/14 X 100% =85,7%
	PM-012	TC01-2	14/14 X 100% = 100%
UC-03	PM-021	TC02-1	10/12 X 100% =83,3%
	PM-022	TC02-1	12/12 X 100% =100%
UC-04	PM-031	TC03-1	14/20 X 100% =70%
	PM-032	TC03-2	17/20 X 100 % =85%
	PM-033	TC03-3	19/20 X 100% =95%
	PM-034	TC03-4	20/20 X 100% =100%
UC-06	PM-041	TC04-1	30/33 X 100% =90,9%
	PM-042	TC04-2	33/33 X 100% =100%

Berdasarkan Tabel 4.10 dapat dilihat bahwa telah dilakukan pengujian *unit testing* karena sudah sesuai tujuan yaitu berfokus menguji unit terkecil dari aplikasi. Pada empat fungsi yang telah dipilih, di mana pada semua fungsi yang diuji tersebut telah berhasil dan memperoleh nilai *coverage* sebesar 100% menggunkan teknik *statement coverage*. Sebelumnya dilakukan pengujian *white box testing* untuk memeriksa jalur yang ditentukan dan memastikan tidak terjadi error. Kemudian setiap unit dites untuk memasikan bahwa fungsi tersebut sesuai dengan spesifikasinya.

Pada tahap pertama pengujian, tidak ada yang bernilai 100%, dikarenakan belum keseluruhan *statement* yang dijalankan. Kemudian dilakukan pengujian selanjutnya, yaitu dengan jumlah *statement* yang telah ditambahkan. Tahapan tersebut dilakukan hingga semua *statement* berhasil dijalankan. Semua fungsi yang memeproleh nilai 100% dikatakan berhasil kerena telah melewati semua *statement* dan kondisi logis pada *source code* telah dieksekusi dengan minimal *test case* yang dijalankan dari jalur yang diperoleh dari perhitungan dengan menggunakan metriks *cyclomatic complexity*.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, dapat ditarik kesimpulan bahwa:

a. Pengujian *unit testing* berhasil dilakukan dengan menggunakan *framework* PHPUnit pada aplikasi *web mobile* bisnis jasa laundry dan menggunakan *white box testing* dengan teknik *statement coverage* sebagai perhitungan tingkat keberhasilan.
b. *Unit testing* dengan PHPUnit menguji aplikasi berdasarkan unit terkecil dan menyatakan setiap pernyataan dengan bentuk *source code* yang kemudian dilakukan pengujian terhadap fungsi yang memiliki percabangan pada setiap kelas. Persentase tingkat keberhasilan pengujian dilakukan menggunakan teknik *statement coverage* sehingga *test case* (kasus uji) dieksekusi minimal satu kali dan harus melewati seluruh *statement* sehingga mencapai nilai keberhasilan 100%.

5.2 Saran

Saran yang diberikan untuk pengembangan dalam pengujian aplikasi bisnis jasa laundry dengan pengujian *unit testing* ini adalah

a. Aplikasi *web mobile* bisnis jasa laundry ini masih bersifat sederhana, sehingga dibutuhkan pengebangan tahapan lebih lanjut untuk perbaikan dan peningkatan fitur.
b. Diharapkan pada pengembangan selajutnya mampu melakukan pengujian pada bagian kelas view dan controller pada aplikasi.
c. Penelitian selanjutnya dapat menguji tingkat keberhasilan dengan metode lain sehingga dapat dibandingkan metode mana yang lebih akurat.

DAFTAR PUSTAKA

Bani, A. A. (2015). IMPLEMENTASI QUALITY ASSURANCE DALAM PENGEMBANGAN MUTU SUMBER DAYA MANUSIA DI FAKULTAS AGAMA ISLAM UNIVERSITAS MUHAMMADIYAH MALANG .

Buchner, F. (2012). Is 100% Code Coverage Enough? Nomenclature, measures and values of code coverage.

Garry. (2010, September). *PHPUnit, Pengecekan Coding yang Lebih Pro*. Diambil kembali dari http://www.computesta.com/blog/2010/09/phpunit-pengecekan-coding-yang-lebih-pro/#.XxrrWDUxXIU.

Hayder, H. (2007). *Object-Oriented Programming With PHP5*. Birmingham: Packt Publishing Ltd.

IEEE. (1990). IEEE Standard Glossary of Software Engineering Terminology.

Indriasari, D., & Shidi, T. A. (2011). SISTEM PENCARIAN ORANG HILANG BERBASIS MOBILE WEB DENGAN SOCIAL NETWORK ANALYSIS. *Seminar Nasional Informatika 2011*.

Jatnika, H., & Irwan, S. (2019). Testing dan Implementasi Sistem.

Kartanti, L. L. (2015). PENGEMBANGAN DAN ANALISIS KUALITAS SISTEM ADMINISTRASI LABORATORIUM KOMPETENSI KEAHLIAN TKJ DI SMK NEGERI 1 KLATEN BERBASIS WEB.

Khan, M. E., & Khan, F. (2012). A Comparative Study of White Box, Black Box and Grey Box Testing Techniques. *Internasional Jurnal of Advanced Computer Science and Applications*.

Khasanah, A. K. (2015). Pengembangan Dan Analisis Kualitas Berdasarkan Iso 9126 Aplikasi Pendeteksi Gaya Belajar Model Vak (Visual, Auditorial, Kinestetik) Berbasis Web.

Kua, P. (2019). Unit Testing. Oracle Australian Development Centre Oracle Corporation.

Laksito, A. D. (2019). API Gateway Menggunakan SlimPHP pada Aplikasi Kantin Amikom. *IPTEK-KOM*.

Novelia, D. P. (2008). IMPLEMENTASI MODEL BASED TESTING UNTUK PEMBANGKITAN TEST CASE (STUDI KASUS : SISTEM RITEL ENTERPISE RESOURCE PLANNING).

Pressman, R. S. (2010). *Software Engineering A Practitioner's Approach, Seventh Edition*. New York: McGraw-Hil.

Rosa, A. S., & Shalahuddin. (2013). *Rekayasa Perangkat Lunak Terstruktur Dan Berorientasi Objek.* Bandung: Informatika.

Sandin, E. V., Yassin, N. M., & Mohamad, R. (2016). Comparative Evaluation ofAutomated Unit Testing Tool for PHP. *International Journal of Software Engineering and Technology*.

Shi, M. (2010). *Software Functional Testing from the Perspective of Business Practice.* Berlin: Canadian Center of Science and Education.

Soeharto, I. (1995). *Manajemen Proyek Dari Konseptual Sampai Operasional, Jilid 2.* Jakarta: Erlangga.

Solecha, I. N. (2016, Desember). *Perbedaan Mobile Web dan Mobile Apps: Keunggulan dan Kekurangan.* Diambil kembali dari https://www.herosoftmedia.co.id/perbedaan-mobile-web-dan-mobile-apps-keunggulan-dan-kekurangan/

Sommerville, I. (2011). *Software Engineering (Rekayasa Perangkat Lunak).* Jakarta: Erlangga.

Studio, F. (2015, August). *Test-Driven Development (TDD) and Behaviour-Driven Development (BDD) in PHP*. Diambil kembali dari https://firebearstudio.com/blog/test-driven-development-tdd-and-behaviour-driven-development-bdd-in-php.html

Usmanto, B., Immawan, R., Fauzi, Sari, K. P., & Mahdi, M. I. (2018). IMPLEMENTASI WEB MOBILE SEBAGAI MEDIAINFORMASI PEMBERDAYAAN MASYARAKAT DI DESA PIRNGADI. *Jurnal Keteknikan dan Sains (JUTEKS) – LPPM UNHAS*.

Wijonarko, D., & Mulya, B. W. (2018). Pengembangan Antarmuka Pemrograman Aplikasi Menggunakan Metode RESTful pada Sistem Informasi Akademik Politeknik Kota Malang. *SMATIKA*.

Wulandari, R. M. (2017). *Perancangan Web Mobile Dengan Metode User Centered Design (Ucd) Untuk Pengelolaan Bisnis Jasa Usaha Laundry.*

Yunisa, R. (2018). *PERBANDINGAN 2 TEKNIK WHITE BOX TESTING (STUDI KASUS : SISTEM INFORMASI REPORTING COMMUNITY TB-HIV 'AISYIYAH TANGGAMUS).*