BAB II
TINJAUAN PUSTAKA

1.1 Laundry

Laundry merupakan salah satu pelayanan jasa di bidang cuci mencuci pakaian dengan memiliki jenis cucian yang telah ditetapkan harga oleh pihak penyedia jasa dan waktu lama cucian biasanya ditentukan oleh penyedia dengan batas minimal dan maksimal selesainya cucian yang dipesan pelanggan (Amnah and Halimah, 2024).

Laundry adalah proses mencuci dan merawat pakaian agar tetap bersih dan terawat. Proses ini melibatkan berbagai langkah, mulai dari memisahkan pakaian berdasarkan warna dan jenis kain, mencuci dengan deterjen yang sesuai, hingga proses pengeringan dan pelipatan pakaian (Rian and Fuadytama, 2019).

1.2 Dijkstra

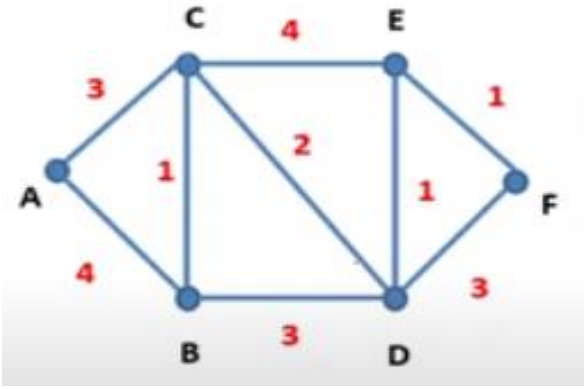
Algoritme Dijkstra adalah sebuah algoritma yang dipakai dalam memecahkan permasalahan jarak terpendek (*shortest path problem*) untuk sebuah *graf* berarah (*directed graph*). Permasalahan rute terpendek dari sebuah titik ke akhir titik lain adalah sebuah masalah klasik optimasi yang banyak digunakan untuk menguji sebuah algoritma yang diusulkan. Permasalahan rute terpendek dianggap cukup baik untuk mewakili masalah optimisasi, karena permasalahannya mudah dimengerti (hanya menjumlahkan seluruh edge yang dilalui) namun memiliki banyak pilihan solusi (Mubarok, Syauqy and Fitriyah, 2021).

Keunggulan dari penggunaan Algoritma Dijkstra yakni meminimalisir biaya yang digunakan dari titik awal menuju titik tujuan dengan cara mencari rute terpendek. Algortma ini lebih intensif dalam komputasi untuk mencari rute optimum dalam suatu jaringan seperti internet dan jalan (Rukmana and Ramdani, 2018).

Diskripsi matematis untuk grafik dapat diwakili G = {V. E}, yang berarti sebuah grafik (G) didefenisikan oleh satu set simpul (Vertex = V) dan koleksi Edge (E).

Algoritma Dijkstra bekerja dengan membuat jarak ke satu simpul optimal pada setiap langkah. Jadi pada langkah ke n, setidaknya ada n node yang sudah kita tahu jarak terpendek.

Sebagai contoh hitunglah Jarak terdekat dari V1 ke V7 pada gambar berikut ini.



Gambar 1.1 Contoh Simpul Jarak

Langkah-langkah algoritma Dijkstra dapat dilakukan dengan langkah-langkah berikut:

1. Tentukan titik mana yang akan menjadi node awal, lalu beri bobot jarak pada node pertama ke node terdekat satu per satu, Dijkstra akan melakukan pengembangan pencarian dari satu titik ke titik lain dan ke titik selanjutnya tahap demi tahap.

V	A	B	C	D	E	F
A	0	4 A, B	3 A,C	∞	∞	∞

2. Beri nilai bobot (jarak) untuk setiap titik ke titik lainnya, lalu set nilai 0 pada node awal dan nilai tak hingga terhadap node lain (belum terisi) 2.

V	A	B	C	D	E	F
C	0	4 A, B	3 A,C	5 A,C,D	7 A,C,E	

3. Set semua node yang belum dilalui dan set node awal sebagai “Node keberangkatan”

V	A	B	C	D	E	F
B	0	4 A, B	ABC=4+1 =5 AC=3 5:3 KECIL 3 3 A,C	ABD=4+3 =7 ACD=5 7:5 KECIL 5 5 A,C,D	7 A,C,E	∞

4. Dari node keberangkatan, pertimbangkan node tetangga yang belum dilalui dan hitung jaraknya dari titik keberangkatan. Jika jarak ini lebih kecil dari jarak sebelumnya (yang telah terekam sebelumnya) hapus data lama, simpan ulang data jarak dengan jarak yang baru.

V	A	B	C	D	E	F
D	0	4 A, B	3 A,C	5 A,C,D	ACDE=3+2+1 =6 ACE=7 6:7 KECIL 6 6 A,C,D,E	ACDF=3+2+3=8 8 ACDF

5. Saat kita selesai mempertimbangkan setiap jarak terhadap node tetangga, tandai node yang telah dilalui sebagai “Node dilewati”. Node yang dilewati tidak akan pernah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.

V	A	B	C	D	E	F
E	0	4 A, B	3 A,C	5 A,C,D	6 A,C,D,E	ACDEF=3+2+1+ 1=7 ACDF=8 7:8 KECIL 7 7 ACDEF

6. Set “Node belum dilewati” dengan jarak terkecil (dari node keberangkatan) sebagai “Node Keberangkatan” selanjutnya dan ulangi langkah e.

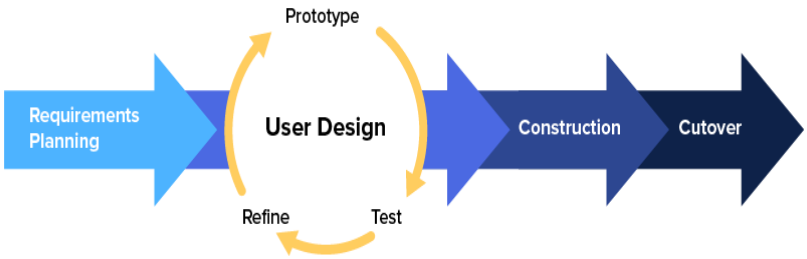
V	A	B	C	D	E	F
HASIL	0	4 A,B	3 A,C	5 A,C,D	6 A,C,D,E	7 ACDEF

Jadi lintasan terpendek adalah : A,C,D,E,F

1.3 *Rapid Application Development* (RAD)

Metode yang berfokus pada pengembangan aplikasi secara cepat, melalui pengulangan dan *feedback* berulang-ulang. RAD diajukan oleh IBM pada tahun 1980 sampai 1990-an, ketika permintaan terhadap aplikasi semakin meningkat (Rosa and Shalahuddin, 2019). Dengan banyaknya *demand*, orang-orang di dunia IT harus mencari solusi untuk memenuhi permintaan tersebut. Metode ini merupakan semacam cikal bakal *agile project management*, karena bisa mengikuti *pace* bisnis yang terus berkembang dan juga kebutuhan pasar yang terus meningkat. Pengembangan software pada umumnya seperti waterfall model membutuhkan perencanaan yang terbilang cukup kaku. Klien atau pelanggan seakan ‘dipaksa’ untuk menyetujui banyak hal di awal, tetapi mereka tidak bisa melihat proses pembuatannya (Mandasari and Kaban, 2022).

Keuntungan utama menjalankan *rapid application development* adalah jangka waktu pengembangan lebih cepat. Hal ini dikarenakan *feedback* dari pelanggan cepat didapatkan dan semua perubahan yang dilakukan akan sesuai hasil tersebut. Akan tetapi, salah satu kekurangan RAD adalah kamu membutuhkan tim berisikan *developer* yang benar-benar memiliki *skill* tinggi dan juga metode ini hanya bisa digunakan untuk proyek yang bisa termodulasi.



Gambar 1.2 *Rapid Application Development* (RAD)
Sumber: (Rosa and Shalahuddin, 2019)

1. Kelebihan Model RAD

   Kelebihan metodologi RAD menurut Marakas (2006):

   a. Penghematan waktu dalam keseluruhan fase projek dapat dicapai.
   b. RAD mengurangi seluruh kebutuhan yang berkaitan dengan biaya projek dan sumberdaya manusia.
   c. RAD sangat membantu pengembangan aplikasi yang berfokus pada waktu penyelesaian projek.
   d. Perubahan desain sistem dapat lebih berpengaruh dengan cepat dibandingkan dengan pendekatan SDLC tradisional.
   e. Sudut pandang user disajikan dalam sistem akhir baik melalui fungsi-fungsi sistem atau antarmuka pengguna.
   f. RAD menciptakan rasa kepemilikan yang kuat di antara seluruh pemangku kebijakan projek.

2. Kelemahan Model RAD

   Kelemahan pada pengembangan tersebut dapat dilihat berdasarkan kesesuaian pengembangan yang dilakukan, berikut adalah kelemahan metode pengembang sistem RAD :

   a. Dengan metode RAD, penganalisis berusaha mepercepat projek dengan terburu-buru.
   b. Kelemahan yang berkaitan dengan waktu dan perhatian terhadap detail. Aplikasi dapat diselesaikan secara lebih cepat, tetapi tidak mampu mengarahkan

penekanan terhadap permasalahan-permasalahan perusahaan yang seharusnya diarahkan.

c. RAD menyulitkan *programmer* yang tidak berpengalaman menggunakan prangkat ini di mana *programmer* dan *analyst* dituntut untuk menguasai kemampuan-kemampuan baru sementara pada saat yang sama mereka harus bekerja mengembangkan sistem

2.3.2 Tahapan Penelitian

Tahapan dalam penelitian sebagai langkah-langkah penelitian yang harus dikerjakan, berikut adalah tahapan penelitian Rapid Application Development.

1. Tahap *Requirements Planning*

   Tahap *Requirements Planning* dimulai dengan menentukan kebutuhan sebuah proyek dengan melakukan pengumpulan data, mengetahui permasalahan pada sistem yang saat ini digunakan serta menentukan kebutuhan pengguna terhadap sistem yang dibutuhkan.

2. Tahap *User Design*

   Tahap *User Design* dilakukan dengan membuat rancangan sistem yang dilakukan dengan menggunakan diagram UML. Tujuannya, untuk mengecek apakah *prototype* yang dibuat sudah sesuai dengan kebutuhan klien. Meski begitu, tahap ini bisa saja dilakukan berulang-ulang. Kadang juga melibatkan user untuk testing dan memberikan feedback.

3. Tahap *Contruction*

   Tahap ini merupakan tahap pembuatan sistem berbasis *mobile* dengan bahasa pemprograman PHP sebagai bagian *back end* dan *mobile bootstrap* serta media penyimpanan Mysql dengan membangun fitur, fungsi, interface, sampai keseluruhan aspek dari produk yang dibuat. jika prosesnya berjalan lancar, developer akan melanjutkan ke langkah berikutnya. Yaitu, finalisasi produk atau implementasi.

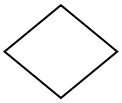
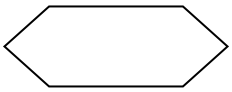
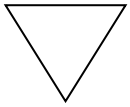
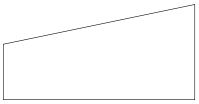
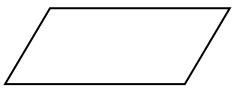
4. Tahap *Cutover*

   Tahap *Cutover* merupakan tahap evaluasi terhadap sistem yang telah dibangun dengan menguji terlebih dulu oleh pengguna sistem untuk memperoleh hasil apakah sistem telah sesuai atau belum.

1.4 *Flowchart*

Bagan alir dokumen (*Document flowchart*) merupakan bagan alir yang menunjukkan arus data dari laporan dan formulir termasuk tembusan-tembusannya (Rosa and Shalahuddin, 2019). Simbol-simbol yang dipergunakan dalam pembuatan bagan alir dokumen dapat dilihat pada Tabel 2.1:

Tabel 1.1 Bagan Alir Dokumen

No.	Simbol	Keterangan
1.		**Simbol *Offline Connector*** Untuk menyatakan sambungan dari satu proses ke proses lainnya dalam halaman/lembar yang berbeda.
2.		**Simbol Manual** Untuk menyatakan suatu tindakan (proses) yang tidak dilakukan oleh komputer.
3.		**Simbol *Decision/Logika*** Untuk menunjukkan suatu kondisi tertentu yang menghasilkan dua kemungkinan jawaban, ya/tidak.
4.		**Simbol *Predefined Proses*** Untuk menyatakan penyediaan tempat penyimpanan suatu pengolahan untuk memberi harga awal.
5.		**Simbol *Terminal*** Untuk menyatakan permulaan atau akhir suatu program.
7.		**Simbol *Off-Line Storage*** Untuk menunjukkan bahwa data dalam simbol ini akan disimpan ke suatu media tertentu.
8.		**Simbol Manual *Input*** Untuk memasukkan data secara manual dengan menggunakan *online keyboard*.
9.		**Simbol *Input-Output*** Untuk menyatakan proses input dan *output* tanpa tergantung dengan jenis peralatanya.

No.	Simbol	Keterangan
10.		**Simbol *Document*** Untuk mencetak laporan ke printer.
11.		**Simbol *Connector*** Untuk menyatakan sambungan dari satu proses ke proses lainnya dalam halaman/lembar yang sama.

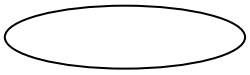
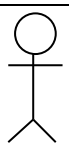
1.5 *Unified Modelling Language (UML)*

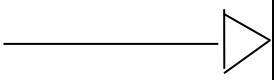
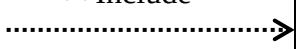
Alat pengembang sistem merupakan konsep desain yang digunakan untuk menggambarkan sistem dengan menggunakan diagram. Penyesuaian alat yang digunakan harus sesuai dengan metode pengembangan yang dilakukan salah satunya adalah penerapan *Unified Modelling Language.* Menurut (Rosa and Shalahuddin, 2019). Berikut ini merupakan penjelasan tentang masing-masing diagram yang ada pada *Unified Modelling Language.*

1.5.1 *Use Case Diagram*

Use Case adalah *use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut (Rosa and Shalahuddin, 2019). Berikut simbol-simbol yang akan digunakan dalam menggambarkan *Use Case Diagram* dapat dilihat pada Tabel 2.2.

Tabel 1.2 Simbol *Use Case Diagram*

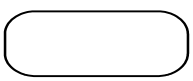
No	Simbol	Deskripsi
1.		*Use case*: Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal *frase* nama *use case*.
2.		Aktor: seseorang/sesuatu yang berinteraksi dengan yang akan dibuat. diluar sistem informasi. Biasanya dinyatakan menggunakan kata benda

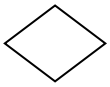
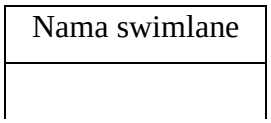
No	Simbol	Deskripsi
3.		Asosiasi *(association)*: merupakan komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau use case memiliki interaksi dengan aktor.
4.		Generalisasi *(generalization):* merupakan hubungan (umum – khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum
5.	<< Include >> 	Include berarti hubungan antara dua *use case* di mana suatu use case (yang disebut *use case* yang termasuk) memasukkan atau mencakup langkah-langkah dari *use case* lain (yang disebut *use case* yang termasuk).
6.	<<Extend>> 	Ekstensi *(extend)* hubungan antara dua *use case* di mana suatu *use case* (*use case* yang memperpanjang) dapat memperluas perilaku *use case* lain (*use case* yang diperpanjang) hanya dalam situasi tertentu.

1.5.2 *Activity Diagram*

Activity diagram adalah *activity* diagram menggambarkan aliran kerja atau aktivitas dari sebuah sistem atau proses bisnis atau menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem (Rosa and Shalahuddin, 2019). Berikut simbol-simbol yang akan digunakan dalam menggambarkan *activity diagram* dapat dilihat pada Tabel 2.3.

Tabel 1.3 Simbol *Activity Diagram*

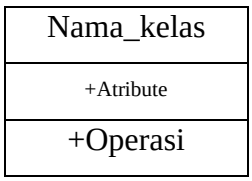
No.	Simbol	Keterangan
1.		Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2.		Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.

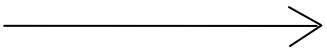
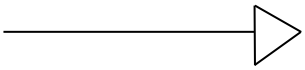
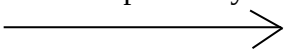
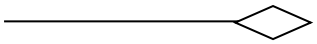
No.	Simbol	Keterangan
3.		Percabangan *(Decision)* merupakan asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4.		Penggabungan *(Join)* merupakan asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5.		Swimlane Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas.
6.		Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

1.5.3 *Class Diagram*

Class Diagram adalah *Class diagram* mengembangkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem (Rosa and Shalahuddin, 2019). Berikut simbol-simbol yang akan digunakan dalam menggambarkan *Class Diagram* dapat dilihat pada Tabel 2.4.

Tabel 1.4 Simbol *Class Diagram*

No.	Simbol	Deskripsi
1.		Kelas pada struktur sistem**.**
2.	Antar Muka/Interface  Nama_Interface	Sama dengan konsep interface dalam pemrograman berorientasi objek.
3.	Asosiasi / Asociation ———————	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *symbol*

No.	Simbol	Deskripsi
4.	Asosiasi Berarah / *Digunakan Association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *symbol*.
5.	Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
6.	Ketergantungan / dependency 	Relasi antar kelas dengan makna ketergantungan antar kelas.
7.	Agregasi / *aggregation* 	Relasi antar kelas dengan makna semua bagian *(whole-part)*

1.6 Penelitian Terdahulu

Penelitian terdahulu digunakan sebagai acuan atau refrensi penelitian yang akan dilakukan seperti berikut:

Tabel 1.5 Penelitian Terdahulu

No	Judul	Penulis	Masalah	Metode	Hasil
1	Sistem Jalur E-Laudry Menggunakan Metode Djikstra	(Setiawan and Nurhidayati, 2017)	Untuk menjaga agar usaha ini tidak sepi oleh pelanggan penyedia jasa laundry harus memiliki ciri khas untuk menarik pelanggan salah satunya menyediakan jasa antar jemput pakaian. Oleh sebab itu diperlukan sebuah aplikasi sistem pengambilan keputusan yang dapat membantu merekomendasikan rute terpendek.	Djikstra	Sistem ini dapat memberikan informasi mengenai urutan customer yang terdekat sampai yang terjauh yang harus dikunjungi menggunakan algoritma Dijkstra, dan memberikan informasi rute jalan yang bisa dilewati menggunakan data dari OpenStreetMaps. Sistem
2	Perancangan Aplikasi Pemesanan Laundry Berbasis Website	(Zain Darma *et al.*, 2020)	Saat ini, semakin banyak orang yang memilih untuk mencuci pakaian mereka di laundry karena banyaknya pekerjaan yang mereka lakukan, sehingga mereka tidak memiliki cukup waktu untuk melakukannya. Oleh sebab itu dibutuhkan inovasi teknologi informasi berbasis web untuk mempermudah pemesanan.	Deskriptif	Website Aplikasi Pemesanan Laundry yang dibuat dapat mempermudah masyarakat dan mahasiswa/i memesan jasa laundry
3	Pembangunan Aplikasi Jasa Laundry Berbasis Android	(Primawaty, 2019)	Permasalahan seperti tidak semua masyarakat mampu untuk melakukan kegiatan mencuci baju karna terbatas akan waktu, tempat (untuk mengeringkan pakaian) dan kemampuan dari_masing-masing individu.	Metode penelitian deskripstif yaitu	Aplikasi dapat membantu dalam pencarian OXY laundry terdekat dari tempat pelanggan, Aplikasi dapat membantu proses antar-jemput laundr, sehingga konsumen jadi lebih mudah dalam melakukan laundry

No	Judul	Penulis	Masalah	Metode	Hasil
4	Implementasi Algoritma Dijkstra Pada Pencarian Rute Terpendek ke Museum di Jakarta	(Cantona, Fauziah and Winarsih, 2020)	Pada implementasi penelitian ini diterapkan pada museum-museum di Jakarta yang memiliki banyak museum yang bersejarah dengan banyaknya museum yang menyimpan sejarah panjang sampai kemerdekaan Indonesia kini	Dijkstra	Dengan metode algoritma Dijkstra yang diterapkan dalam smartphone pengguna dapat mempersingkat efektifitas waktu untuk mencari museum di Jakarta. Rute
5	Penerapan Algoritma Dijkstra Untuk Menentukan Rute Terpendek Dari Pusat Kota Surabaya Ke Tempat Bersejarah	(Bunaen, Pratiwi and Riti, 2022)	Permasalahannya adalah rute dari masing- masing lokasi tempat bersejarahnya sendiri dan juga ketidaktahuan akan rute-rute terpendek dan tercepat yang lebih efisien. Penelitian	Dijkstra	Hasil yang diperoleh dari penelitian ini terdapat 5 rute terpendek yang bisa dilalui untuk menuju tempat bersejarah melalui titik awal yaitu Stasiun Gubeng

Berdasarkan penelitian terdahulu yang telah dijelaskan dan dibandingkan dengan penelitian yang peneliti kembangkan terdapat kelebihan seperti informasi laundary mencakup wilayah Kota Bandar Lampung, proses transaksi pesanan dan pembayaran secara online, terdapat informasi lokasi dan pencarian tempat laundry terdekat dengan djikstra