BAB II
TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Pada tesis yang berjudul ““Follow the World's Creators": Negotiating the Value of GIF Art and Artists”, McCutchin (2017) membahas mengenai nilai dari gambar animasi GIF sebagai bentuk seni digital pada *platfom* media sosial Tumblr, berdasarkan analisis serta wawancara dengan 15 seniman GIF. Dengan berbagai fitur yang mendukung sirkulasi, imaterialitas serta budaya *remix* karya seni, Tumblr menajdi salah satu *platform* media sosial di mana seni gambar animasi GIF mendapatkan nilai kultural yang unik. Nilai tersebut juga berhubungan dengan limitasi dari Tumblr dalam meng-*upload* gambar animasi GIF yang membatasi maksimal *file size* 2MB. Hal ini menjadi tantangan bagi para seniman GIF untuk menjadi kreatif dalam memastikan karya seni mereka dapat mematuhi limitasi yang telah ditetapkan Tumblr. Limitasi tersebut tidak menghalangi, melainkan membentuk karya seni GIF yang ada di Tumblr.

Sebuah *conference proceeeding* yang berjudul “A Study on image compression techniques” oleh Garba, Jauro dan Bala pada tahun 2016 menyatakan peran dari *compression* pada media gambar digital. Walaupun teknologi penyimpanan dan pengiriman data berkembang dengan pesat, permintaan atas kapasitas penyimpanan data serta *bandwith* dari komunikasi data melebihi yang tersedia. Data multimedia seperti gambar digital mengonsumsi sebagian besar kuota penyimpanan dan pengiriman data mayoritas pengguna, dan dalam zaman sekarang, sosial media mustahil berjalan tanpa adanya gambar digital. Untuk menghemat kuota penyimpanan data digital dan pengiriman, *compression* pada gambar digital terbukti menjadi teknik yang berguna. Tujuan dari *compression* pada gamabar digital adalah mengurangi *file size*, sambil mempertahankan kualitas gambar tersebut. Ada dua teknik *compression*, yang pertama merupakan *lossless compression* di mana gambar yang dikompres dapat dikembalikan menjadi semula. Metode ini umumnya digunakan pada beberapa *use case* di mana kualitas sangat penting seperti gambar hasil pencitraan medis. Yang kedua merupakan *lossy compression* di mana gambar yang dikompres tidak dapat dikembalikan menjadi semula, karena beberapa data warna yang dekat pada warna lain dibuang.

Umumnya teknik *lossy compression* menghasilkan pengurangan *file size* yang lebih besar daripada *lossless compression*.

Framework *cross-platform* Electron dibahas oleh Alymkulov pada tahun 2019 di tesis yang berjudul "Desktop Application Development Using Electron". Native application merupakan aplikasi yang dikembangkan dengan bahasa serta API (Application Programming Interface) yang kompatibel dengan OS (Operating System) target *platform* aplikasi tersebut. Pendekatan ini mempunyai satu kekurangan yang besar, yaitu jika *developer* menargetkan OS ataupun platform yang lainnya, ia harus mengembangkan ulang aplikasi tersebut dengan bahasa dan API yang kompatibel. Ini menyebabkan beberapa implikasi, waktu pengembangan dapat menjadi lambat, dan waktu yang diperlukan untuk melakukan *testing* juga menjadi lambat dikarenakan dua *codebase* yang terpisah harus diuji. Di masa lalu, hanya pendekatan *native* yang tersedia. Namun, seiring berjalannya waktu, berbagai perangkat keras mulai berkembang. Waktu dan biaya yang diperlukan untuk mengembangkan aplikasi *native* juga meningkat. Karena teknologi sudah mendunia dan menjadi lebih umum dan mudah diakses, ada tekanan yang besar pada para *developer* untuk merilis produk mereka ke pasar sebelum pesaing mereka. Dengan faktor-faktor tersebut, aplikasi *cross-platform* mulai muncul. Aplikasi *cross-platform* memiliki keunggulan atas kelemahan dari aplikasi *native* di mana *developer* hanya perlu mengembangkan aplikasi dengan satu bahasa, yang dapat kemudian dijalankan pada lebih dari satu *platform*. Dengan keunggulan itu, waktu pengembangan aplikasi menjadi lebih cepat dan tidak perlu memerlukan banyak sumber daya. Para *developer* dapat menjangkau lebih banyak pengguna dalam waktu yang lebih singkat, serta *update* pada aplikasi menjadi lebih mudah.

Pada *conference proceeding* yang berjudul "Optimizing the Performance in Web Browsers Through Data Compression: A Study" oleh Thapar et al. (2016), dilakukan analisis terhadap format gambar animasi APNG. Format animasi yang berdasarkan PNG ini memiliki seluruh fitur yang dimiliki format gambar PNG biasa. APNG memiliki "*backwards compatibility*" di mana jika ada *web browser* yang tidak dapat menampilkan gambar APNG, hanya *frame* yang pertama saja yang ditampilkan. APNG memiliki keunggulan atas format animasi GIF dalam rangka kualitas. Walaupun demikian, APNG tidak sepopuler GIF. Ini disebabkan karena

tools untuk membuat dan memanipulasi APNG tidak tersedia secara luas, sehingga menyebabkan sedikitnya jumlah gambar animasi APNG yang beredar di internet.

Jurnal "Comprehensive Analysis of Software Development Life Cycle Models" oleh Modi, Singh dan Chauhan (2017) membahas mengenai pentingnya metodologi pengembangan *software*. Pada era awal *software development*, *code* diketik dan di-*debug*. Inilah pendekatan yang diterapkan pada saat itu, tidak ada pendekatan formal untuk desain dan analisis. Seiring berjalannya waktu, kebutuhan atas *software* yang kompleks membuat pendekatan lama tersebut semakin tidak optimal. Karena pendekatan dalam mengembangkan *hardware* yang kompleks sudah tergolong formal dan konkrit, model tersebut menjadi patokan dalam pendekatan pengembangan *software*. Model SDLC (Software Development Life Cycle) menjadi sangat penting dalam mengembangkan *software* dengan cara yang sistematis di mana *software* tersebut dapat di-*deliver* dengan kualitas yang sesuai. Ada beberapa model SDLC yang ada, salah satunya merupakan Waterfall model. Model ini optimal saat dipakai untuk mengembangkan proyek berskala kecil.

Tabel kesimpulan di bawah ini adalah rangkuman seluruh kesimpulan dari kelima penelitian yang telah diuraikan di atas untuk dibandingkan dengan proyek skripsi yang akan dilakukan:

Tabel 2.1 Rangkuman kesimpulan dari kelima hasil penelitian yang telah ditinjau

No.	Nama pengarang	Judul Penelitian	Tahun	Kesimpulan Penelitian
1.	McCutchin, Carla	"Follow the World's Creators": Negotiating the Value of GIF Art and Artists	2017	Format gambar animasi GIF mempunyai nilai artistik yang mempengaruhi serta dipengaruhi oleh media sosial.
2.	Garba, Alhassan Jauro, Ali Ibrahim Bala, Alhassan	A study on image compression techniques	2016	Kompresi pada media digital masih diperlukan walaupun teknologi penyimpanan dan pengiriman data sudah maju.
3.	Alymkulov, Daniyar	Desktop Application Development Using Electron	2019	Electron sebagai *framework* dapat memudahkan *developer* untuk mengembangkan aplikasi *cross-platform*.
4.	Thapar, Suraj	Optimizing the Performance in Web	2016	Format gambar animasi APNG mempunyai

	Chowdhary, Sunil Kumar Bahri, Devashish	Browsers Through Data Compression: A Study		kualitas yang lebih unggul daripada GIF, namun tidak lebih populer dikarenakan sedikitnya *tools* yang tersedia untuk membuatnya.
5.	Modi, Harshad S. Singh, Nikhil Kumar Chauhan, Harsha Praddepbhai	Comprehensive Analysis of Software Development Life Cycle Models	2017	Metode pengembangan aplikasi SDLC Waterfall optimal untuk aplikasi berskala kecil.

Kumpulan penelitian yang tertera pada tabel di atas merupakan inspirasi bagi penulis dalam melakukan skripsi ini. Gambar animasi telah menjadi budaya di internet (McCutchin, 2017), dan dalam rangka mengembangkan aplikasi yang dapat membuat gambar animasi GIF, serta menyediakan fitur untuk mendukung pembuatan gambar animasi APNG (Thapar et al., 2016), penulis mengembangkan aplikasi *cross-platform* Electron (Alymkulov, 2019) yang mempunyai fitur kompresi (A. Garba et al., 2016) agar dapat menghasilkan gambar animasi yang mudah disebarkan di internet. Metode SDLC Waterfall model akan digunakan untuk mengembangkan aplikasi ini (Modi et al., 2017).

2.2 Landasan Teori

Dalam perancangan dan pengembangan aplikasi desktop manipulasi GIF dan APNG, penulis membuat landasan teori, yang merupakan teori-teori beserta tinjauan yang diperlukan untuk merancang dan membuat sistem tersebut serta digunakan untuk memperkuat teori penelitian. Berikut adalah berbagai teori yang digunakan:

2.2.1 Sistem Informasi

Pada jurnal yang berjudul "Journal of the Midwest Association for Information Systems" oleh Power dan Hadidi (2018), sistem informasi dari segi teknis dapat didefinisikan sebagai berbagai komponen-komponen dalam sebuah rangkaian yang saling terintegrasi antara satu dengan yang lain, dengan fungsi mengumpulkan, memproses, menyimpan dan menampilkan data dalam tujuan mendukung keputusan, menyediakan informasi, pengetahuan ataupun produk digital. Pendorong utama dari sistem informasi adalah data, serta pendukung

perkembangan sistem informasi merupakan teknologi dalam komputasi dan komunikasi.

Sebuah jurnal yang ditulis oleh Sharipov, Begmatov, Sa'dullayeva dan Ismailova (2019) menjelaskan bahwa konsep dari sistem informasi memiliki interpretasi yang berbeda-beda, tergantung pada konteks pemakaiannya. Secara garis besar sistem informasi memiliki komponen integral yang berupa data, perangkat keras, perangkat lunak serta pengguna. Dalam sebuah organisasi, sistem informasi tergolong sebagai perangkat lunak yang diimplementasi untuk merealisasikan strategi bisnis. Terdapat 3 klasifikasi sistem informasi yaitu:

1. Functional purpose: Untuk hal produksi, komersial, finansial ataupun marketing.
2. Objects of management: Dipakai dalam hal mengenai manajemen enterprise (pengaturan kantor, badan usaha, korporasi ataupun organisasi), computer-aided design, dan atau pengendalian proses bisnis.
3. Nature of the use of the resulting information: Sistem informasi yang tujuannya tergantung pada pemakaian informasi yang disediakan. Sebagai contoh adalah information retrieval yang dirancang untuk mengumpulkan, menyimpan serta mengirimkan informasi atas request dari pengguna.

Manfaat dari sistem informasi tergantung pada tujuan sistem tersebut dibuat. Berbagai macam sistem informasi dibuat untuk memenuhi kebutuhan seseorang atau sebuah perusahaan yang spesifik. Tidak hanya itu, manfaat-manfaat yang dihasilkan dari sebuah sistem informasi tidak dapat dinyatakan secara jelas (Ahlin, 2019). Tetapi, secara umum manfaat utama dari sebuah sistem informasi adalah untuk menyajikan informasi yang telah dikumpulkan dari satu atau lebih sumber dengan efisien dan efektif terhadap pengguna sistem informasi tersebut (Yatskiv & Gromule, 2016).

Perkembangan sistem informasi pada masa awal dimulai dari kehadiran komputer dalam berbagai bisnis. Tidak lama kemudian pertumbuhan teknologi komputer membuat integrasi sistem informasi tidak sebatas proses yang teknikal dan spesifik, dan mulai memainkan peran yang penting dalam segi manajemen

perusahaan. Lalu, perkembangan teknologi internet memungkinkan implementasi sistem informasi pada tingkat *website*. Seiring berjalannya waktu, teknologi *mobile* berkembang dengan pesat dan dengan dukungan teknologi internet, dan munculnya infrastruktur berdasarkan jasa mendorongnya perkembangan teknologi *cloud*, di mana sistem informasi dapat disediakan sebagai jasa perangkat lunak (SaaS, Software-as-a-Service) (Romero & Vernadat, 2016).

Menurut jurnal yang ditulis oleh Kiplie, Yatin, Angutim dan Hamid (2018), perancangan sistem (*system design*) menerangkan berbagai komponen, modul ataupun arsitektur fundamental dari sebuah sistem dalam bentuk desain diagram. Proses ini merupakan sinonim terhadap tahap kedua dari SDLC (System's Development). Pada jurnal ini, System Design dan System Development adalah sama dengan tahap kedua dan tahap ketiga SDLC: System Development dan System Implementation and Coding. Proses yang sama, namun dengan penamaan yang berbeda, di mana System Development pada jurnal tersebut adalah tahap *coding*, dan System Development pada SDLC merupakan tahap desain dari sistem

Pengembangan sistem (*system development*) merupakan serangkaian proses yang digunakan untuk mendefinisi, merancang, menguji-coba dan mengimplementasi sebuah aplikasi *software* yang baru. Proses ini termasuk segala pengembangan internal yang diperlukan untuk membangun sistem yang diperlukan (Kiplie et al., 2018).

Berdasarkan jurnal yang berjudul "Best Practices In Systems Development Lifecycle: An Analyses Based On The Waterfall Model" oleh Kramer (2018), pengembangan sistem (System Development) merupakan salah satu tahap dalam metode pengembangan SDLC, di mana merupakan sebuah proses yang spesifik dalam tahapan keseluruhan pengembangan sistem. Tahap ini dilaksanakan setelah tahap pertama (Requirements Gathering and Analysis), di mana hasil analisis terhadap kebutuhan sistem diubah menjadi desain sistem yang lengkap. Desain ini mendeskripsikan komponen-komponen utama sistem, dan fokus terhadap bagaimana mengimplementasi fitur-fitur yang diperlukan oleh sebuah sistem. Desain yang telah didapatkan akan digunakan dalam tahap berikutnya yaitu tahap implementasi sistem (Systems Implementation and Coding).

Ada dua kategori dari *programming language* jika dilihat dari tingkat abstraksi pada instruksi mesin, yaitu *low-level programming language* dan *high-level programming language*. *Low-level programming language* adalah kategori bahasa pemrograman di mana bahasa tersebut memiliki sedikit atau tidak ada abstraksi sama sekali. Ini memungkinkan sebuah mesin dapat langsung mengerti bahasa tersebut. Karena hal itu struktur bahasanya pun secara sintaks mirip dengan instruksi mesin murni. Contoh dari *low-level programming language* adalah *machine code* dan *assembly*. *High-level programming language* merupakan bahasa pemrograman yang memiliki tingkat abstraksi yang tinggi. Secara struktur bahasa, *high-level programming language* lebih mendekati bahasa manusia daripada instruksi mesin murni, membuatnya mudah dipakai untuk pengembangan aplikasi modern. Beberapa contoh dari *high-level programming language* adalah C, C++, C#, Python, Pascal, FORTRAN dan LISP (Trivedi, 2017).

2.2.2 SDLC (Software Development Life Cycle)

Prosedur yang tetap diperlukan dalam mengembangkan sebuah perangkat lunak dari awal hingga akhir. SDLC merupakan dasar panduan dari seluruh proses perancangan serta semua aktivitas yang perlu dilakukan oleh *developer* . Karena kebutuhan setiap perangkat lunak berbeda dan bervariasi, maka ada lebih dari satu variasi dari model SDLC yang digunakan untuk setiap jenis kebutuhan yang berbeda. Tentunya keterampilan untuk melakukan *coding* adalah dasar dari karir seorang *developer*. Namun dengan mengerti SDLC, ada banyak keterampilan baru yang dapat dipelajari dan diterapkan dalam mengembangkan perangkat lunak, di mana keterampilan baru tersebut bersifat menguntungkan dalam proses *engineering* perangkat lunak (Lemke, 2018).

Pada jurnal berjudul "Best Practices In Systems Development Lifecycle: An Analyses Based On The Waterfall Model" oleh Kramer (2018), metode SDLC Waterfall model merupakan salah satu metodologi pengembangan sistem yang paling *straight-forward*. Struktur dari model ini merupakan 6 tahapan yang merupakan: *Requirements Gathering and Analysis*, *Systems Development*, *Systems Implementation and Coding*, *Testing, Deployment* dan *Systems Operation and Maintenance*. Berikut merupakan garis besar dari keenam tahap tersebut:

1. Requirements Gathering and Analysis: Tahap di mana setiap kebutuhan dari sebuah aplikasi dikumpulkan dan dianalisis.
2. Systems Development: Merupakan tahap di mana dari seluruh kebutuhan yang telah dikumpulkan pada tahap pertama, desain dari aplikasi dibuat yang mengilustrasikan komponen-komponen, alur kerja serta proses berjalannya sistem.
3. Systems Implementation and Coding: Pada tahap ini, proses perkembangan aplikasi berjalan dalam bentuk penulisan kode dari aplikasi, dan mengikuti desain yang telah ditetapkan.
4. Testing: Merupakan tahap uji coba aplikasi dalam rangka memastikan sistem berjalan tanpa bug dan sesuai tujuan.
5. Deployment: Merupakan tahap di mana aplikasi dirilis dalam bentuk archive atau installer yang bersifat standalone. Hasil akhir tersebut dapat disebarkan ke pada pengguna.
6. Systems Operation and Maintenance: Adalah tahap di mana aplikasi sudah siap dipakai. Berbagai masalah teknikal (bug) yang ditemukan dapat diurus oleh developer dan perbaikan dirilis dalam versi baru aplikasi.

Dalam mengembangkan perangkat lunak, metode SDLC Waterfall model memiliki beberapa keunggulan (Kumar, 2018), yaitu:

1. User-friendly, mudah digunakan dan dimengerti
2. Mudah dalam melakukan management karena infleksibilitas model
3. Fase tidak melewati apa yang sudah dispesifikasi dalam tahapan model Waterfall
4. Model ini umum dan efisien jika digunakan dalam proyek berskala kecil di mana requirement sudah dimengerti dengan benar.

2.3 Sistem yang digunakan

Berikut merupakan berbagai *software*, *tools*, *framework*, dan bahasa pemrograman yang dipakai dalam proses merancang dan mengembangkan aplikasi desktop manipulasi GIF dan APNG:

2.3.1 Python

Dengan fitur dynamic-*typing,* sintaks yang singkat serta ekspresif, Python merupakan bahasa pemrograman populer yang telah digunakan pada berbagai sistem untuk mendukung segala bentuk penelitian dalam bidang ilmu pengetahuan. Bahasa ini dapat dipakai untuk berbagai kebutuhan dengan skala yang bervariasi, mulai dari *script* kecil, sampai ke aplikasi berskala besar seperti sebuah sistem ataupun *library*. (Blomqvist, Dulak, Friis, & Hargus, 2017).

Sebagai bahasa pemrograman yang bersifat *interpreted*, Python memudahkan proses diagnosa variabel pada saat *runtime* dengan cepat. (Short, Bayer, & Burns, 2018). Sistem operasi *Windows, Linux* dan *Mac OS* mendukung pengembangan dan *runtime* aplikasi berbasis Python (Zhang, Moynihan, Ernest, & Gutenson, 2017).

Dengan koleksi *standard library* yang sangat besar, Python memiliki dukungan *library* untuk berbagai *task* yang bervariasi, termasuk RegEx (Regular Expression), *unit testing*, *database*, *multithreading*, CGI (Common Gateway Interface), email, kriptografi dan manipulasi gambar (Guan, Zhou, & Zhou, 2019).

2.3.2 Pillow

Pillow merupakan *library* dari bahasa pemrograman Python yang mendukung pemrosesan, manipulasi dan pembuatan gambar digital dalam banyak format. Pillow dikembangkan sebagai *fork* dari proyek PIL (Python Imaging Library), di mana Pillow menyediakan dukungan untuk versi Python 3.x (Nair, Athul, & Kartha, 2018).

2.3.3 HTML, CSS dan JavaScript

Teknologi web tidak akan ada tanpa HTML (HyperText Markup Language), CSS (Cascading Style Sheets) dan JavaScript. Ketiga bahasa tersebut digunakan dalam membuat, mendesain dan mempresentasikan sebuah website. HTML adalah bahasa *scripting* yang digunakan untuk mendefinisikan struktur dan konten dari sebuah website dengan menggunakan ‘tag’, ‘attribute’ dan ‘element’. CSS merupakan bahasa yang memodifikasi tampilan dari konten dan struktur web yang telah didefinisikan oleh HTML. Dengan CSS, website dapat didesain dengan

spesifikasi font, warna, posisi, margin dan berbagai atribute lain pada setiap elemen HTML yang ada. Sedangkan JavaScript adalah bahasa *scripting* pada sisi *client* yang menyediakan interaktivitas dan dinamisasi konten dari website (S. A. Garba, 2017).

2.3.4 Electron Framework

Electron adalah *framework open-source*, yang dikembangan dan di-*maintain* oleh Github. Electron dikembangkan dengan tujuan agar *developer* dapat mengembangkan aplikasi desktop GUI *cross-platform* dengan menggunakan bahasa pemrograman web yaitu HTML, CSS dan JavaScript. Aplikasi yang dikembangkan oleh Electron dapat berjalan pada sistem Windows, Linux dan MacOS (Kredpattanakul & Limpiyakorn, 2019).

2.3.5 Vue.js Framework

Vue.js merupakan *framework front-end* JavaScript yang berbasis model MVVM (Model-View-ViewModel). Vue.js menyediakan implementasi data-binding dua arah di antara data dan DOM. Fitur components dari Vue.js menyediakan abstraksi setiap elemen tampilan menjadi sebuah *component* yang dapat dipakai berulang-kali, sehingga mengurangi repetisi penulisan kode untuk mendefinisikan tampilan yang sama (Song, Zhang, & Xie, 2019).

2.3.6 Webpack

Webpack adalah *module bundler*, di mana secara konsep berfungsi mengumpulkan semua *module* dari *source code* yang ada menjadi sebuah file statis. Jenis *source code* dan hasil output file statis yang diperlukan ditentukan oleh empat konfigurasi di Webpack yaitu *entry*, *loader*, *plugin*, dan *output*. Entry merupakan titik awal, yang mencakup *source code* serta *dependency* yang digunakan dalam aplikasi JavaScript tersebut. Loader merupakan konfigurasi di mana kita dapat menambahkan *static asset* seperti CSS, HTML ataupun gambar. Konfigurasi Plugin menyediakan tambahan fungsionalitas kepada Webpack. *Output* meng-*compile* semua yang telah dikonfigurasi di Entry, Output dan Loader menjadi sebuah *bundle* berupa satu file JavaScript (Hoque, 2017).

2.3.7 ZeroRPC

ZeroRPC merupakan sebuah *library* yang dikembangkan dari *framework* komunikasi ZeroMQ. Tersedia untuk bahasa pemrograman Python dan Node.JS, ZeroRPC menyediakan protokol komunikasi antar aplikasi dengan menggunakan *interface* yang berupa *socket*. Komunikasi berbasis protokol TCP ini dapat menghubungkan dua aplikasi baik dalam komputer yang sama, ataupun komputer yang berbeda (Gaspari, 2016). Informasi yang dikirim menggunakan ZeroRPC harus dapat diserialisasikan berdasarkan format JSON (Stipe, Eugen, & Zeljko, 2017).