BAB 2

LANDASAN TEORI

2.1 Freelance

Freelance adalah pekerjaan lepas yang bekerja tanpa terikat kontrak pada perusahaan dalam jangka waktu yang panjang. Pekerja lepas atau freelancer dapat bekerja dimanapun dan kapanpun tanpa terikat jam kerja harian. Pekerja lepas dapat dianggap profesi dengan tipe *hybrid* dari karyawan dan pengusaha. Di satu sisi, mereka adalah karyawan karena mereka hampir selalu disewa oleh perusahaan (besar) untuk bekerja selama periode yang tidak menjual apa pun selain pengetahuan profesional yang tak berwujud, yang berbeda dari pengusaha lain dan wiraswasta yang menjual produk berwujud kepada pelanggan [1].

Pekerjaan yang dapat diambil oleh para pekerja lepas berbagai macam tergantung pengetahuan, kemampuan, dan profesi asli yang ada pada para pekerja. Beberapa kategori yang biasa popular di kalangan Freelance adalah akunting, SEO, Multimedia, Penulis konten, Pembuat digital konten, dan sebagainya. Kategori pekerjaan telah dirangkum dalam distribusi 2000 proyek acak yang berlabel manual pada **Gambar 2.1.**

Category	Job Type	Description	Count	%
Legitimate [§A.1]	Web Design/Coding	Create, modify, or design a Web site	769	38.5
	Multimedia Related	Complete multimedia-related task (e.g., Flash)	265	13.2
	Private Jobs	Jobs designated for a particular worker	138	6.9
	Desktop/Mobile Applications	Create a desktop or mobile application	100	5.0
	Legitimate Miscellaneous	Miscellaneous jobs	177	8.8
Accounts [§A.2]	Account Registrations	Create accounts with no defined requirements	22	1.1
	Human CAPTCHA Solving	Requests for human CAPTCHA solving	19	0.9
	Verified Accounts	Create verified accounts (e.g. phone)	14	0.7
SEO [§A.3]	SEO Content Generation	Requests for SEO content (e.g., articles, blogs)	195	9.8
	Link Building (Grey Hat)	Get backlinks using grey hat methods	53	2.6
	Link Building (White Hat)	Get backlinks using no grey/black hat methods	20	1.0
	SEO Miscellaneous	Nonspecific SEO-related job postings	61	3.0
Spamming [§A.4]	Ad Posting	Post content for human consumption	25	1.2
	Bulk Mailing	Send bulk emails	8	0.4
OSN Linking [§A.5]	Create Social Networking Links	Get friends/subscribers/fans/followers/etc.	33	1.7
Misc [§A.6]	Abuse Tools	Tools used for abuse (e.g., CAPTCHA OCR)	41	2.1
	Clicks/CPA/Leads/Signups	Get clicks, emails, zip codes, signups, etc.	32	1.6
	Manual Data Extraction	Manually visit websites and scrape content	21	1.1
	Gather Email/Contact Lists	Research contact details for targeted people	17	0.9
	Academic Fraud	Write essays, code homework assignments, etc.	10	0.5
	Reviews/Astroturfing	Create positive reviews	1	0.1
	Other Malicious	Miscellaneous jobs with malicious intentions	35	1.8

Gambar 2.1 Tabel Kategori Pekerjaan Freelance terpopuler

Sumber : Dirty Jobs: The Role of Freelance Labor in Web Service Abuse

2.2 Marketplace

Marketplace adalah sebuah website atau aplikasi yang memfasilitasi proses jual – beli antara toko dengan customer. Marketplace memiliki konsep yang sama namun proses transaksi yang terjadi bersifat online dan menggunakan pihak ketiga (third party) untuk keamanan transaksi yang ditawarkan. Pada dasarnya, pemilik marketplace tidak bertanggung jawab atas barang – barang dan value yang dijual pada marketplace karena marketplace hanya menyediakan fasilitas untuk menghubungkan penjual dan pembeli secara cepat dan efisien. Namun pemilik marketplace biasanya bertanggung jawab atas setiap transaksi yang terjadi pada platform marketplacenya yang dibangun. Pemilik marketplace dapat bertanggung jawab atas kesalahan atau transaksi yang terjadi karena tujuan utama marketplace sendiri diantaranya keamanan bertransaksi dan menyediakan fitur transaksi dengan pihak ketiga yaitu marketplace itu sendiri. Salah satu alasan mengapa marketplace online digandrungi

banyak orang adalah karena kemudahan dan kenyamanan dalam penggunaan. Online marketplace bisa digambarkan dengan department store.

Marketplace menjadi popular pada tahun 1995 yang dimana Amazon dan Ebay mulai terkenal dan dipakai semua orang. Di tahun yang sama, sebuah bank Amerika dengan nama The Presidential Bank meluncurkan online banking pertama di dunia. Pada tahun 1998, PayPal diluncurkan dan memberi kemudahan lebih banyak untuk transaksi online. Di Asia sendiri, Jack Ma meluncurkan Alibaba di China pada tahun 1999. Menurut Pew Internet Research Center pada tahun 2016, 79% orang Amerika melakukan pembelian online dibandingkan hanya 22% di tahun 2000 [14]. Peningkatan dramatis tersebut menunjukan pengaruh, pertumbuhan, dan kekuatan marketplace dan e-commerce online secara pesat.

2.3 Teknologi Web

Teknologi Web adalah teknologi yang berhubungan dengan antarmuka untuk menghubungkan server dan client. Web aplikasi membutuhkan tool utama sebagai jembatan untuk mulai berselancar (browsing) pada jaringan internet yaitu Browser. Banyak beberapa jenis web browser diantaranya yang paling sering digunakan adalah Chrome, Firefox, Opera, dan IE. Dasar – dasar pada teknologi web diantaranya:

1. HTML

   Hypertext Markup Language adalah bahasa yang digunakan untuk mendeskripsikan dan menentukan konten dari halaman web.
2. CSS

   Cascading Style Sheet digunakan untuk menentukan tampilan/gambar pada konten web
3. HTTP

   Hypertext Transfer Protocol digunakan untuk mengantar HTML dan komponen konten lainnya
4. Web API

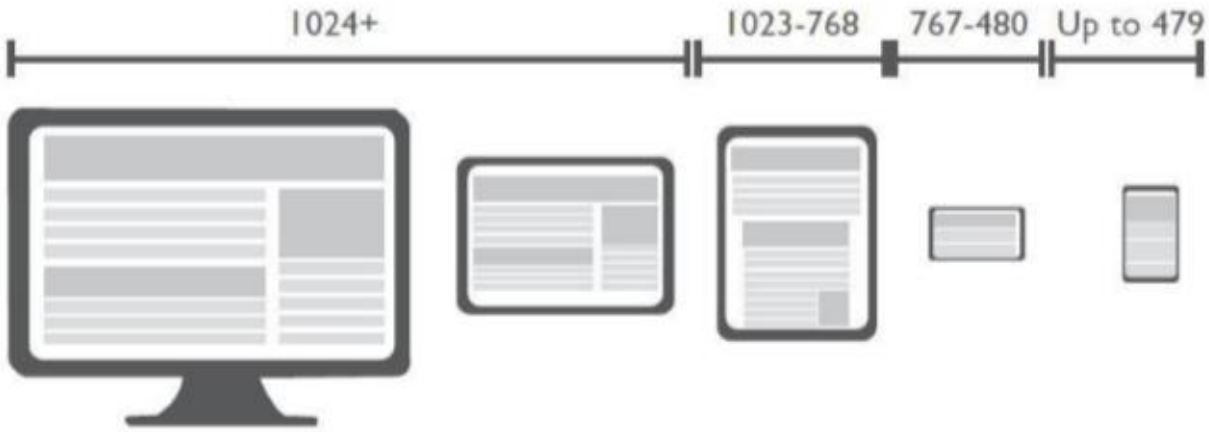
   Web Application Programming Interface digunakan untuk melaksanakan tugas - tugas beragam seperti memanipulasi DOM, dan memainkan audio maupun video.

Dalam artian abstrak, Web adalah kumpulan dokumen yang sangat banyak, beberapa di antaranya dihubungkan oleh tautan link [15]. Dokumen-dokumen ini diakses oleh browser

Web. Web berisikan semua kumpulan perangkat lunak dan protokol yang telah diinstal pada sebagian besar.

2.4 Desain Web Responsive

Responsive Web Design merupakan sebuah situs atau web app yang memiliki domain yang sama, konten yang sama, dan sintaks yang sama, kurang lebih dimanipulasi oleh Javascript atau CSS3 Media Queries. Responsive Web Design menjadi dasar dari penyebaran sebuah situs web. RWD adalah teknologi client-server. Konsep utama dari Responsive Web Design yaitu mengadaptasi content layout dengan minimum request dari server [16]**.** RWD bekerja dengan maksimal saat merender konten pada website untuk setiap perangkat gadget dan mobile. Secara umum, konten pada situs website tidak boleh dikurangi begitu banyak karena akan menjadi sulit dibaca. Lebih baik disesuaikan dimensi layar seperti pada Gambar 2.2. Dengan kata lain, RWD harus terotomatisasi dalam membentuk kembali kontennya untuk kegunaan dan dampak maksimal.



Gambar 2.2 Content Layout di Media yang Berbeda (Musti, Kashyap, 2013)

Membuat situs web yang responsive membutuhkan proportion-based grid, gambar yang fleksibel, dan CSS3 Media-Queries. Proportion-based grid sering disebut "*Grid Fluida*". Gagasan dasarnya adalah bahwa dimensi semua elemen harus diberikan dalam satuan relatif, yaitu dalam persen (%), sedangkan satuan tetap seperti piksel harus dihindari. CSS Media-Quieries memungkinkan untuk membuat style yang berbeda bagi berbagai dimensi dan layar perangkat. Dengan cara itu, halaman web ditampilkan menggunakan gaya yang paling disesuaikan dengan dimensi layar pengguna.

2.5 Freelancing Marketplace

Freelancing marketplace adalah situs web yang mencocokan dengan pembeli layanan yang dapat dikirim secara elektronik dengan penjual atau freelancer yang menawarkan layanan berdasarkan per-pekerjaan atau dengan tarif per jam tetap [2]. Penggunaan freelancing marketplace telah melonjak pesat di beberapa tahun terakhir. Situs – situs website seperti ini menawarkan pekerja yang berkemampuan khusus untuk mendapatkan uang tanpa kewajiban dan potensi bias social yang terkait dengan kerangkan kerja biasa serta tradisional.

Freelancing marketplace menawarkan dua (2) manfaat yang didapatkan bagi pekerja atau Freelancer. Yang pertama fleksibilitas, fleksibilitas dalam mengambil pekerjaan sesuai kemauan, kontrak yang ingin dinegosiasikan, dan bagaimana pekerjaan akan diselesaikan [17]**.** Memang freelancing marketplace menyediakan kesempatan kerja bagi pekerja yang mungkin kehilangan haknya karena masalah pasar tenaga kerja tradisional. Manfaat kedua adalah perjanjian kesetaraan. Banyak penelitian telah mengungkap diskriminasi di pasar tenaga kerja tradisional [18], di mana sadar dan tidak sadar malah dapat membatasi peluang kerja yang tersedia bagi para pekerja dari kelompok yang terpinggirkan. Sebaliknya, platform online dapat bertindak sebagai perantara netral yang mencegah hal tersebut terjadi**.**

2.6 Blockchain

Pada tahap ini akan diterangkan mengenai definisi Blockchain, sejarah Blockchain, dan cara kerja Blockchain

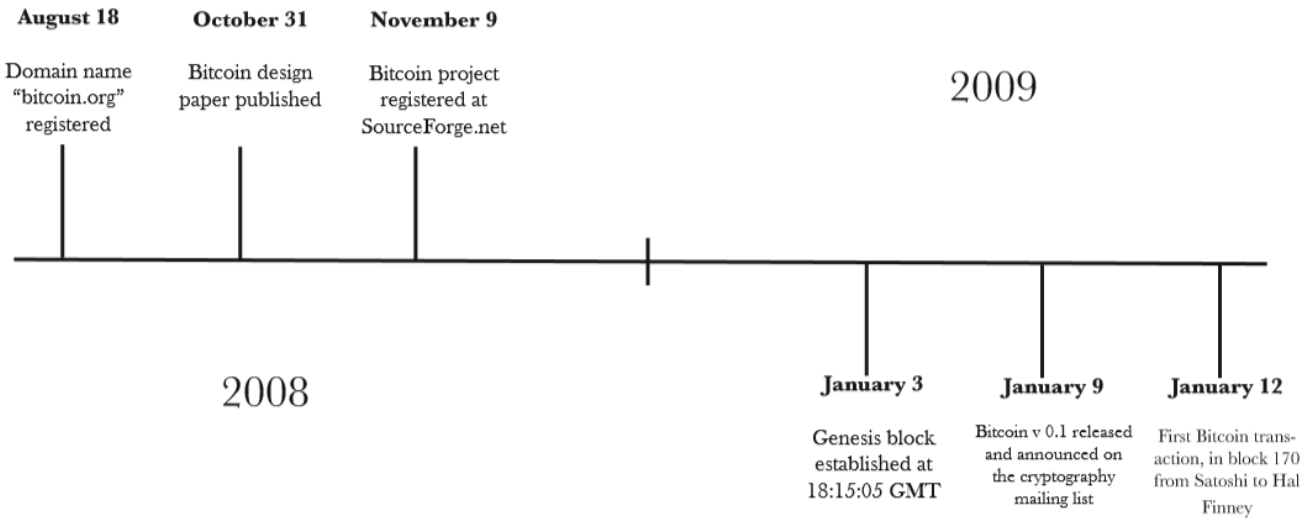
2.6.1 Definisi Blockchain

Blockchain merupakan struktur data yang memungkinkan membuat buku besar digital (*hyperledger*) dan membagikannya diantara jaringan dengan pihak independent [19]. Setiap transaksi dalam buku besar publik diverifikasi oleh consensus mayoritas peserta dalam sistem. Setelah dimasukan, informasi tidak akan pernah bisa diubah atau dihapus. Blockchain berisi catatan tertentu dan dapat diverifikasi dari setiap transaksi yang pernah dilakukan. Bitcoin, mata uang digital peer-to-peer terdesentralisasi, adalah contoh paling populer yang menggunakan teknologi blockchain. Bitcoin mata uang digital itu sendiri sangat kontroversial

tetapi teknologi blockchain yang mendasarinya telah bekerja dengan sempurna dan menemukan berbagai aplikasi baik di dunia finansial maupun non-finansial.

2.6.2 Sejarah Blockchain

Pada tahun 2008, seorang individu (atau grup) yang menulis dengan nama Satoshi Nakamoto menerbitkan sebuah makalah berjudul "Bitcoin: A Peer-To-Peer Electronic Cash System". Makalah ini menjelaskan versi peer-to-peer dari uang tunai elektronik yang akan memungkinkan pembayaran online untuk dikirim langsung dari satu pihak ke pihak lain tanpa melalui lembaga keuangan. Bitcoin adalah realisasi pertama dari konsep ini. Sekarang "cryptocurrency" adalah label yang digunakan untuk menggambarkan semua jaringan dan media pertukaran yang menggunakan kriptografi untuk mengamankan transaksi seperti terhadap sistem-sistem di mana transaksi disalurkan melalui entitas tepercaya yang terpusat. Penulis makalah ingin tetap menjadi anonim dan karenanya tidak ada yang tahu Satoshi Nakamoto sampai hari ini. Beberapa bulan kemudian, sebuah program open source yang mengimplementasikan protokol baru dirilis. Siapa pun dapat menginstal program sumber terbuka ini dan menjadi bagian dari jaringan peer-to-peer bitcoin. Ini telah tumbuh dalam popularitas sejak itu. Popularitas Bitcoin tidak pernah berhenti meningkat sejak saat itu. Selain itu, teknologi BlockChain yang mendasari sekarang menemukan berbagai aplikasi baru di luar keuangan.



Gambar 2.3 Sejarah Bitcoin

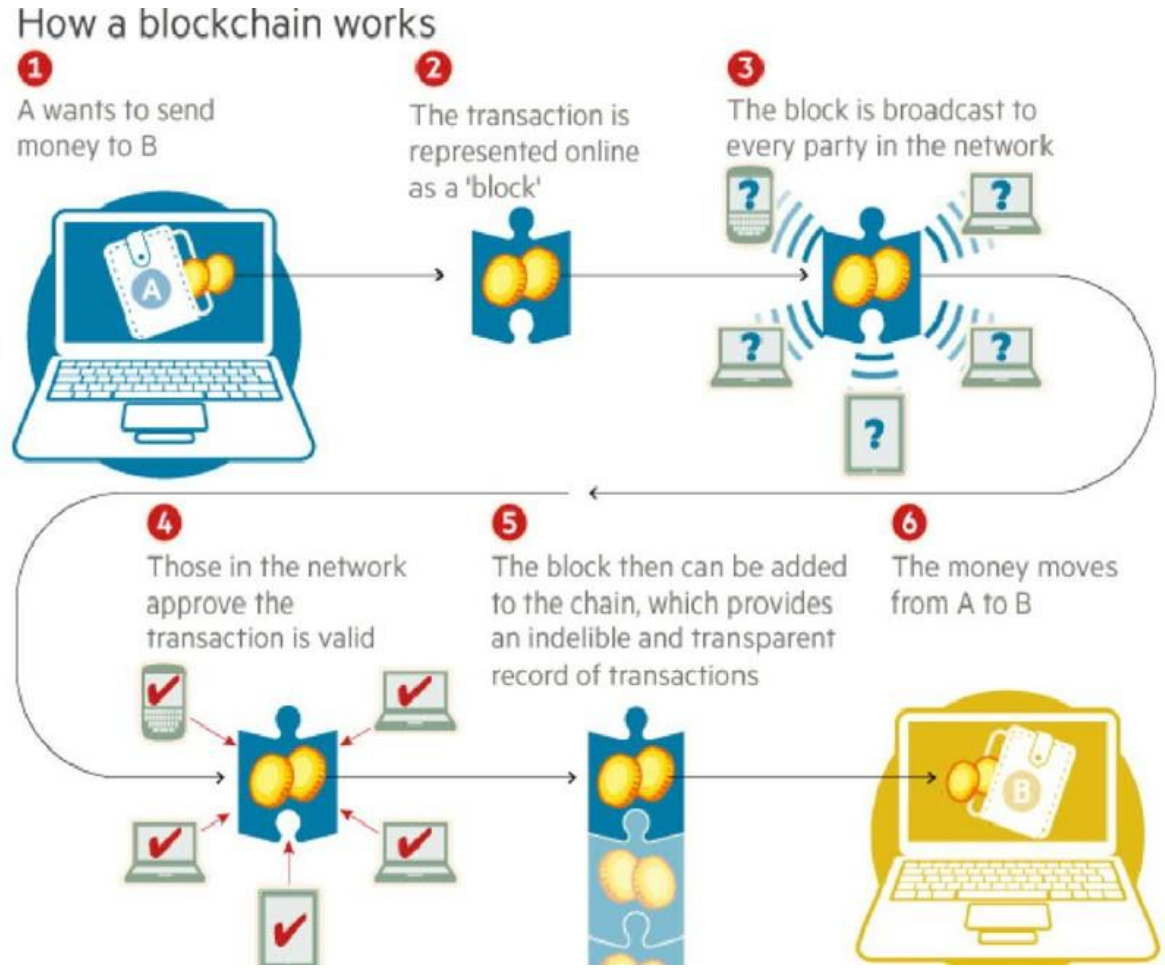
Sumber: Blockchain Technology: Beyond Bitcoin

2.6.3 Cara Kerja Blockchain

Bitcoin menggunakan bukti kriptografi alih-alih mekanisme trust-in-the-third-party untuk dua pihak yang bersedia melakukan transaksi online melalui Internet. Setiap transaksi dilindungi melalui tanda tangan digital, dikirim ke "public key" penerima, dan ditandatangani secara digital menggunakan "private key" pengirim. Untuk menghabiskan uang, pemilik cryptocurrency perlu membuktikan kepemilikannya atas "private key". Entitas yang menerima mata uang digital kemudian memverifikasi tanda tangan digital, yang menyiratkan kepemilikan "private key" yang sesuai, dengan menggunakan "private key" pengirim pada transaksi masing-masing. Kriptografi di belakang protokol didasarkan pada modulo enkripsi asimetris matematika di mana 'kunci' untuk mengenkripsi pesan atau transaksi berbeda dari 'kunci' untuk mendekripsi [5]. Sistem desentralisasi blockchain berjalan dengan jaringan peer-to-peer dimana semua pengguna dengan node block berperan penting dalam penyimpanan dan distribusi data pada satu jaringan tanpa memanfaatkan distribusi server secara sentral. Selain itu, tiap block melakukan verifikasi dan dikelola menggunakan otomatisasi dan protokol tata kelola bersama [6].

Setiap transaksi dikirim ke setiap node yang ada dalam jaringan Blockchain dan kemudian dicatat dalam buku besar publik (*Hyperledger*) setelah diverifikasi. Setiap transaksi perlu diverifikasi untuk validasi sebelum dicatat dalam buku besar. Node verifikasi perlu memastikan dua hal sebelum merekam transaksi apa pun :

1. Pengirim memiliki cryptocurrency melalui verifikasi tanda tangan digital pada transaksi.
2. Pengirim memiliki jumlah cryptocurrency yang cukup dalam akunnya untuk melakukan pengiriman, melalui pemeriksaan transaksi terhadap akun pengirim atau “publick key”, yang terdafar di *hyperledger* (buku besar). Langkah tersebut memastikan bahwa ada saldo yang cukup diakunnya sebelum menyelesaikan transaksi.



Gambar 2.4 Cara Kerja Bitcoin

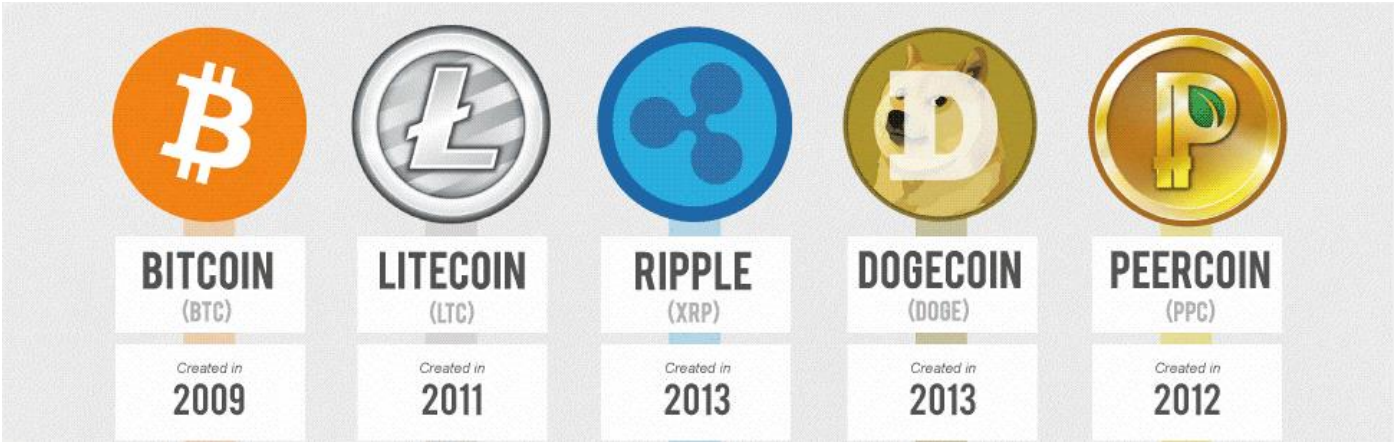
Sumber : A. I. Review, "Applied Innovation Review," no. 2, 2016

2.7 Cryptocurrency

Cryptocurrency merupakan mata uang virtual yang digunakan sebagai mata uang alternatif dimana mata uang tersebut dihasilkan dan diperdagangkan melalui proses kriptografi. Kebanyakan dari Cryptocurrency tersebut bersifat desentralisasi dalam jaringan berbasis computer peer-to-peer dan berdasarkan pada teknologi kriptografi open source yang tidak bergantung pada otoritas pusat seperti bank pusat atau institusi lainnya.

Sebelum muncul era digital, alat pembayaran masih berupa benda fisik. Peranan uang memiliki 3 fungsi yaitu sebagai alat pembayaran, satuan unit, penyimpan nilai (Conway, 2014). Berdasarkan jurnal yang ditulis oleh Joey Conway yang berjudul Beginners Guide to Cryptocurrencies, pada kisaran tahun 1982, David Chaum dari University of california pertama kali mempublikasikan mengenai ide pembuatan sebuah metode pembayaran berbasiskan kriptografi yang dapat menjaga kerahasiaan data pemiliknya. Dan pada tahun

1990, David Chaum membuat perusahaan yang diberi nama "DigiCash", dengan produk utamanya yaitu membuat sebuah alat pembayaran menggunakan smart card dan electronic cash (e-cash). Jenis pembayaran digital sendiri terdiri dari 2 macam, yang pertama virtual currency dalam bentuk uang digital seperti uang yang digunakan pada aplikasi video game, t-cash (telkonsel cash), XL tunai, Indosat Dompetku, dan beberapa alat pembayaran digital lainnya. Jenis virtual currency ini bersifat sentralisasi, diatur dan dikelola oleh suatu lembaga maupun perusahaan (Conway, 2014). Yang kedua adalah virtual currency yang menggunakan teknologi kriptografi atau dikenal dengan sebutan cryptocurrency dimana untuk setiap transaksi data – data dan informasi akan dienkripsi atau disandikan menggunakan algoritma kriptografi tertentu. Blockchain diawali dengan penerapan mata uang cryptocurrency. Bitcoin merupakan cryptocurrency pertama dan terpopuler yang pernah diciptakan. Selain Bitcoin, mata uang cryptocurrency yang lain adalah Litecoin, Ethereum, Dogecoin, Ripple, NXT, Peercoin, dan masih banyak lagi.



Gambar 2.5 Contoh Mata Uang Digital Cryptocurrency

Sumber: https://kampungpasarmodal.com/

Teknologi *cryptocurrency* menggunakan bitcoin menawarkan alternatif yang canggih sehingga bila diterapkan efisiensi mudah tercapai. Berikut tabel perbandingan antara sistem uang elektronik yang saat ini digunakan dengan konsep uang elektronik menggunakan teknologi *cryptocurrency* berdasarkan hasil analisis perbandingan dalam Jurnal penelitian Ferry Mulyanto dengan judul "Pemanfaatan Cryptocurrency Sebagai Penerapan Mata Uang Rupiah Kedalam Bentuk Digital Menggunakan Teknologi Bitcoin" [20].

Tabel 2.1 Analisis Perbandingan Uang Elektronik dan Cyptocurrency

Faktor Penilaian	Uang Elektronik Saat Ini	Uang Elektronik Menggunakan Cryptocurrency
Keamanan	Rentan terjadi manipulasi data, tergantung teknologi masing – masing penyedia layanan	Cukup aman karena menggunakan kriptografi
Kecepatan	Relative terhadap penyedia layanan namun cenderung lebih cepat	Cenderung lebih lama dibandingkan dengan uang elektronik umumnya seluruh perbankan yang terdaftar pada Bank Indonesia.
Biaya	Biaya setiap penyedia beragam	Biaya cenderung lebih murah karena penyedia tidak perlu membangun infrastruktur masing – masing
Kompatibilitas	Tidak semua penyedia layanan bisa saling menukung transaksi finansial	Semua penyedia layanan perbankan dapat saling sinkronisasi data nasabah menggunakan konsep shared ledger
Kemudahan	Cukup mudah dan cepat untuk melakukan transaksi, karena alat pembayaran cukup didekatkan dengan terminal akses pembayaran	Lebih cepat dari uang elektronik saat ini, cukup memasukan *public address* tujuan pengiriman dana.

2.8 Ethereum

Ethereum adalah proyek yang mencoba untuk membangun teknologi umum; teknologi dimana semua konsep mesin berbasis transaksi dapat dibangun [21]. Selain itu

Ethereum juga bertujuan untuk memberikkan pengembang akhir (end developer) sistem end-to-end yang terintegrasi secara ketat untuk membangun perangkat lunak.

2.8.1 Block, State, dan Transaksi

2.8.1.1 World State

World State (state) adalah pemetaan antara alamat (identifier 160 bit) dan state akun (struktur data yang direalisasikan). Meskipun tidak disimpan di blockchain, diasumsikan bahwa implementasi akan mempertahankan pemetaan ini. State akun terdiri dari 4 bidang berikut:

a. Nonce

   Nilai skalar sama dengan jumlah transaksi yang dikirim dari alamat ini atau, dalam kasus akun dengan kode terkait, jumlah contract creation yang dibuat oleh akun.

b. Balance

   Nilai skalar sama dengan jumlah *Wei* yang dimiliki oleh alamat ini.

c. storageRoot

   Hash 256-bit dari simpul akar pohon Merkle Patricia yang mengkodekan konten penyimpanan akun (pemetaan antara nilai-nilai integer 256-bit), dikodekan ke dalam trie sebagai pemetaan dari hash Keccak 256-bit dari 256 kunci integer bit ke nilai integer 256-bit yang dikodekan RLP.

d. codeHash

   Hash dari kode EVM akun ini adalah kode yang dieksekusi jika alamat ini menerima panggilan pesan; itu tidak berubah dan dengan demikian, tidak seperti semua bidang lainnya, tidak dapat diubah setelah konstruksi. Semua fragmen kode seperti itu terkandung dalam database negara di bawah hash yang sesuai untuk pengambilan nanti.

2.8.1.2 Transaksi

Suatu transaksi adalah instruksi tunggal yang ditandatangani secara kriptografis yang dibangun oleh seorang aktor secara eksternal dengan ruang lingkup Ethereum. Meskipun diasumsikan bahwa aktor eksternal utama akan bersifat manusia, alat perangkat lunak akan digunakan dalam konstruksi dan penyebarannya. Ada dua jenis transaksi: transaksi yang menghasilkan panggilan pesan dan transaksi yang menghasilkan akun baru dengan kode

terkait (dikenal secara informal sebagai ‘contract creation’). Kedua jenis tersebut dispesifikasikan dalam beberapa bidang:

a. Nonce

   Nilai skalar sama dengan jumlah transaksi yang dikirim oleh pengirim.

b. gasPrice

   Nilai skalar sama dengan jumlah *Wei* yang harus dibayarkan per unit gas untuk semua biaya perhitungan yang timbul sebagai akibat dari pelaksanaan transaksi ini

c. gasLimit

   Nilai skalar sama dengan jumlah gas maksimum yang harus digunakan dalam melaksanakan transaksi ini. Ini dibayar di muka, sebelum perhitungan dilakukan dan mungkin tidak akan ditingkatkan nanti.

d. To

   Alamat 160-bit dari penerima panggilan pesan atau, untuk transaksi contract creation.

e. Value

   Nilai skalar sama dengan jumlah *Wei* yang akan ditransfer ke penerima panggilan pesan atau, dalam hal pembuatan kontrak, sebagai dana abadi ke akun yang baru dibuat.

f. v, r, s

   Nilai yang sesuai dengan tanda tangan transaksi dan digunakan untuk menentukan pengirim transaksi.

2.8.1.3 Block

Blok dalam Ethereum adalah kumpulan potongan informasi yang relevan (dikenal sebagai header blok), bersama dengan informasi yang berkaitan dengan transaksi yang terdiri, dan satu set header blok U lainnya yang diketahui memiliki induk sama dengan induk blok sekarang. Header block mengandung beberapa informasi yaitu:

a. parentHash

   Hash Keccak 256-bit dari header blok induk, secara keseluruhan.

b. ommersHash

   Hash Keccak 256-bit dari bagian daftar ommers dari blok ini.

c. Beneficiary

Alamat 160-bit di mana semua biaya yang dikumpulkan dari keberhasilan penambangan blok ini akan ditransfer

d. stateRoot

Hash Keccak 256-bit dari node root dari state trie, setelah semua transaksi dieksekusi dan finalisasi diterapkan.

e. transactionsRoot

Hash Keccak 256-bit dari simpul akar dari struktur trie diisi dengan setiap transaksi di bagian daftar transaksi blok.

f. receiptsRoot

Hash Keccak 256-bit dari node root dari struktur trie diisi dengan penerimaan dari setiap transaksi di bagian daftar transaksi blok.

g. logsBloom

Filter Bloom terdiri dari informasi yang dapat diindeks (alamat log dan topik log) yang terkandung dalam setiap entri log dari penerimaan setiap transaksi dalam daftar transaksi.

h. Difficulty

Nilai skalar yang sesuai dengan tingkat kesulitan blok ini. Ini dapat dihitung dari tingkat kesulitan blok sebelumnya dan cap waktu.

i. Number

Nilai skalar sama dengan jumlah blok leluhur. Blok genesis memiliki angka nol.

j. gasLimit

Nilai skalar sama dengan batas pengeluaran gas saat ini per blok.

k. gasUsed

Nilai skalar sama dengan total gas yang digunakan dalam transaksi di blok ini.

l. Timestamp

Nilai skalar sama dengan output pada waktu Unix () pada awal blok ini.

m. extraData

Byte array acak yang berisi data yang relevan dengan blok ini. Harus 32 byte atau kurang.

n. mixHash

Hash 256-bit yang terbukti dikombinasikan dengan notce bahwa jumlah komputasi yang cukup telah dilakukan pada blok ini.

o. Nonce

Hash 64-bit yang terbukti dikombinasikan dengan campuran-hash bahwa jumlah komputasi yang cukup telah dilakukan pada blok ini.

2.8.2 Gas dan Payment

Untuk menghindari masalah penyalahgunaan jaringan dan untuk menghindari pertanyaan yang tak terhindarkan yang berasal dari kelengkapan Turing, semua perhitungan yang dapat diprogram dalam Ethereum dikenakan biaya.

Setiap transaksi memiliki jumlah gas tertentu yang terkait dengannya: gasLimit. Ini adalah jumlah gas yang secara implisit dibeli dari saldo akun pengirim. Pembelian terjadi di gasPrice yang sesuai, juga ditentukan dalam transaksi. Transaksi dianggap tidak sah jika saldo akun tidak dapat mendukung pembelian semacam itu. Ini bernama gasLimit karena setiap gas yang tidak digunakan pada akhir transaksi dikembalikan (dengan tingkat pembelian yang sama) ke akun pengirim. Gas tidak ada di luar pelaksanaan transaksi. Jadi untuk akun dengan kode tepercaya yang terkait, batas gas yang relatif tinggi dapat ditetapkan dan dibiarkan sendiri [21].

Secara umum, Ether yang digunakan untuk membeli gas yang tidak dapat dikembalikan dikirim ke alamat penerima, alamat akun yang biasanya di bawah kendali penambang (miner). Transactor bebas menentukan gasPrice yang mereka inginkan, namun penambang bebas mengabaikan transaksi yang mereka pilih. Oleh karena itu, harga gas yang lebih tinggi pada transaksi akan membebani pengirim lebih dalam hal Ether dan memberikan nilai lebih besar kepada penambang dan dengan demikian akan lebih mungkin dipilih untuk dimasukkan oleh lebih banyak penambang. Para penambang, secara umum, akan memilih untuk mengiklankan harga gas minimum yang mereka akan lakukan transaksi dan transactor akan bebas menentukan harga ini dalam menentukan berapa harga gas yang akan ditawarkan. Karena akan ada distribusi (berbobot) dari harga gas minimum yang dapat diterima, para transaktor tentu akan memiliki trade-o untuk dilakukan antara menurunkan harga gas dan memaksimalkan peluang bahwa transaksi mereka akan ditambang pada waktu yang tepat.

2.9 Hyperledger

Hyperledger adalah proyek open source yang dibuat untuk membantu memajukan teknologi blockchain lintas industri (*cross-industry*). Hyperledger merupakan kolaborasi open

source global yang melibatkan pemimpin dari berbagai industri [22]. Teknologi open source seperti *Hyperledger* dan lainnya memberikan beberapa manfaat untuk industri antara lain yaitu meminimalisir biaya konsumsi software, inovasi dan ekstentabilitas, pengembangan yang berlanjut, keamanan, dan kecepatan pengembangannya. Proyek teknologi Hyperledger adalah upaya komunitas open source terbaru yang didukung oleh puluhan bank, perusahaan asuransi, dan produsen besar yang membahas masalah ini. Ini termasuk fitur untuk memastikan kepercayaan, transparansi, dan smart contract. Teknologi *HyperLedger* dapat mengurangi waktu dalam menyelesaikan transaksi besar dari hitungan hari ke detik. Teknologi *HyperLedger* juga dapat mengurangi pos biaya dengan menghilangkan perantara yang menambah overhead dan pengolahan serta untuk mengurangi risiko dari gangguan, penipuan dan cybercrime, karena terdapat perhitungan matematis yang mendasari pada lapisan perlindungan untuk setiap transaksi. Sebuah catatan keuangan yang didistribusikan bersama dapat meningkatkan kepercayaan karena direplikasi oleh semua pihak dalam transaksi. Peserta hanya dapat menambahkan ke registry ini, namun tidak dapat melakukan modifikasi. Dalam model transaksi tradisional, hacker hanya perlu mengubah entri dalam database untuk mengalihkan dana. Dalam pendekatan blockchain, perubahan hanya ditambahkan ke registri, daripada ditulis langsung ke database.

Catatan besar didistribusikan dengan menetapkan bagian historis dalam rantai pasokan. Jika ditemui masalah di bagian tertentu seperti cacat produk, produsen bisa cepat merespon pemasok, melacak tanggal pembuatan dan jalur produksi yang bertanggung jawab. Hal tersebut akan mempermudah dalam mempersempit ruang lingkup penarikan produk. Selanjutnya, pemasok yang kurang bertanggung jawab tidak akan mampu membuat perubahan ke database lokal mereka untuk menyembunyikan kesalahan karena catatan dibagi bersama semua mitra dalam transaksi. Proyek *Hyperledger* juga memiliki dukungan untuk transaksi yang memerlukan kerahasiaan khusus. Identitas merupakan hal yang penting. Teknologi *Hyperledger* memungkinkan otomatisasi kontrak yang dilaksanakan oleh transaksi yang kompleks pada tanggal tertentu atau ketika kondisi lain terpenuhi.

2.10 Smart Contract

Pada tahap ini akan diterangkan mengenai definisi *smart contract,* jenis – jenis *smart contract,* syarat keabsahan kontrak, dan perlindungan hukum penggunaan *smart contract* dalam jual beli melalui *e-commerce.*

2.10.1 Definisi Smart Contract

Smart contract merupakan suatu bentuk perjanjian elektronik yang berkaitan erat dengan teknologi blockchain. Kim mendefinisikan *blockchain* sebagai "*a distributed database that maintains a continuously-growing list of data records secured from tampering and revision. It consists of blocks, holding batches of individual transactions. Each block contains a timestamp and a link to a previous block*" [23]. Dalam perkembangannya, blockchain dibagi menjadi 4 generasi, yaitu *Blockchain* 1.0, 2.0, 3.0., dan X.0.10. *Blockchain* 1.0 merupakan penerapan teknologinya pada *cryptocurrency* yang berkaitan dengan mata uang digital seperti transfer, pengiriman, dan sistem pembayaran digital [24]. *Blockchain* 1.0 diawali dengan penerapan *blockchain* pada *cryptocurrency*, dan Bitcoin merupakan *cryptocurrency* pertama dan terbesar diciptakan pada masa blockchain 1.0, diikuti oleh Litecoin, Dogecoin, Ripple, NXT dan Peercoin.

Pada blockchain 2.0 yaitu penerapan dalam kontrak seperti crowdfunding, financial technology, dan smart contract. Dalam generasi ini blockchain dikembangkan lagi supaya bisa dimanfaatkan lebih jauh dari sekedar *cryptocurrency*. Yaitu, dalam jasa keuangan seperti derivatif, options, swaps¸ dan obligasi [25]. Blockchain 3.0 menerapkan *blockchain* dalam aspek di luar bidang keuangan dan pasar, seperti pemerintahan, kesehatan, supply chain logistic, seni, dan budaya.

Smart contract merupakan sebuah gagasan dari Nick Szabo pada tahun 1994 dengan penjelasan yang tertulis dalam sebuah artikel dengan judul "Smart Contracts" yang ditulis oleh Nick Szabo yaitu *"....a computerized transaction protocol that executes the terms of a contract*" atau dengan kata lain merupakan sebuah rangkaian perintah terkomputerisasi untuk menjalankan ketentuan dari perjanjian. Tujuan umum *Smart Contract* dijelaskan oleh Nick Szabo dalam naskah berjudul "Smart Contracts" yaitu:

> "*The general objectives of smart contract design are to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement), minimize exceptions both malicious and accidental, and minimize the need for trusted intermediaries. Related economic goals include lowering fraud loss, arbitration and enforcement costs, and other transaction costs*"

Smart contract memungkinkan untuk melakukan transaksi yang kredibel secara transparan tanpa pihak ketiga. Transaksi ini dapat dilacak dan tidak dapat diubah. *Smart*

contract berisi informasi tentang semua ketentuan kontrak dan menjalankan semua tindakan yang direalisasikan secara otomatis [26].

2.10.2 Jenis – jenis Smart Contract

Smart contract terbagi menjadi 5 (lima) macam bentuk dengan fungsi dan penerapan yang berbeda [27]. Kelima macam bentuk tersebut adalah basic token contract, crowd sale contract, mintable contract, refundable contract, dan terminable contract. Dari kelima macam bentuk *smart contract*, empat bentuk pertama merupakan smart contract yang biasa digunakan dalam jual beli *cryptocurrency*. Sedangkan *Terminable Contract* merupakan bentuk *smart contract* yang dapat digunakan untuk sistem *blockchain* dalam jual beli barang secara online dan eksekusi program *blockchain* dalam jasa keuangand dan perbankan.

2.10.3 Syarat Keabsahan Kontrak

Pasal 1233 BW menyatakan bahwa suatu perikatan dapat lahir karena adanya persetujuan atau karena disebutkan dalam undang-undang. Pasal 1234 BW menyatakan bahwa perjanjian dapat bertujuan untuk memberikan sesuatu, untuk berbuat sesuatu, atau untuk tidak berbuat sesuatu. Perjanjian kemudian dijelaskan lebih detil dalam Pasal 1313 BW sebagai "…Suatu perbuatan dimana satu orang atau lebih mengikatkan diri terhadap satu orang lain atau lebih." K.R.M.T Tirtodiningrat memberikan definisi perjanjian adalah suatu perbuatan hukum yang berdasar dengan kata sepakat diantara dua pihak atau lebih untuk menimbulkan akibat-akibat dan kejadian hukum yang dapat dipaksakan oleh undang-undang [28].

Agar perjanjian dapat dilakukan, maka perjanjian tersebut harus memenuhi persyaratan perjanjian sebagaimana diatur dalam Pasal 1320 BW yaitu [29]:

a. Kesepakatan bagi mereka yang mengikat dirinya
b. Kecakapan untuk membuat suatu perikatan
c. Suatu hal tertentu
d. Suatu sebab yang tidak terlarang

Kesepakatan atau konsensus merupakan salah satu syarat keabsahan pada sebuah kontrak perjanjian. Kesepakatan yang dimaksud dalam Pasal 1320 adalah persesuaian kehendak antara para pihak, yaitu bertemunya antara penawaran dan penerimaan [30].

2.10.4 Perlindungan Hukum Pengguna *Smart Contract* dalam Jual – Beli Melalui *E-commerce*

Perlindungan hukum adalah perlindungan akan harkat dan martabat, serta pengakuan terhadap hak-hak asasi manusia yang dimiliki oleh subyek hukum berdasarkan ketentuan hukum dari wewenang atau sebagai kumpulan peraturan ataupun kaidah yang dapat melindungi suatu hal dari hal lainnya. Berkaitan dengan konsumen, yang ada artinya bahwa hukum memberikan perlindungan terhadap hak-hak pelanggan dari sesuatu yang mengakibatkan tidak terpenuhinya hak-hak tersebut [31].

Menurut Muchsin, perlindungan hukum merupakan suatu hal yang melindungi subyek hukum melalui peraturan perundang-undangan yang berlaku dan dipaksakan pelaksanaannya dengan sanksi. Perlindungan hukum dapat dibedakan menjadi dua, yaitu [32]:

a. Perlindungan Hukum Preventif Perlindungan yang diberikan oleh pemerintah dengan tujuan untuk mencegah sebelum terjadinya pelanggaran. Hal ini terdapat dalam peraturan perundang-undangan dengan maksud untuk mencegah suatu pelanggaran serta memberikan rambu-rambu atau batasan-batasan dalam melakukan suatu kewajiban.
b. Perlindungan Hukum Represif Perlindungan hukum represif merupakan perlindungan akhir berupa sanksi seperti denda, penjara, dan hukuman tambahan yang diberikan apabila sudah terjadi sengketa atau telah dilakukan suatu pelanggaran.

Asas proporsionalitas atau keseimbangan merupakan asas dalam suatu perjanjian yang bertujuan untuk menjamin keseimbangan proses pertukaran hak dan kewajiban sehingga perjanjian dapat mencerminkan *fairness* atau kesetaraan.

2.11 Decentralized App

DApps atau aplikasi terdesentralisasi (Decentralized App) adalah generasi baru dari aplikasi yang tidak dikendalikan atau dimiliki oleh otoritas tunggal, tidak dapat dimatikan atau tidak dapat mengalami downtime. Konsep DApp masih pada tahap awal. Berikut ini merupakan 4 karakteristik utama yang harus ada pada DApp yaitu [33]:

1. Open source

   Open source merupakan karakteristik paling penting karena aplikasi tersebut harus membuat core source code (kode inti) bisa tersedia untuk semua orang. Karena karakteristik inti dApps merupakan kesepakatan konsensus yang bulat, pada

dasarnya perubahan harus diputuskan oleh semua atau sebagian besar pengguna. Dan juga kode harus tersedia untuk semua orang untuk check out.

2. Lingkungan terdesentralisasi

   Seperti namanya, aplikasi yang terdesentralisasi menyimpan semuanya di blockchain yang terdesentralisasi atau teknologi kriptografi apa pun untuk menyelamatkan aplikasi dari bahaya otoritas yang terpusat (sentralisasi) dan menekankan pada sifat otonom atau independen.

3. Algoritma

   Dapp perlu memiliki mekanisme konsensus yang menggambarkan bukti nilai dalam sistem kriptografi. Pada dasarnya, hal itu memberikan nilai pada token kriptografi dan membuat protokol konsensus yang disetujui pengguna untuk menghasilkan token kripto yang berharga.

4. Intensif

   Karena aplikasi ini didasarkan pada blockchain yang didesentralisasi, validator catatan di jaringan harus dihargai / diberi insentif dengan token kriptografi atau segala bentuk aset digital yang memiliki nilai.

Secara garis besar, *dApp* adalah aplikasi yang berjalan pada jaringan P2P terdesentralisasi yang diatur oleh semua anggota dan bukan otoritas pusat tunggal. *Dapp* pertama yang telah popular adalah Bitcoin. Bitcoin popular sebagai cryptocurrency yang paling puncak, bitcoin memecahkan masalah sentralisasi dan memberi pengguna kekuatan untuk melakukan transaksi tanpa perantara atau otoritas pusat melalui *Hyperledger* (buku besar) publik yang mandiri. Penggunaan *Dapps* terbagi menjadi tiga segmen yang telah dikategorikan yaitu:

1. Manajemen uang dan transfer

   Aplikasi terdesentralisasi dapat digunakan untuk memperlancar transfer uang di dunia. Bukti nyata manfaatnya dalam bentuk keberhasilan bitcoin dan cryptocurrency lainnya. Menggunakan jaringan blockchain dan token crypto-nya sendiri, dApps dapat mempercepat pengelolaan uang, transfer, dan pinjaman dengan menghilangkan perantara dan meningkatkan keamanan karena mekanisme konsensus yang tidak mungkin diubah tanpa mayoritas.

2. Manajemen bisnis proses

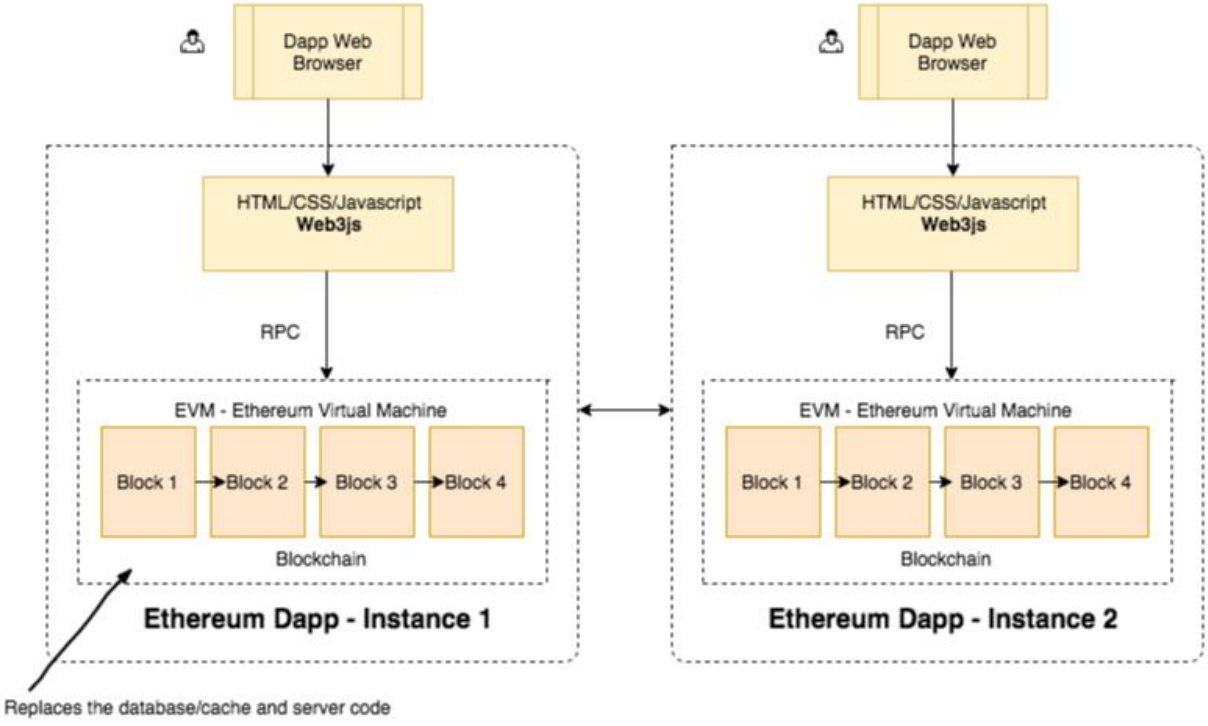
   Perusahaan dapat mengintegrasikan aplikasi terdesentralisasi untuk merampingkan proses tanpa campur tangan manusia. Dengan bantuan kontrak pintar - roda penggerak penting dalam jaringan blockchain, masalah kritis dapat diselesaikan

dan efisiensi proses dapat ditingkatkan. Misalnya, perusahaan logistik dapat mengintegrasikan chip RFID ke dalam pengiriman mereka yang dapat dipindai di pelabuhan tujuan tempat pembayaran dapat diselesaikan secara otomatis melalui kontrak pintar antara pembeli dan penjual.

3. DAO (Decentralized Autonomous Organization)

   DAO adalah fenomena yang sama sekali baru dalam memulai organisasi tanpa wajah tanpa pemimpin. Organisasi-organisasi ini dapat bekerja sebagai perusahaan dan dijalankan melalui aturan yang ditentukan oleh bahasa pemrograman di blockchain. Bagaimana anggota akan memilih, segmen bisnis apa yang akan dioperasikan organisasi, siapa yang dapat menjadi anggota, bagaimana nilai token akan dipertukarkan, semuanya dapat diprogram pada blockchain yang akan menjalankan organisasi. Organisasi ini tidak dapat berhenti setelah dikerahkan dan dapat bekerja di seluruh dunia, tanpa kesulitan menaiki kereta jaringan blockchain.

Arsitektur Dapps dapat dilihat pada **Gambar 2.6**



Gambar 2.6 Arsitektur *Dapps* (Aplikasi yang terdesentralisasi)

Sumber: https://www.commonlounge.com/

Setiap clinet-browser berkomunikasi dengan sistem aplikasi itu sendiri. Tidak ada server utama yang terkonek dengan browser client. Artinya bahwa, setiap orang yang ingin

berinteraksi denga Dapps harus mempunyai full copy Blockchain untuk menjalankan pada perangkat masing-masing. Namun dalam kenyataannya hal tersebut tidak diperlukan karena banyak beberapa solusi dan optimasi agar Dapps bisa selalu berinteraksi dengan cepat dan mudah. Dalam mekanismenya, Dapps pada Blockchain terdiri dari semacam 2 hal ini yaitu:

1. Database

   Setiap transaksi yang muncul pada jaringan Ethereum di kemas menjadi block dan tiap block terhubung satu sama lain. Jaringan block tersebut membawa semua data transaksi pada Blockchain. Untuk memastikan semua node block membawa salinan data transaksi tersebut maka Ethereum menggunakan algoritma yang disebut Proof os Work

2. Code

   Logika untuk beli, jual, cancel, pengembalian uang dibuat melalui smart contract dengan bahasa pemrograman Solidity. Kemudian, Solidity dapat dikompilasi menggunakan Solidity Compiler yang menghasilkan Ethereum Byte Code yang biasa disebut Application Binary Interface yang dapat di-deploy ke jaringan Blockchain Public

Dapps memiliki dua mekanisme secara umum dimana dalam tahap membangun pelaku consensus: mekanisme Proof of Work dan mekanisme Proof of Stake [33]. Dengan mekanisme Proof of Work, keputusan tentang perubahan dalam DApp dibuat berdasarkan jumlah pekerjaan yang berkontribusi masing-masing *stakeholder* terhadap operasi DApp. Bitcoin menggunakan pendekatan itu untuk operasi sehari-harinya. Mekanisme untuk membangun konsensus melalui POW umumnya disebut penambangan.

Dengan mekanisme Proof of Stake, keputusan tentang perubahan DApp dibuat berdasarkan persentase kepemilikan yang dimiliki berbagai *stakeholder* atas aplikasi tersebut. Sebagai contoh, suara pemegang saham yang mengontrol 10% token yang dikeluarkan oleh DApp, memiliki bobot 10%.

2.12 Truffle Suite Framework

Truffle adalah lingkungan kerangka kerja (framework) untuk pengembangan, pengujian dan asset pipeline Blockchain untuk Ethereum Virtual Machine (EVM), yang bertujuan untuk mempermudah developer untuk pengembangan platform Blockchain, Dapps, dan Smart Contract [34]. Truffle merupakan IDE paling interaktif dan paling popular saat ini di komunitas Ethereum.

Truffle dioperasikan di terminal, dank arena itu memiliki serangkaian perintah praktis untuk digunakan pada berbagai tahap pengembangan Dapps. Jika ingin membuat project dari nol, maka perintah yang harus dilakukan adalah “truffle init”. setelah meng-init maka akan muncul project directory yaitu contracts, migrations, dan tests. Termasuk dengan file config yang telah disediakan yaitu “truffle-config.js”. Pada tahap akan menjelaskan mengenai struktur project pada Truffle dan cara mengkonfigurasikan jaringan Ethereum dalam Truffle Config [34].

2.12.1 Struktur Project

Structural project pada Truffle terbagi menjadi 3 directory yaitu:

a. Contracts

Semua file yang terkait kontrak berbasis “.sol” dan “.vy” disimpan pada directory ini.

b. Migrations

File migrasi yang berguna untuk men-deploy kontrak yang telah dibuat tersimpan pada directory ini. Migrations adalah file yang tertulis dalam Javascript yang memprogram penyebaran (deploy) *Smart Contract* ke dalam jaringan Ethereum Blockchain. Contoh sederhana penggunaan Migrations.js adalah:

```
Var MyContract = artifacts.require(“MyContract”);
module.exports – function(deployer){
        //deployment steps
        deployer.deploy(MyContract);
}
```

c. Test

Test unit berguna untuk men-test *smart contract* yang dapat ditulis dalam Javascript, Solidity, Vyper, dan lainnya. Semua disimpan pada directory yang sama

2.12.2 Konfigurasi jaringan Ethereum dalam Truffle Config

"Truffle-config.js" memungkinkan kita untuk mengkonfigurasikan jaringan yang akan di deploy kedalam jaringan Ethereum lokal. Contoh sederhana penggunaannya dalam jaringan Ethereum lokal adalah:

```
module.exports  = {
        networks: {
                development:{
                        host: "127.0.0.1",
                        port: 8545,
                        network_id: "*" // match any network id
                }
        }
};
```

Jaringan untuk pengembangan sudah dikonfigurasi secara default dengan localhost. Secara default Truffle menjalankan localhost pada "localhost:9545". Namun Ganache dijalankan pada "localhost:8545". Jaringan Blockchain lokal yang dibuat juga diberikan ID acak yang tidak berhubungan langsung dengan jaringan Ethereum Public.

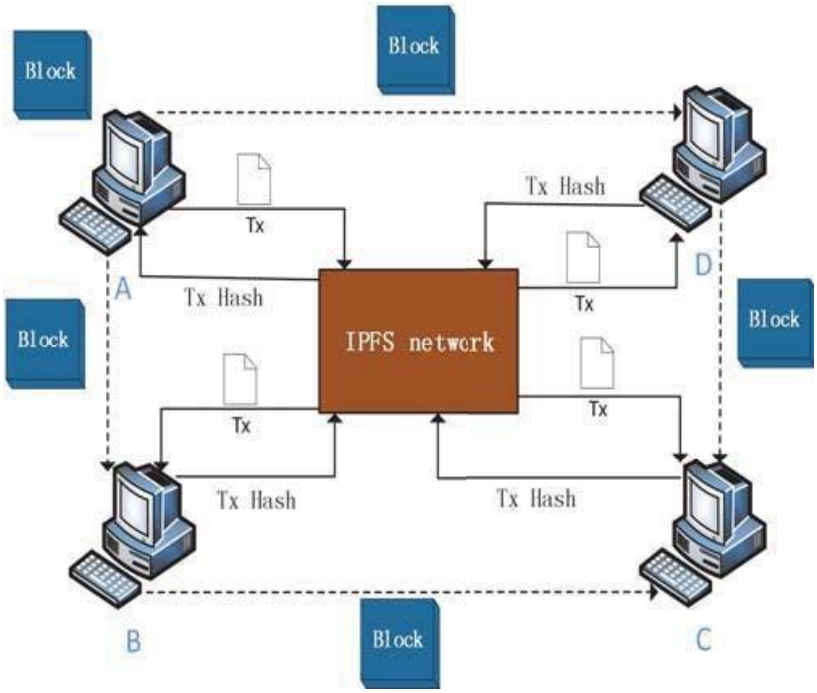
2.13 Interplanetary File System

Interplanetry File System (IPFS) adalah distribusi file sistem berbasis protokol peer-to-peer untuk menyimpan file. Tiap node pada IPFS menyimpan objek IPFS dalam penyimpanan lokal. Node terhubung satu sama lain dan mentransfer objek. Objek – objek ini mewakili file dan struktur data lainnya [35]. IPFS merupakan platform untuk menulis, menggunakan aplikasi, dan sistem baru untuk mendistribusikan dan membuat versi data yang besar.

Interplanetary File mendistribusikan protokol penyimpanan. Konsepnya mempunyai mekanisme deduplikasi tanpa batasan dari server pusat yang tersentralisasi. Selain itu, data yang telah diupload ke sistem dapat disimpan secara permanen. Untuk data dengan request yang tinggi, IPFS akan membuat duplikasi data di sepanjang request path, yang berarti dapat langsung dibaca secara lokal pada request selanjutnya. IPFS terjamin aman, hasil yang tinggi,

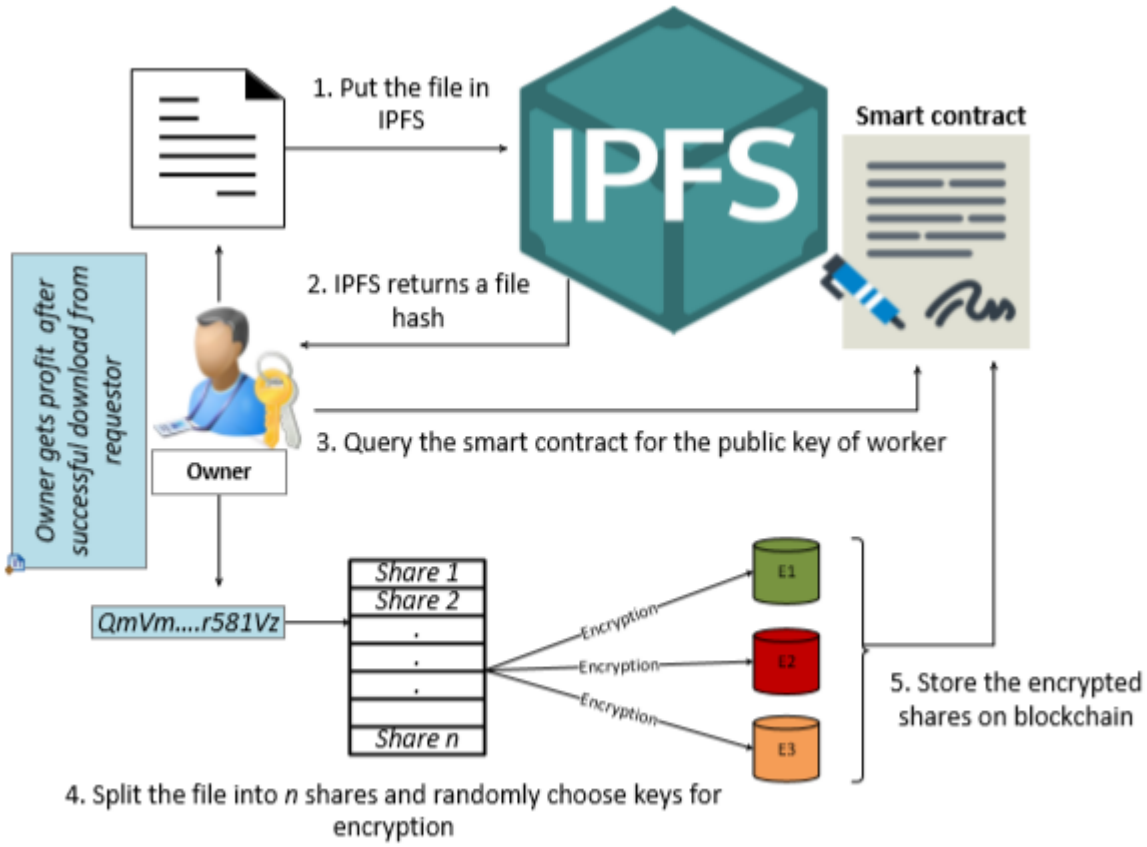
memiliki model penyimpanan *content addressed block* yang mendukung kapasitas tinggi dan akses bersamaan.

Dalam model, tiap penambang memeriksa transaksi yang diterima, menempatkan transaksi yang valid ke dalam node dan disimpan pada IPFS, serta mengamankan transaksi yang dikembalikan hash IPFS. Saat menghitung blok berikutnya, setiap penambang mengemas hash IPFS dari transaksi yang diverifikasi ke dalam blok baru. Struktur blok ditunjukkan pada Gambar. 2. Jika penambang A berhasil menghitung hash blok yang, blok akan disiarkan ke penambang B, C, D, dll. Setelah penambang B, C dan D menerima blok, mereka perlu memverifikasi transaksi dan blok hash. Bahkan, sebagian besar transaksi yang diterima oleh penambang B, C dan D selama proses penambangan sama dengan penambang A. Hanya beberapa transaksi dalam kumpulan transaksi mereka berbeda karena penundaan transmisi jaringan. Oleh karena itu, sebagian besar transaksi hash IPFS di blok baru sama dengan yang ada di kumpulan transaksi lokal penambang B, C dan D. Jika hash IPFS dari transaksi di blok yang dikirim oleh penambang A dapat ditemukan di jaringan lokal kumpulan transaksi, maka transaksi telah dikonfirmasi oleh B, C dan D sebelumnya dan tidak perlu diunduh dari IPFS. Untuk sisa transaksi, data harus diminta dari jaringan IPFS melalui hash IPFS yang sesuai. Kemudian validitas transaksi dan blok akan dikonfirmasi [36].



Gambar 2.7 Model Penyimpanan IPFS

Setelah file diunggah ke IPFS, hash data itu dihasilkan oleh IPFS dan dikembalikan ke pemiliknya. Pada **Gambar 2.8**, node pengirim dihasilkan dan pasangan public-private key disimpan di smart contract. Saat penerima mendapatkan hash dari IPFS, hash tersebut memverifikasi node pengirim yang ada di smart contract, yang bertanggung jawab dalam melaksanakan dekripsi untuk penerima. Hanya pengirim yang bersangkutan yang bisa melaksanakan servis, yaitu pengirim yang dipilih oleh penerima. Saat penerima telah menerima block hash, algoritma *SSS* digunakan untuk membagi file hash IPFS ke jumlah share k. Selama semua pembagian terenkripsi, maka disimpan di blockchain bersama dengan informasi penting lainnya seperti otorisasi file penerima. Jika ada penerima yang tidak terotorisasi, siapapun yang belum mengirimkan deposit untuk konten digital hak akses ke hash, maka file IPFS yang telah digabungkan dapat diambil. Dalam kata lain, penerima resmi akan kehilangan datanya. Aliran keseluruhan file yang diupload di IPFS dapat dilihat pada **Gambar 2.8** [37].

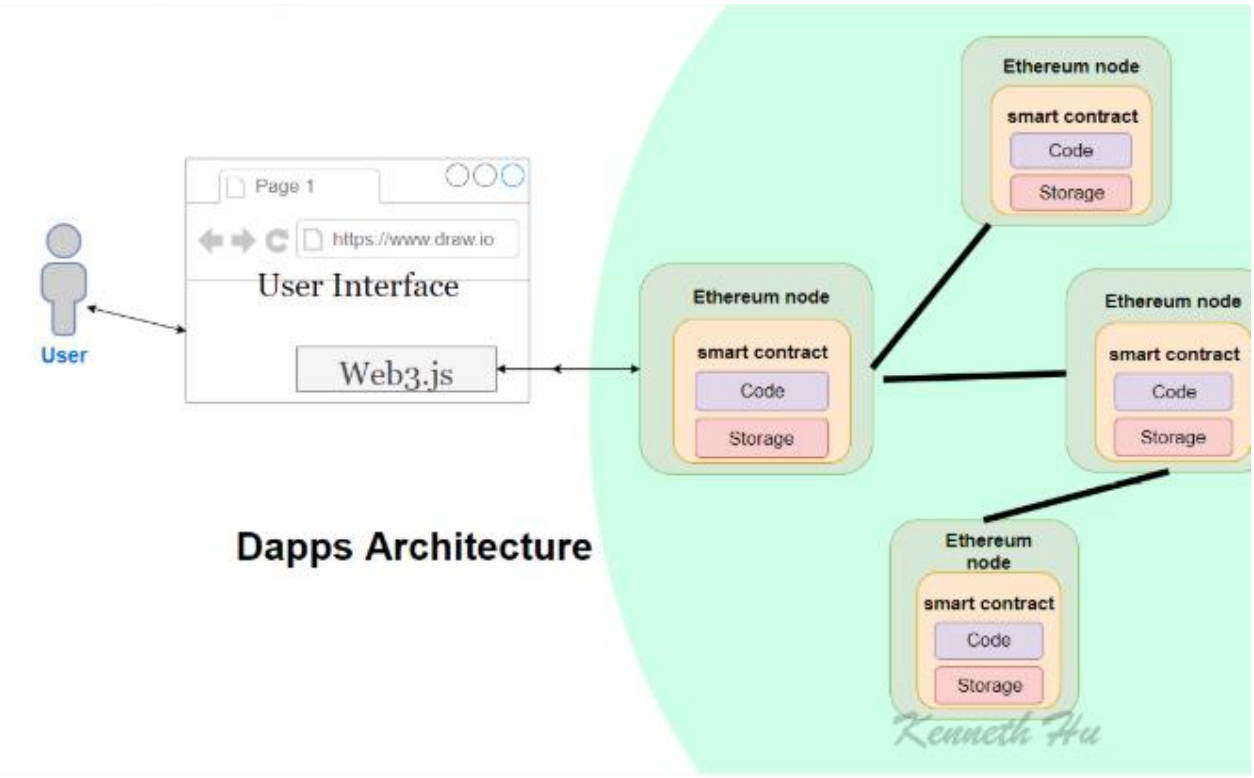


Gambar 2.8 Skema Data Sharing IPFS

2.14 Web3.js

Web3.js adalah kumpulan library Javascript yang memungkinkan program untuk berinteraksi dengan node Ethereum lokal menggunakan HTTP, WebSocket, atau IPC [38]. Dengan API Web3.js, front-end bisa berinteraksi dengan Smart Contract. Web3.js merupakan jembatan bagi JSON RPC Ethereum menggunakan bahasa pemrograman Javascript sebagai atarmukanya, yang memuatnya langsung dapat digunakan dalam aplikasi web karena Jvascript mendukung secara native hamper di semua web browser. Web3.js juga dingunakan pada server-side dengan Node.js dan pada desktop berbasis Electron. Web3.js dapat digunakan untuk terhubung ke jaringan Ethereum melalui node Ethereum yang memungkinkan akses melalui HTTP.

Salah satu cara umum untuk mengintegrasikan aplikasi browser web dengan Ethereum adalah dengan menggunakan ekstensi browser Metamask dalam kombinasi dengan Web3.js. Metamask adalah wallet (dompet) Ethereum pada web browser yang dapat men-inject object Web3 provider ke dalam browser. Web3 Provider adalah struktur data yang menyediakan tautan ke node Ethereum yang dapat diakses publik. Menggunakan Metamask memungkinkan pengguna untuk mengelola private key dan menandatangani transaksi dalam browser web mereka. Menggunakan Metamask dalam kombinasi dengan Web3.js, dalam antarmuka web, menyediakan cara mudah untuk berinteraksi dengan jaringan Ethereum.



Gambar 2.9 Arsitektur Dapps dengan Web3js

Sumber: https://medium.com/coinmonks/web3-js-ethereum-javascript-api-72f7b22e2f0a

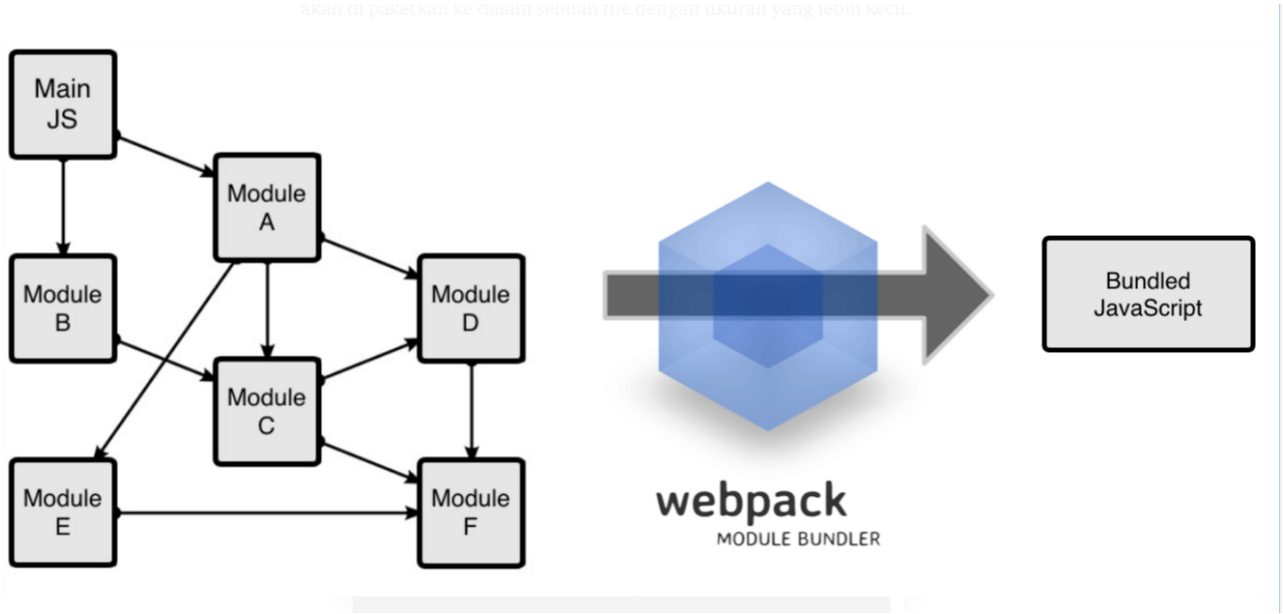
Contoh menginisiasi penggunaan Web3:

```
If (window.ethereum){
        App.Web3Provider = window.ethereum;
        Try{
                Await window.ethereum.enable();
        } catch (error){
                Console.error("User denied account access")
        }
} else if (window.web3){
        App.web3Provider = window.web3.currentProvider;
} else {
        App.web3Provider = new
        Web3.provider.HttpProvider('http://localhost:7545');
}
Web3 = new Web3(App.web3Provider);
```

Pertama, kita cek dulu apakah menggunakan browser dapp modern atau sesuai versi dari Metamask dimana ethereum provider di-inject kedalam objek window. Jika iya, maka kita buat objek web3nya dan juga kita membutuhkan akses khusus ke akun kita dengan ethereum.enable(). Jika objek ethereumnya tidak ada, maka kita mengecek apakah web3nya sudah di-inject atau belum. Jika objek ethereumnya ada, maka terindikasi bahwa kta sedang menggunakan browser dap pang versi lama (seperti versi lama Metamask). Maka dari itu, kita menggunakan providernya dan pakai untuk membuat objek web3nya. Apabila web3 yang di-inject tidak muncul, kita buat iobjek web3nya pada local provider.

2.15 Webpack

Webpack merupakan modul bundler yang memuat semua depedensi dari Javascript dan mem-packagenya untuk mengoptimalkan waktu pemuatan di browser [39]. Modul bundler bekerja dengan cara menelusuri suatu module lalu mencarikan modul apa saja yang dibutuhkan oleh modul tersebut kemudian seluruh modul tersebut digabung menjadi satu file dikenal sebagai bundle. Ada beberapa konsep utama yang perlu ada pada modul bundler Webpack. Beberapa diantaranya yaitu *entry*, *output*, *loaders*, *plugin*, dan *mode*.



Gambar 2.10 Skema Webpack

Sumber: http://www.pro-react.com/materials/appendixA/images/picture2.png

2.15.1 Entry

Entry adalah suatu modul yang akan digunakan sebagai titik awal oleh webpack ketika memulai melakukan proses penelusuran. Secara default entry yang akan digunakan oleh webpack adalah "`./src/index.js`" namun dapat juga diatur modul mana yang akan digunakan sebagai entry. Entry bisa terdiri dari satu modul atau beberapa modul. Contoh penggunaan entry adalah:

```
Module.exports = {
        Entry: {
                Entry1: './path/to/entry1.js',
                Entry2: './path/to/entry2.js'
        }
}
```

2.15.2 Output

Output akan mengatur bagaimana webpack menyimpan semua bundle yang telah dihasilkan. Secara default webpack akan menyimpan semua bundle tersebut di directory "`./dist`" namun dapat juga diletakan di directory lain sesuai dengan keinginan melalui properti `path`. Selain itu output juga dapat mengatur format yang diinginkan untuk nama dari bundle melalui properti `filename`. `Contoh penggunaan output pada module bundler Webpack yaitu:`

```
Const path = require('path');

Module.exports = {
        output: {
                path: path.resolve(_dirname, 'dist'),
                filename: '[name].[contenthash].js'
        }
}
```

2.15.3 Loaders

Webpack akan memproses setiap modul terlebih dahulu sesuai dengan jenisnya sebelum dapat digunakan oleh modul yang membutuhkannya. Webpack hanya mengetahui dua jenis modul yaitu `JavaScript` dan `JSON` selain itu maka harus menggunakan package tambahan dengan mengaturnya pada pengaturan module. Contoh penggunaan loaders adalah:

```
module.exports = {
        module: {
                rules:[{
                        test: /\.txt$/,
                        use: 'raw-loader'
                }]
        }
}
```

Pada pengaturan tersebut digunakan `module.rules` untuk memberitahu webpack agar setiap modul dengan ekstensi `.txt` diproses terlebih dahulu oleh package `raw-loader`. Pada `module.rules` harus terdapat dua properti yang tersedia yaitu yang pertama adalah test yang diisi dengan regex untuk mengidentifikasi modul yang ingin diproses. Yang kedua adalah use diisi dengan nama package yang digunakan untuk memproses modul.

2.15.4 Plugins

Loaders hanya memproses modul tertentu namun plugins dapat melakukan optimasi, manajemen asset dan memanipulasi environment variable yang tersedia. Untuk menjadikan package sebagai plugin maka sebelumnya harus menggunakan `require()` lalu menambahkannya ke properti plugins. Metode seperti ini membuat plugin mudah untuk dikostumisasi. Contoh penggunaan plugins HTML yang berguna untuk membuat file html yang berisikan seluruh bundle yang tersedia.

```
Const htmlplugin = require('html-webpack-plugin')

Module.exports = {
        Plugins: [
                new htmlplugin({
                        template: './src/index.html'
                })
        ]
}
```

2.15.5 Mode

Mode membantu Webpack dalam melakukan optimasi berdasarkan environment yang dipilih. Mode dapat diterapkan dengan penambahan property mode yang diisi dengan

environment development, production atau none. Webpack menggunakan environment production secara default.

2.16 Redux

Redux adalah library Javascript open-source untuk memanajemen state pada aplikasi. Redux paling banyak digunakan pada React untuk mebangun user interface (antar muka pengguna). Redux merupakan wadah bagi state yang dapat diprediksi pada aplikasi Javascript dan dapat membantu dalam penulisan kode secara konsisten, bisa dijalankan di environment berbeda (client, server, native), dan mudah di testing. Redux merupakan implementasi dari Flux. Meskipun terkait dengan React, Redux adalah library [40]. Pada dasarnya peran Redux dalam sebuah aplikasi adalah melakukan perubahan state yang didbutuhkan oleh setiap fungsi pada aplikasi. Terdapat tiga komponen utama pada Redux yaitu *action, reducer,* dan *store* [41].

2.16.1 Action

Terdiri dari *list action* dan *action creator*, keduanya saling berhubungan karena *list* berarti daftar dan *creator* berarti pembuatan. Contoh penggunaan action yaitu:

```
Export const NEW_TODO = ‘NEW_TODO’

Function newTodo(parameter){
        Return{
                Type: NEW_TODO,
                //parameter
        }
}
Export{ newTodo }
```

Baris pertama merupakan *list action* yang berisi daftar tipe action yang akan dilakukan dalam aplikasi. Function newTodo merupakan *action creator* dan baris terakhir *export action* agar bisa digunakan pada file lain.

2.16.2 Reducer

Reducer berguna untuk mengubah atau menjalankan perintah yang diberikan. Contoh penggunaan Reducer pada Redux ialah:

```
Import {NEW_TODO} from ‘./actions’
Import {combineReducers} from ‘redux’
Function newTodo(state = [], action){
        Switch(action.type){
                Case NEW_TODO:
                        Return[ …state, {
                                Text: action.parameter,
                                Completion: false
                        }] default: return state
        }
}
Const appState = combineReducers({ newTodo })
```

Library *combineReducers* digunakan untuk menyatukan banyak reducer. Pada *function newTodo* terdapat cara melakukan perubahan state berdasarkan *action type* yang sudah di list.

2.16.3 Store

Fungsi – fungsi dari Store adalah :

1. Menyimpan state aplikasi
2. Mendapatkan akses ke dalam state, menggunakan getState()
3. Mendapatkan listener menggunakan subscribe(listener)
4. Menangani listener yang belum teregistrasi dari nilai dari subscribe(listener)

Contoh penggunaan Store adalah sebagai berikut:

```
Import { createStore } from ‘redux’
Import todoApp from ‘./reducers’
Import { newTodo } from ‘./actions’
Let store = createStore(todoApp)

Console.log(store.getState())
Let unsubscribe = store.subscribe(() => console.log(store.getState()))

Store.dispatch(newTodo(‘Testing’))
```

Baris pertama memanggil library createStore dari redux dan baris kedua memanggil reducer yang telah dikombinasi, kemudian untuk action yang digunakan dipanggil pada baris ketiga. Sebelum dapat menjalankan aplikasi, store redux harus di start dengan memanggil

library createStore dengan parameter reducer yang telah dibuat.Baris selanjutnya untuk menangani listener dalam aplikasi kemudian untuk mengubah *state* aplikasi.

2.17 Application Binary Interface (ABI)

Application Binary Interface adalah cara standar untuk berinteraksi dengan smart contract pada ekosistem Ethereum, baik dari luar maupun interaksi antara kontrak dengan kontrak. ABI menggunakan sebuah cara abstrak yang bukan bagian dari protokol inti (core) Ethereum, tapi dapat digunakan untuk mengakses kode byte pada Smart Contract sebagai standar. Tujuan dari ABI adalah sebagai berikut [42] :

1. Apa dan bagaimana function pada smart contract akan dipanggil.
2. Format biner yang mengandung informasi harus dipanggil sebagai input dari function pada smart contract.
3. Format biner akan menghasilkan output dari eksekusi function setelah dipanggil.

Pengkodean tidak menggambarkan diri dan karenanya membutuhkan skema untuk memecahkan kodenya. Dalam pengertian umumnya, ABI merupakan antarmuka antara 2 module program, salah satunya biasanya berada di level kode mesin. Pada Ethereum, biasanya itu merupakan cara bagaimana kita dapat meng-encode panggilan dari kontrak Solidity untuk Ethereum Virtual Machine dan bagaimana untuk membaca data transaksinya. Seperti bagaimana kita memanggil fungsi – fungsinya pada kontrak dan mengambil datanya kembali.

Smart Contract Ethereum berisikan *bytecode* ang di-deploy kedalan Ethereum Blockchain. Mungkin ada beberapa fungsi dalam kontrak. ABI termasuk kedalamnya jadi kita dapat memanggil fungsi spesifik di dalam kontrak [42]. ABI berisi kumpulan semua fungsi dan argument pada kontrak dengan format JSON. Pengkodean ABI biasanya diotomatisasi menggunakan alat yang merupakan bagian dari compiler atau perangkat lunak lainnya, seperti wallet yang bisa berinteraksi dengan smart contract. Encoder ABI setidaknya membutuhkan deskripsi antarmuka kontrak termasuk nama fungsi dan tipe parameternya. Cara umumnya untuk memberikan definisi seperti itu yaitu dalam bentuk file JSON. Untuk memanggil metod dari kontraknya, *string byte* pada format spesifik harus dibangun. Untuk memanggilnya, string itu dikirim melalui transaksi *field data* menuju alamat (*address*) smart contract.

2.18 Object-Oriented Programming (OOP)

OOP (Object Oriented Programming) atau yang dikenal dengan Pemrograman Berorientasi Objek merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus ke dalam kelas-kelas atau objek-objek. Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki manager, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bag administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bag administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bias mendapatkan data tersebut melalui objek petugas administrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek-objek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri.

2.19 UML

UML (*Unified Modeling Language*) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem. UML adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya. UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat tool untuk mendukung pengembangan system tersebut. UML mulai diperkenalkan oleh Object Management Group, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. Sekarang UML sudah mulai banyak digunakan oleh para praktisi OOP [43].

Unified Modeling Languange (UML) membahas tentang pengembangan perangkat lunak dan perlunya dilakukan pemodelan untuk membantu. UML adalah salah satu standar

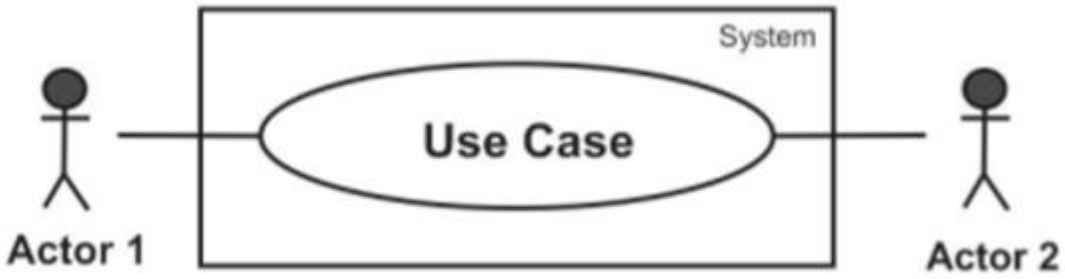
yang banyak digunakan di dunia industri untuk mendefenisikan requirement, membuat analisis dan desain. Dalam mendefenisikan requirement , membuat analisis dan desain serta menggambarkan arsitektur dalam pemograman berorientasi objek [43]. Penggunaan UML dalam industri terus meningkat, ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri perangkat lunak dan pengembangan sistem.

2.19.1 Komponen UML

Pada bagian ini akan dijelaskan mengenai definisi *use case, activity diagram, class diagram,* dan *sequence diagram*.

2.19.1.1 Use Case Diagram

Diagram use case merupakan pemodelan untuk kelakuan (behavior) sistem yang akan dibuat. Use Case mendeskripsikan sebuah interaksi atara satu atau lebih actor dengan sistem yang akan dibuat . Syarat penamaan pada use case adalah nama didefenisikan sesederhana mungkin dan dapat dipahami. Ada dua hal utama pada use case yaitu pendefenisian apa yang disebut aktor dan use case.



Gambar 2.11 Diagram Use Case

1. Aktor

   Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem yang akan dibuat di luar sistem yang akan dibuat itu sendiri, walaupin dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang
2. Use Case

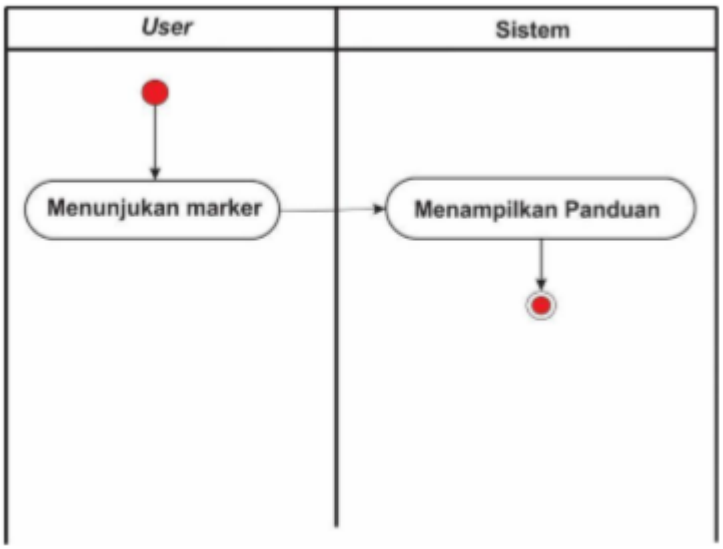
Use case merupakan fungsionalitas yang disediakan sistem sebagai unitunit yang yang saling bertukar pesan antar unit ataupun aktor. Berikut adalah simbol-simbol yang ada pada diagram use case.

Use case digunakan untuk mendeskripsikan interaksi antara actor dan mendeksripsikan fungsi – fungsi yang ada pada sistem [43].

2.19.1.2 Activity Diagram

Menggambarkan rangkaian aliran kerja (workflow) atau aktivitas dari sebuah sistem atau proses bisnis yang ada pada perangkat lunak. Bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor. Diagram aktivitas banyak digunakan untuk mendefinisakn hal-hal:

1. Rancangan proses bisnis dimana setiap utrutan aktivitas yang digambarkan merupakan proses bisnis sistem yang akan didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan [43].



Gambar 2.12 Diagram Activity

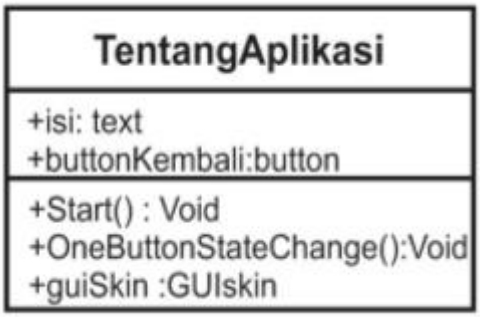
2.19.1.3 Class Diagram

Class Diagram menggambarkan struktur sistem dari segi pendefenisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan

metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas, sedangkan operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas Adapun susunan pada diagram kelas adalah memiliki jenis-jenis sebagai berikut :

1. Kelas main
   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
2. Kelas yang menangani tampilan sistem (view)
   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai
3. Kelas yang diambil dari pendefinisian use case
   Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefenisian use case, kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.
4. Kelas yang diambil dari pendefinisian data
   Kelas yang digunakan untuk memegang dan menghubungkan data menjadi sebuah yang diambil maupun akan disimpan ke basis data.

Pada sistem yang dibangun class diagram digunakan mendefenisikan kelaskelas yang dibuat dalam membangun sistem. Semua class berserta dengan atribut dan methodnya akan didefenisikan dengan menggunakan class diagram [43].

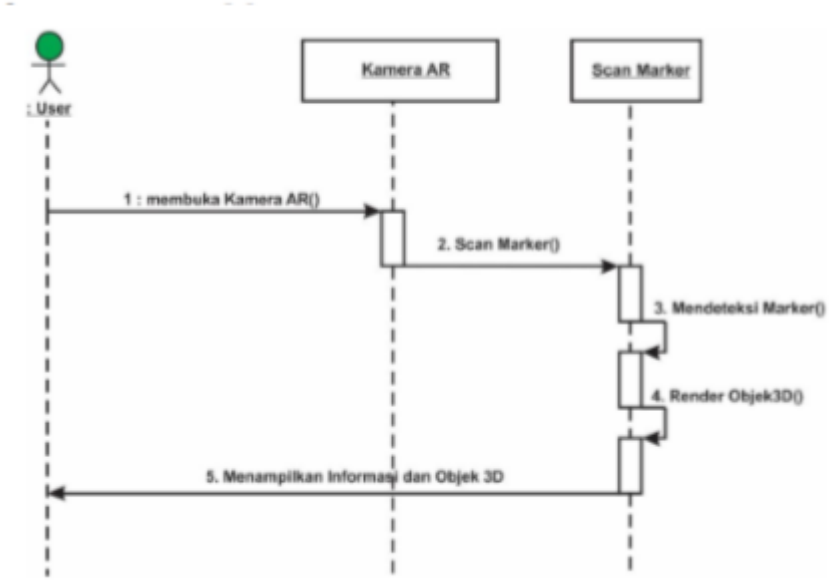


Gambar 2.13 Diagram Class

2.19.1.4 Sequence Diagram

Sequence Diagram menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek dan pesan yang diterima antar objek Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objekobjek yang terlibat dalam use case beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisan use case yang memiliki proses sendiri atau yang penting semua use case yang telah didefinisakan interkasi jalannya pesan sudah dicakup pada diagram [43].



Gambar 2.14 Diagram Sequence

2.20 Metode Pengujian Sistem

Metode pengujian sistem untuk mengetahui efektifitas dari software yang dibangun selain memberikan kesempatan pada para pengguna untuk mengoperasikan dan melakukan pengecekan pada software. Metode pengujian sistem terdiri dari *White-Box* dan *Black-Box*.

2.20.1 Pengujian Black-Box

Pengujian kotak hitam atau *Black-Box* testing yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian Black-box dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian Black-box harus dibuat dengan kasus benar dan kasus salah [44].

Pengujian *black-box* berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya:

1. Fungsi – fungsi yang salah atau hilang,
2. Kesalahan interface,
3. Kesalahan dalam struktur data atau *database* eksternal,
4. Kesalahan performa,
5. Kesalahan inisialisasi dan terminasi.

2.20.2 Skala Data yang digunakan

Skala data yang digunakan untuk mengukur variabel yang didapat, menggunakan skala likert. Skala likert adalah skala respon psikometri terutama digunakan dalam kuesioner untuk mendapatkan preferensi responden atas sebuah pernyataan atau serangkaian laporan. Setelah menyelesaikan definisi operasional variabel maka langkah selanjutnya menyusun item-item. Sebuah skala menjadi penting untuk mengukur derajat pendapat dan data kuantitatif berarti analisis relatif mudah dilakukan. Prinsip pengukuran sikap yaitu meminta orang untuk menanggapi serangkaian pernyataan tentang suatu topik. Sejauh mana mereka setuju dengan memasuki komponen kognitif dan afektif. Data yang telah terkumpul melalui angket, kemudian diolah kedalam bentuk kuantitatif [44], yaitu dengan sebuah kontinum dari sangat setuju sampai sangat tidak setuju yang dapat dilihat pada Tabel

Tabel 2.2 Penilaian Skala Likert

Jawaban	Skor
Sangat Setuju	5
Setuju	4
Ragu – ragu	3
Tidak Setuju	2
Sangat Tidak Setuju	1

Kemudian dengan teknik pengumpulan data kuisioner, maka instrument tersebut misalnya diberikan kepada 30 orang yang diambil secara random. Dari 30 orang tersebut setelah dilakukan analisis didapat hasil sebagai berikut:

1. Sebanyak 15 orang menjawab Sangat Setuju (SS)
2. Sebanyak 10 orang menjawab Setuju (SS)
3. Sebanyak 2 orang menjawab Ragu - ragu (SS)
4. Sebanyak 2 orang menjawab Tidak Setuju (SS)
5. Sebanyak 1 orang menjawab Sangat Tidak Setuju (SS)

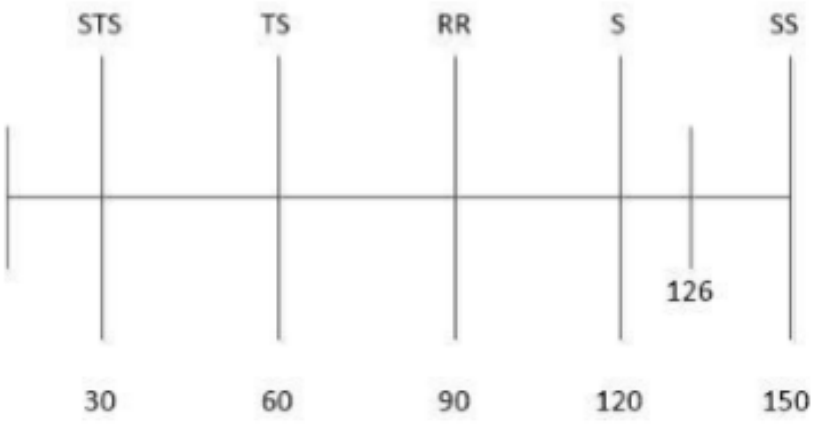
Berdasarkan data tersebut 25 orang (15 + 10) atau 83.3% *stakeholder* menjawab setuju dan sangat setuju.

Data Interval tersebut kemudian dianalisis dengan menghitung rata-rata jawaban berdasarkan skoring setiap jawaban dari responden. Berdasarkan skor yang telah didapat dari analisis sebelumnya dapat dihitung sebagai berikut:

Jumlah skor untuk 15 orang yang menjawab SS = 15x5 = 75
Jumlah skor untuk 10 orang yang menjawab S = 10x4 = 40
Jumlah skor untuk 2 orang yang menjawab RR = 2x3 = 6
Jumlah skor untuk 2 orang yang menjawab TS = 2x2 = 4
Jumlah skor untuk 1 orang yang menjawab STS = 1x1 = 1
Jumlah Total Nilai = 126

Jumlah ideal (kriterium untuk seluruh item = 5x30 = 150 (seandainya semua menjawab SS). Jumlah skor yang diperoleh dari penelitian = 126. Jadi berdasarkan data itu maka tingkat persetujuan stakeholder terhadap penggunaan media pembelajaran interaktif = (126:150) x 100% = 84% dari yang diharapkan (100%).

Secara kontinum dapat dilihat seperti Gambar.



Gambar 2.15 Pengolahan Hasil Kuisioner secara Kontinum

Jadi berdasarkan data yang diperoleh dari 30 responden maka rata-rata 126 terletak pada daerah mendekati setuju.

Alasan mengapa menggunakan skala likert untuk mengukur variabel adalah sebagai berikut:

1. Skala likert memudahkan responden untuk menjawab kuesioner apakah setuju atau tidak setuju.
2. Skala likert mudah digunakan dan mudah dipahami oleh responden.

Skala likert secara visual lebih menarik dan mudah diisi oleh responden.

2.21 Bahasa Pemrograman

Pada tahap ini akan menjelaskan mengenai bahasa pemrograman yang dipakai. Terdapat 2 bahasa pemrograman yaitu Javascript dan Solidity. Javascript menurut (Sunyoto,2007:17) adalah bahasa scripting yang popular di internet dan dapat bekerja di sebagian besar browser popoler seperti Internet Explorer (IE), Mozilla Firefox, Netscape dan Opera. Kode Javascript dapat disisipkan dalam halaman web menggunakan tag SCRIPT.

Beberapa hal tentang Javascript:

1. *Javascript* didesain untuk menambah interaktif suatu web
2. *Javascript* merupakan sebuah bahasa *scripting.*
3. Bahasa *scripting* merupakan bahasa pemrograman yang ringan.
4. *Javascript* berisi baris kode yang dijalankan di computer (web browser).
5. *Javascript* biasanya disisipkan (embedded) dalam halaman HTML.

6. *Javascript* adalaha bahasa interpreter (yang berarti sintaks dieksekusi tanpa proses kompilasi)

2.21.1 React Js

React.js adalah library (pustaka) UI yang merupakan bagian dari proyek dan dikembangkan salah satu perusahaan teknologi terbesar di dunia yaitu Facebook untuk berfokus pada kompabilitas dan kecepatan kinerja pada bagian tampilan aplikasi Front-End [45]. React dapat melakukan rendering user interface (UI) yang kompleks dengan kinerja yang tinggi [46]. Hal yang mendasar dibalik React adalah Virtual DOM. React JS secara efektif menggunakan Virtual DOM yang dapat membuat fungsi baik dari client-side atau server-side. Virtual DOM membuat subtree dari node berdasarkan perubahan status dan dapat memanipulasi DOM sesedikit mungkin untuk menjaga tiap komponen tetap di-update.

React memanfaatkan ekstensi berbasis XML yang nyaman untuk JavaScript, yang dikenal sebagai JSX, untuk menghasilkan komponen sebagai kombinasi dari beberapa node XML [47].

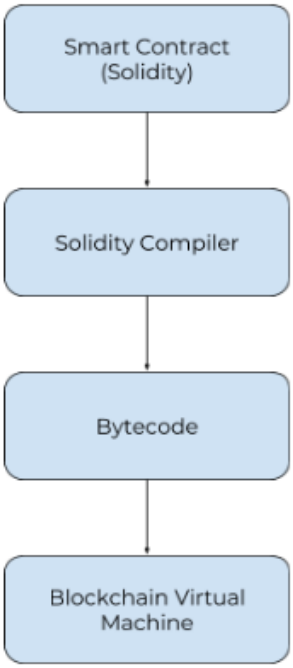
2.21.2 Solidity

Solidity adalah bahasa berorientasi objek (Object Oriented Programming) yang dirancang untuk menulis Ssmart Contract dalam Ethereum dan merupakan bahasa tingkat tinggi serta bisa dikatakan sebagai bahasa berorientasi kontrak (Contract Oriented Programming) dengan eksperimental pertama [48]. Solidity ditulis secara statis, mendukung inheritance (pewarisan) library, dan tipe kompleks yang ditentukan pengguna di antara fitur-fitur lainnya. Bahasa pemrograman Solidity mirip dengan javascript karena pada javascript terdapat class, class yang terdapat pada soldity disebut contract. Syntax dan function nya pun banyak kemiripan dengan javascript. Contoh penggunaan syntax pada Solidity:

```
pragma solidity >=0.4.0 < 0.7.0;
contract SimpleStorage{
        uint storedData;
        function set(uint x) public {
                storedData = x;
        }
        function get() public view returns (uint){
                return storedData;
        }
}
```

Penulisan syntax solidity pada baris pertama source code dimulai dengan syntax "pragma solidity" untuk memulai dengan bahasa pemrograman solidity dilanjutkan dengan range versi dari solidity "0.4.0 < 0.7.0".

Solidity ditulis dengan format file **.sol**. Sebenarnya solidity bukanlah bahasa yang sebenarnya yang dieksekusi pada Blockchain Virtual Machine. Solidity adalah bahasa yang bertujuan untuk mempermudah pembuatan smart contract. Saat akan compile ataupun deploy smart contract, dibutuhkan yang namanya Solidity Compiler. Melalui solidity compiler tersebut smart contract akan di compile ke dalam bytecode yang nantinya bytecode tersebut lah yang akan dieksekusi oleh virtual machine.



Gambar 2.16 Flowchart Solidity

Sumber: https://medium.com/@emurgo_id/bahasa-pemrograman-solidity-6cd597503a6f

Ada beberapa tipe data basic pada solidity, seperti hal nya bahasa pemrograman lain nya. Dalam solidity ada string, integer, array, dan lain-lain.

a. String

Sama seperti bahasa lain nya, tipe data string digunakan untuk membuat variabel yang berupa sekumpulan karakter.

b. Bool

Bool adalah tipe data Boolean.

c. Int

Int tau biasa disebut integer. Dalam solidity, tipe data ini harus berupa angka positif atau negatif. Tidak ada desimal untuk tipe data ini. Pada tipe data **int** ini dibagi menjadi int8, int16, int32, ..., int256. Angka pada tipe data int tersebut adalah sebuah angka bit yang akan digunakan untuk menyimpan angka yang dispesifikasikan seberapa besar angka yang dimiliki oleh variabel.

Tabel 2.3 Spesifikasi Besaran Int

Integer

Nama	Batas Bawah	Batas Akhir
Int8	-128	127
Int16	-32.768	32.767
Int32	-2.147.48.648	-2.147.483.647
…	…	…
Int256	Angka yang sangat kecil	Angka yang sangat besar

d. Uint

Uint atau biasa disebut unsigned integer. Dalam solidity, tipe data ini harus berupa angka positif kebalikan dari int. Tidak ada angka negatif dan desimal untuk tipe data ini. Sama seperti tipe data **int**, pada tipe data **uint** ini dibagi menjadi **uint8**, **uint16**, **uint32**, **…**, **uint256**. Angka pada tipe data uint tersebut adalah sebuah angka bit yang akan digunakan untuk menyimpan angka yang dispesifikasikan seberapa besar angka yang dimiliki oleh variabel.

Tabel 2.4 Spesifikasi Besaran Uint

Integer		
Nama	**Batas Bawah**	**Batas Akhir**
uint8	0	255
uint16	0	65.535
uint32	0	4.294.967.295
…	…	…
uint256	0	Angka yang sangat besar

e. Fixed / ufixed

Fixed / ufixed adalah tipe data untuk variable angka yang mempunyai nilai dan bilangan decimal.

f. Address

Address dalam solidity untuk membuat membuat variabel yang berisi address account ataupun address contract account. Solidity dibuat khusus untuk mengembangkan smart contract sehingga tipe data address dibuat. Address account dibuat dalam bentuk angka acak dengan tipe hexadecimal. Contoh address dalam hexadecimal adalah "0x2a9054870cBB655ae3cd5F8231199dA7aBb6b0b4".

2.21.3 Node Js

Node.js adalah teknologi I / O asinkron open source dan event-driven untuk membangun server web yang skalabel dan efisien [49]. Node.js merupakan platform yang tangguh untuk mengembangkan aplikasi yang tak terhitung jumlahnya. Developer dapat menulis aplikasi dalam satu bahasa yaitu Javascript yang mampu berjalan di browser, server, dan berbentuk mobile [49]. Javascript hanya berjalan pada client-side, makan Node Js ada untuk melengkapi peran Javascript pada sisi server-side. Node Js memiliki library server HTTP sendiri sehingga memungkinkan untuk menjalankan server web tanpa menggunakan program server seperti Apache atau Nginx.

Berbeda dengan bahasa pemrograman sisi server pada umumnya yang bersifat *blocking*, Node.js bersifat *non-blocking*, sebagaimana halnya JavaScript bekerja. Node.js berjalan dengan basis event (*event-driven*). Maksud dari *Blocking* secara sederhana adalah, bahwa suatu kode program akan dijalankan hingga selesai, baru kemudian beralih ke kode program selanjutnya.

2.22 Sublime Text

Sublime Text adalah sebuah teks editor yang digunakan untuk membuka file apapun, menulis bahasa pemrograman yang mensupport banyak bahasa pemrograman, melakukan markup, dan lainnya. Antarmuka Sublime Text sangat sederhana sehingga pengguna bisa berinteraksi dengan nyaman. Sebagai teks editor yang open source, Sublime Text juga memiliki lisensi berbayar untuk fitur kenyamanan lebih. Selain itu, fungsinya yang dirasa ringan dan ramah pada tiap perangkat Personal Computer ataupun Laptop menjadi

keunggulan utama dalam teks editor ini. Sublime Text kompatibel untuk setiap bahasa pemrograman dan project yang akan dibuat sehingga waktu dan efisiensi menjadi manfaatnya. Beberapa kelebihan yang menjadi fitur utama pada Sublime Text yaitu:

1. Goto anything

   Digunakan untuk membuka file diawali dengan menarik satu project file yang sedang dikerjakan kemudian tekan ctrl + p maka kita dapat mencari file yang akan dibuka dengan menuliska nama filenya.
2. Multiple Selection

   Berfungsi untuk perubahan pada sintaks pada saat yang sama dalam beberapa baris yang berbeda.
3. Command Pallete

   Berfungsi untuk menutup semua file, convert cas, lower case, remove tag, dan lainnya.
4. Distraction Free Mode

   Digunakan untuk merubah tampilan menjadi layar penuh untuk menghilangkan segala distraksi pada layar desktop.
5. Split Editing

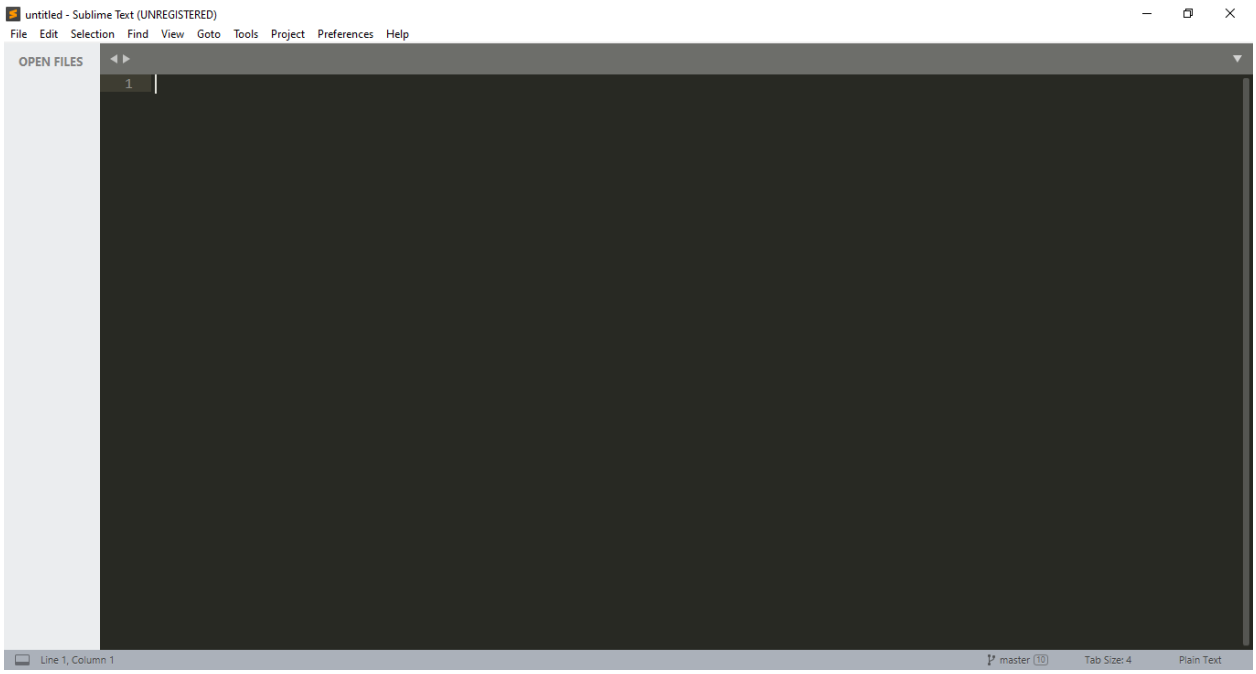
   Membagi dua lokasi file bersebelahan pada satu file dengan baris atau kolom yang diinginkan.
6. Instant Project Switch

   Mengcapture semua pekerjaan pada file project yang sedang dibuka termasuk file yang dirubah dan belum disave.
7. Plugin API Switch

   Sublime Text berbasis Python Plugin API maka konsol built in Python secara interaktif dan real time.
8. Customize Anything

   Bindings Key, Menu, Snippets, Macro, dan lainnya disesuaikan dengan file JSON memberikan fleksibilitas pengaturan.
9. Cross Platform

   Sublime Text tersedia di berbagai platform yaitu Windows, Mac, Linux, OS X dengan satu lisensi untuk semua sistem operasi.



Gambar 2.17 Panel Sublime

2.23 Ganache

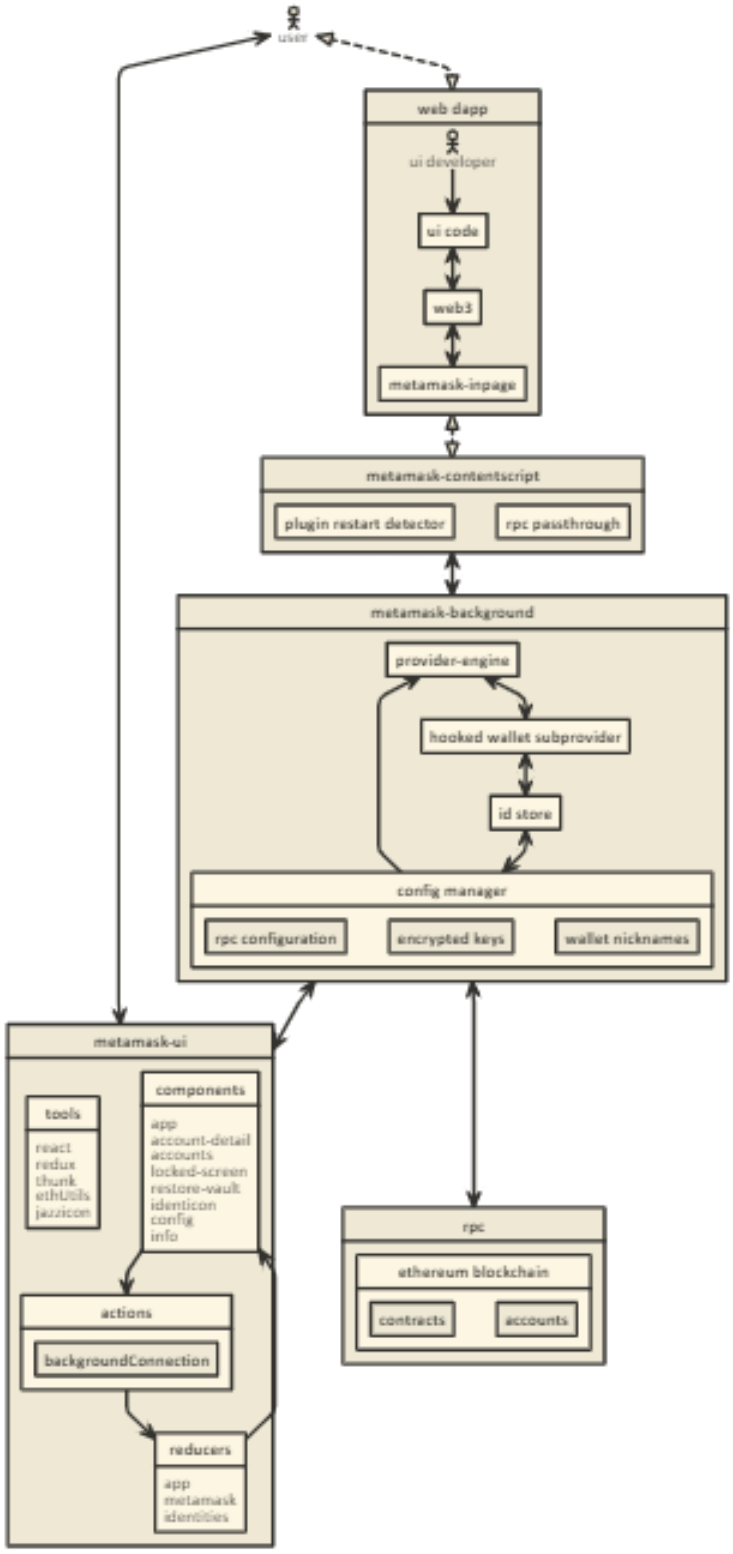
Ganache adalah jaringan Blockchain lokal (virtual) bagi pengembangan Ethereum yang dapat digunakan untuk men-deploy kontrak, mengembangkan aplikasi, dan menjalankan beberapa unit testing berbasis desktop dan command line tool. Ganache menyiapkan 10 akun Ethereum address secara default, lengkap dengan private key dan balance dengan tiap akun address mendapatkan 100 ETH.

Dalam Ganache tidak ada miner sehingga tidak dapat digunakan untuk mining dan hanya bisa digunakan untuk melakukan transaksi. Ganache bisa digunakan untuk melakukan unit test yang akan dieksekusi pada jaringan Blockchain, men-deploy Smart Contract, memanggil function, dan mereset semuanya dari awal dengan keadaan 100 ETH. Dengan menjalankan Ganache kita mempunyai node Ethereum node sendiri dalam bentuk virtual. Jaringan Blockchain Public memerlukan biaya gasLimit dan gasPrice untuk setiap unit testing sehingga tidak efektif jika menggunakannya dalam tahap pengembangan.

2.24 Metamask Chrome Extension

Metamask adalah dompet (wallet) yang berbasis extension yang berjalan pada browser (Chrome, Firefox, Opera, atau Brave Browser) dapat dengan mudah digunakan dan nyaman untuk pengujian karena dapat terhubung ke berbagai node Ethereum untuk menguji

Blockchain [50]. Metamask merupakan dompet (wallet) Ethereum berbasis web. Metamask bisa juga disebut portal bagi web2 dan web3 sehingga biasa disebut “web3 provider”. Dengan kata lain, metamask memungkinkan penggunanya untuk menyimpan data terkait Ethereum seperti alamat public dan private key seperti dompet Ethereum lainnya, dan memungkinkan pengguna untuk berinteraksi dengan situs web yang menjalankan aplikasi berbasis Ethereum dan smart contract (web browser menjadi Ethereum browser).



Gambar 2.18 Arsitektur Metamask Extension

Sumber: https://awesomeopensource.com/project/MetaMask/metamask-extension

Bagi pengembang, Metamask memungkinkan untuk merancang dan menjalankan Ethereum Dapps di web browser tanpa menjalankan node Ethereum sepenuhnya. Metamask

terintegrasi dengan blockchain Ethereum untuk aplikasi web. Metamask mempunyai konsep secure sign-on, menyediakan user interface untuk mengelola identitas dari berbagai situs yang berbeda dan melakukan transaksi blockchain.