BAB 2
LANDASAN TEORI

2.1 Udang Vaname

Udang putih merupakan udang introduksi yang secara resmi ditetapkan sebagai salah satu komoditas unggulan perikanan budidaya oleh Menteri DKP pada tahun 2001, dan sejak itu perkembangan budidaya sangat cepat. Saat ini budidaya udang putih telah dikomersialkan dan berkembang sangat pesat, dikarenakan peminatnya yang semakin meningkat baik dari dalam negeri maupun dari luar negeri. Selain Indonesia, negara-negara yang telah mengembangkan udang putih antara lain China, Taiwan, dan Thailand. Udang putih mempunyai ciri-ciri mampu hidup pada kisaran salinitas 5-45 ppt dengan salinitas optimal 10-30 ppt; kisaran suhu 24-32 ^{0}C dengan suhu optimal 28-30 ^{0}C; mampu bertahan pada oksigen 0,8 ppm selama 3-4 hari tetapi disarankan DO 4 ppm. pH air 7-8,5; kebutuhan protein rendah yaitu 32 % dengan FCR <1,5>.

Tabel 2.1 Standar Kualitas Air

No	Parameter	Status	
		Standar	Optimal
1	pH air	7 – 8,5	7 – 8,5
2	Salinitas air	5 – 45 ppt	10 – 30 ppt
3	Suhu air	24 – 32 ^{0}C	28 – 30 ^{0}C
4	Tinggi air	90 – 130 cm	100 – 120 cm

Kehadiran varietas udang putih tidak hanya menambah pilihan bagi petani tetapi juga dapat menopang kebangkitan usaha udang di Indonesia, akan tetapi budidaya udang putih tidak semudah dibayangkan. Kegiatan pembesaran merupakan bagian penting dalam budidaya udang putih yang harus diperhatikan dengan baik, karena banyak kegagalan dalam budidaya udang putih diakibatkan oleh kelalaian dalam proses pembesaran, terutama dari manejemen pakan, kualitas air media pemeliharaan, penanganan maupun genetiknya, sehingga serangan penyakit tidak dapat dihindarkan [4].

2.1.1 Proses Pembesaran Udang Vanamei di Tambak

a) Persiapan tambak

Persiapan tambak bekas yang perlu dilakukan yaitu pembersihan dan pengeringan tambak dengan bantuan sinar matahari. Sinar matahari berfungsi sebagai desinfektan, membantu proses oksidasi yang dapat menetralkan sifat keasaman tanah, menghilangkan gas-gas beracun dan membantu membunuh telur-telur hama yang tertinggal (Suwanto, 2000)

Persiapan sarana tambak meliputi: a) penataan dan pemasangan pompa air; b) setting kincir air; c) pemasangan (perbaikan) PVC *central drain* /pintu monik dan saringan pembuangan air; d) pembuatan dan pemasangan jembatan untuk kontrol pemberian pakan dan kondisi udang; e) Pengisian air laut didalam tambak sampai kedalaman 130-150 cm bila lahan sudah siap; f) setting dan pemasangan sarana dan fasilitas lainnya (Suwanto, 2000)

Persiapan plankton segera tumbuh meliputi: 1) Menggunakan pupuk organik. Selain itu, bisa ditambahkan bahan probiotik (bakteri pengurai) yang mengandung *Bacillus sp.* karena secara tidak langsung bisa mempercepat proses pembentukan plankton; 2) Bila kondisi cuaca mendukung lingkungan tambak, plankton akan terbentuk dalam waktu 3-5 hari (Suwanto, 2000)

b) Memilih dan penebaran benur

Menurut Suwanto (2000), tips memilih benur vaname yaitu:

1) Benur vaname dipanen setelah mencapalai PL 10 atau organ insang telah sempurna Benur sehat dengan ciri-ciri tubuh transparan, bergerak aktif,
2) Saat berenang di wadah, benur melaju melawan arus air
3) Ukuran benur harus seragam (80 %)
4) Benur dinyatakan lolos uji virus dan bebas patogen
5) Benih yang sudah terseleksi di angkut ke tambak dan kemudian sebelum dilepas terlebih dahulu diadaptasikan terhadap parameter kualitas air yaitu suhu, salinitas, pH, dan parameter kualitas air lainnya

secara perlahan-lahan, lamanya adaptasi berkisar 5-15 menit. Waktu penebaran yang baik diusahakan pagi (jam 05-07.00 Wib). Padat penebaran yang optimal pada pembesaran udang dengan teknologi intensif pada sistem ini berkisar antara 25-50 ekor per m^2 (tergantung faktor dukung lahan dan sarana penunjang lainnya) (Suwanto, 2000)

c) Pengelolaan pakan

Pakan yang umum diberikan berupa pakan buatan dengan jenis crumble dan pellet dan dapat diberikan jenis pakan tambahan lainnya (pakan segar). Pemberian pakan dimulai sejak udang ditebar ke tambak hingga pemanenan hasil. Pengaturan dan pemberian pakan disesuaikan berdasarkan hasil pengamatan dan sampling di lapangan. Selain pakan buatan yang di berikan, diberikan pula pakan segar berupa cumi segar dengan dosis 2-4 % (Suwanto, 2000)

Pemberian pakan berlebih bisa menimbulkan pencemaran air. Akibatnya, udang mudah stress sehingga pertumbuhan udang terhambat. Selain itu, daya tahan udang terhadap penyakit pun menurun sehingga angka mortalitasnya meningkat (Suwanto, 2000)

Keberadaan plankton dalam air media pemeliharaan udang khususnya jenis phytoplankton yang menguntungkan dan sangatlah dibutuhkan, baik dari segi keanekaragamannya maupun kelimpahannya. Fungsi dan peran plankton pada air media pemeliharaan diantaranya yaitu: 1) Sebagai pakan alami untuk pertumbuhan awal udang yang dipelihara; 2) Sebagai penyangga (buffer) terhadap intensitas cahaya matahari; dan 3) Sebagai indikator kestabilan lingkungan air media pemeliharaan (Suwanto, 2000)

d) Pengelolaan kualitas air

Untuk mendapatkan parameter kualitas air yang optimal dan kondisi prima, maka selama masa pemeliharaan dilakukan penggantian volume air secara terprogram dengan memperhatikan kualitas air yang penting seperti suhu, O_2, pH, nitrit, salinitas, dan gas-gas beracun lainnya (Ditjenkan, 2003)

Parameter-parameter tersebut akan mempengaruhi proses metabolisme tubuh udang, seperti keaktifan mencari pakan, proses pencernaan, dan pertumbuhan udang (Suwanto, 2000)

e) Sampling

Pengamatan udang selama masa pemeliharaan merupakan suatu kegiatan yang harus dilakukan untuk mengetahui; 1) Kesehatan dan kondisi udang; 2) Pertambahan berat harian; 3) Tingkat kelangsungan hidup; dan 4) Biomasa. Pengamatan pertumbuhan umumnya dilakukan pengamatan pada anco dan penjalaan dengan cara mengambil beberapa contoh sampel udang (Suwanto, 2000)

f) Pengendalian hama dan penyakit

Penyakit dapat muncul dan menyerang udang vanamei. Beberapa jenis penyakit yang menyerang udang vaname disebabkan oleh predator, parasit, bakteri dan jamur serta virus.

Untuk meminimalkan infeksi hama dan penyakit pada udang vaname meliputi:1) menggunakan benih udang vaname yang berkualitas baik; 2) mendeteksi dan memonitoring kesehatan udang secara rutin dan teratur; 3) menjaga kualitas air agar tetap stabil sehingga udang tidak stres; 4) mengaplikasikan probiotik dan imunostimulan untuk meningkatkan imunitas udang terhadap serangan penyakit; 5) Terapkan prinsip-prinsip *biosecuritas* (Suwanto, 2000)

g) Pemanenan

Pada umumnya pemanenan udang dilakukan setelah umur pemeliharaan > 100 hari, akan tetapi pelaksanaan panen dapat memperhatikan pertumbuhan serta harga udang di pasaran. Teknik panen yang sering dilakukan adalah dengan cara menurunkan volume air secara bertahap dengan gravitasi atau pompa air, bersamaan itu, dilakukan penangkapan udang secara bertahap pula dengan kemampuan peralatan yang tersedia dan akhirnya dilakukan penangkapan secara manual apabila kontruksi dasar tambak tidak tuntas keringnya. Pemanenan dilakukan

malam hingga pagi hari atau diusahakan pada suhu rendah dengan tujuan untuk mengurangi resiko kerusakan mutu udang (Suwanto, 2000)

2.2 *Internet of Things* (IoT)

Internet of Things (IoT) merupakan sebuah konsep yang bertujuan untuk memperluas manfaat dari konektivitas internet yang tersambung secara terusmenerus. Adapun kemampuan seperti berbagi data remote control, dan termasuk juga pada benda dunia nyata. Bahan pangan, elektronik, peralatan apa saja, koleksi termasuk benda hidup, yang semuanya tersambung ke jaringan lokal dan global melalui sensor tertanam dan selalu "on" [11].

Interaksi antara manusia dengan manusia sudah sangat biasa sejak zaman dahulu kala. Interaksi antara manusia dengan mesin sudah biasa pula, sejak adanya penemuan teknologi seperti komputer atau gadget devices lainnya. Namun, Menurut analisa McKinsey Global Institute, *internet of things* adalah sebuah teknologi yang memungkinkan kita untuk menghubungkan mesin, peralatan, dan benda fisik lainnya dengan sensor jaringan dan aktuator untuk memperoleh data dan mengelola kinerjanya sendiri, sehingga memungkinkan mesin untuk berkolaborasi dan bahkan bertindak berdasarkan informasi baru yang diperoleh secara independen. Sebuah publikasi mengenai *Internet of things* in 2020 menjelaskn bahwa *internet of things* adalah suatu keadaan ketika benda memiliki identitas, bisa beroperasi secara intelijen, dan bisa berkomunikasi dengan sosial, lingkungan, dan pengguna.

Gambar 2.1 Ilustrasi dari *Internet of Things*[11]

Dapat kita simpulkan bahwa *internet of things* membuat suatu koneksi antara mesin dengan mesin, sehingga mesin-mesin tersebut dapat berinteraksi dan bekerja secara independen sesuai dengan data yang diperoleh dan diolahnya secara mandiri. Tujuannya adalah untuk membuat manusia berinteraksi dengan benda dengan lebih mudah, bahkan supaya benda juga bisa berkomunikasi dengan benda lainnya.

Teknologi *internet of things* sangat luar biasa. Jika sudah direalisasikan, teknologi ini tentu akan sangat memudahkan pekerjaan manusia. Manusia tidak akan perlu lagi mengatur mesin saat menggunakannya, tetapi mesin tersebut akan dapat mengatur dirinya sendiri dan berinteraksi dengan mesin lain yang dapat berkolaborasi dengannya. Hal ini membuat mesin-mesin tersebut dapat bekerja sendiri dan manusia dapat menikmati hasil kerja mesin-mesin tersebut tanpa harus repot-repot mengatur mereka.

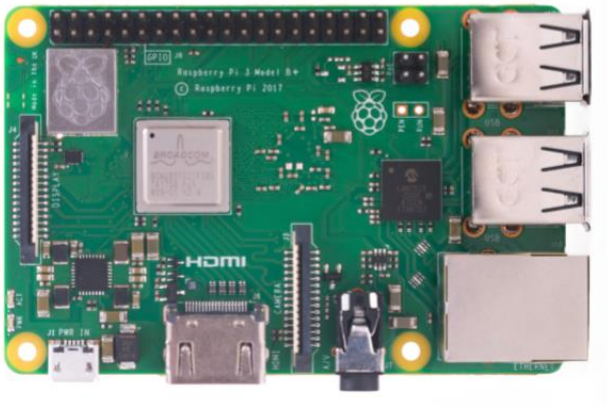
Cara kerja dari *internet of things* cukup mudah. Setiap benda harus memiliki sebuah IP Address. IP Address adalah sebuah identitas dalam jaringan yang membuat benda tersebut bisa diperintahkan dari benda lain dalam jaringan yang sama. Selanjutnya, IP address dalam benda-benda tersebut akan dikoneksikan ke jaringan internet. Saat ini, koneksi internet sudah sangat mudah kita dapatkan. Dengan demikian, kita dapat memantau benda tersebut bahkan memberi perintah kepada benda tersebut. Sebagai contoh, jika ada speaker yang memiliki IP address dan terkoneksi internet di Amerika Serikat, kita dapat memerintahkan speaker tersebut untuk menyalakan musik walaupun kita berada di Indonesia. Yang kita perlukan hanyalah koneksi internet. Lalu apa hubungannya dengan *internet of things*, Setelah sebuah benda memiliki IP address dan terkoneksi dengan internet, pada benda tersebut juga dipasang sebuah sensor. Sensor pada benda memungkinkan benda tersebut memperoleh informasi yang dibutuhkan. Setelah memperoleh informasi, benda tersebut dapat mengolah informasi itu sendiri, bahkan berkomunikasi dengan benda-benda lain yang memiliki IP address dan terkoneksi dengan internet juga. Akan terjadi pertukaran informasi dalam komunikasi antara benda-benda tersebut. Setelah pengolahan informasi selesai, benda tersebut dapat bekerja dengan sendirinya, atau bahkan memerintahkan benda

lain juga untuk ikut bekerja. Jadi, dalam *internet of things* manusia akan bertindak sebagai raja dan akan dilayani oleh benda-benda disekitarnya [11].

2.3 *Raspberry Pi 3 Model B+*

Raspberry Pi 3 merupakan generasi ketiga dari keluarga *raspberry pi. Raspberry pi 3 Model B+* adalah prduk terbaru dalam jajaran seri *raspberry pi 3*, memiliki *RAM* 1 GB dengan chipset Broadcom BCM2873B0 Cortex A53 64-bit berkecepatan 1,4GHz. Chipset ini memiliki manajemen suhu yang lebih baik sehingga dapat berjalan pada kecepatan penuh dengan lebih lama sebelum mengalami throttling akibat panas. Perangkat ini menggunakan koneksi wireless dual band yang mendukung 802.11ac yang lebih kencang dibanding generasi sebelumnya serta dilengkapi juga dengan Bluetooth 4,2/BLE, jaringan Ethernet yang lebih cepat, dan kemampuan melakukan PoE melalui HAT PoE yang terpisah. *Raspberry Pi* 3 juga memiliki 4 *USB port*, 40 *pin GPIO*, *Full HDMI port*, *Port Ethernet*, *Combined 3.5mm audio jack and composite video*, *Camera interface* (CSI), *Display interface* (DSI), slot kartu *Micro SD* (Sistem tekan-tarik, berbeda dari yang sebelum nya ditekan-tekan), dan V*ideoCore* IV *3D graphics core*.

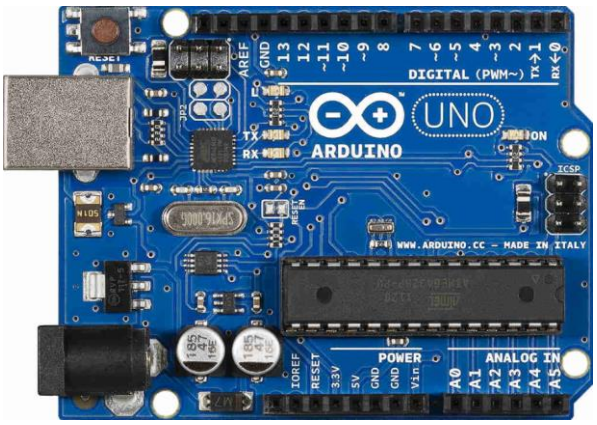
LAN nirkabel dual-band hadir dengan sertifikasi penyesuaian modular, memungkinkan *board* dirancang untuk menjadi produk akhir dengan pengujian kualitas LAN nirkabel yang berkurang secara signifikan, meningkatkan biaya dan waktu untuk memasarkan. *Raspberry Pi 3 Model B+* mempertahankan jejak mekanis yang sama seperti *Raspberry Pi 2 Model B* dan *Raspberry Pi 3 Model B* [21]. Pada penelitian ini *raspberry pi* digunakan sebagai mini pc untuk server website dan juga untuk pengolahan data yang didapat dari sensor melaui Arduino. Gambar adalah bentuk *raspberry pi 3 model b+*.



Gambar 2.2 Raspberry Pi 3 Model B+ [21]

2.4 Arduino Uno

Arduino Uno adalah board mikrokontroler berbasis dari *datasheet* ATmega328. Memiliki 14 pin *input* dari *output digital* dimana 6 pin *input* tersebut dapat digunakan sebagai *output* PWM dan 6 pin *input analog*, 16 MHz osilator kristal, koneksi USB, jack power, ICSP *header*, dan tombol reset. Untuk mendukung mikrokontroler agar dapat digunakan, cukup hanya menghubungkan *Board* Arduino Uno ke komputer dengan menggunakan kabel USB atau listrik dengan AC ke adaptor DC atau baterai untuk menjalankannya [22].



Gambar 2.3 Arduino Uno [22]

Arduino Uno berbeda dengan semua *board* sebelumnya dalam hal koneksi *USB-to-serial* yaitu menggunakan fitur Atmega8U2 yang diprogram sebagai konverter *USB-to-serial* berbeda dengan *board* sebelumnya yang menggunakan *chip* FTDI *driver USB-to-serial*. Nama “Uno” berarti satu dalam bahasa Italia,

untuk menandai peluncuran Arduino 1.0. Uno dan versi 1.0 akan menjadi versi referensi dari Arduino. Uno adalah yang terbaru dalam serangkaian *board* USB Arduino, dan sebagai model referensi untuk *platform* Arduino, untuk perbandingan dengan versi sebelumnya, lihat indeks *board* Arduino. Arduino uno memiliki memori ATmega328 memiliki 32 KB (dengan 0,5 KB digunakan untuk *bootloader*), 2 KB dari SRAM dan 1 KB EEPROM (yang dapat dibaca dan ditulis dengan EEPROM *liberary*). Arduino dapat diaktifkan melalui koneksi USB atau dengan catu daya eksternal (otomatis). Untuk memberikan perinta operasi-operasi logika, Arduino menggunakan program yang berasal dari hardware dengan prosessor Atmel AVR, sedangkan softwarenya memiliki bahasa pemrograman sendiri yang menyerupai bahasa C.

2.5 Sensor

Sensor adalah suatu perangkat yang mendeteksi perubahan energi yang berada di alam seperti energi listrik, energi fisika, energi kimia, energi biologi, energi mekanik dan sebagainya. Di dalam sebuah sensor terdapat transduser yang berfungsi untuk mengubah besaran mekanis, magnetis, panas, sinar, dan kimia menjadi besaran listrik berupa tegangan, resistansi dan arus listrik.

2.5.1 Sensor pH SEN0161

Sensor pH SEN0161 adalah sebuah sensor yang dapat mengukur tingkat keasaman dan kebasaan suatu larutan. Prinsp kerja utama pH meter adalah terletak pada sensor *probe* berupa elektroda kaca (*glass electrode*) dengan cara mengukur jumlah ion H_3O^+ di dalam larutan. Ujung electrode kaca adalah lapisan kaca setebal 0,1 mm yang berbentuk bulat (*bulb*). Bulb ini dipasangkan dengan silinder kaca non-konduktor atau plastic memanjang, yang selanjutnya diisi dengan larutan HCL (0,1 mol/dm^3). Di dalam larutan HCL, terendam sebuah kawat electrode panjang berbahan perak yang pada permukaannya terbentuk senyawa setimbang AgCl. Konstantanya jumlah larutan HCl pada system ini membuat electrode Ag/AgCl memiliki nilai potensial stabil [23].

Jika larutan bersifat asam, maka ion H^+ akan terikat ke permukaan *bulb*. Hal ini menimbulkan muatan positif terakumulasi pada lapisan “gel”. Sedangkan jika

larutan bersifat basa, maka ion H^+ dari dinding *bulb* terlepas untuk bereaksi dengan larutan tadi. Hal ini menghasilkan muatan negative pada dinding *bulb.* Pertukaran ion hydronium (H^+) yang terjadi antara permukaan *bulb* kaca dengan larutan sekitarnya inilah yang menjadi kunci pengukuran jumlah ion H_3O^+ di dalam larutan. Kesetimbangan pertukaran ion yang terjadi di antara fase dinding kaca *bulb* dengan larutan, menghasilkan beda potensial di antara keduanya [23]. Pada penelitian ini sensor pH SEN0161 digunakan untuk mendapatkan nilai pH air di tambak udang vaname. Gambar merupakan bentuk sensor pH SEN0161.

Gambar 2.4 Sensor pH SEN0161 [23]

2.5.2 Sensor Salinitas

Salinitas adalah tingkat keasinan atau kadar garam terlarut dalam air. Sensor Salinitas adalah sensor kadar garam air, memiliki *output* sinyal analog yang dikalibrasi dengan sensor salinitas yang kompleks. Prinsip kerja sensor salinitas didasarkan pada konduktivitas listrik pada air. Dalam pengukurannya, sensor salinitas menggunakan sifat dari air, yaitu air sebagai konduktor listrik yang baik. Misalnya dalam pengukuran salinitas air laut, diketahui bahwa air laut berisi banyak kotoran seperti natrium klorida, magnesium klorida, kalsium klorida dan sebagainya. Ion-ion klor membantu dalam konduksi dan karenanya kotoran ini meningkatkan konduktivitas air [24]. Pada penelitian, sensor

salinitas digunakan untuk mengukur kadar garam air di tambak udang vaname. Gambar merupakan bentuk sensor salinitas.

Gambar 2.5 Sensor Konduktivitas Air [24]

2.5.3 Sensor Suhu DS18B20

Sensor suhu DS18B20 adalah suatu komponen yang dapat mengubah besaran panas menjadi besaran listrik sehingga dapat mendeteksi gejala perubahan suhu pada obyek tertentu [24]. Sensor suhu melakukan pengukuran terhadap jumlah energi panas/dingin yang dihasilkan oleh suatu obyek sehingga memungkinkan kita untuk mengetahui atau mendeteksi gejala perubahan-perubahan suhu tersebut dalam bentuk *output* analog maupun digital. Sensor ini bekerja dengan protokol komunikasi satu kabel / *one wire* dan mempunyai kemampuan untuk mendeteksi suhu dari -10 sampai +85 derajat Celsius. Gambar merupakan betuk sensor suhu DS18B20.

Gambar 2.6 Sensor Suhu DS18B20 [24]

2.5.4 Sensor Ultrasonik HC-SR04

Sensor ultrasonik adalah sensor yang bekerja berdasarkan prinsip pantulan gelombang suara dan digunakan untuk mendeteksi keberadaan suatu objek tertentu di depannya, frekuensi kerjanya pada daerah diatas gelombang suara dari 40 KHz hingga 400 KHz. Sensor ultrasonik terdiri dari dari dua unit, yaitu unit pemancar dan unit penerima. Sensor jarak ultrasonik HC-SR04 adalah sensor 40 KHz. HC-SR04 merupakan sensor ultrasonik yang dapat digunakan untuk mengukur jarak antara penghalang dan sensor [25]. 2 Komponen utama sebagai penyusunnya yaitu *ultrasonic transmitter* dan *ultrasonic receiver*. Fungsi dari *ultrasonic transmitter* adalah memancarkan gelombang ultrasonik dengan frekuensi 40 KHz kemudian *ultrasonic receiver* menangkap hasil pantulan gelombang ultrasonik yang mengenai suatu objek. Waktu tempuh gelombang ultrasonik dari pemancar hingga sampai ke penerima sebanding dengan 2 kali jarak antara sensor dan bidang pantul. Gambar merupakan betuk sensor ultrasonic HC-SR04.

Gambar 2.7 Sensor Ultrasonik HC-SR04 [25]

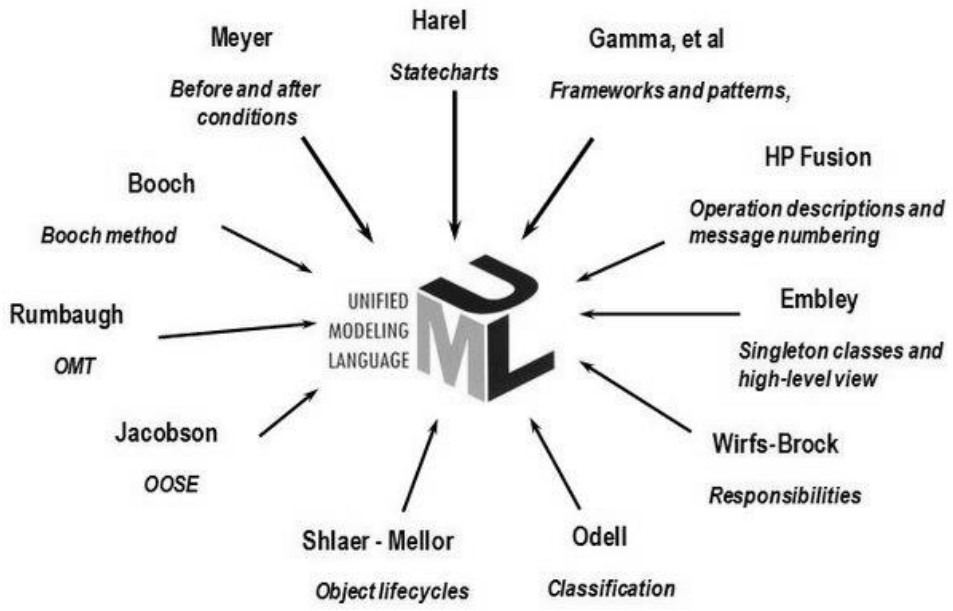
2.6 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa permodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atau svisi mekanisme yang efektif untuk berbagi dan mengkomunikasikan rancagan mereka dengan yang lain.

Unified Modeling Language (UML) merupakan kesatuan dari Bahasa permodelan yang dikembangkan booch, *Object Modeling Technique* (OMT) dan

Object Orented Software Engineering (OOSE) [12]. Metode ini menjadikan proses analisis dan design kedalam empat tahapan iterative, yaitu identifikasi kelas-kelas, dan objek-objek, identifikasi semantic dari hubungan objek dan kelas tersebut, perincian *interface* dan implementasi. Keunggulan metode Booch adalah pada detail dan kayanya dengan notasi dan elemen, permodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstruktur dan pemodelan entity-relationship. Tahapan utama dalam metodologi ini adalah nalisis, design sistem, design objek dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung konsep OO. Metode OOSE dari Jacobson lebih memberi menekanan pada use case. OOSE memiliki tiga tahapan yaitu membuat model requirement dan analisis, design dan implementasi, dan model pengujian. Keunggulan metode ini adalah mudah dipelajari karena memiliki notasi yang sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak.

Dengan UML, metode Booch, OMT dan OOSE digabungkan dengan membuang elemen-elemen yang tidak praktis ditambah dengan elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspresif dan seragam daripada metode lainnya []. Gambar 2.1 berikut adalah unsur-unsur yang membentuk UML.



Gambar 2.8 Unsur-unsur pembentuk UML [12]

Sebagai sebuah notasi grafis yang relative sudah dibakukan (*open standard*) dan dikontrol oleh OMG (*Object Management Group*) mungkin lebih dikenal sebagai badan yang berhasil membakukan CORBA (Common Object Request Broker Architecture), UML menawarkan banyak keistimewaan. UML tidak hanya dominan dalam penotasian di lingkunan OO tetapi juga popular di luar lingkungan OO. Paling tidak ada tiga karakter penting yang melekat di UML yaitu sketsa, cetak biru dan Bahasa pemograman. Sebagai sebuah sketsa, UML bisa berfungsi sebagai jembatan dalam mengkomunikasikan beberapa aspek dari sistem. Dengan demikian semua anggota tim akan mempunyai gambaran yang sama tentang suatu sistem. UML bisa juga berfungsi sebagai sebuah cetak biru karena sangat lengkap dan detil. Dengan cetak biru ini maka akan bisa diketahui informasi detail tentang coding program (*forward engineering*) atau bahkan membaca program dan menginterpretasikannya kembali kedalam diagram (*reverse engineering*). *Reverse engineering* sangat berguna pada situasi dimana code program yang tidak terdokumentasi asli hilang atau bahkan elum dibuat sama sekali. Sebagai bahasa pemograman, UML dapat menterjemahkan diagram yang ada di UML menjadi code program yang siap untuk di jalankan [12].

2.6.1 Diagram UML

UML menyediakan berbagai macam diagram untuk memodelkan aplikasi perangkat lunak berorientasi objek [12]. Namun, pada penelitian ini hanya menggunakan 4 macam diagram saja untuk memodelkannya. Yaitu:

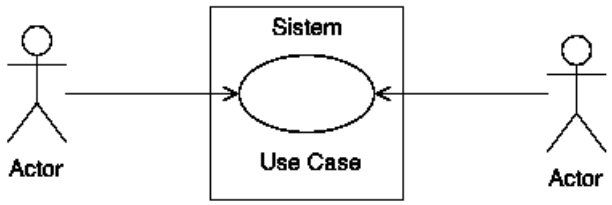
1. *Use Case Diagram*

   Use case diagram adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara user (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut *scenario*. Setiap *scenario* mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, system yang lain, perangkat keras atau urutan waktu [12]. Dengan demikian secara singkat bisa dikatan *use case* adalah

serangkaian *scenario* yang digabungkan Bersama-sama oleh tujuan umum pengguna.

Use case adalah alat bantu terbaik guna menstimulasi pengguna potensial untuk mengatakan tentang suatu system dari sudut pandangnya. Tidak selalu mudah bagi pengguna untuk menyatakan bagaimana mereka bermaksud menggunakan sebuah system. Karena system pengembangan tradisional sering cerboh dalam melakukan analisis, akibatnya pengguna seringkali susah menjawabnya tatkala dimintai masukan tentang sesuatu [12].

Diagram use case menunjukan 3 aspek dari sistem yaitu : *actor*, *use case* dan *sistem* atau *sub sistem boundary*. *Actor* mewakili peran orang, sistem yang lain atau alat ketika berkomunikasi dengan use case. Gambar 2.2 mengilustrasikan *actor, use case* dan *boundary*.

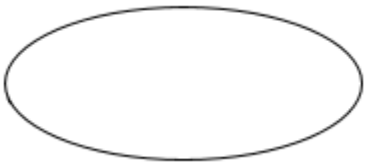


Gambar 2.9 *Use Case Model* [12]

Berikut ini adalah bagian dari sebuah use case diagram :

a. Use Case

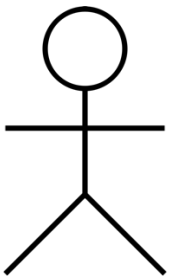
Use case adalah abstraksi dari interaksi antarasistem dengan *actor*. Oleh karena itu sangat penting untuk memilih abstraksi yang cocok. *Use case* dibuat berdasarkan keperluan *actor*. *Use case* harus merupakan ‘apa’ yang dikerjakan software aplikai, bukan ‘bagaimana’ software aplikasi mengerjakannya. Setiap *user case* harus diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan *actor*. Nama *use case* boleh terdiri dari beberapa kata dan tidak boleh ada dua *use case* yang memiliki nama yang sama. Gambar 2.3 menunjukan bentuk *Use Case* dalam UML.



Gambar 2.10 *Use Case* [12]

b. *Actors*

Actors adalah *abstraction* dari orang dan sistem yang lain yang mengaktifkan fungsi dari target sistem. Orang atau sistem bisa muncukl dalam beberapa peran. Bahwa actor berinteraksi dengan *use case*, tetapi tidak memiliki control atas *use case*. Gambar 2.4 menunjukan bentuk *actor* dalam UML.



Gambar 2.11 *Actor* [12]

c. *Relationship*

Relationship adalah hubungan antara *use cases* dengan *actors*. *Relationship* dalam *use case* diagram meliputi:

a. Asosiasi antara *actor* dan *use case*.

Hubungan antara *actor* dan *use case* yang terjadi karena adanya interaksi antara kedua belah pihak. Asosiasi tipe ini menggunakan garis lurus dari actor menuju *use case* baik dengan menggunakan mata panah terbuka ataupun tidak.

b. Asosiasi antara 2 *use case*

Hubungan antara *use case* yang satu dan *use case* lainnya yang terjadi karena adanya interaksi antara kedua belah

pihak. Asosiasi tipe ini menggunakan garis putus-putus/garis lurus dengan mata panah terbuka di ujungnya.

c. Generalisasi antara 2 *actor*

Hubungan *inheritance* (pewarisan) yang melibatkan *actor* yang satu (*the child*) dengan *actor* lainnya (*the parent*). Generalisasi tipe ini menggunakan garis lurus dengan mata panah tertutup di ujungnya.

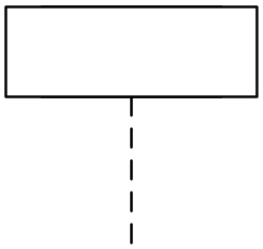
d. Generalisasi antara 2 *use case*.

Hubungan *inheritance* (pewarisan) yang melibatkan *use case* yang satu (*the child*) dengan use case lainnya(*the parent*). Generalisasi tipe ini menggunakan garis lurus dengan mata panah tertutup di ujungnya.

2. *Sequance Diagram*

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah *scenario*. Diagram ini menunjukan sejumlah contoh obyek dan *message* (pesan) yang diletakan diantara obyek-obyek ini di dalam use case. Komponen utama *sequence diagram* terdiri atas objek yang ditulisan dengan kotak segiempat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukan dengan *progress vertical* [12].

Objek diletakan di detak bagian atas diagram dengan urutan dari kiri ke kanan. Mereka diatur dalam urutan guna menyederhanakan diagram, istilah objek dikenal juga dengan *participant*, setiap *participant* terhubung dengan garis titik-titik yang disebut *lifeline*. Sepanjang *lifeline* ada kotak yang disebut *activation*, *activation* mewakili sebuah eksekusi operasi dari *participant*. Panjang kotak ini berbanding lurus dengan durasi *activation* [12].

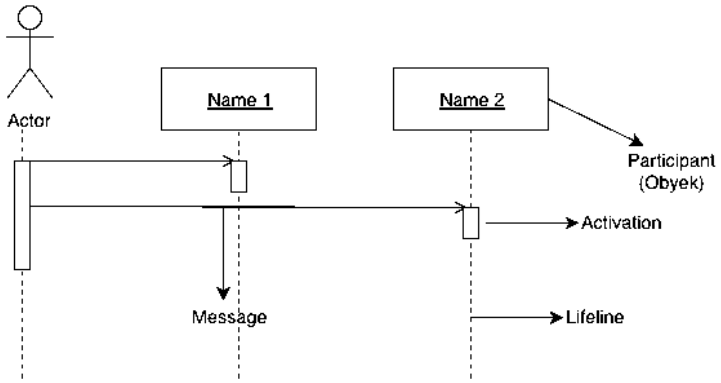


Gambar 2.12 Participant pada sebuah sequence diagram [12]

Sebuah *message* bergerak dari satu *participant* ke *participant* yang lain dan dari satu *lifeline* ke *lifeline* yang lain. Sebuah *participant* bisa mengirim sebuah *message* kepada dirinya sendiri. Sebuah *message* bisa jadi simple, *synchronous* atau *asynchronous. Message* yang simple adalah sebuah perpindahan (*transfer*) *control* dari satu *participant* ke *participant* yang lainnya. Jika sebuah *participant* mengirimkan sebuah message *synchronous*, maka jawaban atas *message* tersebut akan ditunggu sebelum diproses dengan ururs*a*nnya. Namun jika message message *asynchronous* yang dikirimkan, maka jawaban atas *message* tersebut tidak perlu ditunggu.

Gambar 2.13 Simbol-simbol *message* [12]

Time adalah diagram yang mewakili waktu pada arah *vertical*. Waktu dimulai dari atas ke bawah. *Message* yang lebih dekat dari atas akan dijadikan terlebih dahulu dibanding *message* yang lebih dekat ke bawah. Gambar 2.7 menunjukan esensi *symbol* dari *sequence diagram* dan *symbol* kerjanya secara bersama-sama.



Gambar 2.14 Simbol-simbol yang ada pada sequence diagram [12]

Berikut ini merupakan komponen dalam *sequence diagram* :

a. *Activations*

Activations menjelaskan tentang eksekusi dari fungsi yang dimiliki oleh suatu objek.

b. *Actor*

Actor menjelaskan tentang peran yang melakukan serangkaian aksi dalam suatu proses.

c. *Collaboration boundary*

Collaboration boundary menjelaskan tentang tempat untuk lingkungan percobaan dan digunakan untuk memonitor objek.

d. *Parallel vertical lines*

Parallel vertical lines menjelaskan tentang suatu garis proses yang menunjuk pada suatu *state*.

e. *Processes*

Processes menjelaskan tentang tindakan/aksi yang dilakukan oleh aktor dalam suatu waktu.

f. *Window*

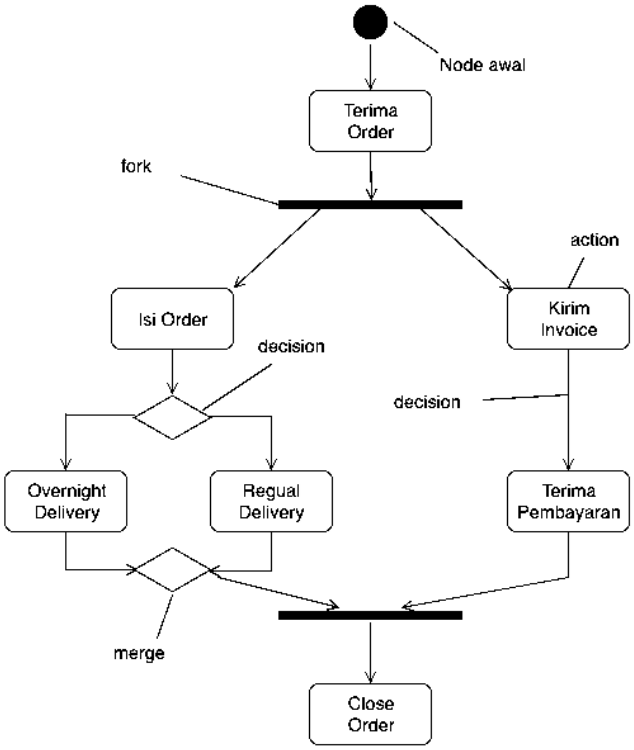
Window menjelaskan tentang halaman yang sedang ditampilkan dalam suatu proses.

g. *Loop*

Loop menjelaskan tentang model logika yang berpotensi untuk diulang beberapa kali.

3. *Activity Diagram*

Activity diagram adalah bagian penting dari UML yang menggambarkan aspek dinamis dari sistem. Logika procedural, proses bisnis dan aliran kerja suatu bisnis bisa dengan mudah dideskripsikan dalam *activity diagram*. *Activity diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaanya dengan *flowchart* adalah *activity diagram* bisa mendukug perilaku pararel sedangkan *flowchart* tidak bisa. Tujuan dari activity diagram adalah untuk menangkap tingkah laku dinamis dari sistem dengan cara menunjukan aliran pesan dari satu aktifitas ke aktifitas lainya. Secara umum activity diagram digunakan untuk menggambarkan diagram alir yang terdiri dari banyak aktifitas dalam sistem dengan beberapa fungsi tambahan seperti percabangan, aliran pararel, swim lane, dan sebagainya. Sebelum menggambarkan sebuah sctivity diagram, perlu adanya pemahaman yang jelas tentang elemen yang akan digunakan di activity diagram. Elemen tuama dalam activity diagram adalah aktifitas itu sendiri. Aktifitas adalah fungsi yang dilakukan oleh sistem Setelah akitifitas terindentifikasi, selanjutnya yang perlu diketahui adalah bagaimana semua elemen tersebut berasosiasi dengan constraint dan kondisi. Langa selanjutnya perlu penjabaran tata letak dari keseluruhan aliran agara bisa ditransofrmasikan ke activity diagram [12]. Gambar 2.8 menunjukan contoh *activity diagram* sederhana.



Gambar 2.15 Contoh activity diagram sederhana [12]

Berikut ini merupakan komponen dalam activity diagram, yaitu :

a. *Activity node*

   Activity node menggambarkan bentuk notasi dari beberapa proses yang beroperasi dalam kontrol dan nilai data

b. *Activity edge*

   Activity edge menggambarkan bentuk *edge* yang menghubungkan aliran aksi secara langsung ,dimana menghubungkan *input* dan *output* dari aksi tersebut

c. *Initial state*

   Bentuk lingkaran berisi penuh melambangkan awal dari suatu proses.

d. *Decision*

   Bentuk wajib dengan suatu *flow* yang masuk beserta dua atau lebih *activity node* yang keluar. *Activity node* yang keluar ditandai untuk mengindikasikan beberapa kondisi.

e. *Fork*

Satu bar hitam dengan satu *activity node* yang masuk beserta dua atau lebih *activity node* yang keluar.

f. *Join*

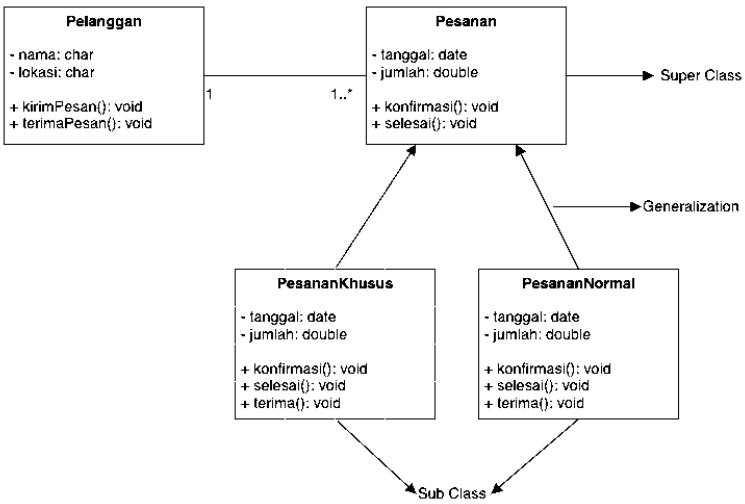
Satu bar hitam dengan dua atau lebih *activity node* yang masuk beserta satu *activity node* yang keluar, tercatat pada akhir dari proses secara bersamaan. Semua *actions* yang menuju *join* harus lengkap sebelum proses dapat berlanjut.

g. *Final state*

Bentuk lingkaran berisi penuh yang berada di dalam lingkaran kosong, menunjukkan akhir dari suatu proses.

4. *Class Diagram*

Class diagram adalah diagram statis. Ini mewakili pandangan statis dari suatu aplikasi. Class diagram tidak hanya digunakan untuk memvisualisasikan, menggambarkan,o dan mendokumentasikan berbagai aspek sistem tetapi juga membangun kode eksekusi dari aplikasi perangkat lunak. *Class diagram* menggambarkan *atribut*, *operation* dan juga *constraint* yang terjadi pada sistem. *Class diagram* banyak digunakan dalam pemodelan sistem OO karena mereka adalah satu-satunya diagram UML, yang dapat dipetakan langsung dengan bahasa berorientasi objek. Class diagram menunjukan koleksi Class, antarmuka, asosiasi, kolaborasi, dan *constraint*. Dikenal juga sebagai diagram structural [12]. Gambar 2.9 menunjukan contoh *class diagram* sederhana.



Gambar 2.16 Contoh class diagram sederhana [12]

Class diagram mempunyai 3 relasi dalam penggunaanya, yaitu :

a. *Assosiation*

Assosiation adalah sebuah hubungan yang menunjukkan adanya interaksi antar *class*. Hubungan ini dapat ditunjukkan dengan garis dengan mata panah terbuka di ujungnya yang mengindikasikan adanya aliran pesan dalam satu arah.

b. *Generalization*

Generalization adalah sebuah hubungan antar *class* yang bersifat dari khusus ke umum

c. *Constraint*

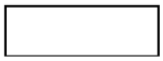
Constraint adalah sebuah hubungan yang digunakan dalam sistem untuk memberi batasan pada sistem sehingga didapat aspek yang tidak fungsional.

2.7 *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram (ERD) merupakan teknik yang digunakan untuk memodelkan kebutuhan data dari suatu organisasi, biasanya oleh *System Analys* dalam tahap analisis persyaratan proyek pengembangan sistem. Sementara seolah-olah teknik diagram atau alat peraga memberikan dasar untuk desain database relasional yang mendasari sistem informasi yang dikembangkan. ERD

bersama-sama dengan detail pendukung merupakan model data yang pada gilirannya digunakan sebagai spesifikasi untuk database [13]. Tabel 2.1 menunjukan beberapa symbol dalam ERD.

Tabel 2.2 Entity Relationship Diagram (ERD) [13]

No	Simbol	Keterangan
1		Entitas
2		Atribut
3		Hubungan
4		Garis

Penjelasan dari tabel adalah sebagai berikut [12] :

1. Entitas

   Objek dalam dunia nyata yang dapat dibedakan dengan objek lain. Entitas terdiri atas beberapa atribut mengidentifikasikan atau membedakan yang satu dengan yang lainnya. Pada setiap entitas baru harus memiliki 1 atribut unik atau yang disebut dengan *primary key*.

2. Atribut

   Isi dari atribut mempunyai elemen yang dapat mengidentifikasikan isi elemen satu dengan yang lain. Ada dua jenis atribut, yaitu:

   a. *Identifier* (*key*) digunakan untuk menentukan suatu *entity* secara unik (*primery key*).
   b. *Descriptor* (*nonkey attribute*) digunakan untuk menspesifikasi karakteristik dari suatu *entity* yang tidak unik.

3. Kardinalitas

   Menyatakan jumlah himpunan relasi antar entitas. Pemetaan kardinalitas terdiri dari:

   a. *One-to-one*, sebuah entitas pada A berhubungan dengan entitas B paling banyak.

b. *One-to-many*, sebuah entitas pada A berhubungan dengan entitas B lebih dari satu.
c. *Many-to-many*, sebuah entitas pada A berhubungan dengan entitas B lebih dari satu dan entitas B berhubungan dengan entitas A lebih dari satu juga.

2.8 *Website*

Website merupakan fasilitas internet yang mengubungkan dokumen dalam lingkup local maupun jarak jauh. Dokumen pada *website* disebut dengan *web page* dan *link* dalam *website* memungkinkan pengguna bisa berpindah dari satu *page* ke *page* lain (*hyper text*), baik diantara *page* yang disimpan dalam *server* yang sama maupun *sever* diseluruh dunia. *Pages* diakses dan dibaca melalui *browser* seperti Netscape Navigator, Internet Explorer, Mozila Firefox, Google Chrome dan aplikasi *browser* lainnya [17].

Berdasarkan sifatnya, suatu *website* dibagi menjadi dua, yakni :

1. *Website Statis*

   Website statis adalah *web* yang halamanya tidak berubah, biasanya untuk melakukan perubahan dilakukan secara manual dengan mengubah kode. *Website statis* informasinya merupakan infromasi satu arah, yakni hanya berasal dari pemilik *software*-nya saja, hanya bisa diupdate oleh pemiliknya saja. Contoh *website statis* ini, yaitu profil perusahaan.

2. *Website Dinamis*

   Website dinamis merupakan *web* yang halaman selalu *update*, biasanya terdapat halaman *backend* (halaman *administrator*) yang digunakan untuk menambah atau mengubah konten. *Web dinamis* membutuhkan database untuk menyimpan. *Website dinamis* mempunyai arus informasi dua arah, yakni berasal dari pengguna dan pemilik, sehingga peng-*update*-nya dapat dilakukan oleh pengguna dan juga pemilik website [17].

2.9 *Database*

Database adalah salah satu koleksi terorganisasi dari data terstruktur, yang disimpan dengan duplikasi item data yang minimum guna memberikan pool

(kelompok) data yang konsisten dan terkontrol. Data ini umum bagi semua sistem, namun independen terhadap program yang menggunakan data itu [14].

Database disimpan di dalam tabel, dan tabel mengandung data yang berhubungan, atau *entity*, seperti misalnya orang, produk, pesanan, dan sebagainya. Tujuannya adalah menjaga table tetap kecil dan dapat dikelola, serta *entity-entity* yang terpisah disimpan dalam tabel-tabel tersendiri. Tentu saja *entity* tidak dapat independen satu sama lain. Di dalam sebuah *database*, setiap tabel memiliki sebuah *field* yang memiliki nilai unik untuk setiap baris [15]. Dalam pengembangan sistem pengarsipan surat pada skripsi ini penulis menggunakan aplikasi *databases MySQL*, adapun beberapa penjelasan tentang *databases MySQL* sebagai berikut :

2.9.1 *MySQL*

MySQL adalah salah satu aplikasi sistem manajemen databases relasional yang handal dalam mengelolah *databases* yang sederhana maupun komplek. *MySQL* mempunyai dua macam lisensi yang dikeluarkan oleh *MySQL AB*, suatu perusahaan Swedia, lisensi tersebut yaitu :

1. Open Source software : *MySQL* tersedia via GNU GPL (*General Public License*) untuk yang gratis.
2. Commercial License : tersedia bagi siapa saja yang menyukai GPL, jika ingin mengembangkan dan menggunakan *MySQL* sebagai bagian dari *software* produk baru maka pengembang harus membeli *license commercial* ini.

2.9.2 Keunggulan *MySQL*

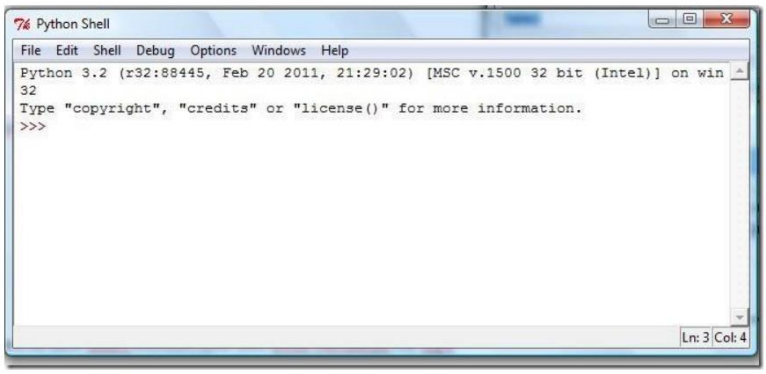
Dibawah ini beberapa keunggulan dari *databases MySQL* [16] :

1. Cepat : tujuan utama dari pengembangan *MySQL* adalah kecepatan dalam mengakses dan mengolah *databases*.
2. Tidak mahal : dibawah *Open Source software license* maka siapapun dapat menggunakan aplikasi *MySQL* secara gratis.
3. Mudah digunakan : kita dapat membangun dan berinteraksi dengan databases MySQL cukup dengan pernyataan sederhana didalam bahasa SQL.

4. Dapat berjalan pada beberapa system operasi : seperti Windows, Linux, Mac OS, Unix (solaris, AIX, DEC unix) FreeBSD, OS/2, Irix, dan lainnya.
5. Aman : MySQL adalah sistem oktorisasi fleksibel yang mengijinkan beberapa atau semua privilege databases untuk pengguna khusus atau kelompok pengguna.

2.10 *Python*

Python adalah bahasa pemrograman interpretatif multiguna. Tidak seperti bahasa lain yang susah untuk dibaca dan dipahami, python lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sintak. Hal ini membuat Python sangat mudah dipelajari baik untuk pemula maupun untuk yang sudah menguasai bahasa pemrograman lain.



Gambar 2.17 Tampilan Utama Python

Bahasa ini muncul pertama kali pada tahun 1991, dirancang oleh seorang bernama Guido van Rossum. Sampai saat ini Python masih dikembangkan oleh Python Software Foundation.Bahasa Python mendukung hampir semua sistem operasi, bahkan untuk sistem operasi Linux, hampir semua distronya sudah menyertakan Python di dalamnya.

Dengan kode yang sederhana dan mudah diimplementasikan, seorang programmer dapat lebih mengutamakan pengembangan aplikasi yang dibuat, bukan malah sibuk mencari syntax error. Saat ini kode python dapat dijalankan di berbagai platform sistem operasi, beberapa di antaranya adalah [18].

a. Linux/Unix

b. Windows
c. Mac OS X
d. Java Virtual Machine
e. OS/2
f. Amiga
g. Palm
h. Symbian (untuk produk-produk Nokia)

2.11 *Hypertext Preprocessor* (PHP)

PHP menurut Anhar [19] adalah bahasa pemrograman web server-side yang bersifat open source, PHP juga merupakan script yeng terintegrasi dengan HTML dan berada pada server (server side HTML embedded script). PHP juga merupakan script yang digunakan untuk membuat halaman website yang sangat dinamis, dinamis berarti halaman tampilan yang akan ditampilkan dibuat saat halaman itu diminta oleh client. PHP pertama kali dibuat oleh Rasmus Lerdorf seorang pemrogram C yang handal dari greenland Denmrak di tahun 1995, PHP diberi nama FI (Form Interpreted) yang digunakan untuk mengelola from dari web. Pada perkembangannya, kode-kode yang digunkan dirilis untuk umum sehingga mulai banyak dikembangkan oleh programer diseluruh dunia. Tahun 1997 PHP dirilis dengan versi 2.0, pada versi ini sudah terintegrasi dengan bahasa pemrograman C dan sudah dilengkapi dengan modul sehingga kualitas kerja PHP lebih meningkat secara signifikan. Ditahun yang sama sebuah perusahaan programbernama Zend merilis ulang PHP versi ini dengan lebih baik, bersih dan cepat. Seiring berkembangnya jaman ditahun 1994 PHP versi 4.0 mulai dirilis dan versi ini paling banyak digunakan pada awal abad 21 karena PHP versi ini sudah mampu membangun web komplek dengan stabilitas kecepatan yang tinggi. Ditahun 2004 perusahaan program Zend merilis PHP lagi dengan versi terbarunya 5.0 yang inti dari interpreter PHPH mengalami perubahan besar. Versi ini juga memasukkan model pemrograman berorientasi objek kedalam PHP untuk menjawab perkembanngan bahasa pemrograman kearah paradigma berorientasi objek.

Bahasa program PHP sering digunakan karena PHP adalah bahasa open source yang memiliki kesederhanaan dan memiliki beberapa fitur built-in yang

berfungsi untuk menangani kebutuhan standart dalam pembuatan aplikasi web. PHP juga merupakan bahasa script yang paling mudah dipahami karena memiliki beberapa referensi. PHP juga dapat digunakan untuk berbagai sistem operasi anatara laina : Unix, Macintosh serta windows. PHP dapat dijalankan secara runtime melalui console serta dapat menjalakan perintah-perintah system. Open source disini memiliki arti code-code PHP terbuka untuk umum dan tidak berbayar atas pembelian dari license. Web server yang mendukung PHP dapat ditemukan dimana-mana, mulai dari Apache, IIS, Lighttpd hingga Xitami dengan configurasi yang relatif mudah. Selain itu PHP juga dilengkapi dengan berbagai macam pendukung lain seperti support langsung keberbagai macam databasea yang populer seperti Oracle, MySQL dan lain-lain.

2.12 *Java Script Object Notation* (JSON)

Java Script Object Notation (JSON) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman JavaScript, Standar ECMA-262 Edisi ke-3 – Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh *programmer* keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll [20]. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data.

Penulis menggunakan metode JSON dalam pengiriman data yang dilakukan, karena JSON memiliki beberapa kelebihan - kelebihan dibandingkan XML, kelebihan – kelebihan tersebut adalah [20] :

1. Format Penulisan

   Untuk merepresentasikan sebuah struktur data yang rumit dan berbentuk hirarkis penulisan JSON relatif lebih terstruktur dan mudah.

2. Ukuran

   Ukuran karakter yang dibutuhkan JSON lebih kecil dibandingkan XML untuk data yang sama. Hal ini tentu berpengaruh pula pada kecepatan pertukaran data, walaupun tidak signifikan untuk data yang kecil, namun cukup berarti jika koneksi yang digunakan relatif lambat untuk mengakses

aplikasi web kaya fitur yang memanfaatkan pertukaran data. Di sini JSON lebih unggul dibandingkan XML, kecuali jika data dikompresi terlebih dahulu sebelum dikirimkan, perbedaan JSON dan XML yang telah dikompresi tidaklah signifikan.

3. Browser

   Parsing Proses parsing merupakan proses pengenalan token atau bagian-bagian kecil dalam rangkaian dokumen XML/JSON. Contohnya, terdapat data text dalam format JSON. Data tersebut harus di-parsing terlebih dahulu sebelum dapat diakses dan dimanipulasi. Browser parsing berarti proses parsing yang terjadi pada sisi client/browser.

Melakukan browser parsing pada JSON lebih sederhana dibandingkan pada XML, JSON menggunakan function JavaScript eval() untuk melakukan parsing. Sementara dokumen XML di-parsing oleh XMLHttpRequest. Rata-rata survei menobatkan JSON sebagai pemenang jika diadu kecepatan parsingnya.

JSON terbuat dari dua struktur [20] :

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakansebagai objek (object), rekaman (record), struktur (struct), kamus (dictionary), tabel hash (hash table), daftar berkunci (keyed list), atau associative array.
2. Daftar nilai terurutkan (an ordered list of values). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (array), vektor (vector), daftar (list), atau urutan (sequence).

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini.

2.13 Metode Pengujian

Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, desain, dan

pengkodean. Sejumlah aturan yang berfungsi sebagai sasaran pengujian pada perangkat lunak adalah [25] :

1. Pengujian adalah proses eksekusi suatu program dengan maksud menemukan kesalahan.
2. Test case yang baik adalah test case yang memiliki probabilitas tinggi untuk menemukan kesalahan yang belum pernah ditemukan sebelumnya.
3. Pengujian yang sukses adalah pengujian yang mengungkap semua kesalahan yang belum pernah ditemukan sebelumnya.

Karakteristik umum dari pengujian perangkat lunak adalah sebagai berikut [25] :

1. Pengujian dimulai pada level modul dan bekerja keluar kearah integrasi pada sistem berbasiskan komputer.
2. Teknik pengujian yang berbeda sesuai dengan poin-poin yang berbeda pada waktunya.
3. Pengujian diadakan oleh software developer dan untuk proyek yang besar oleh group testing yang independent.
4. Testing dan Debugging adalah aktivitas yang berbeda tetapi debugging harus diakomodasikan pada setiap strategi testing.

Metode pengujian perangkat lunak ada 3 jenis, yaitu [25] :

1. White Box/Glass Box - pengujian operasi.
2. *Black Box* - untuk menguji sistem.
3. Use case - untuk membuat input dalam perancangan *black box* dan pengujian statebased

2.13.1 *White Box Testing*

Pengujian *white box* adalah pengujian yang meramalkan cara kerja perangkat lunak secara rinci, karenanya logikal *path* (jalur logika) perangkat lunak akan di-test dengan menyediakan test case yang akan mengerjakan kumpulan kondisi atau pengulangan secara spesifik. Secara sekilas dapat diambil kesimpulan *white box* testing merupakan petunjuk untuk mendapatkan program yang benar secara 100%.

Menurut Pressman [10], pengujian *White box* atau *Glass box* adalah metode *test case* desain yang menggunakan struktur kontrol desain *procedural* untuk memperoleh *test-case*. Dengan menggunakan metode pengujian *white box*, analis sistem akan dapat memperoleh *test case* yang:

a. Memberikankan jaminan bahwa semua jalur *independent* pada suatu modul telah digunakan paling tidak satu kali.
b. Menggunakan semua keputusan logis dari sisi *true* dan *false.*
c. Mengeksekusi semua batas fungsi *loops* dan batas operasionalnya.
d. Menggunakan struktur *internal* untuk menjamin validitasnya.

Ujicoba basis *path* adalah teknik uji coba *white box* yang diusulkan Tom Mc Cabe. Metode ini memungkinkan perancang *test case* mendapatkan ukuran kekompleksan logical dari perancangan prosedural dan menggunkan ukuran ini sebagai petunjuk untuk mendefinisikan basis set dari jalur pengerjaan. *Test case* yang didapat digunakan untuk mengerjakan basis set yang menjamin pengerjaan setiap perintah minimal satu kali selama uji coba.

Terdapat beberapa proses yang harus di lakukan dalam uji coba basis *path* yaitu diantaranya :

1. Notasi Diagram Alir

   Sebelum metode basis *path* diperkenalkan, terlebih dahulu akan dijelaskan mengenai notasi sederhana dalam bentuk diagram alir (grafik alir). Diagram alir menggambarkan aliran kontrol logika yang menggunakan notasi.

2. Kompleksitas Siklomatis

   Kompleksitas siklomatis adalah metriks perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu program. Bila metriks ini digunakan dalam konteks metode pengujian basis *path*, maka nilai yang terhitung untuk kompleksitas siklomatis menentukan jumlah jalur independen dalam basis set suatu pemrograman memberi batas atas bagi jumlah pengujian yang harus dilakukan untuk memastikan bahwa semua statemen telah dieksekusi sedikitnya satu kali.

Jalur independen adalah jalur yang memlalui progarm yang mengintroduksi sedikitnya satu rangkaian statemen proses baru atau suatu kondisi baru. Bila dinyatakan dengan terminologi grafik alir, jalur independen harus bergerak sepanjang paling tidak satu *edge* yang tidak dilewatkan sebelum jalur tersebut ditentukan.

3. Melakukan Test Case

   Metode uji coba basis *path* juga dapat diterapkan pada perancangan prosedural rinci atau program sumber. Pada bagian ini akan dijelaskan langkah-langkah uji coba basis *path*.

4. Matriks Grafis

   Prosedur untuk mendaptakan grafik alir dan menentukan serangkaian basis *path*, cocok dengan mekanisasi. Untuk mengembangkan peranti perangkat lunak yang membantu pengujian basis *path*, struktur data yang disebut matriks grafis dapat sangat berguna.

   Matriks garfis adalah matriks bujur sangkar yang ukurannya sama dengan jumlah simpul pada grafik alir. Masing-masing baris dan kolom sesuai dengan simpul yang diidentifikasikan dan *entry* matriks sesuai dengan *edge* diantara simpul.

2.13.2 *Black Box Testing*

Pengujian menggunakan sekumpulan aktifitas validasi, dengan pendekatan *black box* testing. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian kotak hitam 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* testing harus dibuat dengan kasus benar dan kasus salah.

Menurut Pressman [10], *black box* testing juga disebut pengujian tingkah laku, memusat pada kebutuhan fungsional perangkat lunak. Teknik pengujian *black box* memungkinkan memperoleh serangkaian kondisi masukan yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Beberapa jenis kesalahan yang dapat diidentifikasi adalah fungsi tidak benar atau hilang, kesalahan

antar muka, kesalahan pada struktur data (pengaksesan basis data), kesalahan performasi, kesalahan inisialisasi dan akhir program.

Equivalence Partitioning merupakan metode *black box testing* yang membagi domain masukan dari program kedalam kelas-kelas sehingga test case dapat diperoleh. Equivalence Partitioning berusaha untuk mendefinisikan kasus uji yang menemukan sejumlah jenis kesalahan, dan mengurangi jumlah kasus uji yang harus dibuat. Kasus uji yang didesain untuk Equivalence Partitioning berdasarkan pada evaluasi dari kelas ekuivalensi untuk kondisi masukan yang menggambarkan kumpulan keadaan yang valid atau tidak. Kondisi masukan dapat berupa spesifikasi nilai numerik, kisaran nilai, kumpulan nilai yang berhubungan atau kondisi boolean.

Kesetaraan kelas dapat didefinisikan menurut panduan berikut [10] :

1. Jika masukan kondisi menentukan kisaran, satu sah dan dua diartikan tidak valid kesetaraan kelas.
2. Jika masukan membutuhkan nilai, kondisi tertentu satu sah dan dua tidak valid kesetaraan kelas diartikan.
3. Jika masukan kondisi menentukan anggota dari set, satu sah dan satu tidak valid kesetaraan kelas diartikan.
4. Jika kondisi yang input, boolean satu sah dan satu tidak valid kelas diartikan.

Menerapkan pedoman untuk derivasi kelas kesetaraan, uji kasus untuk setiap masukan domain item data dapat dikembangkan dan dilaksanakan. Uji kasus dipilih sehingga jumlah terbesar dari atribut dari kelas kesetaraan tersebut dilakukan sekaligus.

Beberapa kata kunci dalam pengujian perangkat lunak yang dapat diperhatikan, yaitu [26] :

1. Dinamis

   Pengujian perangkat lunak dilakukan pada masukan yang bervariasi. Masukan ini ditentukan sebelum pengujian dilakukan dengan batasan yang disesuaikan dengan kemampuan perangkat lunak. Masukan tidak harus sesuatu yang dimungkinkan terjadi pada penggunaan program lebih lanjut, melainkan meliputi keseluruhan batasan yang dapat dijangkau perangkat

lunak dan dilakukan pemercontohan (sampling) secara acak untuk proses pengujian.

2. Terbatas

   Meskipun pengujian dilakukan pada perangkat lunak sederhana sehingga rumit sekalipun, pengujian dilakukan dengan memenuhi batasan-batasan tertentu sesuai dengan kemampuan program. Batasan ini juga diberlakukan pada masukan-masukan yang dipilih untuk pengujian. Tidak semua kemungkinan masukan diujika pada perangkat lunak karena akan memakan waktu yang cukup panjang mengingat begitu banyaknya kemungkinan yang bisa terjadi. Untuk mengatasi hal ini, pemilihan masukan-masukan pada proses pengujian secara acak yang diperkirakan mampu memenuhi kebutuhan pengujian perangkat lunak akan dilakukan.

3. Tertentu

   Pengujian dilakukan dengan batasan tertentu disesuaikan dengan harapan pada fungsi, respon, dan karakteristik perangkat lunak tersebut. Batasan tersebut akan disesuaikan dengan teknik-teknik pengujian yang ada. Pemilihan kriteria pengujian yang paling tepat merupakan hal yang kompleks. Dalam praktiknya, analisis risiko pengujian dan pengalaman terhadaop pengujian-pengujian sejenis akan diperlukan.

4. Harapan

   Kata kunci ini memiliki keadaan-keadaan yang diharapkan, baik berupa respon sistem terhadap masukan maupun karakteristik responnya. Dalam hal ini, batasan-batasan hasil pengujian yang diharapkan harus ditentukan. Dengan demikian, dapat diketahui apakah perangkat lunak tersebut telah memenuhi hasil pengujian yang diharapkan atau memerlukan pembenahan kembali, baik berupa perbaikan maupun pengembangan perangkat lunak.