**BAB II**

**LANDASAN TEORI**

* 1. **Aplikasi**

Aplikasi menurut (Dhanta dikutip dari Sanjaya, 2015) adalah *software* yang dibuat oleh suatu perusahaan komputer untuk mengerjakan tugas-tugas tertentu, misalnya *Microsoft Word*, *Microsoft Excel*. Aplikasi berasal dari kata *application* yang artinya penerapan lamaran penggunaan. Menurut (Jogiyanto dikutip oleh Ramzi, 2013) aplikasi merupakan penerapan, menyimpan sesuatu hal, data, permasalahan, pekerjaan kedalam suatu sarana atau media yang dapat digunakan untuk menerapkan atau mengmplementasikan hal atau permasalahan yang ada sehingga berubah menjadi suatu bentuk yang baru tanpa menghilangkan nilai-nilai dasar dari hal data, permasalahan, dan pekerjaan itu sendiri.

* 1. ***Face Recognition***

Identifikasi (pengenalan) wajah atau *face recognition* adalah sebuah tugas yang dikerjakan oleh manusia secara rutin dan mudah dalam kehidupan seharihari. Penelitian dan pengembangan ilmu pengenalan wajah berkembang secara otomatis atas dasar ketersediaan desktop kuat dan rendah biaya serta *embedded system* yang telah menciptakan minat yang sangat besar dalam pengolahan citra digital dan video. Motivasi penelitian dan pengembangan dari pengenalan wajah termasuk dalam lingkup otentikasi *biometric*, pengawasan, interaksi manusia komputer, dan manajemen multimedia (Li dan Jain, 2005:1).

Sistem *face recognition* pada umumnya mencakup empat modul utama menurut (Li dan Jain, 2005:2), yaitu: deteksi, *alignment*, ekstraksi fitur dan pencocokan. Proses lokalisasi dan normalisasi (deteksi wajah dan *alignment*) adalah langkah-langkah sebelum proses pengenalan wajah (ekstraksi fitur wajah dan pencocokan) dilakukan.

* 1. ***Internet of Things* (*IoT*)**

*IoT* adalah sebuah konsep yang menggunakan internet untuk menjadi sarana segala aktifitas yang pelakunya saling berinteraksi. *IoT* mengacu pada milaran perangkat yang saling terhubung atau bisa disebut dengan “Objek Cerdas” atau “Smart Things” (Cirani et al.,2015). Dengan adanya *IoT* segala kegiatan dan aktifitas dimudahkan melalui *online* dan lebih efisien (Sulaiman dan Widarma 2017). *IoT* merupakan inti dari industri teknologi informasi generasi baru.

Dampak *IoT* pada evolusi internet menuju lingkungan cerdas generasi berikutnya yang sangat bergantung pada integrasi *IoT* dengan *cloud computing*. Saat *IoT* terhubung dengan *cloud* sejumlah data besar yang telah dikumpulkan dari banyak tempat, dapat diolah dan dianalisis untuk membuat makna informasi ke *end-user* (Barcelo et al., 2016).

Penelitian ini akan memanfaatkan *IoT* sebagai alat simulasi yang terhubung dengan pendeteksi masker pada wajah menggunakan metode *Face recognition*.

* 1. ***NodeMCU***

*NodeMCU* adalah sebuah platform *IoT* yang bersifat *opensource*. Terdiri dari perangkat keras berupa *System OnChip ESP8266*. dari *ESP8266* buatan *Espressif System*, juga firmware yang digunakan, yang menggunakan Bahasa pemrograman scripting Lua. Menurut (Sumardi, 2016) Istilah NodeMCU secara *default* sebenarnya mengacu pada *firmware* yang digunakan dari pada perangkat keras *development kit NodeMCU* bisa dianalogikan sebagai *board* Arduino-nya *ESP8266*.

* 1. ***NodeJS***

*Nodejs* dikembangan dari *engine* javaScript yang dibuat oleh Google untuk *Browser* Chrome / Chronium (V8) ditambah dengan lib UV serta beberapa 6 pustaka internal lainnya. Dengan menggunakan *Nodejs* semua pengembangan akan dilakukan dengan *javascript*, baik pada sisi *client* maupun *server*. Pengembangan aplikasi dengan menggunakan *Nodejs* dapat dilakukan secara moduler yaitu dengan memisahkan berbagai komponen kedalam pustaka (*library*). Pustaka tersebut dapat dikelola dengan npm yang terdapat di *Nodejs*. Pada dasarnya, *Nodejs* sebuah *runtime environment* dan *script* *library*. Sebuah *runtime environment* adalah sebuah *software* yang berfungsi untuk mengeksekusi, menjalankan dan mengimplementasikan funsi-fungsi serta cara kerja inti dari suatu bahasa pemograman. Sedangkan *script library* adalah kumpulan, kompilasi atau bank data berisi skrip/kode-kode pemograman. (Equan Pr, 2013).

* 1. ***Web Service***

*Web service* adalah sistem perangkat lunak yang dirancang untuk mendukung interaksi yang bisa beroperasi *machine to machine* diatas jaringan. *Web service* mempunyai alat penghubung yang diuraikan di dalam format *machine-processable* (secata spesifik WSDL). Sistem lain saling berhubungan dengan *Web service* di dalam cara yang ditentukan oleh deskripsinya menggunakan pesan SOAP REST yang secara khas disampaikan menggunakan HTTP dengan serialisasi XML atau JSON bersama dengan standar lain yang terkait dengan *web* (Booth et al., 2004).

* 1. ***Application Programming Interface* (API)**

API adalah singkatan dari *Application Programming Interface*, dan memungkinkan *developer* untuk mengintegrasikan dua bagian dari aplikasi atau dengan aplikasi yang berbeda secara bersamaan. API terdiri dari berbagai elemen seperti *function*, *protocols*, dan *tools* lainnya yang memungkinkan *developers* untuk membuat aplikasi. Menurut (Abdul Kadir, 2016) Tujuan penggunaan API adalah untuk mempercepat proses *development* dengan menyediakan *function* secara terpisah sehingga *developer* tidak perlu membuat fitur yang serupa. Penerapan API akan sangat terasa jika fitur yang diinginkan sudah sangat kompleks, tentu membutuhkan waktu untuk membuat yang serupa dengannya. Terdapat berbagai jenis sistem API yang dapat digunakan, termasuk *system* operasi, *library*, dan web.

* 1. ***Arduino IDE* (*Intergrated Development Environment*)**

*IDE* merupakan kependekan dari *Integrated Developtment Enviroenment*, atau secara bahasa mudahnya merupakan lingkungan terintegrasi yang digunakan untuk melakukan pengembangan. Menurut (Muhammad Syahwil, 2015) Disebut sebagai lingkungan karena melalui *software* inilah Arduino dilakukan pemrograman untuk melakukan fungsi-fungsi yang dibenamkan melalui sintaks pemrograman. *Arduino* menggunakan bahasa pemrograman sendiri yang menyerupai bahasa C. Bahasa pemrograman *Arduino* (*Sketch*) sudah dilakukan perubahan untuk memudahkan pemula dalam melakukan pemrograman dari bahasa aslinya. Sebelum dijual kepasaran, IC *mikrokontroler* *Arduino* telah ditanamkan suatu program bernama *Bootlader* yang berfungsi sebagai penengahantara *compiler* *Arduino* dengan *mikrokontroler*.

* 1. **UML (*Unified Modeling Language*)**

Menurut (Sri Dharwiyanti, 2003) dalam bukunya “Pengantar *Unified Modeling Language*” menuliskan bahwa UML adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

UML mendefinisikan diagram-diagram sebagai berikut :

1. *Use Case Diagram*

Tabel 2.1 Simbol *Use Case Diagram*

|  |  |  |
| --- | --- | --- |
| **Gambar** | **Nama** | **Keterangan** |
|  | Actor | Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *Use*  *Case.* |
|  | Dependency | Hubungan dimana perubahan yang terjadi pada suatu elemen  mandiri akan mempengaruhi elemen yang bergantung padanya  elemen yang tidak mandiri. |
|  | Generalization | Hubungan dimana objek anak berbagi prilaku dan stuktur data dari objek yang ada diatasnya  objek induk. |
|  | Include | Menspesifikasikan bahwa Use Case sumber secara eksplisit. |
|  | Extend | Menspesifikasikan bahwa Use Case target memperluas perilaku dari Use Case sumber dari suatu  titik yang diberikan. |
|  | Association | Apa yang menghubungkan antara  objek satu dengan objek lainnya. |
|  | System | Menspesifikasikan paket yang menampilkan sistem secara terbatas. |
|  | Use Case | Deskripsi dari urutan aksi – aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang  terstruktur bagi suatu aktor. |
|  | Collaboration | Interaksi aturan – aturan dan elemen lain yang bekerjasama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen  – elemennya. |
|  | Note | Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya  komputasi. |

1. *Class Diagram*

Tabel 2.2 Simbol *Class Diagram*

|  |  |  |
| --- | --- | --- |
| **Gambar** | **Nama** | **Keterangan** |
|  | Association | Realisasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity. |
|  | Direct Association | Relasi antar kelas dengan makna kelas yang satu digunakan oleh  kelas yang lain. |
|  | Generalization | Relasi antar kelas dengan makna  generalisasi – spesialisasi. |
|  | Dependency | Relasi antar kelas dengan makna  kebergantungan antar kelas. |
|  | Aggregation | Relasi antar kelas dengan makna  semua – bagian. |

1. *Activity Diagram*

Tabel 2.3 Simbol *Activity Diagram*

|  |  |  |
| --- | --- | --- |
| **Gambar** | **Nama** | **Keterangan** |
|  | Start | Status awal aktivitas sistem, subuah *Activity Diagram* mempunyai sebuah status awal. |
|  | Activity | Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan  kata kerja. |
|  | Decision | Asosiasi percabangan dimana jika  ada pilihan aktivitas lebih dari satu. |
|  | Join | Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu. |
|  | End | Status akhir yang dilakukan sistem,  sebuah Activity Diagrammemiliki sebuah status akhir. |
|  | Swimlane | Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi. |
|  | Fork | Digunakan untuk menunjukan  Kegiatan yang dilakukan secara paralel. |
|  | Join | Digunakan untuk menunjukan  kegiatan yang digabungkan. |

1. *Sequence Diagram*

Tabel 2.4 Simbol *Sequence Diagram*

|  |  |  |
| --- | --- | --- |
| **Gambar** | **Nama** | **Keterangan** |
|  | Actor | Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi dan mendapat manfaat dari sistem. |
|  | Object | Berpartisipasi secara berurutan dengan mengirimkan dan atau menerima pesan. |
|  | Life Line | Menandakan kehidupan objek selama urutan, diakhiri tanda x pada titik dimana kelas tidak lagi berinteraksi |
|  | Destroy | Menyatakan suatu objek mengakhiri  hidup objek yang lain, arah panah mengarah pada objek yang diakhiri. |