# BAB II
# LANDASAN TEORI

## 2.1 Landasan Teori

Teori-teori yang mendasar sebagai penunjang proses penelitian yaitu sebagai berikut :

### 2.1.1 Aplikasi

Aplikasi berasal dari kata "Application" yang merujuk pada penerapan, lamaran, dan penggunaan. Secara umum, aplikasi mengacu pada program siap pakai yang diciptakan untuk menjalankan fungsi tertentu bagi pengguna dan dapat digunakan untuk mencapai tujuan yang ditentukan. Menurut Kamus Besar Bahasa Indonesia (KBBI), aplikasi didefinisikan sebagai penerapan dari rancangan sistem yang mengolah data dengan menggunakan aturan atau ketentuan bahasa pemrograman tertentu. Dalam konteks komputer, aplikasi merupakan program yang dirancang untuk melakukan tugas-tugas khusus sesuai dengan kebutuhan pengguna. Aplikasi ini sering kali mencakup antarmuka pengguna yang mempermudah interaksi pengguna dengan sistem dan memungkinkan manipulasi data serta eksekusi proses secara efektif dan efisien [1].

### 2.1.2 Tiket

Tiket adalah kartu kertas atau slip yang sering juga disebut karcis, yang berfungsi sebagai tanda pembayaran atau biaya lainnya. Secara umum, tiket juga dapat didefinisikan sebagai tanda yang menunjukkan bahwa seseorang telah membayar untuk masuk ke sebuah tempat seperti teater, bioskop, taman hiburan, kebun binatang, museum, konser, atau transaksi lainnya. Tiket juga berfungsi sebagai bukti pembayaran dan memberikan hak akses bagi pengunjung untuk memasuki tempat atau menggunakan layanan yang terkait. Dengan adanya tiket, penyelenggara acara atau tempat wisata dapat mengatur dan mengontrol jumlah pengunjung serta memastikan bahwa semua pengunjung telah membayar dengan benar sebelum memasuki area atau menikmati layanan yang disediakan. Selain itu, tiket juga bisa memiliki berbagai bentuk dan desain yang unik, kadang-kadang dilengkapi dengan informasi tambahan seperti waktu acara, nomor kursi, atau instruksi khusus lainnya, yang membuatnya lebih dari sekadar alat pembayaran, tetapi juga sebagai bagian dari pengalaman pengguna dalam menghadiri suatu acara atau menikmati suatu tempat

wisata [2].

### 2.1.3 *E-Ticketing*

*E-Ticketing*, atau tiket elektronik, adalah metode yang digunakan untuk mencatat proses penjualan dalam perjalanan pelanggan tanpa memerlukan dokumen fisik seperti kertas tiket. Dokumen digital yang diperlukan untuk transaksi dalam layanan transportasi publik seperti bus rapid transit, feri, trem, dan lainnya dapat diperoleh melalui *electronic ticket*. Informasi sistem pada E-Ticketing disimpan dalam format digital, yang dibuktikan dengan pemberian barcode atau *QR code* kepada penumpang sebagai akses ke stasiun, pelabuhan, dan lainnya. *E-Ticketing* memberikan keuntungan dalam hal efisiensi, kemudahan akses, dan pengurangan penggunaan kertas secara signifikan, mendukung upaya untuk menjaga lingkungan dengan mengurangi jejak karbon. Dengan adopsi teknologi ini, penggunaan tiket fisik yang rawan hilang atau rusak dapat diminimalisir, sementara pengalaman perjalanan pelanggan dapat ditingkatkan melalui integrasi teknologi yang lebih canggih dan aman. [3].

### *2.1.4 Website*

*Website* adalah kumpulan halaman yang memuat informasi dalam bentuk digital. Data ini dapat berupa teks, gambar, suara, video, animasi atau kombinasi dari semuanya. Sebuah situs web mampu dianggap berhasil jika mudah dipakai dan dipahami oleh pengguna. Selain itu, desain yang responsif dan antarmuka yang intuitif dapat meningkatkan pengalaman pengguna secara keseluruhan. Konten yang relevan dan mudah dinavigasi juga menjadi faktor penting dalam menjaga minat pengunjung terhadap situs web. Selain memuat informasi, situs web juga dapat berfungsi sebagai platform interaktif yang memungkinkan pengguna untuk berinteraksi, memberikan umpan balik, atau bahkan bertransaksi secara *online.* [4].

### 2.1.5 Pemrograman Berorientasi Objek

*Object-Oriented Programming* merupakan metode pemrograman yang berorientasi kepada objek. Dalam OOP, *programmer* yang sama atau berbeda dapat mengembangkan sistem yang ada dimana class dan objek dapat digunakan berulang-ulang atau menambahkan objek baru pada sistem yang ada. Jika terjadi permasalahan, sistem lebih mudah diatasi karena struktur hierarkis dan modularitas

yang dimiliki oleh OOP. Untuk pengembangan sistem yang kompleks, bahasa pemrograman yang mendukung *Object Oriented Programming* sangat efektif digunakan, karena memungkinkan pengorganisasian kode yang lebih baik dan pemeliharaan yang lebih mudah di masa mendatang [5].

Berikut konsep dasar dalam mendalami Pemprograman Berbasis Objek :

a. Objek

Objek dalam paradigma pemrograman berbasis objek (OOP) adalah representasi konkret dari sebuah kelas. Setiap objek memiliki atribut yang mendefinisikan karakteristiknya dan metode yang mendefinisikan perilaku atau tindakannya. Objek secara langsung mewakili instansi dari konsep atau entitas tertentu yang didefinisikan oleh kelasnya. Dalam OOP, objek dapat dibuat berdasarkan blueprint atau cetak biru yang disebut kelas, yang menentukan struktur dan perilaku objek tersebut. Sebagai contoh, sebuah kelas "Mobil" dapat memiliki objek konkret berupa mobil dengan warna, merek, dan fungsi-fungsi seperti "Maju" atau "Mundur". Objek memungkinkan untuk memodelkan dan mengelola berbagai aspek dunia nyata secara sistematis dan modular dalam pengembangan perangkat lunak [6].

b. Kelas

Kelas dalam pemrograman berbasis objek (OOP) adalah struktur atau cetak biru yang digunakan untuk membuat objek-objek dengan karakteristik tertentu. Setiap kelas mendefinisikan atribut-atribut dan metode-metode yang akan dimiliki oleh objek-objek yang dibuat berdasarkan kelas tersebut. Dalam konteks OOP, kelas merupakan abstraksi dari objek yang mencerminkan konsep atau entitas tertentu. Misalnya, kelas "Mobil" dapat memiliki atribut seperti warna, merek, dan metode seperti "Maju" atau "Mundur" yang menggambarkan perilaku mobil tersebut. Dengan menggunakan kelas, kita dapat membuat banyak objek yang serupa dengan struktur dan perilaku yang sudah ditentukan secara konsisten. Pada dasarnya, kelas membantu dalam mengorganisir dan mengelompokkan kode yang terkait untuk membangun sistem yang lebih terstruktur dan mudah dipahami [6].

c. *Encapsulation* (Pembungkusan)

*Encapsulation* atau pembungkusan dalam pemrograman berbasis objek (OOP) adalah konsep yang mengatur cara atribut (variabel) dan metode (fungsi) dalam suatu

kelas diorganisir dan diakses. Dengan encapsulation, atribut-atribut kelas biasanya dideklarasikan sebagai private atau protected untuk mengontrol akses langsung dari luar kelas tersebut. Metode-metode publik (public methods) yang disediakan oleh kelas digunakan untuk mengakses dan memanipulasi atribut-atribut tersebut secara aman. Konsep ini mirip dengan meletakkan semua benda terkait dalam sebuah kotak agar lebih mudah diatur dan dilindungi dari akses yang tidak sah atau kesalahan penggunaan. Melalui *encapsulation*, OOP mempromosikan prinsip pengkapsulan data (data hiding) yang membantu mencegah modifikasi langsung terhadap data internal sebuah objek tanpa menggunakan metode-metode yang disediakan. Dengan menggunakan teknik ini, kode program menjadi lebih modular dan terstruktur, memfasilitasi pengembangan dan pemeliharaan sistem yang lebih mudah dan efisien [6].

d. *Polymorphism* (Perbedaan Bentuk)

*Polymorphism* dalam pemrograman berbasis objek (OOP) mengacu pada kemampuan suatu objek untuk memiliki banyak bentuk atau perilaku yang berbeda tergantung pada konteksnya. Konsep ini memungkinkan objek dari kelas yang berbeda untuk merespons atau menanggapi metode dengan cara yang berbeda, tergantung pada implementasi masing-masing kelas. Misalnya, dalam OOP, terdapat konsep *overloading* dan *overriding* yang memungkinkan kelas yang sama memiliki beberapa metode dengan nama yang sama namun dengan parameter yang berbeda atau perilaku yang berbeda. *Polymorphism* juga memungkinkan untuk menggunakan pola desain seperti *polimorfisme subtype*, di mana objek dari kelas turunan dapat digunakan secara polimorfik melalui referensi kelas dasarnya. Dalam praktiknya, ini mirip dengan situasi di mana kita dapat menggunakan berbagai jenis kendaraan yang berbeda (mobil, sepeda, atau pesawat) untuk melakukan perjalanan, meskipun cara mereka beroperasi mungkin berbeda. Dengan menggunakan polimorfisme, OOP memungkinkan pengembang untuk membuat kode yang lebih fleksibel, reusable, dan mudah dimodifikasi, sesuai dengan prinsip-prinsip desain yang baik dalam pengembangan perangkat lunak [6].

e. Abstraksi

Abstraksi dalam pemrograman berbasis objek (OOP) merujuk pada proses menyembunyikan detail tertentu dari sebuah objek atau fungsi dan hanya

menunjukkan informasi yang penting bagi pengguna. Dalam konteks OOP, abstraksi memungkinkan pengembang untuk fokus pada fitur penting dari suatu objek tanpa harus memperhatikan detail implementasinya. Misalnya, ketika mendefinisikan kelas "Mobil", kita dapat mengabstraksi fitur-fitur penting seperti penggerakannya (maju, mundur) dan atribut utama seperti warna atau mereknya. Abstraksi membantu dalam menggambarkan konsep-konsep dunia nyata ke dalam kode program, sehingga memudahkan pemahaman dan pengelolaan program secara keseluruhan. Dengan menggunakan abstraksi, kita dapat membuat model yang lebih sederhana dan efisien, serta meningkatkan tingkat modularitas dan reusable dalam pengembangan perangkat lunak. Selain itu, abstraksi juga mendukung konsep *polimorfisme* dan *inheritance* dalam OOP, yang mengarah pada penciptaan hierarki kelas yang lebih terstruktur dan mudah diatur [6].

### 2.1.6 *Database*

Database adalah suatu susunan atau kumpulan catatan data yang tersimpan di dalam komputer. Hubungan antar entri dalam database dapat digunakan sebagai sumber informasi bagi pengguna. Sampai saat ini, masih banyak record database yang ditampilkan dalam bentuk teks sebagai informasi kepada pengguna. Ini adalah salah satu kerentanan yang dimiliki analis kriptografi dalam mengakses, memanipulasi atau membocorkan dan mendistribusikan catatan basis data. Database juga disebut kumpulan data dan deskripsi yang terhubung secara logis yang digunakan bersama dan dirancang untuk memenuhi kebutuhan informasi di suatu area tempat tertentu [7].

### 2.1.7 MySQL

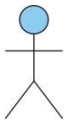
MySQL Karena karakteristik arsitektur dari MySQL yang flexible, MySQL dapat berjalan dengan baik di desktop application maupun web application. MySQL dapat digunakan untuk embedded application, data warehouses, content indexing and delivery software, highly reduntant systems, online transaction proccesing(OLTP), dan masih banyak lagi. Dengan konfigurasi yang benar, MySQL dapat digunakan pada berbagai macam perangkat keras, dan juga MySQL mendukung banyak tipe data. Salah satu fitur penting dari MySQL adalah *storage engine architecture* yang designnya memisahkan *query processing* dan pekerjaan pekerjaan server lainnya dari penyimpanan dan pengambilan data. Dengan adanya separasi ini dapat membuat pengguna memilih bagaimana data disimpan, performa

apa yang ingin dicapai, fitur - fitur apa aja yang diinginkan, dan karakteristik lainnya. (open source) [8].

### 2.1.8 *Unified Modeling Languange* (UML)

Unified Modeling Language adalah sebuah alat desain yang digunakan untuk membuat gambaran tentang bagaimana sebuah sistem bekerja. Gambaran ini mencakup bagian-bagian seperti layar yang dilihat pengguna, tempat penyimpanan data, cara pengelolaan dokumen, dan bagian-bagian lain yang terlibat. Proses-proses tersebut dijelaskan menggunakan gambaran alur kerja dalam UML. Model logis sistem adalah cara untuk melihat bagaimana komputer mengelola informasi dan melakukan tugas [9]. Salah satu penerapan UML adalah *Use Case* diagram. *Use Case* diagram adalah cara untuk menjelaskan bagaimana sistem bekerja dari perspektif pengguna atau aktor, dengan cara menceritakan interaksi antara pengguna melalui cerita tentang bagaimana sistem digunakan [10]. Selain itu, *Use Case* diagram juga menunjukkan bagaimana hubungan antara aktor (yaitu pihak yang berinteraksi dengan sistem) dan fungsi atau tindakan dalam sistem yang akan dibuat atau dikembangkan. *Use Case* Diagram memiliki simbol – simbol [11]. Berikut adalah simbol-simbol untuk membuat *Use Case* Diagram :

**Tabel 2. 1** Simbol Use Case Diagram

| No. | Nama | Simbol | Fungsi |
|---|---|---|---|
| 1. | Aktor |  | Aktor/peran adalah entitas manusia atau sistem lain yang terlibat dalam interaksi dengan sistem saat ini. |
| 2. | *Include* |  | Menspesifikasikan dengan jelas sumber use case merupakan bagian dari use case lain. |
| 3. | *Extend* |  | Menentukan bahwa suatu use case tambahan mengarah pada situasi di mana use case yang ditambahkan dapat berdiri sendiri tanpa bergantung pada use case yang sudah ada sebelumnya. |
| 4. | *Association* | ——— | Menghubungkan koneksi antara satu objek dengan objek lainnya. |

| | | | |
|---|---|---|---|
| 5. | *System Boundary* |  | Batasan sebuah sistem |
| 6. | *Use Case* |  | Menjelaskan yang dilakukan actor dari sistem untuk mencapai tujuan tertentu. |

### 2.1.9 Data Definition Language (DDL)

Data Definition Language (DDL) adalah sebuah paket bahasa yang digunakan dalam sistem manajemen basis data (DBMS) untuk mendefinisikan struktur dan skema dari basis data. Perintah-perintah DDL mencakup operasi seperti Create Table untuk membuat tabel baru dalam basis data, Create Index untuk membuat indeks pada tabel, Alter Table untuk mengubah struktur tabel yang sudah ada, Drop View untuk menghapus sebuah view dari basis data, dan Drop Index untuk menghapus indeks dari sebuah tabel. Dengan menggunakan perintah DDL, struktur basis data dapat disesuaikan dengan kebutuhan aplikasi dan penggunaannya dapat dioptimalkan untuk kinerja dan keamanan yang lebih baik. [12]

### 2.1.10 *Data Manipulation Language (DML)*

DML (Data Manipulation Language)Data manipulation languagemerupakan satu paket DBMS yang memungkinkan pengguna untuk mengakses atau memanipulasi data sebagaimana yang telah diatur sebelumnya dalam model data yang sesuai. Dengan DML kita dapat:

a. Mengambil informasi yang tersimpan dalam database.
b. Menyisipkan informasi baru ke database.
c. Menghapus data dari tabel.Ada dua jenis DML, yaitu prosedural dan non prosedural.

Teknik DML mengharuskan pengguna untuk menentukan data apa yang dibutuhkan dan bagaimana cara mengambilnya. Contoh bahasa prosedural adalah FoxBase, dBase III, FoxPro. DML non-prosedural mengharuskan pengguna untuk menentukan data apa yang dibutuhkan tanpa harus mengetahui cara mendapatkannya. Contoh bahasa non-prosedural adalah SQL (Structured Query Language) atau QBE (Query By Example). Contoh perintah DML adalah insert, select, updatedan delete. Ketiga perintah bahasa ini (DDL, DML, DCL) saat ini telah dibentuk menjadi sebuah paket bahasa yang disebut sebagai SQL (Structured

Query Language), yang implementasinya sangat berbeda dengan SQL. Tidak semua fitur SQL didukung oleh vendor software. Beberapa contoh software databaseyang menggunakan SQL seperti DB2, Ingres, Informix, Oracle, MS-Access, MySQL, PostgreSQL, Rdb, dan Sybase [12].

### *2.1.11 Entity Relationship Diagram (ERD)*

Dengan kata lain, ERD adalah suatu model jaringan yang menggunakan susunan data yang disimpan dalam sistem secara abstrak. Jadi, jelalah balwa ERD ini berbeda dengan DFD yang merupakan suatu model jaringan fungsi yang akan dilaksanakan oleh sistem, sedangkan ERD merupakan model jaringan data yang menekankan pada struktur-struktur dan relationship data (Ladjamudin, Al-Bahra. 2005). Berikut ini elemen-elemen dari ERD :

a. Entitas

Entitas dalam Entity-Relationship (E-R) diagram direpresentasikan dengan bentuk persegi panjang dan mewakili suatu objek atau konsep yang ada dalam sistem basis data. Entitas dapat berupa sesuatu yang nyata atau abstrak di mana data disimpan atau terkait dengan proses penyimpanan data. Biasanya, entitas dinamai menggunakan kata benda dan dapat dikelompokkan ke dalam empat jenis berdasarkan sifatnya, yaitu orang, benda, lokasi, dan kejadian, yang mencakup aspek waktu. Pengidentifikasian entitas dalam sebuah diagram membantu dalam merancang struktur basis data dengan jelas, memastikan bahwa semua objek yang relevan dengan sistem tercakup dalam model.

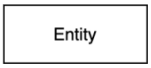
b. Relationship

Pada E-R diagram, Relationship dapat digambarkan dengan sebuah bentuk belah ketupat. Relationship adalah hubungan alamiah yang terjadi antara entitas. Pada umumnya penghubung (Relation-ship) diberi nama dengan kata kerja dasar, sehingga memudahkan untuk melakukan pembacaan relasinya bisa dengan kalimat aktif atau kalimat pasif). Penggambaran hubungan yang terjadi adalah sebuah bentuk belah ketupat dihubungkan dengan dua bentuk empat persegi panjang.

Contohnya Entitas Mahasiswa dengan NIM ="14534" dan Nama Mhs ="Yudin" yang mempunyai relasi dengan Entitas Kuliah dengan Kode_Kul ="SI-140" dan Nama_MK=" Basis Data", sehingga Struktur data dari Relasi ini bahwa mahasiswa

tersebut mengambil mata kuliah pada suatu perguruan tinggi.

**Tabel 2. 2** Simbol Entity Relationship Diagram

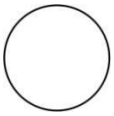
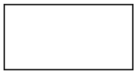
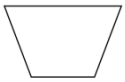
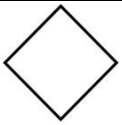
| No. | Nama | Simbol | Fungsi |
|---|---|---|---|
| 1. | Himpunan Entitas atau Entitas E |  | Entitas merupakan informasi inti yang mencakup manusia, peristiwa, atau objek yang hendak disimpan; merupakan sekumpulan entitas yang dapat dikenali dengan jelas. |
| 2. | Atribut a sebagai key |  | Atribut adalah data yang diperoleh mengenai suatu entitas. |
| 3. | Himpunan Relasi atau Relasi R |  | Relasi merujuk pada hubungan antara satu entitas dengan entitas lainnya. |
| 4. | *Link* |  | Penghubung antara relasi dan entitas. |

### 2.1.12 Flowchart

Flowchart adalah bagan-bagan yang mempunyai arus yang menggambarkan langkah- langkah penyelesaian suatu masalah. Flowchart merupakan cara penyajian dari suatu algoritma dan Bagan yang memperlihatkan urutan proses dalam system dengan menunjukkan alat media input, output serta jenis media penyimpanan dalam proses pengolahan data (Ladjamudin, Al-Bahra. 2005).

**Tabel 2. 3** Simbol Flowchart

| No. | Nama | Simbol | Fungsi |
|---|---|---|---|
| 1. | Arus/*Flow* |  | Untuk menyatakan jalannya arus suatu proses. |
| 2. | Terminal |  | Untuk menyatakan permulaan atau akhir suatu program. |

| | | | |
|---|---|---|---|
| 3. | *Connector* |  | Untuk menyatakan sambungan dari satu proses ke proses lainnya dalam halaman/lembar yang sama. |
| 4. | *Process* |  | Untuk proses operasional komputer. |
| 5. | *Manual Operator* |  | Untuk menyatakan suatu tindakan (proses) yang tidak dilakukan oleh komputer (manual). |
| 6. | *Decision/* logika |  | Untuk menunjukkan suatu kondisi tertentu yang akan menghasilkan dua kemungkinan jawaban, ya/tidak. |
| 7. | *Input-output* |  | Untuk menyatakan proses input dan output tanpa tergantung dengan jenis peralatannya, |
| 8. | *Document* |  | Untuk mencetak laporan ke printer. |

### 2.1.13 *Framework*

*Framework* adalah komponen pemrograman yang siap *re-use* (bisa digunakan ulang) kapan saja, sehingga *programmer* tidak harus membuat skrip yang sama untuk tugas yang sama. Misalkan *programmer* ingin halaman-halaman web menampilkan data dengan paginasi (paging) halaman, *Framework* telah menyediakan fungsi paging tersebut sedangkan *programmer* cukup menggunakan fungsi tersebut pada saat coding, tetapi tentu dengan kaidah- kaidah yang ditetapkan oleh masing - masing *Framework* [13]. *Framework* adalah kumpulan intruksi-intruksi yang dikumpulkan dalam class dan function-function dengan fungsi masing-masing untuk memudahkan developer dalam memanggilnya tanpa harus menuliskan syntax program yang sama berulang-ulang serta dapat menghemat waktu [14].

### 2.1.14 Laravel

Laravel adalah sebuah *Framework* web berbasis PHP yang open-source dan

tidak berbayar, diciptakan oleh Taylor Otwell dan diperuntukkan untuk pengembangan aplikasi web yang menggunakan pola MVC. Struktrur pola MVC pada laravel sedikit berbeda pada struktur pola MVC pada umumnya. Di laravel terdapat routing yang menjembatani antara request dari user dan controller. Jadi controller tidak langsung menerima request tersebut [13].

#### 2.1.15 *Hypertext Prepocessor (PHP)*

PHP adalah bahasa pemrograman script server-side yang didesain untuk pengembangan web. Selain itu, PHP juga bisa digunakan sebagai bahasa pemrogramanumum. PHP diciptakan oleh Rasmus Lerdorf pertama kali tahun 1994. Saat ini PHP adalah singkatan dari PHP: Hypertext Preprocessor, sebuah kepanjangan rekursif, yakni permainan kata dimana kepanjangannya terdiri dari singkatan itu sendiri: PHP: Hypertext Preprocessor. PHP dapat digunakan dengan gratis (free) dan bersifat Open Source. PHP dirilis dalam lisensi PHP License, sedikit berbeda dengan lisensi GNU General Public License (GPL) yang biasa digunakan untuk proyek Open Source [15].

#### 2.1.16 *HyperText Markup Language (HTML)*

*Hypertext Markup Language* (HTML) adalah script pemrograman yang mengatur bagaimana kita menyajikan informasi di dunia internet dan bagaimanainformasi itu membawa kita melompat dari satu tempat ke tempat lainnya. HTML dibuat oleh Tim Berners-Lee ketika masih bekerja dengan CERN dan dipopulerkan pertama kali oleh browser Mosaic. Awal tahun 1990 HTML mengalamiperkembangan yang sangat maju. Setiap pengembangan HTML pasti akan menambahkan kemampuan dan fasilitas yang lebih baik dari versi sebelumnya. (sini) Bahasa pemrograman yang digunakan untuk mengembangkan website dengan HTML adalah PHP dan Javascript [15].

#### 2.1.17 *Cascading Style Sheets (CSS)*

*Cascading Style Sheet* (CSS) merupakan salah satu kode pemrograman yang bertujuan untuk menghias dan mengatur gaya tampilan atau layout halaman web agar lebih elegan dan menarik. CSS adalah sebuah dokumen yang berdiri sendiri dan dapat dimasukkan dalam kode HTML atau sekedar menjadi rujukan oleh HTML dalam pendefinisian style. Ada banyak hal yang dapat di lakukan menggunakan CSS dibandingkan dengan bahasa pemrograman inti seperti HTML dan PHP. Ketika menggunakan CSS, dapat mengatur warna teks, jenis font, baris

antar paragraf, ukuran kolom, dan jenis background yang dipakai.Tidak hanya itu CSS juga bisa untuk mendesain layout, variasi tampilan di berbagai perangkat yang berbeda, dan berbagai efek yang dipakai di dalam website.CSS sangat mudah dipelajari, tapi juga powerful karena dapat mengontrol penyajian tampilan dari dokumen HTML. Mulai dari yang sederhana sampai kompleks. Tidak heran jika saat ini CSS hampir dipakai di berbagai website untuk dikombinasikan dengan HTML maupun PHP [15].