

BAB II

TINJAUAN PUSTAKA

2.1 Praktikum

Praktikum adalah subsistem dari perkuliahan yang merupakan kegiatan terstruktur dan terjadwal yang memberi kesempatan kepada mahasiswa untuk mendapatkan pengalaman yang nyata dalam rangka meningkatkan pemahaman mahasiswa tentang teori atau agar mahasiswa menguasai keterampilan tertentu yang berkaitan dengan suatu pengetahuan (Akib et al., n.d.).

2.2 Framework

Framework adalah suatu kumpulan kode berupa pustaka (*library*) dan alat (*tool*) yang dipadukan sedemikian rupa menjadi satu kerangka kerja (*Framework*) guna memudahkan dan mempercepat proses pengembangan aplikasi web (Rachman et al., 2020). Jadi, *Framework* adalah kumpulan-kumpulan potongan program yang dipadukan menjadi satu kerja kerja yang digunakan untuk membantu dalam pembuatan sebuah aplikasi. Proses pengembangan web dapat dilakukan dengan beragam bahasa pemrograman seperti PHP, Python, Ruby, Perl, C++, JAVA dan sebagainya. Saat ini, banyak bermunculan *Framework* web yang dirancang untuk bahasa bahasa pemrograman tersebut. Salah satunya adalah *CodeIgniter*.

2.3 CodeIgniter

CodeIgniter adalah sebuah *web application Framework* yang bersifat *open source* digunakan untuk membangun aplikasi PHP dinamis (Abidilah, n.d.). Tujuan utama pengembangan *CodeIgniter* adalah untuk membantu *developer* untuk mengerjakan aplikasi lebih cepat daripada menulis semua *code* dari awal. *CodeIgniter* menyediakan berbagai macam *library* yang dapat mempermudah dalam pengembangan.

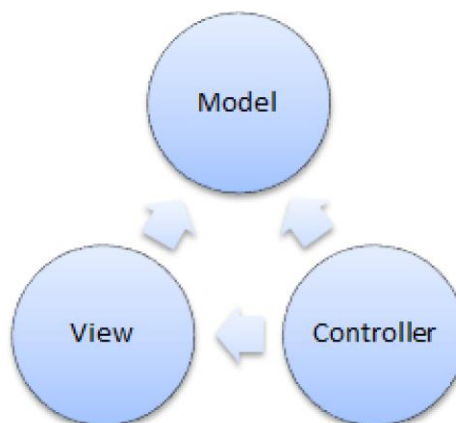
2.4 Kelebihan *CodeIgniter*

Kelebihan dari penggunaan *Framework CodeIgniter* yang merupakan sebuah *Framework* yang *open source* dengan menggunakan bahasa pemrograman PHP dengan menerapkan *Model View Controller* (MVC), diantaranya

1. Cepat : *Framework CodeIgniter* memiliki ukuran data bawaan yang kecil, hal ini dapat diakses dengan cepat oleh user.
2. *Resource*, untuk dapat menggunakan *Framework* ini, kita hanya memerlukan spesifikasi *server/hosting* yang kecil, dibandingkan dengan *Framework* lain seperti *laravel*.
3. Lebih mudah, hal ini berarti *Framework CodeIgniter* tidak perlu melakukan instal hal-hal lainnya (Syafitri et al., 2021).

2.5 MVC (*Model View Controller*)

Model-View-Controller, atau MVC, adalah pola arsitektur yang digunakan untuk membantu memisahkan lapisan antarmuka pengguna dari bagian sistem lainnya (LETHBRIDGE, n.d.). MVC tidak hanya membantu menegakkan kohesi lapisan dari lapisan antarmuka pengguna, tetapi juga membantu mengurangi sambungan antara lapisan itu dan sistem lainnya, serta antara berbagai aspek UI itu sendiri. Pola MVC memisahkan lapisan fungsional sistem (model) dari dua aspek antarmuka pengguna, tampilan dan pengontrol. Hal ini diilustrasikan pada Gambar 2.1



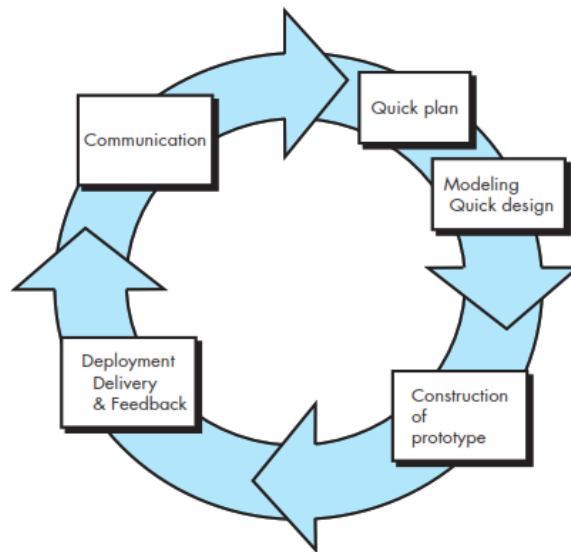
Gambar 2. 1 Kosep MVC

2.6 Github

GitHub adalah situs web hosting kode yang sangat populer dan menggunakan Manajemen Kode Sumber *Git* (SCM). *Git* memungkinkan untuk berbagai jenis alur kerja yang dapat disesuaikan dengan kebutuhan individu proyek (Peterson, 2013). Selain itu, *GitHub* juga menggabungkan aspek sosial dalam pengembangan perangkat lunak. Pengguna *GitHub* memiliki profil yang terlihat oleh komunitas, dan tindakan pengguna dapat dilacak dan diikuti oleh anggota komunitas lainnya. Jenis integrasi sosial dari identitas dan tindakan pengguna ini merupakan keunikan dari *GitHub*. Sebagai platform *open source*, *GitHub* memungkinkan pengguna untuk membagikan, mengunduh, dan memodifikasi kode secara bebas..

2.7 Metode *Prototype*

Prototype merupakan metode yang efektif dalam merancang perangkat lunak. *Prototype* dimulai dengan mengumpulkan kebutuhan yang akan dirancang. Pengembangan mendefinisikan objek keseluruhan dari perangkat lunak, mengidentifikasi segala aktifitas yang diketahui dan kemudian melakukan “perancangan kilat”. Perancangan kilat berfokus pada penyajian dari aspek-aspek perangkat lunak tersebut yang akan nampak bagi pelanggan atau pemakai (contohnya pendekatan input dan format output) (Roger S. Pressman & Bruce R. Maxim, 2015). Gambar 2.2 berikut adalah metode *Prototype* :



Gambar 2. 2 Metode *Prototype*

Tahapan-tahapan dalam metode *Prototype* adalah sebagai berikut :

1. *Communication*, yaitu komunikasi dengan klien dan user untuk menentukan kebutuhan.
2. *Quick Plan*, yaitu pembuatan perencanaan analisis terhadap kebutuhan pengguna.
3. *Modeling Quick Design*, yaitu membuat rancangan desain program.
4. *Construction of Prototype*, yaitu pembuatan aplikasi berdasarkan dari pemodelan desain yang telah dibuat.
5. *Development Delevery and Feedback*, yaitu memproduksi perangkat secara benar sehingga dapat digunakan oleh pengguna.

2.8 Unified Modeling Language (UML)

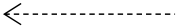
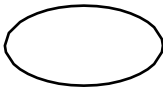
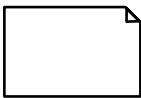
R. A. Sukamto dan Shalahuddin menjelaskan bahwa “UML (*Unified Modeling Language*) adalah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun menggunakan teknik pemograman berorientasi objek (Sukamto & Shalahuddin, 2013). UML (*Unified Modeling Language*) muncul karena adanya kebutuhan pemodelan

visual untuk menspesifikasi, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak.”

1. *Use Case Diagram*

John W Satzinger, Robert B Jackson menguraikan bahwa “*Use Case Diagram* merupakan rangkaian tindakan yang dilakukan oleh sistem, aktor mewakili user atau sistem lain yang berinteraksi dengan sistem yang dimodelkan.” (Satzinger et al., 2005) Tabel 2.1 berikut merupakan simbol-simbol *use case diagram* disertai dengan penjelasan kegunaan dari simbolnya :

Tabel 2. 1 Simbol *Use Case Diagram*

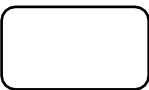
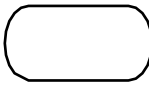
Gambar	Nama	Keterangan
	<i>Actor</i>	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan <i>use case</i> .
	<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.
	<i>System</i>	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
	<i>Generalization</i>	Hubungan dimana objek anak (<i>descendent</i>) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (<i>ancestor</i>).
<<Include>>	<i>Include</i>	Menspesifikasikan bahwa <i>use case</i> sumber secara <i>eksplisit</i> .
<<Extend>>	<i>Extend</i>	Menspesifikasikan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan.
	<i>Use Case</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
	<i>Collaboration</i>	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (<i>sinergi</i>).
	<i>Note</i>	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

Tabel 2. 1 diatas merupakan *Use Case Diagram* menjelaskan simbol-simbol yang digunakan untuk membuat *use case diagram* beserta nama dari setiap simbol dan dilengkapi dengan penjelasan kegunaan dari setiap simbol. Dengan demikian dapat disimpulkan bahwa *use case diagram* digunakan untuk mengetahui alur atau tindakan yang akan terjadi pada sebuah aplikasi yang akan kita bangun. *Use case diagram* menggambarkan bagaimana user atau aktor akan berinteraksi dengan aplikasi yang dibangun.

2. Activity Diagram

R. A. Sukamto dan Shalahuddin menguraikan bahwa “diagram aktivitas atau *activity diagram* adalah menggambarkan aliran kerja atau aktifitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Diagram aktifitas menggambarkan aktifitas sistem bukan apa yang dilakukan oleh aktor.” (Sukamto & Shalahuddin, 2013)

Tabel 2. 2 Simbol Activity Diagram

Gambar	Nama	Keterangan
	<i>Activity</i>	Memperlihatkan bagaimana masing-masing kelas antarmukasaling berinteraksi satu sama lain.
	<i>Action</i>	<i>State</i> dari sistem yang mencerminkan eksekusi dari suatu aksi.
	<i>Initial Node</i>	Bagaimana objek dibentuk atau diawali.
	<i>Activity Final Node</i>	Bagaimana objek dibentuk dan dihancurkan
	<i>Fork Node</i>	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

Tabel 2.2 tersebut diatas merupakan Simbol *Activity Diagram* menjelaskan simbol-simbol yang digunakan untuk membuat *activity diagram* beserta nama dari setiap simbol dan dilengkapi dengan penjelasan kegunaan dari setiap simbol. Dengan demikian dapat disimpulkan bahwa *activity diagram* adalah penjelasan tentang alur kerja dari setiap menu yang ada pada program. Menggambarkan bagaimana sebuah menu akan bereaksi. Pada *Diagram Activity* lebih menekankan pada penjelasan alur atau aktifitas menu dari aplikasi bukan aktifitas yang dilakukan oleh user atau aktor.

2.9 User Experience Questionnaire (UEQ)

Kuesioner UEQ atau *User Experience Questionnaire*, termasuk salah satu kuesioner yang hasilnya dapat digunakan dalam usability testing untuk mengukur tingkat *user experience* suatu produk dengan cepat. UEQ dapat diunduh di www.ueq-online.org. Terdapat 6 skala dengan total 26 elemen yang dikategorikan berdasarkan skala-skala pengukuran yang terdapat pada UEQ (D. M. Schrepp, 2014). Skala *User Experience* dalam kuesioner, yaitu:

- a. *Attractiveness* (Daya tarik): Kesan terhadap produk. Apakah pengguna menyukai atau tidak menyukai produk? Seberapa besar daya tarik dari sebuah produk. Misal: bagus atau jelek, atraktif atau tidak atraktif.
- b. *Efficiency* (Efisiensi): Kemungkinan suatu produk dapat digunakan dengan cepat dan efisien. Seberapa besar pengguna dapat menyelesaikan tugasnya tanpa usaha yang besar atau efisien. Misal: cepat atau lambat, praktis atau tidak praktis.
- c. *Perspicuity* (Kejelasan): Apakah memungkinkan menggunakan produk ini? Apakah mudah bagi pengguna untuk membiasakan diri menggunakan produk ini? Seberapa besar kejelasan dari sebuah produk. Misal: mudah dipahami atau sulit dipahami.
- d. *Dependability* (Ketepatan): Apakah pengguna merasa dalam kontrol interaksi? Apakah interaksi dengan produk aman dan dapat diprediksi? Seberapa besar ketepatan yang dirasakan oleh pengguna melalui

kontrol yang ia miliki. Misal: dapat diprediksi atau tidak dapat diprediksi, mendukung atau menghalangi.

- e. *Stimulation* (Stimulasi): Apakah menarik dan menyenangkan untuk pengguna dalam menggunakan produk tersebut? Apakah pengguna merasa termotivasi untuk terus menggunakan produk? Seberapa besar motivasi untuk menggunakan produk. Misal: bermanfaat atau kurang bermanfaat, menarik atau tidak menarik.
- f. *Novelty* (Kebaruan): Apakah desain produk inovatif dan kreatif? Apakah produk mampu mendapatkan perhatian pengguna? Seberapa kreatif atau besar kebaruan dari produk.

Enam skala UEQ dapat dikelompokkan menjadi 3, yaitu aspek daya tarik (*attractiveness*), aspek *pragmatic quality*, dan aspek *hedonic quality*. Aspek *pragmatic quality* berkaitan dengan manfaat yang dirasakan, efisiensi dan kemudahan digunakan. Kejelasan (*perspicuity*), efisiensi (*efficiency*) dan ketepatan (*dependability*) termasuk kedalam *pragmatic quality aspect*. Aspek *hedonic quality* berkaitan dengan stimulasi (*stimulation*) dan kebaruan (*novelty*) (D. M. Schrepp, 2014).

Data UEQ dapat diolah dengan menggunakan *UEQ tool analysis*. Untuk mengetahui makna dari nilai yang dihasilkan oleh setiap skala UEQ dilakukan dengan membandingkan nilai hasil perhitungan setiap skala dengan nilai pada Tabel 2.3 berikut.

Tabel 2. 3 Benchmark Interval untuk Skala UEQ (M. Schrepp et al., 2017)

	Daya Tarik (attractiveness)	Kejelasan (perspicuity)	Efisiensi (efficiency)	Ketepatan (dependability)	Stimulasi (stimulation)	Kebaruan (novelty)
<i>Excellent</i>	$\geq 1,75$	$\geq 1,9$	$\geq 1,78$	$\geq 1,65$	$\geq 1,55$	$\geq 1,4$
<i>Good</i>	$\geq 1,52$ $< 1,75$	$\geq 1,56$ $< 1,9$	$\geq 1,47$ $< 1,78$	$\geq 1,48$ $< 1,65$	$\geq 1,31$ $< 1,55$	$\geq 1,05$ $< 1,4$
<i>Above Average</i>	$\geq 1,17$ $< 1,52$	$\geq 1,08$ $< 1,56$	$\geq 0,98$ $< 1,47$	$\geq 1,14$ $< 1,48$	$\geq 0,99$ $< 1,31$	$\geq 0,71$ $< 1,05$
<i>Below Average</i>	$\geq 0,7$ $< 1,17$	$\geq 0,64$ $< 1,08$	$\geq 0,54$ $< 0,98$	$\geq 0,78$ $< 1,14$	$\geq 0,5$ $< 0,99$	$\geq 0,3$ $< 0,71$
<i>Bad</i>	$< 0,7$	$< 0,64$	$< 0,54$	$< 0,78$	$< 0,5$	$< 0,3$

2.10 Penelitian Terkait

Penelitian ini mereferensi penelitian- penelitian terkait sesuai dengan masalah yang didapat dari latar belakang peneltian ini.

Tabel 2. 4 Penelitian Terkait

No	Judul, Penulis, Tahun	Metode	Hasil
1.	IMPLEMENTASI <i>FRAMEWORK CODEIGNITER</i> DALAM PEMBUATAN SISTEM INFORMASI PENCATATAN DAN PENDATAAN PENDUDUK DESA BERBASIS WEB, oleh Rakhmad Hidayat, Dony Alfian Rosid, Mochamad, tahun 2020.	Metode penelitian yang digunakan dalam pengumpulan data yaitu dengan teknik observasi berdasarkan jalannya pencatatan, pendataan, serta Sensus Penduduk 2020.	sistem informasi pencatatan dan pendataan penduduk berbasis web
2.	Pengembangan Sistem Pengelolaan Praktikum berbasis Web (Studi Kasus: Fakultas Ilmu Komputer Universitas Brawijaya) oleh A. Kartika Indrajati, D. S. Rusdianto, E. Muhammad, and A. Jonemaro, tahun 2019	siklus pengembangan dilakukan dengan metode Waterfall	Menghasilkan URL yang clean dan <i>search- engine friendly</i> .
3.	PENERAPAN <i>FRAMEWORK CODEIGNITER</i> PADA SISTEM ABSENSI QR CODE DISKOMINFO KABUPATEN LABUHANBATU SELATAN. Oleh D. Wahyudi and A. Putra Juledi, tahun 2021	metode water- fall	Dari hasil penelitian didapatkan bahwa <i>framework codeigniter</i> dapat diterapkan pada sistem absensi yang menggunakan QR Code dengan proses pengerjaan kode program yang lebih efisien.

4.	Implementasi Model View Controller Dengan <i>Framework CodeIgniter</i> Pada E-Commerce Penjualan Kerajinan Bali, oleh I. Ketut Suharsana, I. Wirarama Wedashwara Wirawan, N. S. Luh Ayu Kartika Yuniastari, tahun 2016	Pada penelitian ini pembangunan E-commerce ditekankan pada penelitian metode MVC dengan <i>framework CodeIgniter</i> .	Penelitian ini menghasilkan sebuah Aplikasi <i>E-commerce</i> kerajinan bali dengan <i>framework CodeIgniter</i> .
5.	Implementasi Model View Controller Dengan <i>Framework CodeIgniter</i> Pada Perpustakaan, oleh A. Dwi Praba, tahun 2018	metode pengembangan pada penelitian ini menggunakan pemrograman berbasis objek (OOP).	Sistem informasi perpustakaan.