BAB II

LANDASAN TEORI

2.1. Pengertian Tagihan

Tagihan merupakan sejumlah kewajiban yang harus dibayarkan oleh pelanggan atas seluruh penggunaan atau pemakaian jasa dan fasilitas tertentu yang biasanya dalam kurun waktu 1 bulan, termasuk juga jumlah denda, bunga, biaya administrasi serta biaya lain apabila ada (Agianto, Somantri, & Sinuraya, 2018).

2.2. Pengertian *Web*

"*Web* dapat diartikan sekumpulan halaman yang terdiri dari beberapa laman yang berisi informasi dalam bentuk data digital baik berupa text, gambar, video, audio, dan animasi lainnya yang disediakan melalui jalur koneksi internet (Abdullah, 2015)". *Web* pada saat ini sudah cukup banyak digunakan untuk keperluan bisnis dikarenakan perkembangan teknologi yang semakin pesat dan juga kebutuhan manusia yang meningkat. Selain itu, *web* juga digunakan sebagai media promosi, media pemasaran, media informasi, media pendidikan dan media komunikasi.

2.2.1. HTML (*HyperText Markup Language*)

"HTML merupakan bahasa pemrograman *web* yang memberitahukan peramban *web* (*web browser*) bagaimana menyusun dan menyajikan konten di halaman *web* (Solichin, 2016)". Pada intinya, HTML digunakan untuk penempatan sebuah konten pada halaman *web* agar di dalam halaman *web* dapat tersusun dengan baik dan rapi.

2.2.2. CSS (*Cascading Style Sheets*)

"CSS adalah suatu teknologi yang digunakan untuk memperindah tampilan halaman *website* (situs)" (Prasetio, 2014). Selain untuk memperindah tampilan pada halaman website, CSS juga memudahkan dalam membuat halaman *website* dikarenakan hanya perlu membuat *style sheet* agar dapat dipakai dalam setiap halaman *website*.

2.2.3. *PHP* (*Hypertext Preprocessor*)

"*PHP Hypertext Preprocessor* adalah suatu bahasa *scripting* khususnya digunakan untuk *web development*" (Hidayatullah & Kawistara, 2017). *PHP* memiliki sifat *server side scripting* sehingga untuk menjalankan *PHP* harus menggunakan *web server*. *PHP* juga dapat digunakan secara gratis dan bersifat *open source*. *PHP* juga masih relevan dan masih menjadi salah satu pilihan untuk pengembangan *web.*

2.2.4. *JavaScript*

JavaScript adalah pemrograman dengan dukungan langsung ke metodologi berorientasi objek (Ahmed, 2014). *JavaScript* memungkinkan aplikasi *web* modern yaitu aplikasi yang dapat digunakan untuk berinteraksi secara langsung tanpa melakukan pemuatan ulang halaman untuk setiap tindakan. *JavaScript* dapat digunakan di situs *web* yang lebih tradisional untuk disediakan berbagai bentuk interaktivitas dan kepandaian (Haverbeke, 2018).

2.2.5. *XAMPP*

Menurut (Solichin, 2016) *xampp* merupakan perangkat lunak yang terdiri dari *PHP*, *Apache*, *MySQL*, dan *phpMyAdmin* sehingga menjadi satu kesatuan atau dikenal dengan *software package/installer* dimana proses dan konfigurasi dilakukan secara otomatis, mudah dan praktis.

2.2.6. *MySQL*

MySQL adalah salah satu aplikasi DBMS yang sudah banyak oleh para pemogram aplikasi *web*. Contoh DBMS lainnya adalah : *PostgreSQL* (*freeware*), *SQL Server*, *MS Access* dari Microsoft, *DB2* dari IBM, Oracle dan Oracle Corp, Dbase, FoxPro, dsb (Hidayatullah & Kawistara, 2017).

2.2.7. *Visual Studio Code*

Visual Studio Code (VS Code) merupakan sebuah teks editor yang dibuat oleh Microsoft untuk sistem operasi multiplatform yaitu untuk Linux, Mac, dan Windows. Teks editor ini mendukung bahasa pemrograman *JavaScript*, *Typescript*, dan *Node.js*, serta bahasa pemrograman lainnya dengan bantuan *plugin* yang dapat dipasang via *marketplace Visual Studio Code* (seperti *C++*, *C#*, *Python*, *Go*, *Java*, dst) (Yulianto W, 2019).

2.3. Basis data

Basis data (*database*) adalah sekumpulan data yang memiliki hubungan secara logika dan diatur menurut susunan tertentu serta disimpan dalam media penyimpanan komputer. Basis data digunakan untuk memproses data untuk menghasilkan informasi tertentu (Aisyah, 2019).

2.4. *WhatsApp*

WhatsApp adalah sebuah aplikasi pesan lintas platform yang memiliki fungsi untuk mengirim dan menerima pesan dengan gratis tanpa dikenakan biaya SMS, hal ini dikarenakan adanya paket data internet yang sama untuk *email*, *browsing web*, berlaku juga untuk penggunaan *WhatsApp* (Anjani, Ratnamulyani, & Kusumadinata, 2018).

2.5. *Application Programming Interface* (API)

Application Programming Interface (API) memiliki manfaat yang dimungkinkan pengembang dapat mengintegrasikan antara 2 bagian aplikasi atau aplikasi yang berbeda. Pengembangan aplikasi yang membutuhkan API yang terdiri dari beberapa element seperti *function, protocols,* dan *tools* (Sunardi, Riadi, & Raharja, 2019). API digunakan untuk mempersingkat proses pengembangan sehingga pengembang tidak pelu membuat fitur yang sama (Sunardi et al., 2019). API adalah sebuah class yang dirancang untuk menghubungkan antara aplikasi *mobile* dengan basis data (Sunardi et al., 2019). Melalui *class* ini dapat bekerja untuk mengakses dan mengeksekusi beragam perintah dari aplikasi *mobile*. (Sunardi et al., 2019).

2.6. Notifikasi dan *Reminder*

Notifikasi adalah pemberitahuan mengenai informasi atau pengumuman dari pihak tertentu kepada pihak yang dituju yang dilakukan melalui media seperti *email, sms,* maupun dari aplikasi pesan instan seperti *whatsapp* (Winandar, 2015). Sedangkan *reminder* adalah sebuah pesan yang menolong seseorang untuk mengingat sesuatu. *Reminder* dapat lebih bermanfaat ketika informasi kontekstual

digunakan untuk menyajikan informasi pada waktu yang tepat dan tempat yang tepat. *Reminder* dapat digunakan sebagai manajemen waktu yang berfungsi untuk memberi alarm peringatan berupa pemberitahuan berbasis lokasi, waktu maupun catatan yang berupa kontekstual (Putri, Arso, & Sriatmi, 2017).

2.7. Pengertian Framework

Framework adalah sekumpulan fungsi, *class*, dan aturan-aturan. *Framework* bersifat menyeluruh mengatur bagaimana kita membangun aplikasi. *Framework* memungkinkan kita membangun aplikasi dengan lebih cepat karena sebagai *developer* kita akan lebih memfokuskan pada pokok permasalahan, sedangkan untuk hal-hal penunjang lainnya seperti koneksi ke *database*, *form validation*, GUI dan *security* umumnya telah disediakan oleh *framework* (Ruli Erinton, Ridha Muldina Negara, 2017) .

2.8. Pengertian Codeigniter

CodeIgniter adalah sebuah *web application framework* yang digunakan untuk membangun aplikasi PHP dinamis yang dibangun menggunakan konsep *Model View Controller development pattern*. *CodeIgniter* menyediakan berbagai macam *library* yang dapat mempermudah dalam pengembangan dan termasuk *framework* tercepat dibandingkan dengan *framework* lainnya (Ruli Erinton, Ridha Muldina Negara, 2017).

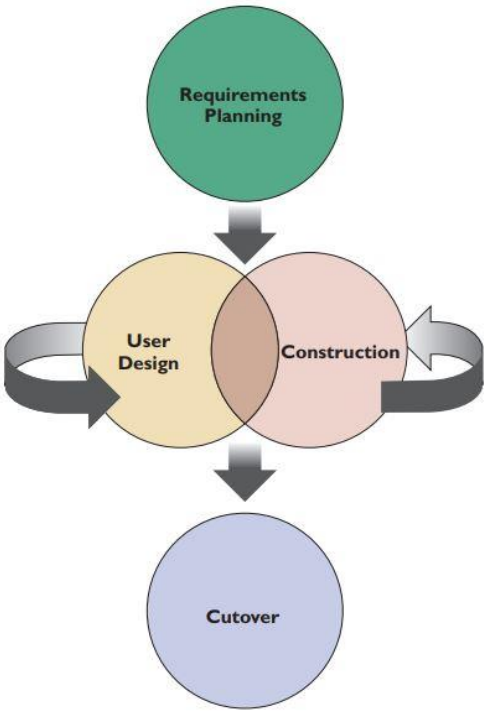
2.9. Pengertian *Blackbox Testing*

Metode *blackbox testing* adalah salah satu metode dari beberapa metode testing lainnya, yang memiliki pengertian yaitu salah satu metode yang mudah

digunakan karena hanya memerlukan batas bawah dan batas atas dari data yang diharapkan (Cholifah, Yulianingsih, & Sagita, 2018). *Blackbox testing* merupakan pengujian yang berorientasi pada fungsionalitas yaitu perilaku dari perangkat lunak atas input yang diberikan dari perangkat lunak atas input yang diberikan pengguna sehingga mendapatkan atau menghasilkan *output* yang diinginkan tanpa melihat proses internal atau kode program yang dieksekusi oleh perangkat lunak (Alfaris, Anam, & Masy'an, 2013).

2.10. Metode *Rapid Application Development*

RAD merupakan sebuah model pengembangan sekuensial linear yang menekankan pada siklus pengembangan yang sangat pendek. Model RAD merupakan penerapan kecepatan tinggi dari model-model tradisional yang mana kecepatan dalam pengembangan dicapai dengan menggunakan pendekatan konstruksi berbasis komponen (Saptono & Anggrainingsih, 2016). Dengan metode RAD, pengembangan suatu sistem informasi dapat dilakukan dengan waktu yang relatif singkat. Jika *requirements* sudah dipahami dengan baik dan ruang lingkup dari proyek dibatasi, dengan metode *Rapid Application Development*, *developer team* dapat menyelesaikan suatu sistem hanya dalam waktu kurang lebih 60-90 hari (Mishra & Dubey, 2013). Metode ini juga cukup tepat jika digunakan untuk pembuatan sistem skala menengah.



Gambar 2.1. Metode RAD

Sumber: (Shelly & Rosenblatt, 2012)

Berdasarkan gambar 2.1. dalam buku *Systems Analysis and Design Ninth Edition* karangan (Shelly & Rosenblatt, 2012), Metode RAD memiliki beberapa tahapan antara lain:

1. *Requirements Planning*

   Tahap *requirements planning* menggabungkan elemen fase perencanaan sistem dan analisis sistem SDLC. *Users*, *managers*, and IT *staff members* mendiskusikan dan menyetujui kebutuhan bisnis, ruang lingkup proyek, kendala, dan persyaratan sistem. Tahap *requirements planning* berakhir ketika tim menyetujui masalah utama dan mendapatkan otorisasi manajemen untuk melanjutkan.

2. *User Design*

Selama fase *user design*, pengguna berinteraksi dengan analis sistem dan mengembangkan model dan prototipe yang mewakili semua proses, output, dan input sistem. Grup atau subkelompok RAD biasanya menggunakan kombinasi JAD *techniques* dan CASE *tools* untuk menerjemahkan kebutuhan pengguna ke dalam *working models*. *User design* adalah proses interaktif dan berkelanjutan yang memungkinkan pengguna untuk memahami, memodifikasi, dan akhirnya menyetujui sebuah *working model* dari sistem yang memenuhi kebutuhan mereka.

3. *Construction*

Tahap *construction* berfokus pada pengembangan program dan aplikasi tugas yang mirip dengan SDLC. Namun dalam RAD, pengguna terus berpartisipasi dan masih dapat menyarankan perubahan atau perbaikan program atau aplikasi tersebut.

4. *Cutover*

Fase *cutover* menyerupai tugas akhir dalam fase implementasi SDLC, termasuk konversi data, pengujian, pergantian ke sistem baru, dan pelatihan pengguna. Dibandingkan dengan metode tradisional, seluruh proses telah dikompresi. Sebagai hasilnya, sistem baru dibangun, dikirim, dan ditempatkan dalam operasi lebih cepat.

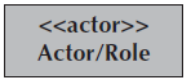
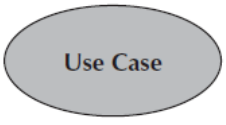
2.11. UML (*Unified Modeling Language*)

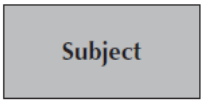
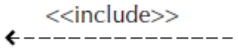
UML (*Unified Modeling Language*) adalah “Sebuah teknik pengembangan sistem yang menggunakan bahasa grafis sebagai alat untuk pendokumentasian dan melakukan spesifikasi pada sistem” (Mulyani, 2016). UML memang biasa digunakan untuk merancang dan mendokumentasikan sistem piranti lunak. UML juga memberikan sebuah standar dalam merancang model sebuah sistem.

2.11.1. *Use Case Diagram*

Use case diagram merupakan cara pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat (Al Faruq, 2015). *Use case diagram* menjelaskan bagaimana sistem digunakan dan merupakan titik awal dari permodelan UML. Adapun simbol-simbol yang digunakan dalam *use case* adalah sebagai berikut:

Tabel 2.1. Simbol *Use Case Diagram*

No.	Nama	Deskripsi	Simbol
1.	*Actor*	• Apakah seseorang atau sistem yang memperoleh manfaat dari dan merupakan eksternal dari subjek. • Digambarkan sebagai figur orang (*default*) atau, jika aktor bukan manusia yang terlibat, dapat menggunakan persegi panjang dengan <<*actor*>> di dalamnya (alternatif). • Diberi label dengan perannya. • Dapat dikaitkan dengan aktor lain menggunakan asosiasi spesialisasi / superclass, dilambangkan dengan panah dengan panah berongga. • Ditempatkan di luar batas subjek.	 Actor/Role 
2.	*Use Case*	• Merupakan bagian utama dari fungsionalitas sistem. • Dapat memperpanjang use case lain. • Dapat menyertakan use case lain. • Ditempatkan di dalam batas sistem. • Diberi label dengan frase kata kerja deskriptif-kata benda.	

No	Nama	Deskripsi	Simbol
3.	*A Subject Boundary*	• Termasuk nama subjek di dalam atau di atas. • Merupakan ruang lingkup subjek, misalnya sistem atau individu proses bisnis.	
4.	*Assosiation relationship*	• Menghubungkan aktor dengan use case yang digunakannya untuk berinteraksi.	
5.	*Include relationship*	• Merupakan penyertaan fungsi dari satu *use case* di dalam yang lain. • Memiliki panah yang ditarik dari *use case* pangkalan ke *use case* yang digunakan.	
6.	*Extend relationship*	• Merupakan perpanjangan dari use case untuk memasukkan perilaku opsional. • Memiliki panah yang ditarik dari *use case* ekstensi ke *use case* dasar.	
7.	*Generalization relationship*	• Merupakan kasus penggunaan khusus untuk yang lebih umum. • Memiliki panah yang ditarik dari *use case* khusus ke *use case* dasar.	

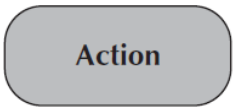
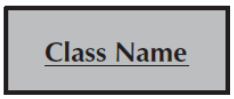
Sumber: (Dennis, Wixom, & Tegarden, 2015)

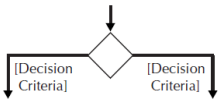
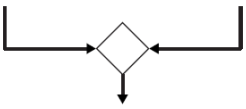
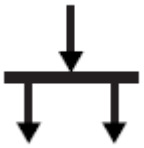
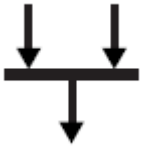
2.11.2. *Activity Diagram*

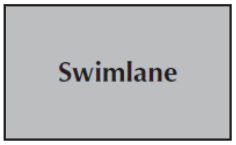
Activity diagram menunjukkan sebuah aktivitas sistem dalam bentuk kumpulan aksi-aksi, bagaimana masing-masing aksi tersebut dimulai,

keputusan yang mungkin terjadi hingga berakhirnya aksi (Suendri, 2018). *Activity diagram* mengilustrasikan alur kegiatan pada sebuah use case. Adapun simbol-simbol yang digunakan dalam *activity diagram* adalah sebagai berikut:

Tabel 2.2. Simbol *Activity Diagram*

No.	Nama	Deskripsi	Simbol
1.	*Action*	• Perilaku yang sederhana, tidak dapat diurai. • Diberi label nama.	
2.	*Activity*	• Digunakan untuk mewakili serangkaian tindakan. • Diberi label nama.	
3.	*Object Node*	• Digunakan untuk mewakili objek yang terhubung ke serangkaian aliran objek. • Diberi label berdasarkan nama kelasnya.	
4.	*Control Flow*	• Menunjukkan urutan eksekusi.	
5.	*Object Flow*	• Memperlihatkan aliran suatu objek dari satu aktivitas (atau aksi) ke aktivitas lain (atau aksi).	
6.	*Initial Node*	• Menggambarkan awal dari serangkaian tindakan atau kegiatan.	
7.	*Final Activity Node*	• Digunakan untuk menghentikan semua aliran kontrol dan objek mengalir dalam suatu kegiatan (atau tindakan).	

No.	Nama	Deskripsi	Simbol
8.	Final-flow Node	• Digunakan untuk menghentikan aliran kontrol tertentu atau aliran objek.	
9.	Decision Node	• Digunakan untuk mewakili kondisi pengujian untuk memastikan aliran kontrol atau aliran objek hanya turun satu jalur. • Diberi label dengan kriteria keputusan untuk melanjutkan ke jalur tertentu.	
10.	Merge Node	• Digunakan untuk menyatukan kembali jalur keputusan berbeda yang dibuat menggunakan simpul keputusan.	
11.	Fork Node	• Digunakan untuk membagi perilaku menjadi serangkaian kegiatan yang paralel atau bersamaan (atau tindakan)	
12.	Join Node	• Digunakan untuk menyatukan kembali serangkaian kegiatan yang paralel atau bersamaan (atau tindakan)	

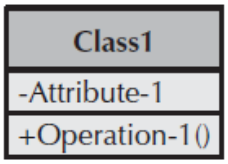
No.	Nama	Deskripsi	Simbol
13.	Swimlane	• Digunakan untuk memecah diagram aktivitas menjadi baris dan kolom untuk menetapkan aktivitas individu (atau tindakan) untuk individu atau objek yang bertanggung jawab untuk menjalankan aktivitas (atau tindakan) • Diberi label nama individu atau objek yang bertanggung jawab	

Sumber: (Dennis et al., 2015)

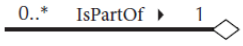
2.11.3. *Class Diagram*

Class diagram merupakan gambaran struktur sistem dari segi pendefinisian kelas- kelas yang akan dibuat untuk membangun sistem (Putra & Andriani, 2019). *Class diagram* menunjukkan kelas-kelas dalam *domain* masalah beserta relasinya. Adapun simbol-simbol yang digunakan dalam *class diagram* adalah sebagai berikut:

Tabel 2.3. Simbol *Class Diagram*

No.	Nama	Deskripsi	Simbol
1.	*Class*	• Merupakan jenis orang, tempat, atau hal tentang yang perlu ditangkap dan disimpan oleh sistem informasi. • Memiliki nama yang diketik tebal dan dipusatkan di atasnya ruang. • Memiliki daftar atribut di ruang tengahnya. • Memiliki daftar operasi di ruang bawahnya. • Tidak secara eksplisit menunjukkan operasi yang tersedia untuk semua kelas.	
2.	*Attribute*	• Merupakan properti yang menggambarkan keadaan suatu obyek. • Dapat diturunkan dari atribut lain, ditunjukkan dengan menempatkan garis miring di depan nama atribut.	attribute name /derived attribute name

No.	Nama	Deskripsi	Simbol
3.	*Operation*	• Merupakan tindakan atau fungsi suatu kelas agar dapat melakukan. • Dapat diklasifikasikan sebagai konstruktor, kueri, atau perbarui operasi. • Termasuk tanda kurung yang mungkin mengandung parameter atau informasi yang diperlukan untuk melakukan operasi.	operation name ()
4.	*Association*	• Merupakan hubungan antara banyak kelas atau kelas dan itu sendiri. • Diberi label menggunakan frasa kata kerja atau nama peran, mana yang lebih baik mewakili hubungan. • Dapat ada di antara satu kelas atau lebih. • Berisi simbol multiplisitas, yang mewakili waktu minimum dan maksimum kelas instance dapat dikaitkan dengan kelas terkait contoh.	AssociatedWith 0..* 1
5.	*Generalization*	• Merupakan sejenis hubungan antara beberapa kelas.	

No.	Nama	Deskripsi	Simbol
6.	*Aggregation*	• Merupakan hubungan logis bagian antara beberapa kelas atau satu kelas dan itu sendiri. • Merupakan bentuk khusus dari suatu asosiasi.	
7.	*Composition*	• Merupakan hubungan fisik bagian antara beberapa kelas atau satu kelas dan itu sendiri • Merupakan bentuk khusus dari suatu asosiasi.	

Sumber: (Dennis et al., 2015)

2.12. Penelitian Terdahulu

Tabel 2.4. Penelitian Terdahulu

1.	**Nama Peneliti**	Ivy Marcia (Marcia, 2019)
	Tahun	2019
	Judul	Rancang Bangun Sistem *Whatsapp Gateway* untuk Pengiriman Pesan Massal Berbasis Website dengan Proses Model Spiral (Studi Kasus: Kompas Gramedia)
	Nama Jurnal	FTI UMN 2019
	Hasil	*WhatsApp Gateway* dibangun dikarenakan terdapat beberapa kendala yang dirasakan baik oleh pihak Kompas Gramedia maupun *user* terkait SMS Gateway. *WhatsApp Gateway* dibuat berbasis website dengan menggunakan framework CodeIgniter3, sedangkan pengirim pesan berbasis Python3 yang dijalankan pada Linux. Sistem dibangun dengan menggunakan proses model spiral dan terdapat enam fitur yang berhasil dikembangkan, yaitu Pesan untuk pengiriman pesan, *Master Data* sebagai pengatur seluruh data pada website, Alamat untuk penyimpanan kontak pengguna, Akun untuk sistem login dan penentuan jenis pengguna, Top-up untuk kredit dan status pembayaran pesan, dan Ticketing untuk pengiriman dan pelaporan permasalahan.

2.	**Nama Peneliti**	Tiara Rizki Wulansari, Woro Isti Rahayu, Noviana Riza (Tiara Rizki Wulansari, Woro Isti Rahayu, 2019)
	Tahun	2019
	Judul	Aplikasi Pemesanan Bahan Bakar Minyak Melalui Media WhatsApp Menggunakan Algoritma *WhatsApp Gateway* (Studi Kasus: PT. Pertamina Patra Niaga)
	Nama Jurnal	FTI Politeknik Pos Indonesia 2019
	Hasil	Pada penelitian ini, berisikan tentang *WhatsApp Gateway* yang diterapkan pada pesan masuk aplikasi pemesanan bahan bakar minyak. Dengan fitur-fitur yang digunakannya, seperti yang pertama-tama adalah fitur permintaan bahan bakar minyak, lalu setelah permintaan diproses, selanjutnya pengiriman bahan bakar minyak. Dari semua proses itu, setiap kegiatan yang dilakukan dalam aplikasi itu, dari mulai pemesanan hingga pengiriman, itu mendapatkan sebuah pesan masuk melalui *WhatsApp*, jikalau pemesanan yang dibuat oleh pengguna telah ditindaklanjuti.
3.	**Nama Peneliti**	Yohannes Kurniawan, Janastasha Christie Parapaga (Kurniawan & Parapaga, 2014)
	Tahun	2014
	Judul	Pengembangan Sistem Informasi Siklus Pendapatan pada PT XYZ(Pendekatan Studi Kasus).
	Nama Jurnal	ULTIMA InfoSys
	Hasil	Sistem ini dirancang untuk memberikan informasi pembayaran angsuran yang telah dibayarkan oleh pelanggan. Sistem ini dibangun karena adanya pelanggan yang langsung membayarkan angsuran dengan melakukan setoran dan informasi pembayaran tidak dicatat langsung oleh bagian finance. Sistem dapat menghasilkan jenis laporan yang lebih beragam mengenai siklus pendapatan. Laporan yang dihasilkan tersebut membantu perusahaan dalam membuat rencana strategis untuk kegiatan perusahaan selanjutnya.

4.	**Nama Peneliti**	Eko Harli, Ahmad Fauzi (Fauzi & Harli, 2017)
	Tahun	2017
	Judul	Rancang Bangun Sistem Informasi Akademik berbasis SMS Gateway dengan Metode Rapid Application Development
	Nama Jurnal	Urecol
	Hasil	Metode Rapid Application Development dalam penggunaannya memberikan fleksibilitas pada saat merancang karena tidak terpaku pada proses saja.

Berdasarkan penelitian terdahulu pada tabel 2.4, terdapat 4 jurnal yang dijadikan referensi dalam studi literatur ini.

Mengacu pada jurnal nomor 1, *WhatsApp Gateway* berbasis *website* itu digunakan untuk berbagai fitur sesuai dengan kebutuhan, yang menggunakan metode *framework CodeIgniter 3* dalam membangun sistem tersebut.

Mengacu pada jurnal nomor 2, membahas perihal pesan masuk ke *whatsapp* yang merekam semua kegiatan mulai dari pemesanan sampai dengan pengiriman.

Mengacu pada jurnal nomor 3, sistem yang dibuat pada jurnal tersebut mengacu pada topik yang memberikan informasi tagihan kepada pelanggan dalam pembayaran angsuran.

Mengacu pada jurnal nomor 4, membahas penggunaan metode RAD yang cukup baik dalam pengembangan sistem dikarenakan *requirements planning* yang ada pada awal itu dapat sesuai dan jika memang ada perubahan pada sistemnya, metode ini dapat membuat *requirements* baru.