BAB II

TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Penelitian yang dilakukan oleh penulis menggunakan beberapa tinjuaan studi yang digunakan sebagai landasan teori serta pembanding dengan penelitan yang dilakukan. Tinjauan studi dalam penelitan ini akan membandingkan dari berbagai sumber. Dalam hal ini, penelitian terdahulu yang dijadikan acuan adalah terkait dengan masalah presensi perkuliahan. Oleh karena itu, peneliti melakukan langkah kajian terhadap beberapa penelitian berupa tesis dan jurnal-jurnal melalui internet. Perbandingan akan berfokus pada teknologi yang dipakai, arsitektur sistem dan metode pengembangan sistem.

Berbagai penerapan teknologi untuk melakukan presensi yang banyak dipakai pada penelitan terdahulu yaitu RFID. Tinjauan studi yang dilakukan menemukan empat penelitian menggunakan teknologi RFID untuk melakukan presensi. Penelitian-penelitian tersebut diantaranya dilakukan oleh Tora Fahrudin (2011), Pratomo dan Prasetyo (2015), Setiawan E.B. Dkk (2015) dan Hasan A. Dkk. (2016). Berdasarkan keempat penelitian tersebut, penerapan teknologi RFID hanya dilakukan di satu Program Studi atau penerapan dalam sekala kecil. Sedangkan presensi perkuliahan pada penelitian ini nantinya akan diimplementasikan di seluruh Program Studi UMY atau dalam sekala besar. Jika penerapan teknologi RFID dipakai dalam penelitian ini maka biaya yang dikeluarkan untuk pembelian mesin *scanner* RFID pada setiap kelas membutuhkan biaya yang besar. Tentu peneliti mempertimbangkan untuk menggunakan RFID

sebagai teknologi untuk melakukan presensi perkuliahaan pada penelitian ini. Teknologi untuk melakukan presensi perkuliahan yang dapat diterapkan dalam skala besar yang membutuhkan biaya kecil yaitu menggunakan teknologi *token*. Teknologi token yang diterapkan menggunakan *smartphone* milik pengguna atau mahasiswa sebagai media untuk melakukan presensi.

Sistem presensi menggunakan teknologi token telah diterapkan di UMY oleh Krisna Nuresa(2017). Penerapan teknologi token pada penelitian tersebut dapat berjalan dengan baik dan dapat digunakan untuk melakukan presensi perkuliahan di UMY. Dengan mempertimbangkan berbagai hal, penelitian ini tetap akan menggunakan teknologi token sebagai teknologi untuk melakukan presensi perkuliahan di UMY. Perbedaan penelitian ini dengan penelitian yang telah dilakukan dalam penelitian tersebut terkait teknologi presensi berbasis token di UMY sebagai berikut:

1. Penelitian ini menggunakan arsitektur *three-tier* sedangkan pada sistem presensi berbasis *Android* masih menggunakan arsitektur two-tier.
2. Penelitian ini menambahkan fitur daftar presensi perkuliahan yang telah dilakukan mahasiswa di sistem presensi ini. Sedangkan pada penelitian sebelumnya belum terdapat fitur tersebut.

Tinjauan studi ini juga dilakukan untuk menunjukan bahwa arsitektur yang dipakai pada sistem presensi di penelitian-penelitian terdahulu masih menggunakan arsitektur *two-tier*. Terdapat dua penelitian yang menggunakan arsitektur two tier yaitu pada penelitian yang dilakukan oleh Febriliyan Samopa.Dkk(2013) dan Muhammad Yusuf. Dkk(2016). Kedua penelitian tersebut masih menggunakan

arsitektur *two-tier* karena aplikasi yang digunakan pengguna untuk melakukan presensi masih terhubung langsung dengan database tanpa adanya perantara. Faktor keamanan database di kedua penelitian tersebut belum begitu diperhatikan dengan diterapkannya arsitektur *two-tier* karena berisiko jika akun database berada pada aplikasi pengguna. Sedangkan pada penelitian ini akan memperhatikan faktor keamanan database, sehingga pada penelitian ini menerapkan arsitektur *three-tier* supaya keamanan database dapat ditingkatkan.

Tinjauan studi ini juga dilakukan untuk mendapatkan metode pengembangan sistem yang baik. Dari beberapa metode pengembangan sistem yang telah dilakukan dalam penelitian yang dilakukan oleh Pratomo dan Prasetyo (2015) dan Norhikmah. Dkk (2016), metode pengembangan sistem menggunakan SDLC dengan model *waterfall* dapat memperoleh hasil yang baik. Sehingga dalam penelitian ini menggunakan SDLC dengan model *Waterfall* sebagai metode pengembangan sistem.

Adanya persamaan dan perbedaan yang terdapat dalam tesis ini dengan penelitian sebelumnya terkait sistem presensi tentu membawa konsekuensi pada hasil penelitian yang diperolehnya. Diharapkan penelitian ini mendapatkan hasil yang dapat mengatasi permasalahan sistem presensi di UMY.

2.2 **Landasan Teori**

Dalam pengembangan sistem presensi berbasis *iOS* dibutuhkan beberapa teori yang akan menjadi acuan dalam penelitian ini. Beberapa teori tersebut akan dijelaskan pada bab ini diantaranya Aplikasi Mobile , iOS beserta Arsitekturnya dan juga metode pengembangan sistem.

2.2.1 **Pengelolaan Data Presensi**

Pengolahan data presensi merupakan proses kegiatan pencatatan yang dilakukan terhadap setiap presensi dengan tujuan untuk mengetahui laporan berkaitan dengan data presensi. Operasi yang dilakukan di dalam sistem pengolahan data presensi antara lain: pencatatan, pemeriksaan, penggolongan, penyusunan, dan penyimpanan (Kristanto, 2008). Proses ini penting karena dapat mempengaruhi hasil laporan presensi. Oleh karena itu didalam pengolahan data presensi diupayakan untuk seminimal mungkin terjadinya kesalahan.

2.2.2 ***Security Token***

Menurut Jones. M (2014), *Security Token* merupakan sebuah kriptografi terenkripsi berupa *keyword* tentang sebuah subjek yang digunakan untuk memeriksa kesesuaian atau kebenaran tentang subjek tersebut.

Karakter didalam *token* dikategorikan sebagai salah satu dari lima kelas *token* yang menggambarkan fungsi mereka (*identifier, keyword, literal string, operator, separator*) sesuai dengan aturan bahasa pemrograman.

Menurut Oktafia (2008), token merepresentasikan nama:

1. *Identifier* (nama variabel, fungsi dan tipe atau nama yang didefinisikan pemakai).
2. *Keyword*
3. *Literal String*
4. *Operator*
5. *Label*
6. Simbol tanda (tanda kurung, koma dan titik koma).

Dalam penelitian ini jenis karakter *token* yang digunakan hanya menggunakan beberapa karakter dalam penerapannya, yaitu:

1. Angka (1, 2, 3, 4, 5, 6, 7, 8, 9, 0)
2. Huruf besar (A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z)
3. Huruf kecil (a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z)

Menurut Wicaksono, A. (2016), Token ini terdiri dari delapan digit *random keyword*. Proses pembuatan *token* selesai setelah pengguna (dosen) membuat sesi perkuliahan. Jika proses pembuatan *token* sama dengan *token* pada sisi perkuliahaan sebelumnya. Sistem akan melakukan pembuatan *token* baru sampai *token* tersebut tidak sama dengan sesi perkuliahaan sebelumnya atau dapat dikatakan unik. Setelah sistem mendapatkan *token* yang unik, *token* akan diperlihatkan oleh sistem di halaman sesi perkuliahan ynag telah dibuat oleh pengguna(dosen). Sehingga *token* tersebut nantinya akan digunakan oleh mahasiswa untuk melakukan presensi.

2.2.3 **Aplikasi *Mobile***

Aplikasi *Mobile* adalah perangkat lunak yang berjalan pada perangkat *mobile* seperti *smartphone* atau *tablet* PC. *Mobile Application* juga dikenal sebagai aplikasi yang dapat diunduh dan memiliki fungsi tertentu sehingga menambah fungsionalitas dari perangkat *mobile* itu sendiri. (*Mobile Marketing Association*, 2008).

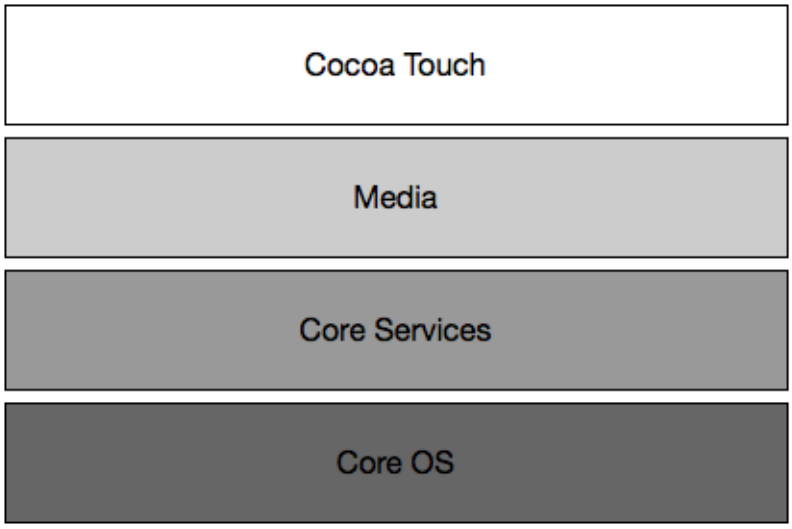
2.2.4 *iOS*

iOS adalah sistem operasi yang dikembangkan dan didistribusikan oleh perusahaan Apple *Inc.* untuk ponsel iPhone. Saat ini *iOS* telah berkembang dan dapat digunakan ke dalam perangkat Apple yang lainnya seperti iPod Touch, Apple TV dan iPad. Sistem operasi ini bersifat tertutup dan hanya bisa digunakan oleh perangkat Apple. Karena tidak menemukan sistem operasi *iOS* pada perangkat serupa dengan merek lain. Didalam *iOS* juga terdapat komponen *abstraction layers*, yaitu lapisan sistem *iOS* yang terbagi menjadi empat bagian, seperti *framework* yang berfungsi untuk interaksi pengguna ke *hardware* (Adelphia, 2015)

2.2.5 *iOS Architecture*

Arsitektur *iOS* mirip dengan arsitektur dasar yang ditemukan di *Mac OS X*. Pada tingkat tertinggi *iOS* bertindak sebagai perantara antara hardware yang mendasari dan aplikasi yang muncul di layar. Aplikasi untuk berkomunikasi dengan perangkat keras melalui serangkaian *interface* sistem yang jelas melindungi aplikasi dari perubahan *hardware*. Abstraksi ini membuat mudah untuk membuat aplikasi yang bekerja secara konsisten pada perangkat dengan kemampuan *hardware* yang berbeda (Apple, 2014).

Arsitektur teknologi *iOS* dapat dilihat dalam beberapa lapisan , seperti yang terlihat pada Gambar 2.1. Pada lapisan tertatas sistem ini adalah aplikasi yang menghubungkan *user* dengan perangkat.



Gambar 2.1 Lapisan dari Arsitektur *iOS* (Apple, 2014)

Layer Core OS dan *Core Services* yang berisi *interface* dasar untuk *iOS*, termasuk yang di gunakan untuk mengakses file, tipe data tingkat rendah, layanan bonjour, soket *network*, dan sebagainya. *Interface-interface* ini sebagian besar adalah C-based, dan memasukkan teknologi-teknologi seperti *Core Foundation*, *CF Network*, *SQLite*, dan akses ke thread *POSIX* dan soket *Unix* dengan yang lain (Apple, 2014).

Pada lapisan *Touch Cocoa*, sebagian besar penggunaan teknologi *Objective-C*. Kerangka kerja di lapisan ini menyediakan infrastruktur dasar yang digunakan oleh aplikasi. Sebagai contoh, kerangka Foundation memberikan dukungan berorientasi objek untuk koleksi, manajemen file, operasi jaringan, dan banyak lagi. Kerangka UIKit menyediakan infrastruktur visual untuk aplikasi, termasuk kelas untuk windows, view, kontrol, dan pengendali yang mengelola objek tersebut. Kerangka kerja lain pada tingkat ini memberikan Anda akses untuk menghubungi pengguna dan informasi foto dan ke *accelerometers* dan fitur perangkat keras lainnya dari perangkat (Apple, 2014).

2.2.6 *iOS App Store*

iOS App Store adalah toko aplikasi dimana pengguna dapat menelusuri dan mengunduh aplikasi *mobile* pilihan mereka sendiri pada *iOS*, yang dikembangkan dengan *Apple iOS SDK*. *iOS App Store* juga dapat dikatakan sebagai salah satu tempat untuk mendistribusikan aplikasi untuk sistem operasi iOS. Aplikasi gratis atau dibayar tergantung pada penerbit dari aplikasi yang tersedia di *app store*. Banyak juga aplikasi memiliki status *in-app purchase* berarti pengguna harus membayar jumlah beberapa untuk mendapatkan akses ke fitur tertentu dari aplikasi (Shivam Jaiswal dan Ajay Kumar, 2014).

2.2.7 **Bahasa Pemrograman *Swift***

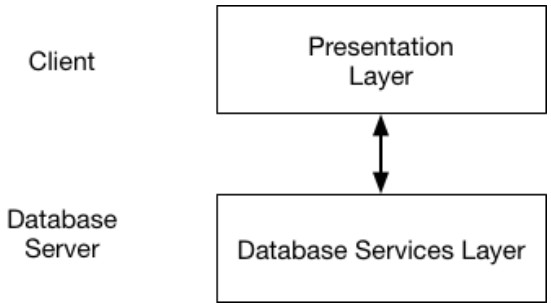
Swift digunakan dalam pembuatan Aplikasi untuk *iOS* dan *macOS.* Menurut Mikey Ward dan Robert Edwards (2017), *Swift* merupakan bahasa pemrograman baru yang di rilis pada tahun 2014 sebagai pengganti Bahasa pemrograman *Objective-C*. Saat ini bahasa pemrograman yang direkomendasikan oleh *Apple Inc.* adalah *Swift*, yang dipasarkan sebagai bahasa pemrograman yang kokoh dan intuitif. Bahasa ini didesain untuk memberikan lebih banyak kebebasan kepada pengembang. *Swift* mudah digunakan dan bersifat sumber terbuka (*Open Source*).

2.2.8 ***Xcode***

Xcode adalah *integrated development environment* untuk *macOS* yang berisi *software development tools* yang dikembangkan oleh *Apple Inc.*, untuk mengembangkan aplikasi untuk *macOS*, *iOS*, *watchOS* dan *tvOS* (Apple, 2017).

2.2.9 **Arsitektur *Two-tier***

Oluwatosin (2014) Arsitektur *Two-Tier* terdiri dari dua layer yaitu *Presentation Layer* dan *Database Services Layer*. Dalam Arsitektur *two-tier*, aplikasi pengguna terhubung langsung ke *database server* tanpa melibatkan perantara apapun. Arsitektur *Two-tier* dapat dilihat pada gambar 2.2.



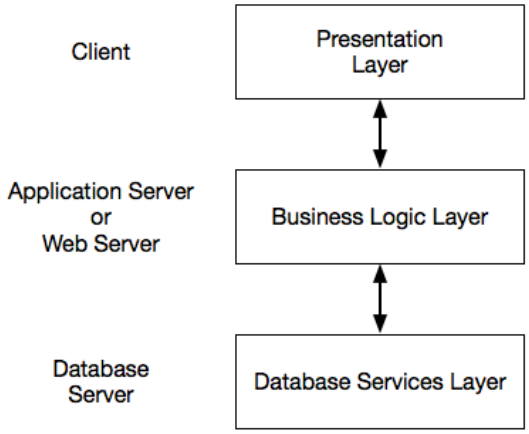
Gambar 2. 2 *Two-tier Architecture* (Oluwatosin, 2014).

2.2.10 **Arsitektur *Three-tier***

Arsitektur *Three-Tier* merupakan arsitektur *client-server* yang menambahkan *application server* atau *web server* antara *client* dengan *database server*. *Application Server* atau *web server* berperan sebagai perantara dengan aplikasi yang dijalankan dan menyimpan aturan bisnis (prosedur atau batasan) yang digunakan untuk mengakses data dari *database server*. Perantara tersebut dapat meningkatkan keamanan *database* dengan memeriksa kredensial pengguna sebelum meneruskan permintaan ke *database server* (Oluwatosin, 2014).

Menurut Gunadi . K, (2001) Arsitektur *Three-Tier* memiliki keunggulan dibandingkan dengan Arsitektur *Two-Tier* adalah sebagai berikut:

1. Menyediakan kemampuan perluasan yang lebih besar.
2. Meningkatkan keamanan.
3. Perawatan yang lebih rendah dan mudah.



Gambar 2.3 *Three-tier Architecture* (Oluwatosin, 2014).

2.2.11 **RESTful Web Service**

Menurut W3C (2014), *Web service* adalah sistem yang di desain untuk mendukung interaksi dari mesin-ke-mesin melalui jaringan. *Web service* merupakan aplikasi sekumpulan data (*database*), perangkat lunak (*software*) atau bagian dari perangkat lunak yang dapat diakses secara remote oleh berbagai piranti dengan sebuah perantara tertentu.

REST(Representational State Transfer) merupakan sebuah teknik di arsitektur *software* untuk sistem terdistribusi seperti *World Web Wide. RESTful web service* atau juga dikenal dengan nama *RESTful Web API* merupakan sebuah *web service* yang di implemantasikan dengan menggunakan *http* dengan menggunakan prinsip-prinsip *REST*.

2.2.12 **Unified Modeling Language (UML)**

Menurut Yulianto, A.A. Dkk. (2009), *UML* merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram

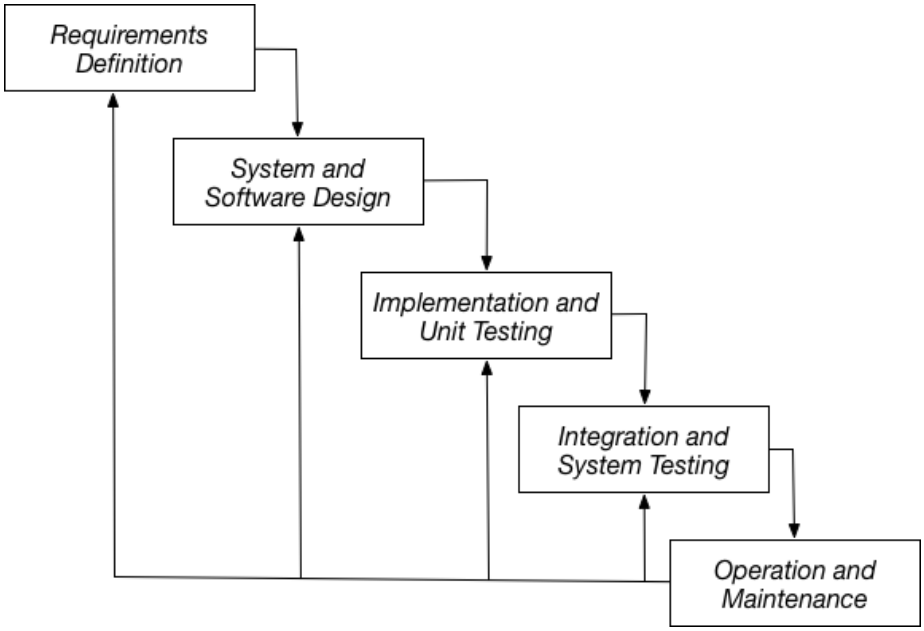
dan teks-teks pendukung. *UML* juga merupakan standarisasi bahasa permodelan untuk membangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek.

2.2.13 **Software Development Life Cycle (SDLC)**

System Development Life Cycle (SDLC) merupakan siklus pengembangan sistem untuk membangun sistem informasi. SDLC dimaksudkan untuk pengembangan sistem informasi dalam cara yang terstruktur dan metodis, yang mengharuskan tahap *life cycle* dari mulai awal hingga final sistem dilaksanakan secara beraturan. Salah satu metode SDLC yang paling banyak digunakan adalah *Waterfall method* (Govardhan .dkk ,2010).

Waterfall method sering dianggap sebagai pendekatan klasik dengan siklus hidup pengembangan sistem. Pembangunan dengan metode *Waterfall* memiliki tujuan yang berbeda untuk setiap fase pembangunan. Setelah fase pembangunan selesai, hasil pengembangan ke tahap berikutnya dan tidak ada jalan kembali.

Tahapan-tahapan pada metode *Waterfall* dapat dilihat pada Gambar 2.4,



Gambar 2. ***4*** *Waterfall* Model menurut Govardhan .dkk (2010)

1. *Requirements Definition*

Pada tahap ini berlangsung proses pengumpulan kebutuhan secara lengkap untuk dianalisis dan didefinisikan kebutuhan apa saja yang harus dipenuhi oleh sistem yang akan dibuat. Seluruh kebutuhan *software* harus bisa didapatkan dalam fase ini, termasuk didalamnya kegunaan *software* yang diharapkan pengguna dan batasan *software*. Informasi ini biasanya dapat diperoleh melalui wawancara, *survey* atau diskusi. Informasi tersebut dianalisis untuk mendapatkan dokumentasi kebutuhan pengguna untuk digunakan pada tahap selanjutnya.

2. *System and Software Design*

Software Design adalah proses dimana analisa diterjemahkan menjadi "*blueprint*" untuk membangun perangkat lunak. Tahap ini bertujuan untuk memberikan gambaran apa yang seharusnya dikerjakan dan bagaimana tampilannya (Pressman 2010, p219). Tahap ini membantu dalam

menspesifikasikan kebutuhan perangkat keras dan sistem serta mendefinisikan arsitektur sistem secara keseluruhan.

Tahap ini dilakukan sebelum melakukan coding. Tahap ini bertujuan untuk memberikan gambaran apa yang seharusnya dikerjakan dan bagaimana tampilannya. Tahap ini membantu dalam menspesifikasikan kebutuhan sistem dan perangkat keras yang digunakan, serta mendefinisikan arsitektur sistem secara keseluruhan.

3. *Implementation*

Dalam tahap ini dilakukan pemrograman. Pengembangan aplikasi dibagi menjadi modul-modul kecil yang nantinya akan digabungkan dalam tahap berikutnya. Selain itu dalam tahap ini juga dilakukan pemeriksaaan terhadap modul yang dibuat, apakah sudah memenuhi fungsi yang diinginkan atau belum.

4. *Integration* & *Testing*

Di tahap ini dilakukan penggabungan modul-modul yang sudah dibuat dan dilakukan pengujian ini dilakukan untuk mengetahui apakah sistem yang dibuat telah sesuai dengan desainnya dan masih terdapat kesalahan atau tidak.

5. *Operation* & *Maintenance*

Ini merupakan tahap terakhir dalam model waterfall. Software yang sudah jadi dijalankan serta dilakukan pemeliharaan. Pemeliharaan termasuk dalam memperbaiki kesalahan yang tidak ditemukan pada langkah sebelumnya. Perbaikan implementasi unit sistem dan peningkatan jasa sistem sebagai kebutuhan baru.

2.2.14 **Pengujian Perangkat Lunak**

Dalam pengujian perangkat lunak ini penulis menggunakan *Black-Box Testing* dan *Real Testing* sebagai metode pengujian. Pengujian perangkat lunak sangat diperlukan untuk memastikan *software*/aplikasi yang sudah/sedang dibuat dapat berjalan sesuai dengan fungsionalitas yang diharapkan. Menurut Khan (2011), *Black-Box Testing* merupakan pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak, tester dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program.