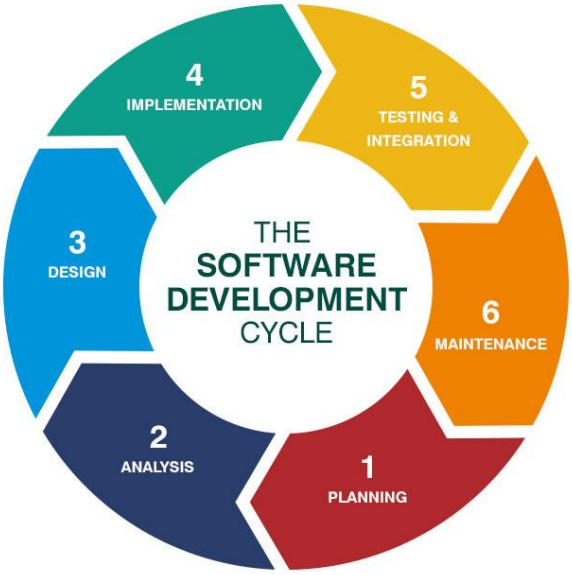
BAB II
LANDASAN TEORI

A. *System Development Life Cycle* (SDLC)

System Development Life Cycle (SDLC) merupakan siklus atau tahapan proses pengembangan sistem perangkat lunak. Siklus pengembangan ini bersifat sistematis sehingga pelaksanaanya dilakukan secara berurutan. Fungsi utama dari SDLC sendiri yaitu untuk mengakomodasi kebutuhan pengguna terhadap sistem yang akan dikembangkan dan menjadi panduan dalam melakukan pembangunan sistem perangkat lunak. Berikut merupakan siklus hidup pengembangan sistem perangkat lunak[6][7].



Gambar 2. 1 System Development Life Cycle (SDLC)[6]

Seperti gambar 2.1 siklus pengembangan sistem perangkat lunak terdiri dari 6 tahapan yang dapat dilakukan secara berulang ketika sistem yang telah dibuat akan dilakukan pengembangan atau pembaharuan. Berikut penjelasan dari tahapan langkah SDLC sebagai berikut:

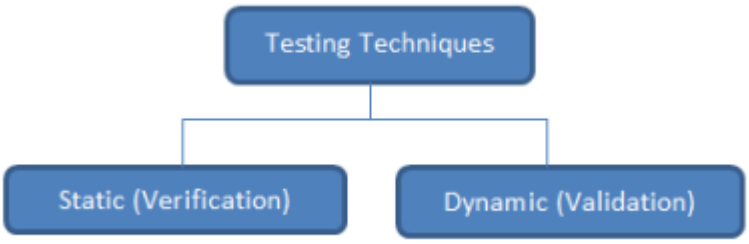
1. *Planning*, merupakan proses perencanaan pembangunan perangkat lunak, dimulai dengan mendeskripsikan sistem seperti apa yang ingin dibuat, target atau sasaran pengguna, jangka waktu pembuatan sistem, dana dan siapa yang terlibat dalam pembangunan sistem tersebut.
2. *Analysis*, merupakan proses menganalisis dan mengidentifikasi sistem serupa atau sistem yang sudah ada dengan tujuan untuk membuat sistem baru atau merancang pembaruan terhadap sistem yang sudah ada.
3. *Design*, merupakan proses penentuan data dan *tools* yang digunakan untuk membangun suatu sistem.
4. *Implementation*, merupakan proses implementasi dari rencana konseptual untuk menghasilkan sebuah sistem. Pada tahapan ini dilakukan dengan melakukan pembuatan kode program.
5. *Testing dan Integration*, merupakan proses untuk menganalisis dan menguji fungsionalitas sistem.
6. *Maintanance*, merupakan proses untuk mengevaluasi dan memelihara sejauh sistem telah dibangun dan dioperasikan sehingga dapat diketahui apabila terdapat kerusakan, diperlukan perubahan atau penambahan fitur baru terhadap sebuah sistem.

B. *Software Testing*

Software Testing atau pengujian *software* merupakan proses peng-aplikasian atau menjalanakan sebuah sistem oleh seorang *software tester* dengan tujuan untuk menghasilkan kasus uji dan mendeteksi *bug* atau kesalahan pada sistem perangkat lunak. Pengujian perangkat lunak dapat dilakukan secara manual atau automasi. Pengujian secara manual dilakukan tanpa menggunakan alat/*tools*, ini merupakan cara tradisional uji sistem namun setiap *software* harus melalui proses manual *testing* sebelum bisa dilakukan automasi *testing*. Manual *testing* dilakukan dengan menyiapkan kasus uji/*test case* kemudian mengeksekusinya untuk mengidentifikasi cacat pada perangkat lunak. Sedangkan pengujian automasi dilakukan menggunakan bahasa *scripting* seperti Python, JavaScript atau *Tool Command Language*[8][9].

C. Software Testing Metodology

Pengujian perangkat lunak dilakukan setelah mengembangan kode, pengujian ini wajib dilakukan untuk mengidentifikasi kesalahan dan *de-buging* sebelum rilis perangkat lunak. Meskipun tidak mungkin untuk mengidentifikasi dan men-*debug* semua kesalahan dalam perangkat lunak signifikan pada setiap fase, semua kesalahan dicoba untuk dihilangkan sebanyak mungkin. Berikut merupakan kategori teknik pengujian yang terbagi dalam 2 bagian[8]:



Gambar 2. 2 Testing Teqniques[8]

1. *Static Testing*, pengujian statis dilakukan sebelum kode program dieksekusi atau dibuat. Pengujian statis ini dikenal juga sebagai pengujian verifikasi yang tujuan utamanya adalah untuk meningkatkan kualitas produk perangkat lunak dengan membantu para profesional perangkat lunak untuk mengidentifikasi dan mengatasi kesalahan mereka diawal proses pengembangan perangkat lunak dengan mengkaji kode program, dokumen dan *file* pengembangan perangkat lunak.
2. *Dynamic Testing*, pengujian dinamis dilakuakn sedang/saat kode program sudah dieksekusi. Pengujian dinamis dikenal juga sebagai pengujian validasi dimana perilaku dinamis dari kode program akan dianalisis dengan cara memasukan input kedalam program dan membandingkan output akhir dengan hasil output yang diharapkan. Pengujian dinamis terbagi lagi menjadi 2 kategori yaitu *Blackbox Testing* dan *Whitebox Testing*. *Black-box Testing* dikenal dengan *Behavioral* atau *Input - Output Testing* yaitu dengan memasukan *input* dan melihat *output* dari sistem

yang dijalankan. Sedangkan *White-box Testing* adalah proses testing yang berfokus menguji struktur kode pada aplikasi.

D. Test Case

Test case merupakan kasus uji atau sekumpulan prosedur *test* yang memuat kondisi operasi yang dibutuhkan untuk melakukan pengujian sistem bersamaan dengan hasil yang diharapkan dari proses operasi tersebut. Seorang *software tester* diharapkan dapat membuat *test case* sebagai panduan untuk mendeteksi sistem dari kegagalan dan menuliskan hasil uji. Dalam hal pembuatan *test case* tidak ada *template* khusus yang secara spesifik harus diikuti namun biasanya terdiri dari kompen utama seperti *section* modul yang berisi nama uji kasus atau *test id* yang berisi kode uji kasus, *section precondition* berisi kondisi sebelum melakukan uji kasus, *section test step* berisi langkah uji kasus dan *section result* berisi hasil dari uji kasus. *Test Case* dapat dibuat menggunakan *tool Microsoft Excel* ataupun *google spreadsheet*[10].

E. *System Point of Sales* (POS)

System Point of Sales (POS) merupakan sistem aplikasi yang digunakan pada bisnis minimarket ataupun pertokoan untuk menangani pengolahan data transaksi pembelian, penjualan, stok dan laporan transaksi, yang mana hal tersebut secara umum dibutuhkan dalam pengambilan keputusan strategic oleh pebisnis atau petinggi perusahaan. Selain itu dengan dibuat sistem POS dapat mempermudah proses transaksi dan mempercepat operasional bisnis. Setiap sistem POS yang siap dipasarkan biasanya terdiri dari *hardware* dan *software* yang terintegrasi sehingga memudahkan *client* untuk menggunakannya. *Hardware* biasanya berupa terminal/PC, *receipt printer, cash drawer*, terminal pembayaran, *Barcode scanner* dan *software* berupa *inventory management*, pelaporan, *purchasing, customer management*, dan *return processing*[11].