BAB II

LANDASAN TEORI

2.1. Tinjauan jurnal

Menurut Sugiman (2014:1) “Pengetahuan masyarakat akan rambu lalu lintas sangat lah minim sehingga perlu adanya panduan tentang berlalu lintas”. seiring semakin banyaknya pengguna smartphone berbasis android, maka dapat dimanfaatkan untuk memperkenalkan rambu lalu lintas dengan cara mengembangkan aplikasi android yang memiliki fitur tentang lalu lintas. Dengan adanya media pada proses belajar mengajar, diharapkan dapat membantu guru dalam meningkatkan prestasi belajar pada siswa.

Media pembelajaran selalu mengalami perkembangan seiring perkembangan teknologi. Sebagai salah satu contohnya penggunaan media pembelajaran interaktif untuk proses pembelajaran, salah satunya adalah pembelajaran tentang pengenalan rambu-rambu lalu lintas.

Menurut Kusrianto (2007:1) “ Media Interaktif adalah cara seseorang menyajikan penjelasan terhadap data, uraian proses, maupun pemblajaran, baik disajikan dimuka *audience* dengan bantuan alat peraga berupa slide show, program aplikasi yang menyajikan informasi interaktif yang dapat diakses secara personal,

maupun presentasi dalam bentuk cetakan yang dibagikan kepada semua penerima informasi".

2.2. Konsep Dasar Program

Menurut Hariyanto (2014:12) "Pemrograman berorientasi objek adalah cara ampuh dalam pengorganisasian dan pengembangan perangkat lunak". Pada orientasi objek, program sebagai sekelompok objek yang saling berinteraksi. Objek-objek ini ada secara independen, mempunyai aturan-aturan berkomunikasi dengan objek lain guna meminta informasi tertentu atau meminta objek lain mengerjakan sesuatu.

Perkembangan gaya pemrograman mencapai gaya pemrograman orientasi objek setelah era pemrograman terstruktur. Pemrograman orientasi objek menggantikan pemrograman terstruktur karena mempunyai banyak keunggulan dalam menangani proyek yang luar biasa kompleks. Pemrograman menggunakan bahasa orientasi objek menawarkan fleksibilitas, dan kemudahan perawatan.

Pembuatan program tentunya tidak lepas dari tahapan-tahapan yang harus dikerjakan secara tersetruktur, untuk membantu pemrogram dalam menyelesaikan programnya dengan baik, untuk lebih jelas nya tahapan-tahapan perancangan program secara umum adalah sebagai berikut:

1. Definisi Masalah

   Tujuanya untuk mendapatkan pemahaman tentang permasalahan yang ada, sehingga akan diperoleh asumsi-asumsi yang benar untuk dapat memecahkan permasalahan.

2. Analisis Kebutuhan

   Langkah ini dilakukan untuk menentukan masukan dan keluaran yang diinginkan serta sebagai gambaran tentang data yang akan diproses sehingga program yang disusun terarah dan menghasilkan informasi yang dibutuhkan.

3. Definisi Algoritma

   Menulis langkah-langkah dalam pemecahan masalah yang ada dengan mengggunakan simbol-simbol untuk menceritakan aktifitas data yang akan diolah menjadi suatu informasi.

4. Pengkodean

   Merupakan pengkodean dari algoritma yang dibuat, diterjemahkan kedalam bentuk *statement* yang sesuai dengan bahasa pemograman yang digunakan.

5. *Testing* Program

   Dari proses yang sudah dibuat, diperiksa apakah program tersebut sudah benar dan bebas dari kesalahan atau masih harus diperbaiki kembali, semua kesalahan yang terjadi deperbaiki agar program komputer dapat dijalankan dan memberi hasil sesuai yang diharapkan.

6. Dokumentasi Program

   Membuat pendokumentasian program sebagai cadangan (*backup*) yang mana proses ini penting untuk dilakukan,untuk usaha pengembangan program selanjutnya.

7. Pemeliharaan

   Pemeliharaan digunakan untuk menjabarkan aktivitas dari analisis sistem pada saat perangkat telah dipergunakan oleh pemakai.

2.2.1. *Android*

1. Mengenal *Android*

Menurut Murya (2014:3) "*Android* adalah sistem operasi berbasis *linux* yang di gunakan untuk telepon seluler *(mobile)* seperti smartphone dan komputer tablet *(PDA)*." *Android* menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri yang digunakan oleh bermacam piranti bergerak. *Android* kini telah menjelma menjadi sistem operasi *mobile* terpopuler di dunia. Perkembangan *Android* tidak lepas dari peran sang raksasa *Google*. *Android* pada mulanya didirikan oleh Andry Rubin, Rich Miner, Nick Sears Dan Chris White pada tahun 2003. Dan pencipta *Android* adalah andy rubin namun telah di ambil alih oleh *Google*.

Pada tahun 2005 Andy Rubin dan larry page melakukan pertemuan di kantor pusat *Google*. Pertemuan tersebut bukanlah pertemuan pertama. Mereka telah berjumpa tiga tahun sebelumnya, ketika Andy Rubin akan merilis *smartphone* yang dibuat dan diberi nama *sidekick* yang memakai mesin pencari *Default Google.*

Pada bulan november 2007, *Google* mengadakan konferensi pers. Ternyata tidak ada *Gphone (google phone)* diluncurkan seperti yang banyak di perkirakan orang. Waktu itu diumumkan terbentuknya *Open Handset Alliace* yang didirikan *Google* bersama 34 perusahaan lain, untuk mengembangkan *OS open source*.

2. Versi Android

a) *Android Versi* 1.0

b) *Android Versi* 1.1

c) *Android Versi* 1.5 *(Cupcake)*

d) *Android Versi* 1.6 *(Donut)*

e) *Android Versi* 2.1 *(Eclair)*

f) *Android Versi* 2.2 *(Froyo* atau *Frozen Yogurt)*

g) *Android Versi* 2.3 *(Gingerbread)*

h) *Android Versi* 3.0-3.1 *(Honeycome)*

i) *Android Versi* 4.0 *(Ice Cream Sandwich)*

j) *Android Versi* 4.1-4.2 *(Jelly Bean)*

k) *Android Versi* 4.4 *(Kitkat)*

l) *nAdroid Versi 4.5 (Lolipop)*

3. Fitur *Android*

a) Fitur yang tersedia di *Android* adalah :

b) Kerangka aplikasi

c) *Delvik Mesin Virtual*

d) *Grafik*

e) *SQLite*

f) *GSM, bluetooth, EDGE, 3G, 4G* dan *Wifi*

g) Kamera, *GPS*, Kompas, *NFC*, dan *Accelelometer*

4. Resourch

Resourch digunakan untuk menyimpan *file-file non-coding* yang diperlukan Pada sebuah aplikasi misalnya *file icon, file picture, file audio, file video*, atau yang lain. Gambar berformat *JPG* dan *PNG* sebuah aplikasi biasnya disimpan

dalam *folder res/drawble, icon* aplikasi disimpan dalam *res/drawable-ldpi* dan *file audio*

Disimpan dalam *folder res/raw*. File XML untuk membentuk sebuah *user interface* disimpan dalam *folder res/layout.*

Dalam *Project Android* kita juga akan menjumpai *direktori res/* yang membuat "sumber-sumber" (*file* tetap seperti gambar yang dikemas bersama aplikasi). Beberapa *sub direktori* yang akan dijumpai atau dibuat dibawah *direktori res/* terdiri dari:

1. *Res/drawable/* untuk gambar (*PNG<JPEG,etc*).
2. *Res/layout/* untuk *spesifikasi UI layout* yang dibuat dengan *XML*.
3. *Res/raw/ for general-purpose files* (misalnya *CSV File Account Informaation*).
4. *Res/values*/ untuk menyimpan *nilai strings, dimensions* dan sejenisnya
5. *Res/xml/ for other general purpose* XML *files you wish to ship.*

2.2.2. *Java*

1. Pengertian *Java*

Menurut Enterprise, (2015:1) "*Java* merupakan bahasa pemrograman yang berbasasis objek". Bahasa pemrograman *Java* di kembangkan oleh Sun microsystem yang dimulai oleh james gosling dan dirilis pada tahun 1995. Saat ini Sun Microsystem telah diakuisisi oleh oracle corporation.

Java bersifat *Write Once, Run Anywhere* (Program yang di tulis satu kali dan dapat berjalan pada banyak *platform*). Dan *Java* mempunyai fitur-fitur yang dapat

menarik perhatian banyak programmer. Berikut ini fitur-fitur *Java* yang perlu diketahui:

a. Berorientasi objek : dalam *Java* semua adalah objek.
b. Bersifat *platform* independent : *Java* di compile dalam bit kode *platform* independent dan bukan pada mesin *platform* spesifik seperti pada *C* dan *C++*.
c. Sedehana : *Java* didesain untuk dapat dengan mudah dipelajari.
d. Aman : dengan fitur keamanan *Java*, anda dapat membuat sistem yang berbasis virus dan powerfull.
e. Bersifat architectural-neutral : compiler *Java* membuat format file objek yang architectural-neutral, yang membuat kode yang di compile dapat di eksekusi pada berbagai *prosesor* yang memiliki sistem runtime *Java*.
f. Portabel :*Java* bersifat portabel karena adanya fitur *platform* independent dan architectural-neutral.
g. Kuat dan powerful : *Java* mengeliminasi eror dengan menjalankan pengecekan pada waktu compile dan runtime.
h. Multithreaded : dengan fitur multitreader *Java*, anda dapat membuat program yang dapat mengerjakan banyak tugas sekaligus.
i. Terinterprestasi : kode bit *Java* ditranslasi secara langsung pada intruksi mesin dan tidak disimpan.
j. Perfoma tinggi : *Java* memiliki perfoma yang tinggi karena menggunakan compiler langsung.
k. Terdistribusi : *Java* didesain untuk lingkungan distribusi internet.
l. Dinamis : *Java* lebih dinamis dari *C* dan *C++* karena *Java* didesain untuk beradaptasi dengan lingkungan pengembang.

2. *Java Development Kit (JDK)*

Java Deployment Kit (JDK) berisi sekumpulan kakas baris perintah *(command-line tool)* untuk menciptakan program *Java*. Rilis terakhir *JDK* dapat di *download* dari alamat *URL* berikut :http://*Java*.sun.com/products/JDK/index.html. *JDK* berisi sekumpulan kakas, utiltas, dan dokumentasi serta kode *applet* contoh untuk pengembangan program *Java*.

Berikut adalah daftar komponen utama *JDK* :

a. *Kompilator (Javac)*
b. *Interpreter program Java (Java)*
c. *Applet viewer (appletviewer)*
d. *Debugger (jdb)*
e. *Class file disassembler (Javap)*
f. *Header and stub file generator (Javah)*
g. *Documentation generator (Javadoc)*
h. *Applet demo*
i. Kode sumber *Java API*

3. *Java Runtime Environtment*

Java Runtime Environtment merupakan perangkat lunak yang digunakan untuk menjalankan aplikasi yang dibangun menggunakan *Java*. *Versi JRE* harus sama atau lebih tinggi dari *JDK* yang digunakan untuk membangun aplikasi agar aplikasi dapat berjalan sesuai dengan yang diharapkan.

2.2.3. Eclipse Integrated Development Enviroment (IDE)

Eclipse atau disebut juga *IDE (Integrated Development Environment)* untuk mengembangkan perangkat lunak dan dapat dijalankan di semua *platform (platform-independent).* Berikut ini adalah sifat dari *Eclipse* :

a. Multi-platform: Target sistem operasi Eclipse adalah *Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.*

b. *Mulit-language : Eclipse* dikembangkan dengan bahasa pemrograman *Java*, akan tetapi *Eclipse* mendukung pengembangan aplikasi berbasis bahasa pemrograman lain seperti *C/C++*, *Cobol*, *Python*, *Perl*, *PHP*, dan lain sebagainya.

c. *Multi-role :* Selain sebagai *IDE* untuk pengembangan *aplikasi*. *Eclipse* pun bisa digunakan untuk *aktivitas* dalam siklus pengembangan perangkat lunak seperti *dokumentasi*, pengujian perangkat lunak, pengembangan *web*, dan lain sebagainya.

Pada saat ini, *Eclipse* merupakan salah satu *IDE* favorit karena gratis dan *open source*. *Open source* berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari *Eclipse* yang membuatnya populer adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan membuat komponen yang disebut *plug-in.*

2.2.4. *Software Development Kit (SDK)*

Menurut Murya (2014:15) "*Android SDK* (Software Development Kit) adalah *tools API (Application Programming Interface)* yang diperluka untuk pengembangan aplikasi pada *platform Android*." *Android SDK* digunakan untuk mengembangkan aplikasi *Android*. *Android SDK* berisi *API*, sample, dan dokumentasi dari *Android* yang akan dikembangkan. *Android SDK* mencakup perangkat *tools* pengembangan yang *komprehensif*. *Android SDK* terdiri dari *debugger, libraries, handset emulator*, *dokumentasi*, contoh kode program dan tutorial. Saat ini *Android* sudah mendukung arsitektur x86 pada *Linux* (distribusi *Linux* apapun untuk *desktop modern*), *Mac OS X* 10.4.8 atau lebih, *Windows XP* atau *Vista*.

IDE yang didukung secara resmi adalah *Eclipse* 3.2 atau lebih dengan menggunakan *plugin Android Development Tools (ADT),* dengan ini pengembang dapat menggunakan *IDE* untuk mengedit dokumen *Java* dan *XML* serta menggunakan peralatan *command line* untuk menciptakan, membangun, melakukan *debug aplikasi Android* dan pengendalian perangkat *Android* (misalnya *reboot*, menginstal paket perangkat lunak).

2.2.5. *Android* Development Tools (*ADT*)

Android Development tools atau lebih dikenal dengan sebutan *ADT* atau plugin eclipse merupakan plugin yang digunakan untuk membuat project berbasis *Android*. *Android* development tools wajib di instal sehingga IDE *eclipse* yang sudah terinstal di komputer dapat digunakan sebagai tempat atau media untuk melakukan pemrograman *Android*.

2.2.6. *Emulator*

Menurut Murya (2014:25) "*Emulator* adalah sebuah perangkat lunak atau sistem yang berlaku seolah-olah seperti sistem yang sesungguhnya". *Emulator* mensimulasikan perangkat *Android* yang sesungguhnya dan digunakan untuk menjalankan aplikasi atau program *Android* yang sedang dikembangkan. Didalam tools *Android* disediakan perangkat *emulator* yang dapat langsung digunakan saat mengembangkan aplikasi *Android*. Konfigurasi emulator akan di simpan didalam AVD atau *Android Virtual Device*.

2.3. Metode *Algoritma*

Algoritma Fisher-Yates Shuffle (dinamai berdasarkan penemunya, *Ronald Fisher dan Frank Yates)* digunakan untuk mengubah urutan masukan yang diberikan secara acak. Permutasi yang dihasilkan oleh algoritma ini muncul dengan probabilitas yang sama *(source*).

Algoritma ini dinyatakan bias karena permutasi yang dihasilkan oleh algoritma ini muncul dengan probabilitas yang sama, hal ini dibuktikan dengan percobaan mengacak suatu set kartu yang dilakukan berulang – ulang.

Maka *algoritma* yang di gunakan oleh penulis yaitu *Algoritma Fisher-Yates Shuffle* yang di terapkan pada salah satu menu pada aplikasi yaitu pada menu *Quiz* untuk mengacak soal-soal agar tidak menampilkan soal yang sama untuk memulai kembali soal yang akan di jawab.

2.4. Pengujian Sistem

Menurut Sukamto, Shalahuddin (2013:272) "Pengujian adalah satu set aktifitas yang direncanakan dan sistematis untuk menguji atau mengevaluasi kebenaran yang diinginkan". Aktifitas pengujian terdiri dari satu set atau sekumpulan langkah dimana dapat menempatkan desain kasus uji yang spesifik dan metode pengujian.

Sering perangkat lunak mengandung kesalahan *(error)* pada proses-proses tertentu pada saat perangkat lunak sudah berada ditangan user. Kesalahan-kesalahan *(error)* pada perangkat lunak ini sering disebut dengan *"bug".* Untuk menghindari banyaknya bug maka diperlukan adanya pengujian perangkat lunak sebelum perangkat lunak diberikan ke pelanggan atau selama perangkat lunak masih terus dikembangkan.

Setelah pengujian sistem selesai dilakukan maka dapat dilakukan pengujian penerimaan perangkat lunak oleh pelanggan *(customer)* atau pemakai perangkat lunak *(user).* Pengujian penerimaan digunakan untuk mengetahui kepuasan pelanggan atau user terhadap perangkat lunak yang telah dibuat. Jika pelanggan sudah puas dengan perangkat lunak, maka perangkat lunak dapat diserahkan kepada pelanggan *(coustamer).*

Pengujian untuk validasi memiliki beberapa pendekatan sebagai berikut :

1.) Pengujian *Black Box*

Menurut Sukamto, Shalahuddin (2013:275) pengujian *black box* "Yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan

kode program". Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang di butuhkan.

Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan *spesifikasi* yang dibutuhkan. Kasus uji yang dibuat dengan kasus benar dan kasus salah, misalkan untuk kasus proses login maka kasus uji yang di buat adalah :

a. Jika *user* memasukkan nama pemakai *(username)* dan kata sandi *(password)* yang benar.
b. Jika user memasukkan nama pemakai *(username)* dan kata sandi *(password)* yang salah, misalnya nama pemakai benar tapi kata sandi salah, atau sebaliknya, atau keduanya salah.

2.) Pengujian *White Box*

Menurut Rosa, Shalahuddin (2013:276) pengujian *White Box* "Yaitu menguji perangkat lunak dari segi desain dan kode program apakah mampu menghasilkan fungsi-fungsi, masukan, dan keluaran yang sesuai dengan spesifikasi kebutuhan". Pengujian kotak putih dilakukan dengan memeriksa lojik dari kode program. Pembuatan kasus uji bisa mengikuti standar pengujian dari standar pemerograman yang seharusnya.

Pengujian terhadap dokumentasi yang dibuat juga harus dilakukan agar dokumentasi yang dibuat tetap konsisten dengan perangkat lunak yang dibuat.

2.5. Peralatan Pendukung

Membuat suatu aplikasi *Android* diperlukan berbagai macam pendukung yang dapat membantu terbentuknya aplikasi mulai dari logika hingga konsep pemrogramannya.

1. *UML(Unified Modeling Language)*

Menurut Rosa, Shalahuddin (2014:137) "*UML* merupakan bahasa *visual* untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung". *UML* hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan *UML* tidak terbatas pada metodologi tertentu, meskipun pada kenyataan *UML* paling banyak digunakan pada metodologi berorientasi objek.

Pada perkembangan teknologi perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang di berbagai negara dapat mengerti pemodelan perangkat lunak. Banyak orang yang telah membuat bahasa pemodelan pembangunan perangkat lunak sesuai dengan teknologi pemrograman yang berkembang pada saat itu, misalnya yang sempat berkembang dan digunakan oleh banyak pihak adalah data *flow diagram (DFD)* untuk memodelkan perangkat lunak yang menggunakan pemrograman prosedural atau struktural, kemudian juga ada *State Transition Diagram (STD)* yang digunakan untuk memodelkan sistem realtime (waktu nyata).

Berikut adalah diagram UML yang digunakan oleh penulis :

a. *Activity Diagram*

Menurut Sukamto, Shalahuddin (2013:161) “Diagram aktivitas atau *Activity diagram* menggambarkan *workflow* (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak”.

Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

b. *Use Case Diagram*

Use Case Diagram merupakan pemodelan untuk kelakuan *(behavior)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk manegetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

c. *Squence diagram*

Squence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirim dan diterima antar objek. Oleh karen itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode

yang dimiliki kelas yang diinstansikan menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case*.

d. *Class Diagram*

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Diagram kelas dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron.

e. *Deployment diagram*

Diagram *Deployment* atau *Deployment Diagram* menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi. Diagram *deployment* juga dapat digunakan untuk memodelkan hal-hal berikut :

1) Sistem tambahan *(embedded system)* yang menggambarkan rancangan *device, node,* dan *hardware.*
2) Sistem *client* atau *server*
3) Sistem terdistribusi murni
4) Rekayasa ulang aplikasi