

BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

Tinjauan pustaka dalam penelitian ini mengenai penilaian tampilan, menu dan fitur-fitur dari beberapa aplikasi yang berupa kamus Android yang diambil dari jurnal maupun naskah publikasi yang sudah ada. Dimulai dari menilai segi tampilan, menu dan fitur dari jurnal ilmiah karangan Sinaga dkk.(2014) dengan judul *Panduan Mobile P3K Dalam Insiden Kecelakaan*, kemudian jurnal Huda (2011), *Pengembangan Aplikasi P3K Berbasis Smartphone Android* dan yang terakhir Chai dkk.(2015), *Pengembangan Aplikasi Mobile Learning Untuk Pertolongan Pertama*.

Panduan Mobile P3K Dalam Insiden Kecelakaan, Sinaga dkk,(2014). Aplikasi ini dapat digunakan sebagai salah satu alat bantu pengenalan pentingnya P3K disetiap waktu dan simulasi pengajaran pertolongan pertama pada kecelakaan dan dengan menggunakan aplikasi mobile dalam bentuk aplikasi android masyarakat dapat lebih mudah untuk mempelajari penanganan kecelakaan. Memiliki beberapa menu yaitu Pengenalan P3K, Alat-alat P3K, Luka Ringan, Luka Berat, About. Aplikasi tersebut terdapat simulasi-simulasi yang dilakukan ketika terjadi kecelakaan sehingga memudahkan pengguna dalam pengoperasian aplikasi tersebut. Kekurangan aplikasi ini yaitu kurangnya video, simulasi tersebut hanya berupa gambar saja.

Pengembangan Aplikasi P3K Berbasis Smartphone Android, Huda (2011). Animasi ini dapat digunakan sebagai salah satu alat bantu pengenalan pentingnya kotak P3K disetiap waktu dan simulasi pengajaran pertolongan pertama pada kecelakaan dan dengan menggunakan aplikasi mobile. Masyarakat dapat lebih mudah untuk mempelajari penanganan kecelakaan. Sistem yang di bangun masih memiliki beberapa kekurangan salah satunya yaitu Aplikasi dapat dikembangkan sebagai sebuah aplikasi berbasis website yang tetap berfungsi sebagai aplikasi mobile. Aplikasi web digunakan sebagai media pengembangan basis pengetahuan sistem yang berfungsi sebagai media *update*.

Pengembangan Aplikasi *Mobile Learning* Untuk Pertolongan Pertama, Chaidkk.(2015), Pengetahuan dan pelatihan mengenai Pertolongan Pertama merupakan hal yang terbilang penting. Oleh karena itu, diperlukan fasilitas yang lebih mudah bagi masyarakat yang tertarik untuk mengikuti pelatihan ini. Mengingat banyak masyarakat yang terhalang jarak dan waktu untuk mempelajarinya, maka dilakukanlah pengembangan metode pelatihan dengan memanfaatkan berbagai teknologi yang ada saat ini sehingga persoalan jarak dan waktu serta informasi dapat terselesaikan. Pengembangan yang dilakukan memungkinkan peserta melaksanakan proses pelatihan secara mobile. Sistem yang di bangun masih memiliki beberapa kekurangan salah satunya yaitu Verifikasi user agar tidak terjadi kecurangan tertentu dan pengembangan fitur secara nyata.

Penelitian pada tugas akhir dengan judul membangun aplikasi panduan pertolongan pertama pada anak-anak saat kecelakaan di PMI Surakarta. Aplikasi ini terdiri dari beberapa menu yaitu menu Pengenalan Pertolongan Petama, menu Alat-alat Pertolongan, menu Luka Ringan, menu Luka Berat, menu Tentang serta beberapa fitur tentang pencarian penyakit, *splash screen*, *welcome screen*, dan keluar. *Software* yang digunakan dalam pembuatan aplikasi ini adalah *Smartphone*, Android Studio, Android kitkat, *CSS*, *Html* dan UML. Tugas akhir ini berbeda dengan penelitian sebelumnya karna lebih spesifik untuk anak-anak dan memiliki fitur video yang bisa dipelajari oleh orang tua masyarakat.

2.2 Landasan Teori

2.2.1 Smartphone

Smartphone merupakan *cellphone* yang menggabungkan fungsi-fungsi *Personal Digital Assistant* (PDA) seperti kalender, *personal schedule*, *address book*, dan memiliki kemampuan untuk mengakses internet, membuka email, membuat dokumen, bermain game, serta membuka aplikasi lainnya. Istilah *smartphone* merupakan istilah yang digunakan untuk mendeskripsikan mobile device yang menggabungkan fungsi *cellphone*, *audio player*, *digital camera*,

camcorder, Global Positioning System (GPS) receiver, dan Personal Computer atau PC (Wijayanto, 2015).

2.2.2 Android Studio

Android Studio adalah sebuah lingkungan pengembangan terpadu (LPT) untuk mengembangkan pada platform Android . Android Studio tersedia secara bebas di bawah Lisensi Apache 2.0. Android Studio berada di awal tahap *preview* akses mulai dari versi 0.1 Mei 2013, kemudian memasuki tahap beta mulai dari versi 0.8 yang dirilis pada bulan Juni 2014. Yang pertama membangun stabil dirilis pada bulan Desember 2014, mulai dari versi 1.0. Berdasarkan software IDEA *JetBrains ' IntelliJ*, Android Studio dirancang khusus untuk pengembangan Android . Ini tersedia untuk di-*download* pada Windows, Mac OS X dan Linux, dan diganti Eclipse Pengembangan Android Tools sebagai IDE utama Google untuk pengembangan aplikasi Android asli .

2.2.3 Android versi 4.4 (Kitkat)

Android Kitkat dirilis pada tanggal 28 Oktober 2013. Android Kitkat mempunyai beberapa pembaruan yang lebih baik dari versi sebelumnya seperti perintah suara lebih sederhana dan pintar karena pengguna tidak harus menyentuh ponsel untuk melakukan pencarian, mengirim pesan, atau menerima arahan dari *Google Map*, pada menu utama pengguna hanya perlu mengatakan ' OK Google ' untuk memberikan perintah dengan kerja spesifik yang presisi kemudian adanya *Multi-Tasking* yang lebih cepat, visual yang lebih baik pada versi Android sebelumnya. Selain itu, ada tombol tambahan pada kunci layar untuk memungkinkan pengguna bermain game dan bisa langsung masuk ke dalam menu galeri pada waktu bersamaan, adanya Photo Editor pada Android Kitkat pun diperbarui. Terdapat beberapa fitur efek, filter, dan tools untuk menyesuaikan gambar. Hebatnya, Photo Editor juga memungkinkan pengguna mengembalikan hasil editan foto ke bentuk semula, *Google Hangouts* juga ditambahkan pada sistem operasi ini. *Hangouts* mampu mengelompokkan semua SMS, MMS, video call, dan chatting dengan *Google Hangouts* pada satu tempat.

2.2.4 CSS (*Cascade Style Sheet*)

CSS (*Cascade Style Sheet*) adalah sebuah fitur yang diperkenalkan sejak HTML versi 4.0 dan berfungsi untuk menangani masalah tampilan pada HTML seperti jenis, ukuran dan warna font, posisi teks, batas tulisan atau margin, warna *background*, dan sebagainya. Dari sisi manajemen dan perawatan, penggunaan CSS dipandang lebih praktis karena para *web developer* tidak perlu membuka setiap file dalam sebuah situs untuk melakukan perubahan. Hal penting yang perlu diperhatikan adalah cara meletakkan CSS dan juga bahasa berbasis web lain untuk memudahkan *manajemen file, editing, dan maintenance*. Penulis menyisipkan CSS, *JavaScript*, VB Script, PHP, maupun ASP langsung ke dalam dokumen HTML *embedded script* (Yuniyanti, 2016).

2.2.5 HTML (*Hypertext Markup Language*)

HTML adalah kependekan dari *Hyper Text Markup Language*, merupakan sebuah bahasa *scripting* yang berguna untuk menuliskan halaman web. Pada halaman web, HTML dijadikan sebagai bahasa *script* dasar yang berjalan bersama berbagai bahasa *scripting* pemrograman lainnya. Semua tag-tag HTML bersifat dinamis, artinya kode HTML tidak dapat dijadikan sebagai *fileexecutable* program. Hal ini disebabkan HTML hanyalah sebuah bahasa *scripting* yang dapat berjalan apabila dijalankan di dalam *browser* (pengaksesan web), *browser-browser* yang mendukung HTML antara lain adalah *Internet Explorer, Netscape Navigator, Opera, Mozilla* dan lain-lain. Jadi pada saat ingin membuka halaman yang berasal dari HTML dapat dilihat dari bentuk pengkodeanya dengan cara mengklik menu *view source*, maka disana akan ditampilkan semua tag beserta isi dari halaman web tersebut (Yuniyanti, 2016).

2.2.6 UML (*Unified Modeling Language*)

Menurut Rosa dan Shalahuddin (2016) dijelaskan 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. Pada perkembangan teknik pemrograman berorientasi objek, muncullah sebuah





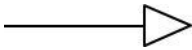
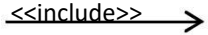
standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modelling Language*.

Ada 6 (tujuh) macam diagram dalam *Unified Modeling Language* (UML), yaitu :

a. Use Case Diagram

Use case diagram adalah teknik untuk merekam persyaratan fungsional sebuah sistem. *Use case diagram* mendefinisikan interaksi tipikal antara para pengguna sistem dengan sistem itu sendiri, dengan memberi sebuah narasi tentang bagaimana sistem tersebut digunakan. *Use case diagram* berhubungan erat dengan kejadian-kejadian. *Use case diagram* dibuat untuk menggambarkan hubungan antara *actor* dan *use case*.

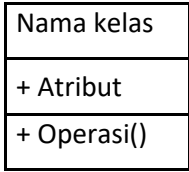



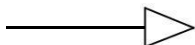
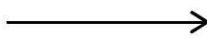
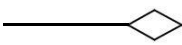
Tabel 2.1. Simbol-simbol Use Case Diagram

NO.	SIMBOL	NAMA	KETERANGAN
1.	 Nama use case	<i>Use case</i>	Fungsional yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan kata kerja di awal di awal frase nama <i>use case</i> .
2.		<i>Actor</i>	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, nama aktor.
3.		<i>Association</i>	Komunikasi antara aktor dan <i>use case</i> yang berpartisipasi pada <i>use case</i> atau <i>use case</i> memiliki interaksi dengan aktor.
4.	 <<extend>>	<i>Extend</i>	Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> dimana <i>use case</i> yang ditambahkan dapat berdiri sendiri walau tanpa <i>use case</i> tambahan itu.
4.		<i>Generalization</i>	Hubungan generalisasi dan spesialisasi (umum – khusus) antara dua buah <i>use case</i> dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
5.	 <<include>>	<i>Include</i>	Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> di mana <i>use case</i> yang ditambahkan memerlukan <i>use case</i> ini untuk menjalankan fungsinya atau sebagai syarat dijalankan <i>use case</i> ini.

b. *Class Diagram*

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode operasi. Kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem sehingga pembuat perangkat lunak dapat membuat kelas-kelas di dalam program perangkat lunak sesuai dengan perancangan diagram kelas, (Rosa & Shalahuddin, 2016). Simbol *class diagram* dapat dilihat pada tabel 2.2.

Tabel 2.2. Simbol-simbol *Class Diagram*

NO.	SIMBOL	NAMA	KETERANGAN
1.		<i>Class</i>	Kelas pada struktur sistem
2.		<i>Interface</i>	Sama dengan konsep interface dalam pemrograman berorientasi objek.
3.		<i>Association</i>	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan <i>multiplicity</i> .
4.		<i>Directed association</i>	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain biasanya juga disertai dengan <i>multiplicity</i> .
5.		<i>Generalization</i>	Relasi antar kelas dengan makna generalisasi - spesialisai (umum – khusus).
6.		<i>Dependency</i>	Relasi antar kelas dengan makna kebergantungan antar kelas.
7.		<i>Aggregation</i>	Relasi antar kelas dengan makna semua bagian (<i>whole-part</i>).

c. Activity Diagram

Tabel 2.3.Simbol-simbol Activity Diagram



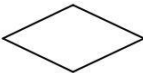


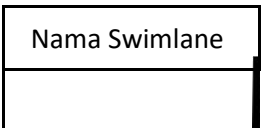

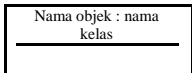

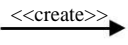
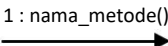
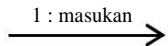
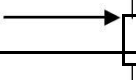

NO	GAMBAR	NAMA	KETERANGAN
1		Status awal	Status awal aktivitas sistem.
2		Aktivitas	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3		<i>Decision</i>	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4		<i>Join</i>	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5.		Status akhir	Status akhir yang dilakukan sistem.
6.		<i>Swimlane</i>	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

Diagram aktivitas atau *Activity diagram* menggambarkan *workflow* atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan actor,(Rosa & Shalahuddin, 2016) Simbol *Activity Diagram* ditunjukan pada tabel 2.3.

d. Sequence Diagram

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek,(Rosa & Shalahuddin, 2016).

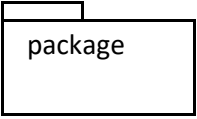
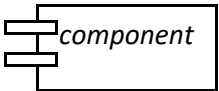



Tabel 2.4. Simbol-simbol Sequence Diagram

NO.	SIMBOL	NAMA	KETERANGAN
1.		Aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
2.		<i>Lifeline</i>	Menyatakan kehidupan suatu objek.
3.		Objek	Menyatakan objek yang berinteraksi pesan.
4.		Waktu aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi.
5.		Pesan tipe create	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
6.		Pesan tipe call	Menyatakan suatu objek memanggil operasi / metode yang ada pada objek lain atau dirinya sendiri.
7.	 	Pesan tipe send	Menyatakan bahwa suatu objek mengirimkan data / masukan / informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim.
8.		Pesan tipe destroy	Menyatakan suatu objek mengakhiri hidup objek lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada <i>create</i> maka ada <i>destroy</i> .

e. *Component Diagram*

Diagram komponen *atau component diagram* dibuat untuk menunjukkan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem, (Rosa & Shalahuddin, 2016). Simbol-simbol yang ada pada *component diagram* dapat dilihat pada tabel 2.5.

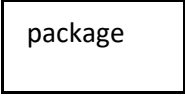
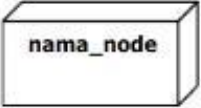


Tabel 2.5 Simbol-simbol *Component Diagram*.

NO	GAMBAR	NAMA	KETERANGAN
1.		<i>Package</i>	<i>Package</i> merupakan sebuah bungkusan dari satu atau lebih komponen
2.		<i>Component</i>	Komponen sistem
3.		<i>Dependency</i>	Kebergantungan antar komponen, arah panah mengarah pada komponen yang dipakai
4.		<i>interface</i>	Sama dengan konsep <i>interface</i> pada pemrograman berorientasi objek, yaitu sebagai antarmuka komponen agar tidak mengakses langsung komponen.
5.		<i>Link</i>	Relasi antar komponen

f. *Deployment Diagram*

Deployment diagram menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi. Diagram *deployment* juga dapat digunakan untuk memodelkan sistem tambahan yang menggambarkan rancangan device, node, dan hardware. Sistem client, sistem terdistribusi murni, rekayasa ulang aplikasi, (Rosa & Shalahuddin, 2016). Simbol – simbol *Deployment Diagram* ditunjukkan pada Tabel 2.6.

Tabel 2.6. Simbol-simbol Deployment

NO	SIMBOL	NAMA	KETERANGAN
1.		<i>Package</i>	<i>Package</i> merupakan sebuah bungkus dari satu atau lebih <i>node</i> .
2.		<i>Node</i>	Biasanya mengacu pada perangkat keras (<i>hardware</i>), perangkat lunak yang tidak dibuat sendiri (<i>software</i>), jika didalam <i>node</i> disertakan komponen untuk mengkonsistenkan rancangan maka komponen yang diikutsertakan harus sesuai dengan komponen yang telah didefinisikan sebelum pada diagram komponen.
3.		<i>Dependency</i>	Kebergantungan antar <i>node</i> , arah panah mengarah pada <i>node</i> yang dipakai.
4.		<i>Link</i>	Relasi antar <i>node</i> .

2.2.7 Black Box Testing

Pengertian *blackbox testing* yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian ini dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.

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 coba yang dibuat untuk melakukan pengujian kotak hitam harus dibuat dengan kasus benar dan kasus salah, misalkan untuk kasus proses *login* maka kasus uji yang dibuat 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, misal nama pemakai benar kata sandi salah atau sebaliknya, atau keduanya salah.

Pengujian *black-box* berfokus pada persyaratan fungsional perangkat lunak. Dengan demikian pengujian *black-box* memungkinkan perancang perangkat lunak mendapatkan serangkaian kondisi *input* yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Pengujian *black-box* bukan merupakan alternatif dari teknik *white-box*, tetapi merupakan pendekatan komplementer yang kemungkinan besar mampu mengungkap kelas kesalahan daripada metode *white-box*.

Pengujian *black-box* berusaha menemukan kesalahan dalam kategori sebagai berikut:

1. fungsi-fungsi yang tidak benar atau hilang,
2. kesalahan *interface*,
3. kesalahan dalam struktur data atau akses *database external*,
4. kesalahan kinerja,
5. inisialisasi dan kesalahan terminasi.

Pengujian *black-box* memperhatikan struktur kontrol, maka perhatian berfokus pada *domain* informasi. Pengujian di desain untuk menjawab pertanyaan-pertanyaan berikut :

1. Bagaimana validitas fungsional diuji ?
2. Kelas input apa yang akan membuat *test case* menjadi baik ?
3. Apakah sistem sangat sensitif terhadap harga input tertentu ?
4. Bagaimana batasan dari suatu data diisolasi ?
5. Kecepatan data apa dan volume data apa yang dapat ditolelir oleh sistem ?

Dengan mengaplikasikan teknik *black-box*, maka serangkaian *test case* akan memenuhi kriteria berikut ini : (1) *test case* yang mengurangi, dengan harga lebih dari satu, jumlah *test case* tambahan yang harus didesain untuk mencapai pengujian yang dapat dipertanggungjawabkan, dan (2) *test case* yang memberi tahu mengenai kehadiran atau ketidakhadiran kelas kesalahan, daripada

memberitahu kesalahan yang berhubungan hanya dengan pengujian spesifik yang ada (Pressman, 2002).

2.2.8 Pengujian dengan Kuesioner

Kuesioner merupakan daftar pertanyaan yang sudah distandarisasikan dan menstrukturkan serta memperluas proses pengumpulan fakta. Kuesioner dapat mengarahkan sistem analisis melalui wawancara dan arena analisis menggunakan pertanyaan-pertanyaan yang sama, maka kuesioner dapat meningkatkan konsistensi pengumpulan fakta.

Teknik kuesioner untuk pengumpulan data merupakan salah satu cara yang baik untuk mendapatkan data yang akurat, dimana daftar pertanyaan yang dibuat adalah daftar yang berisi pertanyaan-pertanyaan untuk tujuan khusus yang memungkinkan sistem analisis untuk mengumpulkan data dan pendapat dari responden-responden yang dipilih. Daftar pertanyaan atau kuesioner ini akan dikirim kepada responden yang akan mengisinya sesuai dengan pendapat mereka. Penggunaan daftar pertanyaan ini mendapat banyak kritik karena diragukan hasilnya. Akan tetapi untuk mengumpulkan data dengan jumlah sumber yang banyak, tidak ada teknik pengumpulan data lainnya yang lebih dibandingkan kuesioner. Adapun petunjuk untuk membuat daftar pertanyaan adalah :

- a. Rencanakanlah terlebih dahulu fakta/opini apa saja yang ingin dikumpulkan.
- b. Berdasarkan opini tersebut di atas, tentukan tipe dari pertanyaan yang paling tepat untuk masing-masing fakta dan opini.
- c. Tulislah pertanyaan-pertanyaan yang diajukan. Pertanyaan itu tidak boleh mengandung kesalahan, serta harus jelas dan sederhana.
- d. Lakukan uji coba atas pertanyaan itu ke beberapa responden terlebih dahulu, misalnya dua atau tiga orang. Apabila responden mengalami kesulitan dalam mengisi daftar pertanyaan itu maka pertanyaan-pertanyaan itu harus diperbaiki lagi.

2.2.9 Kerangka pemikiran

Kerangka pemikiran dari tugas akhir ini dapat dijelaskan atau didefinisikan sebagai berikut:

1) Latar belakang masalah

Pokok permasalahan yang mendasari perlunya dibangun aplikasi panduan pertolongan pertama saat kecelakaan pada anak-anak di PMI Surakarta.

2) Perumusan masalah

Perumusan masalah Bagaimana membangun Aplikasi panduan pertolongan pertama saat kecelakaan pada anak anak di PMI Surakarta

3) Pengumpulan data tertulis dan tidak tertulis

Pada penelitian dilakukan pengumpulan data secara tertulis dan tidak tertulis pada PMI Surakarta. Pengumpulan data penelitian ini menggunakan metode observasi, dokumentasi, dan wawancara, dan menguasai aplikasi android, PHP, MYSQL dan CSS untuk membangun aplikasi.

4) Analisis dan perancangan sistem

Pada penelitian dilakukan menganalisa dan merancang bagaimana sistem nantinya akan dibuat untuk membantu memecahkan permasalahan yang ada.

5) Implementasi sistem

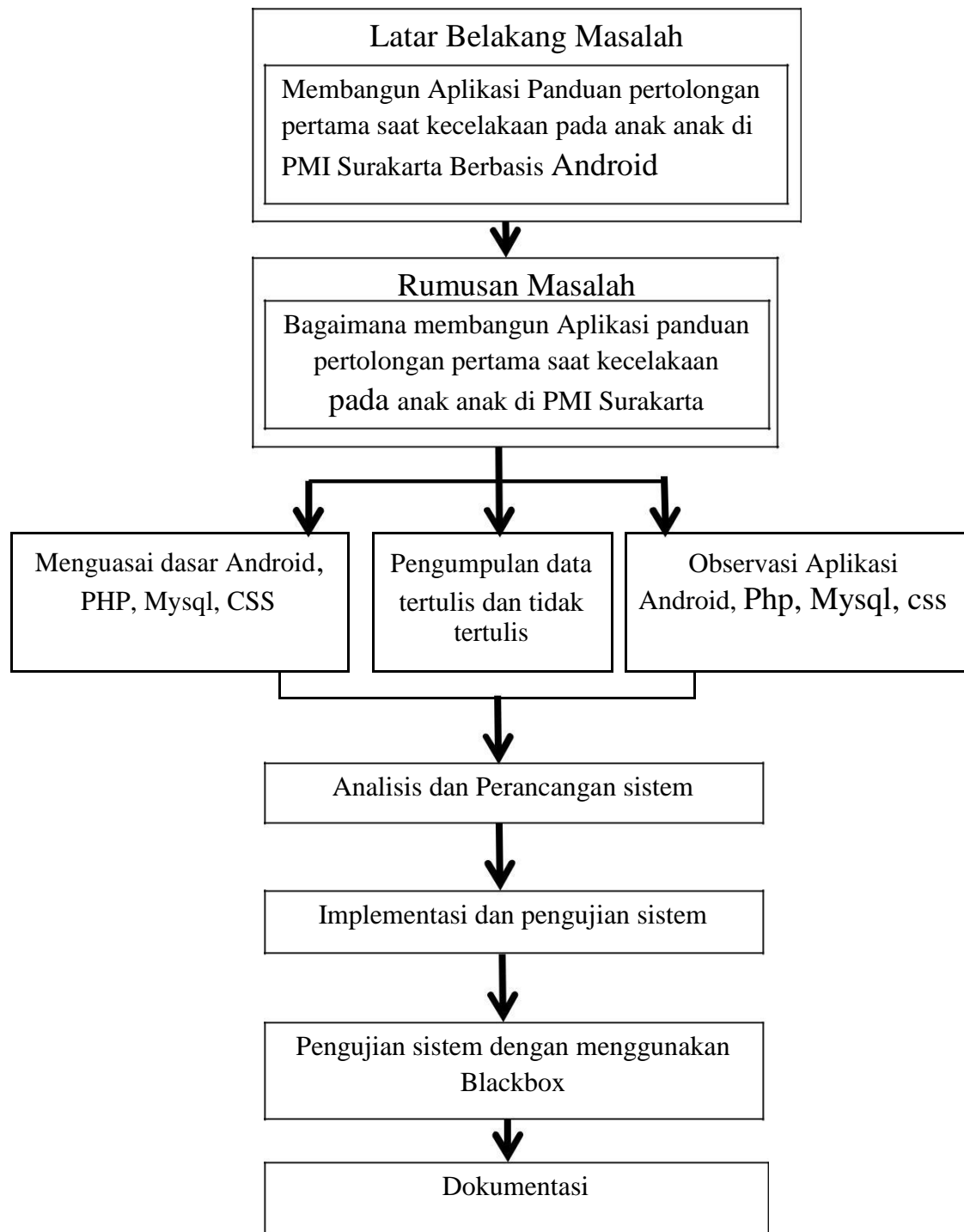
Pada penelitian dilakukan implementasi apa yang sudah dirancang untuk membangun aplikasi panduan pertolongan pertama saat kecelakaan pada anak anak di PMI Surakarta.

6) Pengujian sistem

Pada penelitian dilakukan uji coba aplikasi apakah masih terjadi kesalahan ataupun kekurangan pada sistem menggunakan blackbox.

7) Penerapan sistem dan Dokumentasi

Sistem yang sudah diimplementasikan dan diuji coba kemudian diterapkan pada PMI Surakarta dan setelah itu dibuatnya dokumentasi dari keseluruhan kegiatan penyusunan Tugas Akhir.



Gambar 2.1 Diagram kerangka Pemikiran