Aplikasi Pendeteksi Citra yang diambil dari Kamera Mobile Device untuk Menghasilkan Ensiklopedia Fauna Berbasis Android

Joshua Caine Buwono[1], Liliana[2], Henry Novianus Palit[3]
Program Studi Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
Jln. Siwalankerto 121-131 Surabaya 60236
Telp. (031)-2983455, Fax. (031)-8417658
jbuwono@gmail.com[1], lilian@petra.ac.id[2], hnpalit@petra.ac.id[3]

ABSTRAK

Menemukan hewan baru terkadang menjadi sesuatu yang baru bagi manusia. Apabila hewan baik jinak maupun buas ditemukan, maka biasanya pencarian informasi tentang hewan tersebut akan dilakukan. Namun, pengetahuan akan hewan tersebut masih cenderung minim. Pertanyaan muncul tentang pengetahuan akan buas atau tidaknya fauna tersebut. Tetapi, pengetahuan tersebut baru datang setelah hewan tersebut melakukan penyerangan. Oleh karena itu, solusi yang diberikan adalah aplikasi yang menghasilkan informasi yang selalu baru dengan teknologi pendeteksi citra. Sehingga, hanya dengan memotret, informasi sudah bisa didapat dengan mudah dan cepat. Aplikasi juga dapat dijalankan pada sistem operasi Android Jellybean API 17 dan seterusnya.

Kata Kunci: hewan, informasi, citra

ABSTRACT

Finding new animals can sometimes be something new for humans. If both tame or wild animal is found, it usually search for information about the animal would do. However, knowledge of these animals still tend to be minimal. Questions arise about knowledge of these animal. However, this knowledge comes after the animal attacked. Therefore, the solution provided is an application that generates information that is always new with image detection technology. Thus, with just a photograph, information can be obtained easily and quickly. Application can be run on Android Jellybean operating system API 17 and so on.

Keyword: *animals, information, image*

1. PENDAHULUAN

Menemukan hewan baru terkadang menjadi sesuatu yang baru bagi manusia. Apabila kita bertemu dengan hewan baik jinak maupun buas, maka biasanya kita akan mencari informasi tentang hewan tersebut. Terutama bila kita bertemu dengan hewan buas, biasanya kita akan mencari informasi apakah hewan ini buas atau jinak dan seberapa buaskah hewan ini. Kita tidak mungkin membuka laptop atau *smartphone* kita lalu *browsing* di internet tentang hewan itu. Belum lagi, hewan biasanya memiliki ketakutan terhadap beberapa hal sehingga membuat hewan ini menjadi siap untuk membela diri. Seperti ular bila dikagetkan, maka ular akan langsung menggigit apapun. Atau seperti kuda bila merasa stres, maka akan menendang ke belakang. Hal seperti ini biasanya kita tidak tahu. Dibutuhkan informasi yang cepat agar hal seperti ini tidak terjadi. Bila terlalu lama mendapatkan informasi maka nyawa pun bisa menjadi taruhan.

Yang lain, kita tidak dapat memberikan kata kunci yang sesuai pada *search engine* atau ensiklopedia bila kita menemukan hewan tertentu. Memberikan deskripsi hewan yang ditemukan kepada *search engine* atau ensiklopedia membutuhkan waktu yang lama. Sedangkan, kita tidak tahu nama hewan tersebut. Deskripsi hewan yang dikirim pun belum tentu akurat karena kita terburu-buru untuk mengetik. Belum lagi informasi yang didapat ternyata tidak sesuai dengan keinginan. Banyak *search engine* atau ensiklopedia yang beredar di internet yang menghasilkan informasi yang tidak teratur atau mungkin tidak sesuai karena terlalu susah untuk dimengerti. Belum lagi *search engine* atau ensiklopedia yang dipilih tidak mencantumkan referensi sehingga keakuratan data pun perlu dipertanyakan.

Salah satu solusi dari masalah diatas adalah dengan menggunakan aplikasi pendeteksi citra sehingga menghasilkan informasi tentang hewan yang ingin kita cari.

2. LANDASAN TEORI

2.1. JSON

JSON adalah data yang berupa text dan dapat dibaca manusia untuk mewakili sejumlah data [9]. Biasanya JSON digunakan untuk sejumlah besar data agar pengolahan data selanjutnya dapat dilakukan dengan mudah.

Ada 2 format dalam mengirim data dari *server* ke *client*, yaitu JSON dan XML. Perbedaan mendasar dari 2 format data tersebut adalah JSON cenderung ringan dalam pengiriman data, sedangkan XML memperhatikan struktur data [8].

JSON berupa *object* yang diawali dan diakhiri dengan tanda kurung kurawal ({}) dan memiliki *object member* dan *value* [4]. *Object member* adalah anggota yang berisi *key* dan *value*. *Object member* biasanya ditemukan tanda titik dua (:) sebagaii pemisah antara *key* dan *value*.

2.2. Java

Java adalah bahasa pemrograman berdasar pada OOP (*Object Oriented Programming*) yang digunakan untuk membuat aplikasi yang dapat berjalan di *computer* ataupun *mobile* [14]. Dalam perkembangannya, pemrograman Java ini berkembang menjadi bermacam-macam, Java EE (*Web app*), Java/Java SE (*Desktop App*), Java ME (*Mobile*), Java FX (*Next Generation*). Program java dibuat melalui *interface*. *Interface* tersebut dapat diimplementasikan dengan berbagai *class* yang tidak ada dalam hirarkinya [1].

2.3. Android *Mobile Device*

Android adalah teknologi yang *open-source* yang digunakan untuk mengembangkan aplikasi pada *mobile device* [11]. Android bekerja pada Linux *kernel*. Linux *kernel* ini men-*support* Java *Virtual Machine* (JVM) yang menggunakan bahasa pemrograman Java sebagai bahasa pemrograman yang dianjurkan untuk pengembangan.

Keamanan untuk *mobile device* menjadi penting karena kemajuan teknologi serta berbagai ancaman dari luar yang turut berkembang pula. Pada sebuah literatur, ancaman terhadap

keamanan tersebut dibagi menjadi 3, yaitu *malware*, *vulnerabilities*, dan *attacks* [16]. *Malware* mempunyai tujuan untuk mencuri data sehingga data menjadi hilang ataupun tidak dapat digunakan lagi. Ada beberapa tipe *malware* salah satunya adalah trojans. *Software* ini sebenarnya sudah sering kita temui di beberapa komputer. Trojans sebenarnya tidak membelah diri namun digunakan untuk mengambil informasi dari perangkat yang sudah diinfeksi [16].

Masalah yang kadang terjadi adalah layar yang kecil dan ruang memori yang kecil pula [5]. Sehingga, pengembangan aplikasi perangkat menjadi terkendala. Sedangkan, keunggulan dari *mobile device* adalah dapat dibawa kemanapun, digunakan untuk berinteraksi dengan orang lain, dapat terkoneksi dengan jaringan, dan dipakai untuk personal [5].

Dalam perkembangannya, *mobile device* digunakan untuk membantu pengguna untuk bisa berpartisipasi dalam *collaborative learning* dengan menggunakan *smartphone* [2]. Peneliti juga mengungkapkan bahwa dengan menggunakan sarana belajar seperti *smartphone*, pengguna akan mendapatkan nilai tinggi untuk dapat berpartisipasi dengan diskusi kelas.

2.4. *Service Oriented Architecture*

Service oriented architecture (SOA) adalah arsitektur yang membentuk sebuah aplikasi, layanan yang sudah terkontrak untuk menjawab berbagai persoalan bisnis [12]. SOA harus menjamin bahwa teknologi yang berdasar SOA bersifat netral dan transparan. SOA juga merupakan model untuk pengembangan sistem yang berbasis layanan yang dapat digunakan untuk berbagai macam bisnis domain [3]. SOA juga merupakan pendekatan dan pelatihan untuk membangun sebuah sistem perangkat lunak *IT*.

Dalam perkembangannya, SOA dapat digunakan untuk mengatur senjata perang dan *battle command* pada sebuah peperangan. Secara singkat, SOA yang dipakai dibagi menjadi 3 level. Level pertama adalah *client* dalam hal ini adalah prajurit. Level kedua adalah *server* yang menyimpan semua layanan yang dapat digunakan oleh *client*. Level ketiga adalah *middleware* yang menyediakan jalur data yang aman antara *server* dan *client*.

2.5. *Web Service*

Web Service adalah teknologi *middleware* yang ditujukan untuk menjembatani antar *platform* sehingga data dapat mengalir ke antar *platform* tersebut [6]. *Web service* didefinisikan oleh WSDL (*Web Service Definition Language*), dijalankan oleh protocol SOAP (*Simple Object Access Protocol*), dan data yang dihasilkan akan dijelaskan pada dokumen XML (*eXtended Markup Language*).

Web service juga menggunakan protocol HTTP untuk mengirim ataupun menerima data [7]. *Web service* juga menjaga agar data yang dikirim atau diterima tetap dijaga kerahasiaannya. *Web service* juga dapat diprogram yang artinya *web service* dapat dikembangkan lebih lanjut.

Selain SOAP, metode lain yang digunakan adalah REST. Arsitektur REST menggunakan arsitektur *client-server* dimana *client* mengirim data pada *server* yang disebut *request* dan *server* akan mengolah dan mengirim hasilnya pada *client* yang disebut *response* [10]. Keunggulan REST adalah memiliki alamat yang jelas, tidak menggunakan *state* yang rumit, dan *interface* yang seragam. Dari sisi besar data, REST sudah diuji 9-10 kali lebih ringan daripada besar data SOAP. Sehingga waktu yang ditempuh 5-6 kali lebih cepat daripada SOAP. [10]

2.6. MVC

MVC (*Model-View-Controller*) adalah sebuah aplikasi yang memisahkan antara input aplikasi, logika bisnis, dan output aplikasi sesuai dengan *model*, *view*, dan *controller* [13]. Keunggulan MVC adalah membantu *developer* untuk fokus pada berbagai fitur aplikasi yang dikembangkannya.

2.7. *Image Recognition*

Image Recognition adalah bagian dari *Computer Vision* (CV) yang dapat dipelajari oleh komputer untuk mengenali berbagai objek dan menganalisanya. CV sendiri dibagi menjadi 3 tugas, yaitu pengambilan (*Acquiring*), memproses (*Processing*), dan menganalisa (*Analyzing*). *Image Recognition* bekerja dengan mendeteksi berbagai *pixel* pada gambar lalu mengenalinya dengan mencocokkan dengan *database*. *Image Acquisition* diambil dari kamera atau *device* pengambil gambar. Segala bentuk informasi tentang *pixel* akan disimpan di dalam gambar tersebut. *Image Pre-processing* adalah metode sebelum *Image Recognition* dilakukan. Metode ini memastikan agar gambar tersebut dapat diterima oleh algoritma. *Image Detection* adalah proses untuk mendeteksi dan merupakan bagian dari *Image Recognition*. Biasanya algoritma ini memiliki toleransi untuk pendeteksi sehingga gambar yang hampir mirip akan langsung disamakan. *Image Decision Making* adalah proses terakhir setelah deteksi selesai. Algoritma akan mengeluarkan hasil berdasarkan *database* yang tersedia. [15]

Image Recognition API dibuat dengan mengirim gambar melalui HTTP POST menuju *endpoint* /image_request dengan menggunakan *multipart-form-encoded*. API tersebut tidak menerima data berupa *base64 encoded*. Selain mengirim data secara local, *user* juga dapat mengirim data melalui link URL untuk dikirim ke *endpoint* yang tersedia. *User* dapat menggunakan salah satu metode namun tidak boleh dua-duanya yang digunakan. Parameter fokus digunakan untuk memberi tahu tempat yang digunakan sebagai detail dari gambar tersebut. *Focus point* dapat berupa koordinat atau koordinat absolut (contohnya gambar 400x400 akan mempunyai koordinat 0 sampai 400 untuk setiap axis). Setelah gambar diterima, responnya akan berupa pesan berhasil atau *error*. Bila pesan berhasil, maka *user* akan mendapatkan URL dan *token* untuk dapat menerima hasil dari proses gambar tersebut. Direkomendasikan gambar yang dikirim memiliki resolusi maksimal 1024px dan level kompresi JPEG antara 5-8.

3. ANALISIS DAN DESAIN

3.1. Analisis Permasalahan

Proses pertama yang dilakukan adalah *user* memasukkan gambar. Ada 2 pilihan untuk *user* dalam rangka memasukkan gambar ke aplikasi tersebut, yaitu menggunakan kamera *mobile* yang sudah disediakan oleh *device* atau menggunakan *gallery mobile device* untuk memilih gambar yang sudah diambil oleh kamera. Pada saat ini ada beberapa masalah yang terjadi. Salah satunya adalah gambar yang terlihat *blur*. Hal ini dapat diantisipasi oleh API sehingga pada sistem yang dibuat nanti akan diperiksa dari hasil yang dikeluarkan API tersebut.

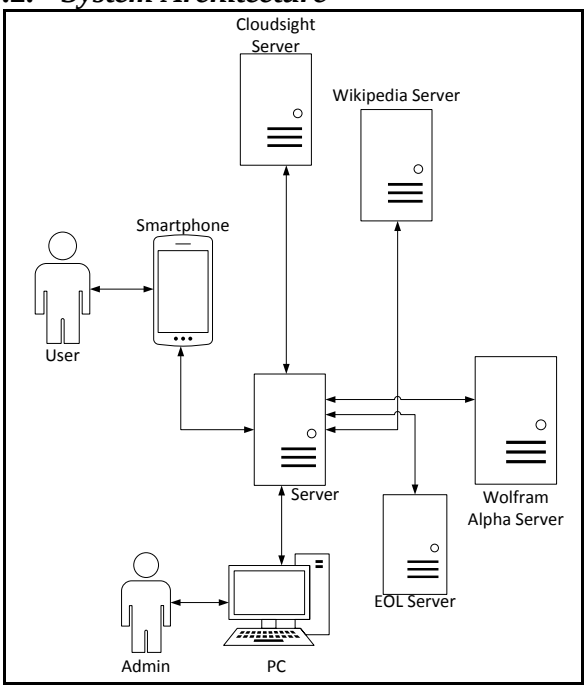
Setelah proses pertama, maka selanjutnya adalah gambar akan dikirim dari *mobile device* menuju ke *server*. Sehingga, masalah selanjutnya adalah *server down*. Solusinya adalah memilih server yang sudah terjamin sehingga waktu untuk server *down* bisa diminimalisir.

Proses pada *server* membutuhkan waktu cukup lama karena selain mendeteksi *image*, *server* juga harus menerima informasi dari hasil pendeteksi gambar yang diterima. Masalah yang terjadi adalah cara untuk mengirim gambar tersebut ke API. Oleh karena itu, dibutuhkan *library* untuk mengirim serta menerima informasi yang sudah dihasilkan oleh API. Selain itu, masalah lainnya adalah status pada API bila gambar tidak dapat diproses. Untuk mengantisipasi hal tersebut, *server* akan memeriksa status API tersebut.

Setelah pendeteksi gambar selesai, maka akan didapat hasil berupa kata kunci. Masalah yang terjadi selanjutnya adalah model dari hasil informasi dari penghasil informasi. Seringkali, penghasil informasi menghasilkan informasi yang kacau. Oleh karena itu, pada *server* semua informasi yang diperoleh akan disusun kembali sehingga memudahkan dalam pengambilan informasi pada *mobile device*.

Proses terakhir adalah pengambilan informasi serta penyusunan informasi yang akan ditampilkan pada *mobile device*. Penyusunan informasi menjadi masalah sendiri. Oleh karena itu, penyusunan informasi dilakukan dengan memberi beberapa halaman pada aplikasi.

3.2. *System Architecture*



Gambar 1 *System Architecture*

Penjelasan dari
Gambar **1** adalah sebagai berikut. Ada 2 jenis hak akses yang diperbolehkan untuk mengakses sistem, yaitu *user* dan *administrator*. *Administrator* mengakses sistem melalui protokol enkripsi yaitu SSH. Protokol ini hanya dimiliki oleh *administrator* dan hanya digunakan untuk men-*develop* sistem sehingga sistem menjadi lebih sempurna. Sedangkan, *user* mengakses sistem menggunakan aplikasi *mobile*. Aplikasi lah yang akan mengakses sistem. *User* tidak dapat mengakses sistem secara langsung. Cloudsight *server*, wikipedia *server*, wolfram alpha *server*, dan EOL *server* berada di luar sistem dan mempunyai *endpoint API* sehingga *administrator* tidak dapat mengatur *server* tersebut. Cloudsight *server* adalah *server* yang akan digunakan untuk mendeteksi gambar yang sudah di-*input*. Wikipedia, Wolfram Alpha, dan EOL *server* adalah *server* yang akan menyediakan informasi untuk dijadikan ensiklopedia.

4. IMPLEMENTASI SISTEM

4.1. *Upload Server*

Segmen Program 1 *Upload Server*berfungsi untuk mengirim gambar yang telah dikirim dari aplikasi *mobile device* kepada *server*. Peletakan pada *server* juga akan mengubah isi dari *database*. Hal ini untuk mencegah agar apabila beberapa *user* mengakses *server* pada waktu yang sama tidak akan tertukar data yang dikirim kepada *server*.

Segmen Program 1 *Upload Server*

```
public function upload(Request $request){
    $file=base64_decode($request->image);
    $image1=Image::orderBy('id','desc')->first();
    if($image1==null)
        $id=1;
    else
        $id=$image1->id+1;
    $image=new Image();
    $image->nama="hewan".$id.".jpg";
    $image->save();
    if(Storage::disk('local')->put($image->nama,$file)){
        $arr="success-";
        $arr.=$image->nama;
    }
    else
        $arr="error-";
    return $arr;
}
```

4.2. *Save to Document Server*

Segmen Program 2 *Save to Document Server*

```
public function saveDocument(Request $request){
    $word=new PhpWord();
    $section=$word->addSection();

    $section->addText($request->dataOver);
    $section->addText($request->dataDes);

    $filename=$request->name.'.docx';
    if($word->save($filename,'Word2007'))
        return "success-".filesize($filename);
    else
        return "error";

}
```

Segmen Program 2 berguna untuk menyimpan semua informasi ke dalam sebuah *word document*. Proses ini menggunakan *library* PHPWord. *Library* ini hanya membantu proses pembuatan *word document* saja. *Word document* yang dibuat akan disimpan terlebih dahulu di *server*. Aplikasi dapat mengambil *document* tersebut pada fungsi *getDocument*.

4.3. *Detect Server*

Segmen Program 3 berguna untuk menjalankan fungsi dari *library* curl. Data gambar diambil dari *server* secara *online*. Setelah proses pengiriman data, *library* akan menunggu untuk mendapatkan *response*.

Segmen Program 5, Segmen Program 4, dan Segmen Program 6 berguna untuk mengecek sukses atau tidaknya *response* yang diterima. Apabila sukses, maka *response* akan dikirim ke *client*. Apabila gagal, maka akan dicek lagi alasan gagal. *Server* akan mengirim alasan gagal tersebut ke *client*.

Segmen Program 3 *Detect Server Part 1*

```
public function detect($img){
    $client = new CloudSight_Http_Client
        ("v6JhsmrE8a9K81lc7cMgxQ");
    $data=array();
    $request = $client->postImageRequests
        ("http://softdev.petra.ac.id/helpdesk/get/img/".$img);
    while(true){
        $result = $client->getImageResponses($request->token);
        // Check if analysis is complete.
        if ($client->isComplete()) {
            break;
        }
    }
```

Segmen Program 4 *Detect Server Part 2*

```
    else if($result->reason=="bright")
        $data['result']= "Image Too Bright to Identify";
    else if($result->reason=="unsure")
        $data['result']= "System is Unsure with the Image";
}
else if($result->status=="timeout"){
    $data['status']="error";
    $data['result']= "Detection exceed TTL setting";
}
```

Segmen Program 5 *Detect Server Part 3*

```
if($result->status=="not found"){
    $data['status']="error";
    $data['result']="Token Mismatch with Image";
}
else if($result->status=="skipped"){
    $data['status']="error";
    if($result->reason=="offensive")
        $data['result']= "Image Content Too Ofensive";
    else if($result->reason=="blurry")
        $data['result']= "Image Too Blurr to Identify";
    else if($result->reason=="close")
        $data['result']= "Image Too Close to Identify";
    else if($result->reason=="dark")
        $data['result']= "Image Too Dark to Identify";
```

Segmen Program 6 *Detect Server Part 4*

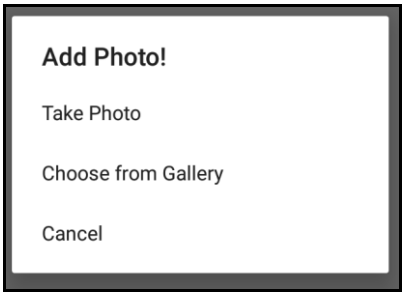
```
    else{
        $data['status']="success";
        $data['result']= $result->name;
    }

    return json_encode($data);
}
```

5. HASIL PENGUJIAN

Pengujian dilakukan dengan meng-*input* sebuah gambar dan *output* yang dihasilkan adalah informasi dalam bentuk *tab*. Gambar 2 adalah tampilan saat *user* mengklik tombol *capture image*. *User* dapat mengambil gambar dari kamera atau mengambil gambar dari *gallery*.

Gambar 3 adalah tampilan setelah gambar dipilih. Pada *mobile device screen* akan ditampilkan *preview* gambar yang sudah dipilih. Tombol *upload* akan *enable* setelah gambar dipilih.

Gambar 4 adalah tampilan hasil informasi setelah semua proses selesai. Hasil informasi ditampilkan dalam beberapa *tabs*. *Tabs* tersebut dapat di-*scroll*. Menu lainnya dapat ditemukan dengan menggeser logo garis-garis.



Gambar 2 Tampilan saat *User* memilih *Capture Images*

Gambar 3 Tampilan saat Gambar sudah dipilih

Gambar 4 Tampilan Hasil Informasi

6. KESIMPULAN DAN SARAN

Dari hasil perancangan dan pembuatan aplikasi deteksi citra yang diambil dari kamera mobile device untuk menghasilkan ensiklopedia fauna, dapat diambil kesimpulan antara lain:

- Aplikasi dapat berjalan dari sistem operasi Android Jellybean API 17 dan seterusnya
- Aplikasi juga menerapkan teknologi kamera untuk mendukung identifikasi.
- Aplikasi dapat memberikan informasi yang akurat yang berasal dari sumber terpercaya.
- Kecepatan aplikasi tidak berpengaruh pada hardware *device*.
- Aplikasi dapat menghasilkan word *document* sebagai dokumentasi dari hasil informasi yang diterima.

Saran yang diberikan untuk menyempurnakan dan mengembangkan aplikasi ini lebih lanjut antara lain:

- Desain *interface* dapat dibuat lebih menarik agar user yang membaca informasi dapat lebih nyaman.
- Dapat dilakukan deteksi fauna berdasarkan ras fauna tersebut. Sebagai contoh anjing chihuahua, anjing golden retriever, anjing herder, dsb.
- Pengembangan aplikasi tidak hanya mendeteksi fauna saja, namun dapat mendeteksi tumbuhan atau objek lainnya.
- Pengembangan aplikasi dalam mengambil informasi dari beberapa sumber yang lebih banyak.
- Pengembangan aplikasi dalam memberikan *alert* saat hewan yang dideteksi adalah hewan buas sehingga user dapat menghindar dari hewan tersebut

7. DAFTAR PUSTAKA

[1] Abdeen, H., & Shata, O. 2014. Type Variability and Completeness of Interfaces in Java Applications. *International Journal of Software Engineering & Applications*, 5(3), 1-7.

[2] Ali, A., Alrasheedi, M., Ouda, A., & Capretz, L. F. 2014. a Study of the Interface Usability Issues of Mobile Learning Applications for Smart Phones from the User's Perspective. *International Journal on Integrating Technology in Education, 3*(4), 1-16.

[3] Bassil, Y. 2012. Service-Oriented Architecture for Weaponry and Battle Command and Control Systems in Warfighting. *International Journal of Information and Communication Technology Research, 2*(2).

[4] Bharthan, A., & Bharathan, D. 2014. RelationalJSON, An Enriched Method to Store and Query JSON Records. *International Journal of Computer Applications, 98*(7), 3-6.

[5] Cheng, S. C., Hwang, W. Y., Wu, S. Y., Shadiev, R., & Xie, C. H. 2010. A Mobile Device and Online System with Contextual Familiarity and its Effects on English Learning on Campus. *Educational Technology & Society, 13*(3), 93-109.

[6] Felhi, F., & Akaichi, J. 2012. Adaptation of Web Services to the Context Based on Workflow:Approach for Self Adaptation of Service-Oriented Architectures to the Context. *International Journal of Web & Semantic Technology, 3*(4), 1-13.

[7] Gashti, M. Z. 2012. Investigating SOAP and XML Technologies in Web Service. *International Journal on Soft Computing, 3*(4), 15-19.

[8] Hengming, F., Jia, C., & Bin, X. 2013. The Interaction Mechanism based on JSON for Android Database Application. *Information Technology Journal, 12*(1), 224-228.

[9] Matos, V., & Grasser, R. 2014. Reactions to Two Software Approaches for Object Persistence. *International Journal of Applied Science and Technology, 4*(5), 20-29.

[10] Mumbaikar, S., & Padiya, P. 2013. Web Services Based On SOAP and REST Principles. *International Journal of Scientific and Research Publications, 3*(5), 1-4.

[11] Ramalingam, A., Dorairaj, P., & Ramamoorthy, S. 2012. Personal Safety Triggering System on Android Mobile Platform. *International Journal of Network Security & Its Applications, 4*(4), 179-197.

[12] Saab, C. B., Coulibaly, D., Haddad, S., Melliti, T., Moreaux, P., & Rampacek, S. 2009. An Integrated Framework for Web Services Orchestration. *International Journal of Web Services Research, 6*(4), 1-29.

[13] Sarker, I. H., & Apu, K. 2014. MVC Architecture Driven Design and Implementation of Java Framework for Developing Desktop Application. *International Journal of Hybrid Information Technology, 7*(5), 317-322.

[14] Wardoyo, O. E. 2012, march 7. *JAVA No.1: Belajar Java Pemula, Apa itu Java, cara install Java??* URI= http://www.mediatutorial.web.id/2012/03/java-no1-belajar-java-pemula-apa-itu.html

[15] Wikipedia, the free encyclopedia. n.d. *Computer vision*. URI= https://en.wikipedia.org/wiki/Computer_vision

[16] Yesilyurt, M., & Yalman, Y. 2016. Security Threats on Mobile Devices and their Effects: Estimations for the Future. *International Journal of Security and Its Applications, 10*(2), 13-26`.