BAB II
TINJAUAN PUSTAKA

II.1 Dodo Kids Browser

Dodo Kids Browser adalah sebuah aplikasi pengawasan internet anak yang digunakan oleh orang tua untuk mengontrol penggunaan internet anaknya dan oleh anak untuk melakukan pencarian informasi yang tidak memiliki konten negatif di internet. Aplikasi ini dibangun di dua *platform* yaitu *Windows Phone* dan *Extension Browser*.

Cara kerja dari Dodo Kids Browser yaitu pertama anak melakukan pencarian informasi di Internet menggunakan aplikasi Dodo Kids Browser di *platform mobile*, ketika kata kunci pencarian dimasukkan baik berdasarkan ketikan langsung ataupun melalui suara, aplikasi ini akan mengecek apakah kata kunci yang dimasukkan itu bersifat negatif atau positif. Jika kata kunci tersebut bersifat positif, maka aplikasi ini akan mengizinkan anak melakukan pencarian kata kunci tersebut di internet, sebaliknya jika kata kunci tersebut bersifat negatif maka aplikasi ini akan menghentikan pencarian kata kunci tersebut dan memberikan peringatan kepada anak dimana kata kunci tersebut bersifat negatif. Selama anak menggunakan aplikasi, riwayat pencarian akan ter rekam dan akan disimpan untuk dijadikan laporan bagi orang tua.

Cara kerja kedua dari Dodo Kids Browser yaitu pertama anak melakukan pencarian informasi di *browser* internet. *Extension Browser* aplikasi Dodo Kids Browser akan mengecek kata kunci pencarian informasi yang dimasukan oleh anak pada *browser* apakah kata tersebut bersifat positif atau negatif berdasarkan saran dari orang tua. Disini orang tua menggunakan aplikasi Dodo Kids Browser pada *platform mobile* untuk memberikan saran dimana setiap saran tersebut terdapat jenis aksi yang diberikan.

II.2 Microsoft Azure

Microsoft Windows Azure adalah sebuah teknologi untuk *cloud computing* yang di kembangkan oleh Microsoft. Dengan menggunakan Windows Azure, pengembang dapat membuat aplikasi web, sebagai tempat penyimpanan data suatu aplikasi, sebagai jembatan koneksi antar *platform*, dan sebagainya. Berikut ini adalah komponen yang terdapat dalam Windows Azure:

1. *Cloud Applications*

   Pada bagian ini terdapat dua tipe peran, yaitu:

   1. Web roles, yaitu peran yang berinteraksi langsung dengan pengguna.
   2. Worker roles, yaitu peran yang lebih dikhususkan dalam pemrosesan data.

2. *Data Management*

   Setiap aplikasi Microsoft Azure dijalankan di satu atau lebih VM (*Virtual Memory*) dimana setiap VM ini memiliki kapasitas penyimpanan lokal. Oleh karena itu, Windows Azure menyediakan lima jenis *data management*, yaitu:

   1. SQL Azure
   2. Tables
   3. Windows Azure Blobs Storage Services
   4. Import/Export
   5. Azure File Service

3. *Networking*

   Karena Microsoft Azure memiliki banyak *datacenters* di dunia maka ketika pengembang akan menjalankan aplikasi atau akan menyimpan data, pengembang dapat memilih satu atau lebih *datacenters* untuk digunakan berikut jenis koneksi yang digunakan, seperti:

   1. Virtual Network
   2. Traffic Manager
   3. Express Route

4. *Identity and Access Control*

Setiap aplikasi yang dijalankan pasti membutuhkan suatu informasi *user* yang menggunakan aplikasi tersebut. Oleh karena itu, Microsoft Azure menyediakan *services* yang digunakan untuk melakukan perekaman aktivitas yang dilakukan oleh *user* dengan menggunakan:

1. Active Directory
2. Multi Factor Authentication

5. *Messaging and Integration*

   Dalam melakukan manajemen pesan yang disampaikan berikut integrasi data antar perangkat/*platform,* maka Microsoft Azure menyediakan beberapa cara, yaitu:

   1. Storage Queue
   2. Service Bus Queue
   3. Service Bus Relay
   4. Service Bus Topics
   5. Biz Talk Hybrid Connection
   6. Biz Talk Service

Dengan menggunakan teknologi Microsoft Azure, didapat beberapa keuntungan yaitu:

1. Pengembang hanya membuat perangkat lunak/sistem karena Microsoft akan melindungi sistem tersebut termasuk jika akan dilakukan pembaharuan atau *backup* data.
2. Mudah dalam menambah jumlah kapasitas penyimpanan data.
3. Terintegrasi dengan aplikasi Microsoft Visual Studio.
4. Adanya manajemen seperti pada *Application Server* yang digunakan. [6]

II.3 Web Service

Web Service adalah sebuah *software* yang dirancang untuk mendukung interoperabilitas interaksi antar mesin melalui sebuah jaringan baik berupa agregasi (pengumpulan) maupun sindikasi (penyatuan). *Web Service* melayani integrasi data dan kolaborasi informasi yang bisa diakses melalui internet oleh pengguna dengan teknologi yang dimilikinya. Pemanggilan *web service* dapat dilakukan oleh bahasa

pemrograman apa saja dan dalam *platform* apa saja. Berikut ini adalah kelebihan *web Service*:

1. Lintas *platform*.
2. *Language Independent*.
3. Jembatan penghubung dengan *database* tanpa perlu *driver database* dan tidak harus mengetahui jenis DBMS yang digunakan.
4. Mempermudah proses pertukaran data.
5. Penggunaan kembali komponen aplikasi.

Dalam perkembangannya, *web service* mengenal dua metode yang berorientasi pada layanan dan sumber daya informasi, yaitu metode SOAP (*Simple Object Access Protocol*) dengan proses layanannya memiliki tiga komponen utama, yaitu:

1. *Service Provider*
2. *Service Requester*
3. *Service Broker*

Serta komponen pendukung yaitu:

1. XML.
2. SOAP-XML (terdiri atas *header* dan *body*).
3. WSDL.
4. UDDI.

Dan metode REST (*REpresentational State Transfer*) yang didasari oleh empat prinsip utama teknologi, yaitu:

1. *Resource identifier through Uniform Resource Identifier* (URI).
2. *Uniform Interface*, yaitu sumber daya CRUD menggunakan operasi PUT, GET, POST, dan DELETE.
3. *Self-descriptive messages*, yaitu sumber daya tidak terikat sehingga dapat mengakses konten HTML, XML, PDF, JPEG, *plain text*, *meta data*, dan lainnya.
4. *Stateful interactions through hyperlinks*.

Metode REST ini lebih sederhana karena menggunakan format standar (HTTP, HTML, XML, URI, MIME), dan jika diperlukan untuk proses pertukaran

data, maka konten hasil eksekusi *web service* dapat diolah dalam format teks (XML atau HTML) dengan menggunakan utilitas komunikasi data berupa koneksi *socket* protokol HTTP yang tersedia pada pustaka komunikasi bahasa pemrograman. [7]

II.4 JSON (*JavaScript Object Notation*)

JavaScript Object Notation adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari bahasa pemrograman JavaScript, Standar ECMA-262 Edisi ke-3 – Desember 1999. JSON tidak bergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh pengembang/*programmer* keluarga bahasa C (C, C++, C#, Java, JavaScript, Perl, Phyton, dan lainnya) sehingga JSON ideal sebagai bahasa pertukaran data.

Struktur JSON sendiri terdiri dari:

1. Kumpulan pasangan nama/nilai seperti objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel *hash* (*hash table*), daftar berkunci (*keyed list*) atau *associative array*.
2. Daftar nilai terurutkan seperti larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

Bentuk dari JSON terdiri dari:

1. **Objek/*Object*** (pasangan nama/nilai yang tidak terurutkan). Dalam objek tersebut terdapat kurung kurawal buka untuk mengawali dan diakhiri oleh kurung kurawal tutup ({ }), dalam kurung kurawal tersebut terdapat nama yang diikuti dengan titik dua (:) dan setiap pasangan nama/nilai dipisahkan oleh koma (,).
2. **Larik/*Array***, kumpulan nilai yang terurutkan. Larik dimulai dengan kurung kotak buka dan diakiri dengan kurung kotak tutup ([ ]) dan setiap nilai dipisahkan oleh koma (,).

3. **Nilai/*Value***, dapat berupa sebuah *string* dalam tanda kutip ganda (“ “) atau angka, atau nilai *true*, *false*, dan *null*, atau objek atau larik. Struktur-struktur tersebut dapat disusun bertingkat.
4. **String**, adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Dalam string dapat digunakan *backslash escapes* “\” untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada *string* dan *string* sangat mirip dengan *string* C atau Java.
5. **Angka**, nilai yang sangat mirip dengan angka di C atau Java, kecuali format Oktal dan Heksadesimal tidak digunakan. Spasi kosong (*whitespace*) dapat disisipkan di antara pasangan tanda-tanda tersebut, kecuali beberapa detail *encoding* yang secara lengkap dipaparkan oleh bahasa pemrograman yang digunakan. [8]

II.5 API (*Application Programming Interface*)

Application Programming Interface adalah sekumpulan perintah, fungsi, komponen dan protokol yang disediakan oleh sistem operasi ataupun bahasa pemrograman tertentu yang dapat digunakan oleh pengembang/*programmer* saat membangun perangkat lunak. Pada API terdapat fungsi-fungsi/perintah untuk menggantikan bahasa yang digunakan dalam *system call* dengan bahasa yang lebih mudah dimengerti oleh *programmer*.

Kelebihan dari API adalah:

1. Profitabilitas, *programmer* yang menggunakan API dapat menjalankan programnya dengan sistem operasi mana saja asalkan sudah ter-*install* API tersebut.
2. Mudah Dimengerti, bahasa yang digunakan lebih terstruktur dan mudah dimengerti oleh *programmer* daripada bahasa *system call*. [9]

II.6 Structured Analysis and Design

Structured Analysis and Design adalah salah satu metode dalam pendekatan analisis dan perancangan suatu perangkat lunak. Pada metode ini dilakukan pendekatan secara terstruktur untuk setiap fungsionalitas yang terdapat pada perangkat lunak. Dalam penerapannya, metode ini menerapkan pemodelan-pemodelan sebagai berikut pada Tabel II.1. [10]

Tabel II.1 Pemodelan pada SAD

Nama Pemodelan	Keterangan
Dataflow diagram (DFD)	Pemodelan yang menggambarkan aliran data untuk setiap proses pada sistem. Penggambaran pemodelan ini dilakukan dari yang sifatnya umum ke khusus.
Data-dictionary	Kamus data yang di dalamnya terdapat informasi mengenai komponen data, *files*, dan proses yang dilakukan.
Entity-relationship diagram (ERD)	Pemodelan yang menggambarkan entitas-entitas yang saling terhubung melalui relasi dan kardinalitas yang mengikat entitas-entitas tersebut.
Hierarchy diagram	Sebuah *simple diagram* yang memperlihatkan *top-to-bottom* dekomposisi hierarki dari data *files* dan data *items*.
Mini-spec	Spesifikasi struktural bahasa Inggris dari pendetailan suatu logika prosedural dalam suatu proses, memiliki fungsi yang sama dengan tradisional *flow-chart*. Pemodelan ini dibentuk untuk setiap proses dalam level sebuah DFD.
State-transition diagram	Menggambarkan perbedaan kondisi dari sistem atau komponen sistem dan kejadian atau pesan yang menyebabkan transisi keadaan.
Structure chart	Menggambarkan arsitektur sistem dalam bentuk hierarki fungsi seperti bentuk pohon struktural. Memiliki informasi seperti identitas, fungsi, *input* dan *output*.

II.7 Algoritma Boyer-Moore

Algoritma Boyer-Moore diciptakan oleh R.M Boyer dan J.S Moore. Algoritma ini terkenal karena banyak diterapkan pada kasus pencocokan *string* yang cepat dalam kinerjanya. Algoritma Boyer-Moore melakukan pencocokan karakter dimulai dari kanan ke kiri. Karakter paling kanan pada pola merupakan karakter pertama yang akan dicocokkan dengan teks. Algoritma ini mempunyai dua fase, yaitu fase *preprocessing* yang terdapat dua buah fungsi untuk menggeser pola ke arah kanan yaitu:

1. *Good-Suffix-Shift*, fungsi ini disimpan ke dalam tabel bmGs (boyer-moore Good suffix) berukuran m+1.
2. *Bad-Character-Shift*, fungsi ini disimpan ke dalam sebuah tabel bmBc (boyer-moore Bad character) yang berukuran n.

Dan fase pencarian *string* tersebut.

Pembentukan tabel bmBc dan bmGs mempunyai kompleksitas waktu $O(m+n)$ dan kompleksitas ruang $O(m+n)$. Kompleksitas waktu untuk fase pencarian adalah $O(mn)$. [11]

II.8 Pengujian Perangkat Lunak

Suatu perangkat lunak yang dibangun sebelum digunakan oleh pengguna, pengembang/*developer* harus melakukan pengujian terhadap perangkat lunak yang dia bangun. Pengujian perangkat lunak dapat mendeteksi dan menemukan suatu kesalahan dalam perangkat lunak. Tujuan adanya pengujian perangkat lunak adalah untuk menunjukkan bahwa kesalahan perangkat lunak ada, sehingga pengujian yang baik bukan untuk memastikan tidak ada kesalahan tetapi untuk mencari sebanyak mungkin kesalahan yang ada dalam perangkat lunak.

Berikut ini adalah beberapa teknik pengujian perangkat lunak:

1. *White Box*

Pengujian dilakukan pada logika internal dan struktur *code* dari perangkat lunak sehingga penguji (*tester*) harus memiliki pengetahuan terhadap *source code* dari perangkat lunak.

2. *Black Box*

Pengujian dilakukan tanpa harus memiliki pengetahuan dalam kerja internal sebuah aplikasi. Pengujian ini hanya dikhususkan pada aspek-aspek dasar dari sistem (fungsional sistem).

3. *Grey Box*

Gabungan dari pengujian *white box* dan *black box*, sehingga dalam penerapannya *tester* harus memiliki pengetahuan tentang kerja internal dan aspek-aspek dasar dari perangkat lunak.

Berikut ini adalah perbedaan dari tiga teknik pengujian pada Tabel II.2. [12]

Tabel II.2 Perbedaan Teknik Pengujian Perangkat Lunak

No	Black Box Testing	Grey Box Testing	White Box Testing
1	Berfokus pada aspek dasar (fungsional sistem)	Beberapa pengetahuan tentang kerja internal sistem	Pengetahuan penuh terhadap kerja internal sistem
2	Granularity rendah	Granularity sedang	Granularity tinggi
3	Dilakukan oleh pengguna (*end users*) dan pengembang (*developer*) juga penguji (*tester*) (*User Acceptance Testing*)	Dilakukan oleh pengguna (*end users*) dan pengembang (*developer*) juga penguji (*tester*) (*User Acceptance Testing*)	Dilakukan oleh pengembang (*developer*) dan penguji (*tester*)
4	Pengujian didasari pada eksepsi eksternal, kerja internal program diabaikan	Pengujian didasari pada tingkatan *database diagram, data flow diagram*¸ *internal states,* pengetahuan algoritma dan arsitektur	Pengujian didasari pada pengetahuan internal sistem
5	Agak lengkap dan memakan waktu	Pertengahan antara *black box* dan *white box*	Berpotensi lebih lengkap dan memakan waktu

No	Black Box Testing	Grey Box Testing	White Box Testing
6	Hanya dapat diuji dengan teknik *trial and error*	Jika diketahui dapat melakukan pengujian *data domains* dan *internal boundaries*	Pengujian lebih baik: *data domains* dan *internal boundaries*
7	Tidak cocok untuk pengujian algoritma	Tidak cocok untuk pengujian algoritma	Cocok untuk pengujian algoritma