3. DESAIN DAN IMPLEMENTASI

Bab ini berisi penjelasan mengenai desain dari sistem secara keseluruhan. Mulai dari alur kerja sistem, hingga fitur-fitur yang dirancang di dalam sistem. Di dalam bab ini pula, akan dibahas cara mengimplementasikan desain dan rancangan dari sistem, sehingga sistem bisa digunakan dengan baik.

3.1. Gambaran Umum Sistem Tugas Akhir

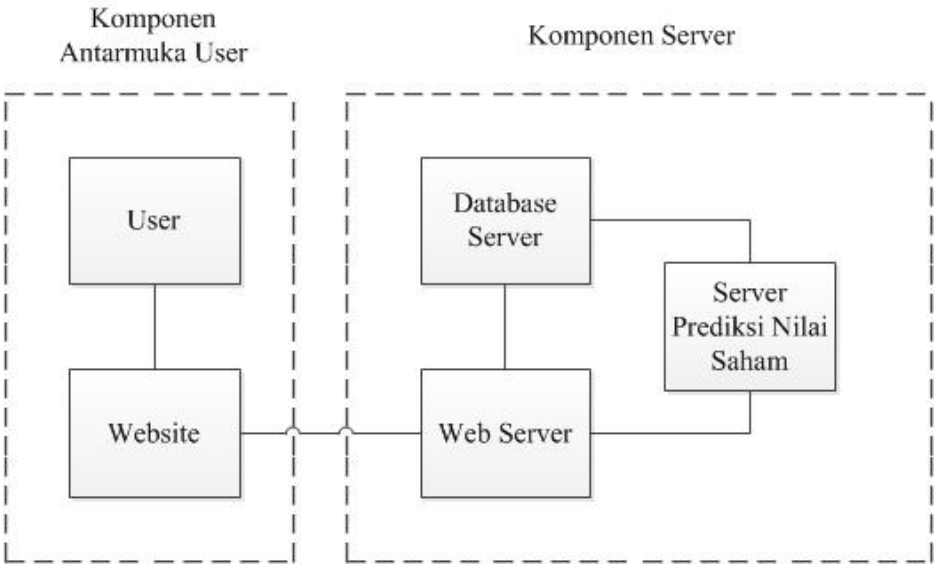
Di dalam Tugas Akhir ini akan dibuat sebuah sistem akses *database* prediksi nilai saham. Saham-saham yang ada di dalam sistem adalah saham-saham dari Bursa Efek Jakarta. Data saham tersebut akan disimpan di dalam *database* sebelum disajikan kepada *user* dalam bentuk *website*. Data prediksi nilai saham yang disediakan oleh *website* tersebut bersifat penting dan komersial. Oleh karena itu tidak semua pengunjung *website* diperbolehkan untuk melihat dan mengetahui nilai prediksi tersebut.

Di dalam sistem dibuat sebuah mekanisme authentikasi yang melakukan suatu pembatasan akses menuju data-data prediksi nilai saham. Tidak semua pengunjung *website* nantinya dapat melihat data prediksi tersebut. Pengunjung yang diperbolehkan adalah mereka yang sudah mempunyai *account* di dalam *website*. Pengunjung yang tidak memiliki *account* masih bisa masuk ke dalam *website* dengan beberapa batasan, yaitu ada beberapa bagian tertentu dari *website* yang tidak bisa diakses.

Bagi pengunjung yang ingin membuat *account*, mereka harus melalukan proses registrasi terlebih dahulu di dalam *website*. Secara garis besar, proses registrasi berlangsung dengan melalui dua tahap. Tahap pertama adalah mengisi data pribadi untuk kepentingan *account*, tahap berikutnya adalah melakukan transaksi pembayaran. *User* dapat memilih paket berlangganan *account* sesuai kebutuhannya. Pembayaran dapat dilakukan dengan dua cara, yaitu dengan menggunakan PayPal dan BCA. Setelah pengunjung selesai melakukan registrasi dan transaksi pembayaran sudah berhasil divalidasi oleh sistem, maka pengunjung bisa mengakses data prediksi nilai saham yang disajikan di dalam *website*.

3.2. Blok Diagram Sistem Tugas Akhir

Secara garis besar, sistem tersusun menjadi dua buah bagian yaitu komponen antarmuka *user* dan komponen *server*. Komponen antarmuka *user* adalah bagian yang berinteraksi langsung dengan *user* yaitu sebuah *website*. Di dalam *website* itulah fitur-fitur yang disediakan oleh sistem disajikan kepada *user*, termasuk informasi data prediksi nilai saham. Selain hal itu, *website* tersebut juga menjadi saluran komunikasi antara *user* dengan sistem. Hal ini membuat peran dari *website* sangatlah penting. Kinerja dan penampilan dari *website* akan sangat mempengaruhi *user* karena hal yang dilihat pertama kali oleh *user* adalah *website* tersebut. Apabila *website* menarik dan dapat melakukan kinerjanya dengan baik, maka *website* akan ramai dikunjungi oleh pengunjung.



Gambar 3.1. Blok diagram dari sistem di dalam Tugas Akhir.

Komponen berikutnya adalah komponen *server*. Komponen *server* merupakan komponen yang menyimpan informasi dan berkas-berkas yang dibutuhkan oleh sistem. Komponen *server* dalam sistem ini dibagi lagi menjadi 3 buah bagian, yaitu web *server*, *database server*, dan *server* prediksi nilai saham. Web *server* yaitu sebuah *server* yang menyimpan data-data dan *file-file* yang dibutuhkan untuk menampilkan *website*, seperti *file* html, php, css, dan sebagainya. Di dalam web *server* berisi kode-kode program yang digunakan untuk menjalankan sistem. Mulai dari program untuk melakukan *update* data saham dan

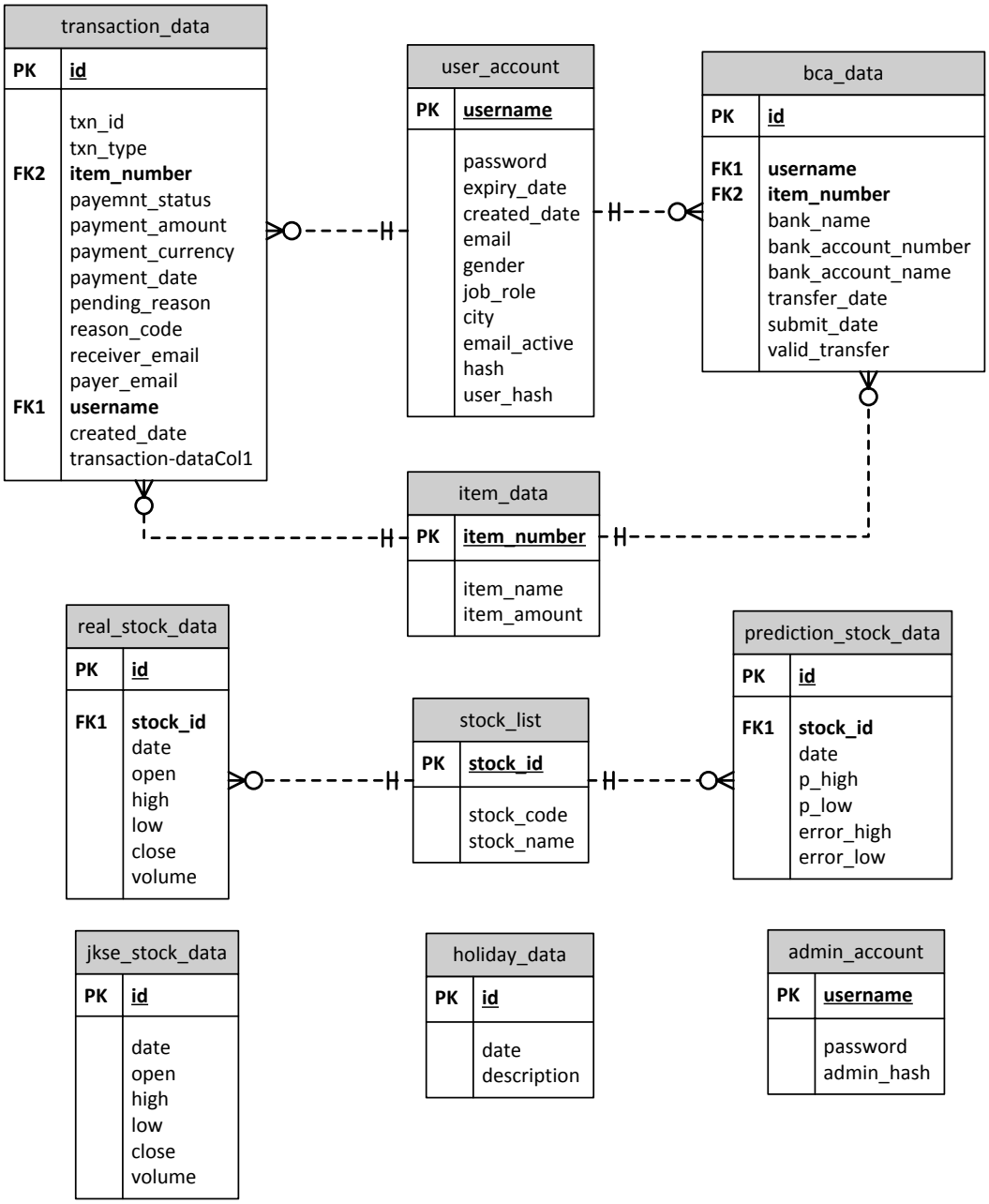
data prediksi saham, program untuk melakukan proses registrasi, hingga menampilkan data-data yang ada di dalam *database* ke dalam *website*.

Database server adalah *server* yang berisi data-data pelengkap yang dibutuhkan oleh sistem. Segala informasi-informasi yang dibutuhkan oleh *website* disimpan di dalam *server* ini, misalnya data mengenai *account* dari *user*, data asli nilai saham, data prediksi nilai saham, data transaksi pembayaran, dan data-data lain yang digunakan di dalam sistem. *Database server* merupakan pusat data informasi dari sistem.

Komponen *server* yang terakhir adalah *server* prediksi nilai saham. Di dalam *server* tersebut, terdapat sistem lain yang akan menghasilkan *file* CSV yang berisi data prediksi nilai saham. *File* CSV tersebut nantinya akan dipindahkan ke dalam *database* dan ditampilkan di dalam *website*.

3.3. Desain *Database*

Untuk menjalankan tugasnya, sistem sangat membutuhkan keakuratan dan kredibilitas dari data. Untuk menunjang hal tersebut, perlu dilakukan desain *database* agar data-data yang disimpan di dalam sistem tidak berantakan dan menjadi tidak efisien. *Database* yang digunakan di dalam sistem ini disimpan dengan menggunakan MySQL pada *server* Ubuntu Linux. Berikut ini adalah susunan dari tabel-tabel didesain di dalam *database*.



Gambar 3.2. Struktur *database* sistem

Database terdiri atas 10 tabel yang masing-masing tabel memiliki kolom yang berbeda-beda. Tabel yang pertama yaitu adalah tabel *user_account*. Tabel ini berisi informasi-informasi yang berkaitan dengan *user*. Tabel ini memiliki kolom-kolom sebagai berikut,

- *Username*, yaitu nama yang digunakan oleh *user* untuk masuk ke dalam *website*. *Username* ini merupakan *Primary key*, sehingga tidak boleh ada *user* yang memiliki *username* yang sama.

- *Password*, merupakan *password* yang digunakan untuk masuk ke dalam *website* pada proses *Sign In*.
- *Expiry_date* merupakan tanggal berakhirnya masa aktif dari *account*.
- *Created_date* merupakan tanggal *account* tersebut di buat.
- *Email* adalah alamat *email* dari *user*. *Email* tersebut akan digunakan untuk melakukan komunikasi dengan *user*.
- *Gender* berisi informasi jenis kelamin *user*.
- *Job Role* berisi informasi pekerjaan dari *user*.
- *City* berisi informasi dari kota temapat tinggal *user*.
- *Email_active* meyimpan data informasi apakah *user* yang bersangkutan sudah melakukan verifikasi *email* atau belum. *Email_active* bernilai 1 bila *user* sudah melakukan verifikasi, dan bernilai 0 bila belum.
- *Hash* merupakan 32 random karakter yang digunakan oleh sistem untuk melakukan verifikasi *user* pada saat melakukan registrasi.
- *User_hash* merupakan 32 random karakter yang digunakan oleh sistem untuk melakukan verifikasi *user* pada saat melakukan *Sign In*.

Tabel berikutnya adalah *transaction_data*. Tabel ini berisi informasi yang berhubungan dengan pembayaran melalui PayPal. Tabel ini berisi,

- *Id*, yaitu *Primary key* dari tabel tersebut.
- *Txn_id* merupakan *id* dari transaksi yang digunakan di dalam PayPal.
- *Txn_type* merupakan jenis pembayaran yang dilakukan di dalam PayPal.
- *Item_number* merupakan jenis *account* berlangganan yang dibeli oleh *user*.
- *Payment_status* adalah status dari pembayaran yang dilakukan di dalam PayPal. Status akan bernilai *Completed* bila transaksi usdah berhasil berjalan.
- *Payment_amount* adalah banyaknya uang yang digunakan dalam transaksi. Banyaknya uang sesuai dengan harga jenis *account* berlangganan yang dipilih,
- *Payment_currency*, mata uang yang digunakan.
- *Payment_date*, tanggal transaksi dilakukan oleh PayPal.

- *Pending_reason* berisi informasi penyebab terjadinya penundaan dalam proses validasi pembayaran. *Pending_reason* ini muncul bila ada pembayaran yang bersifat *pending*.
- *Reason_code* berisi informasi penyebab terjadinya kegagalan dalam transaksi. *Reason_code* ini muncul hanya bila pembayaran yang dilakukan gagal.
- *Receiver_email* adalah *email* PayPal dari penjual atau sistem.
- *Payer_email* adalah *email* PayPal dari pembeli.
- *Username* adalah Foreign *Key* dari tabel suer_*account*
- *Created_date* menunjukkan tanggal dimana informasi yang berhubungan dengan transaksi PayPal tersebut disimpan dalam *database*.

Tabel berikutnya adalah *bca_data*. Pada tabel ini berisi informasi yang berhubungan dengan pembayaran melalui BCA. Tabel inii berisi,

- *Id* merupakan *Primary Key* dari tabel tersebut.
- *Username* adalah *Foreign Key* dari tabel *user_account*
- *Item_number* adalah jenis *account* yang dipilih.
- *Bank_name* adalah nama bank pembeli yang digunakan untuk pembayaran
- *Bank_account_number* adalah nomor rekening pembeli.
- *Bank_account_name* adalah nama rekening dari pembeli.
- *Transfer_date* adalah tanggal proses transfer dilakukan.
- *Submit_date* merupakan tanggal *username* mengisi form konfirmasi pembayaran.
- *Valid_transfer* adalah informasi apakah data yang dikirim oleh *user* benar atau salah. *Valid_transfer* bernilai 1 bila data yang dikirim benar dan pembayaran sudah dikonfirmasi oleh admin. Untuk pembayaran yang belum dikonfirmasi akan bernilai 0.
- Admin adalah nama dari admin yang melakukan konfirmasi pembayaran.

Tabel *item_data* berisi tentang informasi jenis dan harga *account* langganan yang disediakan oleh sistem. Tabel tersebut berisi,

- *Item_number* merupakan nomor dari produk. Nomor ini juga merupakan *Primary key* dari tabel.
- *Item_name* adalah nama dari jenis *account* langganan.

- *Item_amount* merupakan harga dari jenis *account* langganan.

Tabel *stock_list* berisi informasi jumlah saham perusahaan yang ada di dalam sistem. Tabel ini berisi,

- *Stock_id* merupakan *Primary key* dari tabel. Menunjukkan id dari saham di dalam sistem.
- *Stock_code* merupakan kode dari saham tersebut.
- *Stock_name* adalah nama perusahaan dari saham.

Tabel *real_stock_data* berisi segala informasi tentang data nilai asli saham dari Yahoo Finance. Tabel ini berisi,

- *Id*, merupakan *Primary key* dari tabel.
- *Stock_id* adalah *Foreign Key* dari tabel *stock_list.*
- *Date* merupakan tanggal dari nilai saham.
- *Open,* yaitu harga pembuka dari saham.
- *High*, yaitu nilai tertinggi dari harga saham.
- *Low*, yaitu nilai terendah dari harga saham.
- *Close,* yaitu harga penutup dari saham.
- *Volume,* yaitu nilai *volume* dari saham.

Tabel *prediction_stock_data* berisi segala informasi mengenai data nilai prediksi saham yang didapatkan dari *file* CSV. Tabel ini berisi,

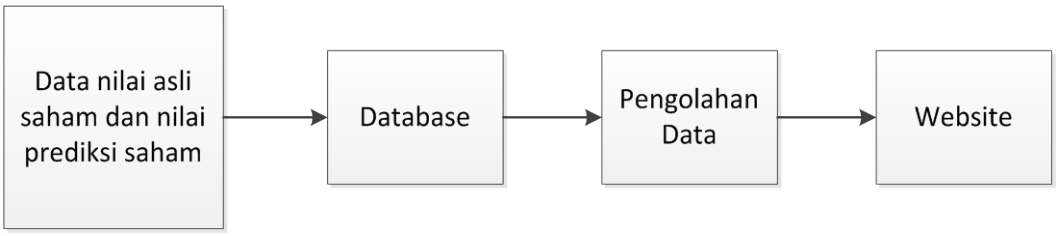
- *Id*, merupakan *Primary key* dari tabel.
- *Stock_id* adalah *Foreign Key* dari tabel *stock_list.*
- *Date* merupakan tanggal dari saham.
- *P_high*, yaitu nilai prediksi harga tertinggi saham.
- *P_low,* yaitu nilai prediksi harga terendah saham.
- *Error_high*, yaitu presentase kesalahan dari nilai prediksi *high*
- *Error_low*, yaitu persentase kesalahan dari nilai prediksi *low*.

Tabel *jkse_stock_data* berisi informasi data nilai asli dari Indeks Harga Saham Gabungan (IHSG) yang didapatkan dari Yahoo Finance. Stuktur dari tabel sama seperti pada tabel *real_stock_data* yaitu *date*, *open*, *high*, *low*, *close*, dan *volume*. Tabel berikutnya yaitu admin_*account*. Tabel ini berisi informasi tentang Admin. Kolom-kolom yang terdapat dalam tabel ini mirip dengan *user_account*, yaitu *username*, *password*, dan *admin_hash*. Tabel yang terakhir yaitu

holiday_date. Tabel ini digunakan untuk menyimpan informasi hari libur dari bursa. Di dalam tabel tersebut berisi kolom *date* yaitu tanggal hari libur, dan *decription* yaitu penjelesan mengenai hari libur tersebut.

3.4. Alur Kerja Sistem Tugas Akhir

Setiap hari sistem mempunyai tugas untuk melakukan *update* data nilai asli saham dan nilai prediksi saham. Tugas ini perlu dilakukan untuk menjaga agar data-data yang ada di dalam sistem adalah data yang terbaru. Sistem melakukan tugasnya dengan mengikuti alur seperti pada Gambar 3.3.



Gambar 3.3. Alur kerja dari sistem di dalam Tugas Akhir.

Data nilai asli saham didapatkan dari Yahoo Finance, sedangkan data nilai prediksi saham didapatkan dari *file* CSV yang ada di dalam *server* sistem. Kedua data tersebut kemudian dimasukkan ke dalam *database* sesuai dengan tabel yang telah dijelaskan pada sub bab sebelumnya. Data yang ada di dalam *database* kemudian dilakukan proses pengolahan data sebelum ditampilkan ke dalam *website*. Pengolahan data yang dilakukan meliputi,

- Menentukan lima saham yang mengalami kenaikan harga saham tertinggi.
- Menentukan lima saham yang mengalami penurunan harga saham terbanyak.
- Menentukan lima saham yang memiliki prediksi terbaik.
- Menentukan persentase *error* harian dan *error* mingguan dari nilai prediksi saham.

Data hasil pengolahan kemudian ditampilkan di dalam *website* sesuai dengan fitur-fitur yang disediakan di dalam *website*. Sistem di dalam Tugas Akhir ini memiliki 10 fitur yaitu,

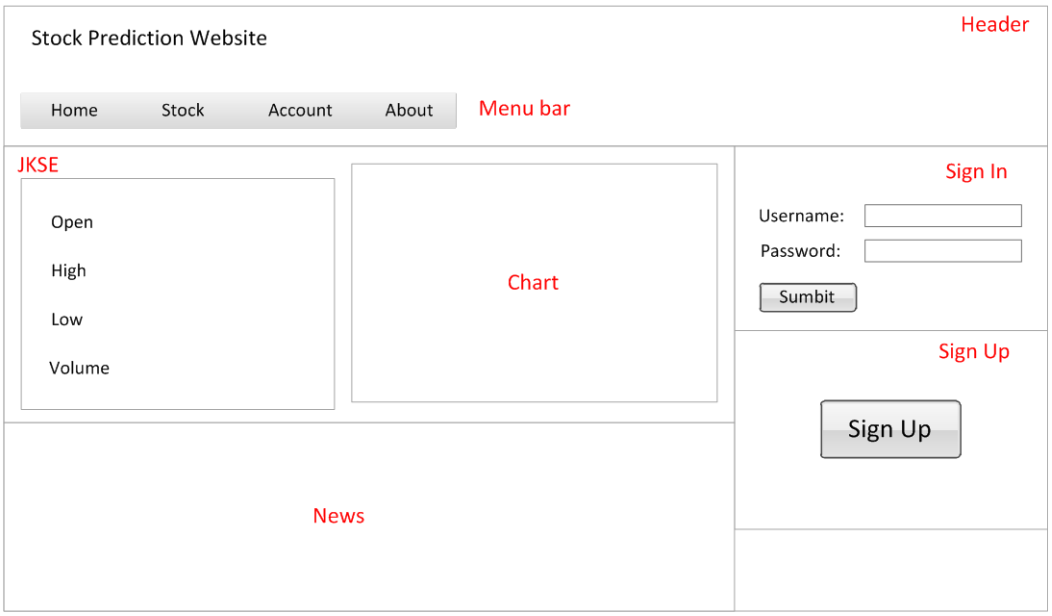
- Fitur *Home*, merupakan tampilan awal dari wesbite.

- Fitur Registrasi, yaitu bagian dari *website* yang menyediakan layanan registrasi atau pendaftaran *account* bagi *user*.
- Fitur *Sign In*, yaitu bagian yang digunakan oleh *user* untuk memasukkan *account* mereka di dalam *website*.
- Fitur *Stock* merupakan bagian yang menampilkan data nilai asli saham dan data nilai prediksi saham.
- Fitur *Chart* merupakan grafik yang menampilkan data nilai saham.
- Fitur *News* menampilkan informasi berita bisnis, ekonomi, dan informasi lain yang berkaitan dengan saham-saham yang ada di dalam sistem.
- Fitur *Auto Update* Data merupakan bagian *back-office* dari sistem, yang melakukan proses *update* data nilai asli dan data nilai prediksi dari saham.
- Fitur *Account*, yaitu bagian yang menampilkan informasi *account* dari *user*.
- Fitur *About*, yaitu menampilkan keterangan dari *website*.
- Fitur *Admin Center*, yaitu bagian *back-office* sistem yang digunakan oleh Administrator untuk melakukan tugasnya.

3.5. Fitur *Home*

Home merupakan tampilan awal dari sebuah *website*. Tampilan awal ini harus dibuat semenarik mungkin agar bisa menarik perhatian pengunjung atau *user*. Di dalam sistem ini tampilan *Home* terdiri dari 4 bagian dasar yaitu,

- Tampilan informasi Indeks Harga Saham Gabungan (IHSG)
- Tampilan Berita dari Yahoo Finance RSS
- Tampilan untuk *Sign In*
- Tampilan untuk *Sign Up* atau mendaftar *account*



Gambar 3.4. Layout tampilan dari fitur *Home*.

Gambar 3.4 adalah desain Layout dari tampilan *Home* yang nantinya akan diterapkan oleh sistem. Di dalam IHSG, terdapat informasi mengenai nilai *Open*, *High*, *Low*, *Close*, dan *Volume* dari IHSG Jakarta. Selain itu juga terdapat grafik yang akan menunjukan perubahan nilai harga saham gabungan secara periodik. Cara untuk menampilkan grafik sama dengan yang dijelaskan pada Fitur Chart. Di dalam tampilan Berita, akan disajikan berita terkini yang didapatkan dari RSS 2.0 Yahoo Finance. URL yang dipakai untuk mengambil data berita terlihat pada Gambar 3.5.

```
parser("http://id.berita.yahoo.com/rss/bisnis");
parser("http://id.berita.yahoo.com/rss/ekonomi/");
```

Gambar 3.5. URL RSS Yahoo Finance yang digunakan untuk menampilkan berita.

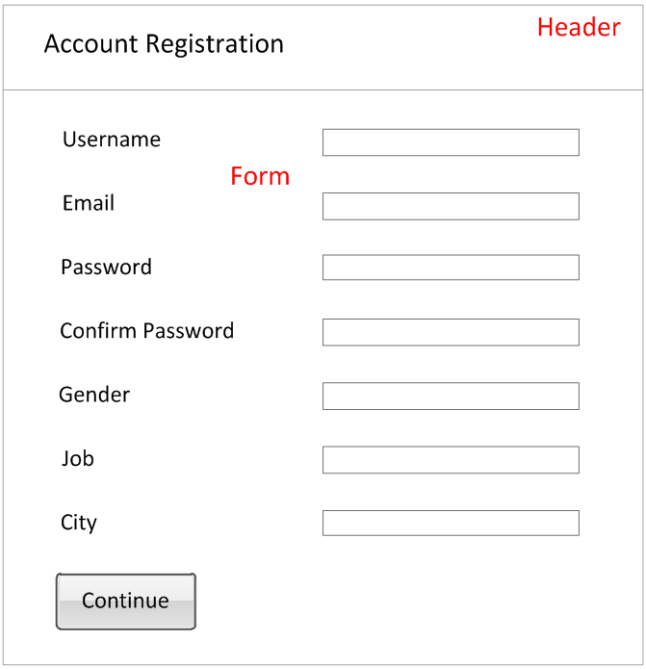
Berita dari URL tersebut kemudian akan diolah dan dipisah-pisah untuk mendapatkan berita yang diinginkan. Masing-masing berita dipisahkan oleh <*tag*> sehingga harus ada program tertentu untuk memisahkannya. Cara yang digunakan untuk memisahkan berita dijelaskan pada Fitur *News*. Tampilan *Sign In* berupa dua buah *textfield* untuk *Username* dan *Password* serta satu buah tombol

untuk Submit. Dari tempat inilah *user* yang sudah mempunyai *account* akan masuk dan dapat mengakses data-data prediksi yang telah disediakan. Tampilan *Sign Up* berupa satu buah tombol yang akan mengantarkan *user* yang ingin membuat *account* ke halaman Registrasi. Setelah itu *user* dapat memulai membuat *account* dengan mengikuti beberapa langkah, seperti yang dijelaskan pada Fitur Registrasi.

3.6. Fitur Registrasi

Sistem membatasi *user* yang dapat melihat data-data prediksi yang disajikan di dalam *website*. Cara membatasinya adalah dengan metode berlangganan *account* berbayar. *User* yang hendak melihat data-data prediksi tersebut harus membuat *account* di dalam *website* terlebih dahulu. Oleh karena itu, di dalam sistem disediakan fitur bagi *user* untuk melakukan registrasi *account* dan melakukan pembayaran. Proses registrasi menjadi bagian yang sangat penting karena bagian ini merupakan sumber dana bagi sistem. Proses ini dibagi ke dalam 4 langkah yaitu,

3.6.1. Langkah pertama, memasukkan data *user*



Gambar 3.6. Layout tampilan form registrasi

Di dalam langkah pertama ini, *user* diminta untuk memasukkan data-data tertentu. Data-data tersebut ada yang bersifat penting dan digunakan oleh sistem, seperti *username*, *email*, dan *password*. Selain itu ada juga data-data yang bersifat pelengkap atau digunakan sebagai data mining. Data tersebut adalah jenis kelamin, pekerjaan, dan kota. Sistem menyediakan sebuah form di dalam *website* sehingga *user* bisa memasukkan data-data tersebut dengan mudah. Desain Layout dari form yang digunakan terlihat pada gambar 3.6.

Pada saat *user* memasukkan data ke dalam form, *user* bisa mengetik apa saja pada form tersebut. Ada kemungkinan bahwa data yang diketik oleh *user* tidak sesuai dengan form yang disediakan. Contohnya pada bagian *email* diisikan alamat *user* atau diisikan alamat *email* yang tidak valid. Ada juga kemungkinan bahwa *user* tidak mengisi form tersebut, misalnya *user* lupa mengisi bagian *username*, padahal *username* dibutuhkan oleh sistem untuk proses *Sign In*. Oleh karena itu, untuk mengatasi hal-hal tersebut sistem perlu mengecek apakah data yang dimasukkan oleh *user* sudah benar atau belum. Secara umum ada 5 bagian yang perlu dicek yaitu,

1. Semua form sudah terisi.
2. Format penulisan *email* harus benar.
3. *Password* yang dimasukkan pada *field password* dan *confirm password* harus sama.
4. *Email* tersebut aktif.
5. *Username* tidak ada di dalam *database*.

Untuk mengecek bagian nomor 1, 2, dan 3, digunakan fungsi *javascript*. Setiap *field* yang digunakan pada form seperti *textfield* untuk *username*, atau *textfield* untuk *email* menggunakan *SpryValidationTextField* yang disediakan oleh Adobe Dreamweaver CS5. Di dalamnya terdapat fungsi *javascript* yang digunakan untuk mengecek apakah *field* tersebut sudah diisi atau belum. Selain itu ada juga fungsi *javascript* untuk mengecek regression *email*, sehingga sistem bisa tahu apakah *email* yang diketikkan oleh *user* sudah benar format penulisannya atau belum. Contohnya apabila *user* mengetikkan *user@example.com* maka *email* tersebut benar penulisannya, namun apabila *user* mengetikkan *user*.example.com maka *email* tersebut salah penulisannya.

```
'email': {
    characterMasking: /[^\s]/,
    validation: function(value, options) {
        var rx = /^[\w\.-]+@[\w\.-]+\.\w+$/i;
        return rx.test(value);
    }
},
```

Gambar 3.7. Potongan kode program *javascript* untuk mengecek format penulisan *email*.

Gambar diatas adalah potongan dari program *javascript* yang digunakan untuk mengecek regression *email*. Di dalam variabel *rx* terdapat *regression* dari *email* yang terdiri dari tiga bagian yaitu nama *email* contohnya *user* kemudian nama penyedia *email* contohnya @example kemudian nama domain contohnya .com. Data *email* yang dimasukkan oleh *user* kemudian akan dibandingkan dengan regression *email* dengan menggunakan fungsi *return rx.test(value).*

Pada bagian *password* disediakan dua buah *field* yaitu *field password* dan *field confirm password*. Kedua *field* ini harus diisi dengan *password* yang sama. Tujuan digunakannya dua buah *field* untuk *password* adalah untuk memastikan *user* memasukkan *password* yang *user* inginkan dengan benar, karena bisa saja terjadi kesalahan ketik saat *user* mengetikkan *password user*. Dengan menggunakan dua buah *field password* maka bila *user* melakukan kesalahan ketik di salah satu *field* dapat dicek dengan cara membandingkan apakah kedua *field* tersebut isinya sama atau tidak, sehingga peluang *user* memasukkan *password* yang salah menjadi berkurang.

Setelah *user* selesai memasukkan semua data, format penulisan *email* sudah benar, dan *password* yang dimasukkan sudah benar. Proses akan dilanjutkan untuk mengecek bagian selanjutnya. Namun bila *user* melakukan kesalahan selama mengisi data tersebut akan muncul pesan *error* dan *user* diminta untuk memasukkan data sampai data yang dimasukkan benar. *User* tidak bisa melanjutkan ke proses berikutnya bila masih terdapat kesalahan.

Proses berikutnya adalah mengecek bagian nomor 4 dan 5. Kedua bagian ini tidak bisa dicek dengan menggunakan fungsi *javascript* seperti pada bagian nomor 1, 2, dan 3, melainkan menggunakan PHP. Pada bagian ini diperlukan

koneksi dengan *server* yang berada di luar dari komputer yang digunakan oleh *user* sehingga digunakanlah PHP.

Hal pertama yang harus diperiksa adalah *email* yang dimasukkan oleh *user*. *Email* tersebut harus berupa *email* aktif, maksudnya *email* tersebut benar-benar ada bukan *email* yang dikarang sendiri. *Email* ini penting diperiksa karena nantinya sistem akan menggunakan *email* tersebut untuk komunikasi dengan *user*. Cara yang digunakan untuk memeriksa *email* adalah dengan memeriksa domain pada *email* tersebut apakah domain tersebut ada atau tidak. Kode program untuk memeriksa domain terlihat pada gambar 3.8.

```
if ($isValid && !(checkdnsrr($domain, "MX") || checkdnsrr($domain, "A")))
  {
    $isValid = false;
  }
```

Gambar 3.8. Kode program untuk mengecek domain *email*.

Pertama-tama *email* harus dipisahkan terlebih dahulu bagian nama dan bagian domain. Kemudian bagian domain tersebut disimpan di dalam variabel domain. Domain tersebut kemudian dibandingkan dengan *database* domain yang terdapat di dalam MX dan A. MX adalah sebuah domain record yang digunakan untuk memeriksa domain dari email. Bila domain *email user* ada di dalam *database* maka diasumsikan bahwa *email* tersebut benar, bila tidak ada berarti *email* tersebut salah.

Bagian terakhir yang harus diperiksa adalah *username* yang dimasukkan oleh *user*. *Username* bersifat unik, sehingga tidak boleh ada *username* yang sama di dalam sistem. Bila *user* kebetulan memasukkan *username* yang sudah ada di dalam *database* sistem maka *user* diminta untuk memasukkan *username* yang lain. Cara untuk mengecek *username* tersebut terlihat pada gambar 3.9.

```
    $select = "SELECT * FROM user_account WHERE username =
'$username'";
    $double = mysql_query($select);
    //check if the username double
    if (mysql_num_rows($double) < 1){
        $insert = "INSERT INTO user_account (username, password,
email, gender, job_role, city, hash) VALUES ('$username',
'$password', '$email', '$gender', '$jobrole', '$city', '$hash')";
```

Gambar 3.9. Kode program untuk mengecek *username*

Proses pemeriksaan *username* menggunakan koneksi dengan *database*, sehingga perlu melibatkan *file* config.php. Hal yang dilakukan untuk memeriksa adalah menggunakan SQL *SELECT* dimana *username* adalah *username* yang dimasukkan oleh *user*, kemudian dilakukan mysql_*query*. mysql_num_rows digunakan untuk mengecek hasil dari *query* SQL terdiri dari berapa baris. Bila myqsl_num_rows bernilai lebih dari 1 artinya *username* yang dimasukkan oleh *user* sudah ada di dalam *database*, bila bernilai kurang dari 1 atau 0 artinya *username* tidak ada di dalam *database*. Oleh karena itu pada gambar di atas, *if(mysql_num_rows($double)<1)* artinya jika *username* tidak ada di dalam *database* maka data-data tersebut akan dimasukkan ke dalam *database* dengan menggunakan fungsi SQL INSERT. Jika *username* sudah ada maka akan muncul pesan *error*.

3.6.2. Langkah kedua, melakukan verifikasi *email*.

Di dalam langkah pertama *email user* telah diperiksa format penulisan dan domain *email*nya, sehingga semakin kecil peluang terjadi kesalahan *email*. Pada langkah ini akan dilakukan proses verifikasi *email* untuk memastikan bahwa *email* tersebut adalah benar milik *user*. Hal pertama yang dilakukan adalah membuat sebuah variabel *hash*. Hash adalah 32 karakter yang dihasilkan secara random, *hash* digunakan sebagai pengaman sehingga proses verifikasi *email* sulit untuk dibobol.

```
$hash = md5( rand(0,1000) ); // Generate random 32 character
```

Gambar 3.10. Kode program untuk membuat *hash*

Setelah *hash* terbentuk, *hash* beserta data dari form pada langkah pertama akan dimasukkan ke dalam tabel *user_account* dengan menggunakan fungsi SQL *INSERT*. Jadi proses pembuatan *account* berada pada langkah ini, namun *account* tidak akan bisa digunakan sebelum *user* melakukan proses pembayaran. Jika proses memasukkan data ke dalam *database* berhasil maka sistem akan mengirim *email* kepada *user*. *Email* yang dikirim berupa *link* verifikasi menuju proses registrasi berikutnya. Jika *email* yang dimasukkan pada langkah pertama bukan milik *user*, maka *user* tidak akan bisa melanjutkan proses registrasi karena tidak bisa membuka *email* tersebut. Jadi *email* yang dimasukkan harus bisa dibuka oleh *user*.

Setelah *link* diklik maka *user* akan masuk ke proses berikutnya. Namun sebelum melalui proses berikutnya, sistem perlu melakukan *update* data di dalam *database* bahwa *user* yang bersangkutan telah melakukan verifikasi *email*. Kode program untuk melakukan *update* data terlihat pada gambar 3.11.

```
    $search = mysql_query("SELECT username, hash, email_active FROM
 user_account WHERE username='".$username."' AND hash='".$hash."'
AND email_active='0'");
    $match  = mysql_num_rows($search);

    if($match > 0){
        // We have a match, activate the account
        $update = mysql_query("UPDATE user_account SET
email_active='1' WHERE username='".$username."' AND hash='".$hash.
"' AND email_active='0'");
```

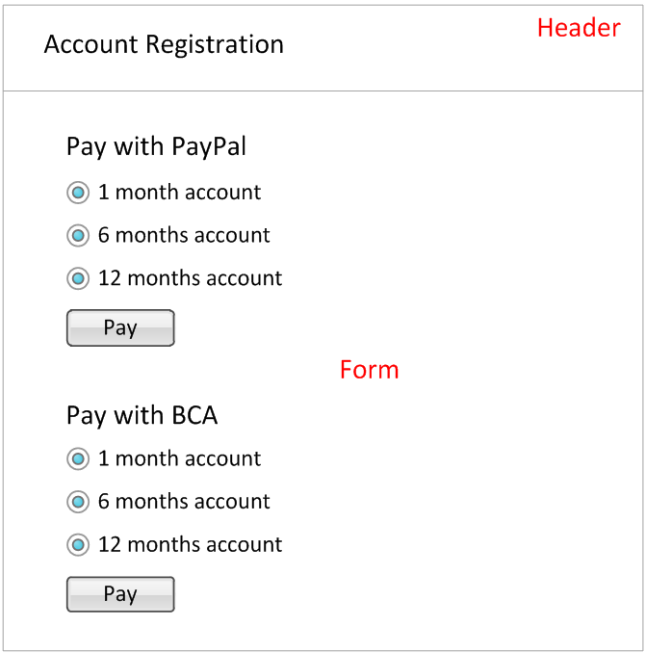
Gambar 3.11. Kode program untuk *update* data *user*

Pertama yang dilakukan adalah memeriksa apakah *username* dan *hash* ada di dalam *database*, bila tidak ada berarti *user* masuk ke dalam sistem melalui jalur yang tidak sesuai dengan prosedur yang disediakan atau *user* mencoba memaksa masuk ke dalam sistem. Perlu diperiksa juga apakah *user* sudah melakukan aktivasi *email* sebelumnya atau tidak, dalam hal ini perlu dipastikan bahwa *email_active* harus berisi 0 artinya *user* belum melakukan aktivasi sebelumnya. Bila hasil pencarian tersebut berhasil *match > 0*, maka sistem akan melakukan *update* data dan merubah nilai *email_active* menjadi 1 artinya *user* telah melakukan aktivasi *email*.

3.6.3. Langkah ketiga, melakukan pembayaran.

Untuk melakukan proses pembayaran sistem menyediakan dua layanan pembayaran yaitu dengan menggunakan PayPal dan BCA. Masing-masing layanan terdiri dari 3 jenis berlangganan yaitu,

1. Berlangganan *account* untuk 1 bulan
2. Berlangganan *account* untuk 6 bulan.
3. Berlangganan *account* untuk 12 bulan.



Gambar 3.12. Layout tampilan form pembayaran.

User dapat memilih salah satu jenis berlangganan yang diinginkan, biaya untuk berlangganan yang lebih lama akan lebh murah harganya. Setelah memilih salah satu jenis berlangganan, maka sistem akan masuk ke dalam proses pembayaran sesuai dengan layanan pembayaran yang dipilihnya. Desain Layout form pembayaran terlihat pada gambar 3.12.

3.6.3.1. Pembayaran dengan PayPal

Pembayaran dengan PayPal mengikuti prosedur pembayaran *Express Checkout* API *for Digital Goods* seperti yang telah dijelaskan pada Bab II. Tampilan pembayaran *Express Checkout* PayPal menggunakan *javascript* yang

telah disediakan oleh PayPal di dalam URL *https://www.paypalobjects.com/js/external/dg.js*. Ketika *user* melakukan klik terhadap tombol Pay With PayPal maka sistem akan menjalankan *file checkout.php.*

SetExpressCheckout

Hal pertama kali yang dilakukan mempersiapkan *field-field* yang akan digunakan untuk melakukan operasi API *SetExpressCheckout*. Persiapan yang dilakukan dapat dilihat pada gambar berikut.

```
$currencyCodeType = "USD";
$paymentType = "Sale";

$returnURL =
"http://203.189.123.200/~m23409017/ta/register/orderconfirm.php";

$cancelURL =
"http://203.189.123.200/~m23409017/ta/register/cancel.php";

$custom = "$username,$item_number,$item_name,$email,$pass";
```

Gambar 3.13. Isi dari variabel yang akan digunakan pada saat operasi API *SetExpressCheckout*

Variabel-variabel yang terlihat pada gambar di atas berisi nilai-nilai yang akan dimasukkan ke dalam operasi *SetExpressCheckout* API. Variabel *currencyCodeType* berisi kode mata uang yang digunakan, karena PayPal tidak menyediakan kode mata uang Rupiah maka sistem menggunakan *US Dollar*. Variabel *paymentType* diisi *Sale*. Variabel *returnURL* merupakan URL yang digunakan setelah PayPal selesai melakukan operasi API *SetExpressCheckout*, pada bagian ini diisikan URL untuk melakukan proses API berikutnya yaitu *GetExpressCheckout* dan *DoExpressCheckout* yang ada di dalam *file orderconfirm.php*. Variabel *cancelURL* berisi URL ketika *user* melakukan pembatalan pembayaran yaitu *file cancel.php*. Variabel custom merupakan *field* custom yang disediakan oleh PayPal untuk memasukkan nilai khusus yang tidak ada hubungannya dengan proses pembayaran, di dalam variabel custom tersebut dimasukkan *username*, jenis item yang dibeli, *email*, dan *password*. Hal-hal

tersebut dimasukkan ke dalam custom karena nantinya akan dipakai pada saat pemrosesan pesan IPN. PayPal. Untuk lebih mengetahui tentang *SetExpressCheckout* API dapat melihat Bab II. Setelah variabel terisi maka sistem memanggil fungsi Set*ExpressCheckout*DG yang ada di dalam *file paypalfunctions.php*

```
$resArray = SetExpressCheckoutDG( $custom, $paymentAmount, $currencyCodeType, $paymentType,
                                  $returnURL, $cancelURL, $items );
```

Gambar 3.14. Kode program yang memanggil Set*ExpressCheckoutDG*.

Di dalam *file paypalfunctions.php* juga berisi beberapa konfigurasi yang digunakan untuk melakukan pembayaran dengan menggunakan PayPal. Ada beberapa hal yang perlu disiapkan terlebih dahulu yaitu seperti mengisi variabel *API_Username*, *API_Password*, dan *API_Signature*. Ketiga buah variabel ini digunakan untuk memanggil API PayPal. *Username*, *Password*, dan *Signature* merupakan API *Credentials* yang didapatkan dari PayPal *Account Business*.

```
$API_UserName="yustus_1456824517_biz_api1.hotmail.com";
$API_Password="1359445567";
$API_Signature=
"AiPC9BjkCyDFQXbSkoZcgqH3hzsfAi6SL3VEPYJ4jQrAtq3xg2yyEBuj";
```

Gambar 3.15. Isi dari variabel mengenai API Credentials

Setelah melakukan konfigurasi API *Credentials* hal penting berikutnya yang harus dilakukan adalah mengatur API *End Point*. API *End Point* adalah alamat URL API PayPal dimana operasi API PayPal, seperti *SetExpressCheckout*, *DoExpressCheckout* akan ditujukan kepada URL tersebut. Alamat URL untuk PayPal *Sandbox* dan PayPal *Live* berbeda, jadi harus dilakukan konfigurasi dengan benar. Pada Gambar 3.16, API *End Point* disimpan di dalam sebuah variabel untuk memudahkan memanggil alamat URL tersebut.

```
if ($SandboxFlag == true)
{
    $API_Endpoint = "https://api-3t.sandbox.paypal.com/nvp";
    $PAYPAL_URL = "https://www.sandbox.paypal.com/cgi-bin/webscr?cmd=_express-checkout&token=";
    $PAYPAL_DG_URL = "https://www.sandbox.paypal.com/incontext?token=";
}
else
{
    $API_Endpoint = "https://api-3t.paypal.com/nvp";
    $PAYPAL_URL = "https://www.paypal.com/cgi-bin/webscr?cmd=_express-checkout&token=";
    $PAYPAL_DG_URL = "https://www.paypal.com/incontext?token=";
}
```

Gambar 3.16. Isi dari variabel mengenai API *End Point*

URL pada variabel API_Endpoint digunakan pada saat sistem memanggil operasi API *SetExpressCheckout*, sedangkan URL pada variabel *PAYPAL_DG_URL* digunakan pada saat memanggil operasi API *GetExpressCheckout* dan *DoExpressCheckout*. Jika sistem menggunakan *Sandbox* atau dalam masa percobaan maka digunakan alamat *website www.sandbox.paypal.com*, jika sistem *Live* atau sudah berjalan secara asli maka digunakan alamat *website www.paypal.com*

```
    function SetExpressCheckoutDG( $custom, $paymentAmount,
$currencyCodeType, $paymentType, $returnURL,
                                        $cancelURL, $items)
    {

//-------------------------------------------------------------
-------------------------------------------------------------
        // Construct the parameter string that describes the
SetExpressCheckout API call in the shortcut implementation

        $nvpstr = "&PAYMENTREQUEST_0_AMT=". $paymentAmount;
        $nvpstr .= "&PAYMENTREQUEST_0_PAYMENTACTION=" .
$paymentType;
        $nvpstr .= "&RETURNURL=" . $returnURL;
        $nvpstr .= "&CANCELURL=" . $cancelURL;
        $nvpstr .= "&PAYMENTREQUEST_0_CURRENCYCODE=" .
$currencyCodeType;
        $nvpstr .= "&REQCONFIRMSHIPPING=0";
        $nvpstr .= "&NOSHIPPING=1";
        $nvpstr .= "&PAYMENTREQUEST_0_CUSTOM=" . $custom;
```

Gambar 3.17. Kode program untuk menyusun pesan *SetExpressCheckout*

Setelah proses konfigurasi selesai maka proses berikutnya adalah menyusun pesan yang akan dikirim ke PayPal. Menyusun pesan yang akan

dikirim tidak secara sembarangan karena penyusunan harus mengikuti parameter yang digunakan oleh API *SetExpressCheckout*. Seperti yang dijelaskan pada Bab II, penyusunan harus mengikuti format *Request Parameter*. Nantinya PayPal akan membalas dengan format *Respone Parameter*. Variabel-variabel yang telah berisi nilai-nilai yang diperlukan untuk operasi *SetExpressCheckout*, disambung menjadi satu buah variabel string yaitu *nvpstr*. Antara *field* yang satu dengan *field* yang lain dipisahkan dengan tanda "&", kemudian isi dari *field* ditulis setelah tanda "=" seperti yang terlihat pada gambar 3.17.

```
        //NVPRequest for submitting to server
        $nvpreq="METHOD=" . urlencode($methodName) . "&VERSION=" .
urlencode($version) . "&PWD=" . urlencode($API_Password) . "&USER="
 . urlencode($API_UserName) . "&SIGNATURE=" . urlencode(
$API_Signature) . $nvpStr . "&BUTTONSOURCE=" . urlencode($sBNCode);

        //setting the nvpreq as POST FIELD to curl
        curl_setopt($ch, CURLOPT_POSTFIELDS, $nvpreq);

        //getting response from server
        $response = curl_exec($ch);

        //convrting NVPResponse to an Associative Array
        $nvpResArray=deformatNVP($response);
```

Gambar 3.18. Kode program untuk mengirim pesan *SetExpressCheckout*

Setelah selesai menyusun pesan, variabel *nvpstr* akan dikirim ke PayPal API *End Point*, proses pengiriman pesan menggunakan CURL, dengan menggunakan CURL pada waktu sistem mengirim pesan ke PayPal, sistem juga dapat mengambil respone yang diberikan oleh PayPal. Hal ini merupakan cara kerja dari API PayPal yaitu request and respone. Sebelum pesan *nvpstr* tersebut dikirim, pesan perlu digabung lagi dengan API Credential sehingga PayPal mengetahui siapakah yang melakukan transaksi pembayaran tersebut. Pesan yang telah digabung tersebut dimasukkan ke dalam variabel *nvpreq* seperti pada gambar 3.18.

Setelah pesan *nvpreq* sudah digabung, pesan tersebut dimasukkan ke dalam CURL kemudian CURL dijalankan dengan perintah *curl_exec*, *ch* merupakan nama inisialisasi dari CURL yang dijalankan saat pertama kali menggunakan CURL dengan menggunakan perintah *curl_init*. Hasil yang

didapatkan dari PayPal disimpan di dalam variabel respone. Isi dari variabel tersebut kemudian dipisah-pisah dengan menggunakan fungsi *deformarNVP* untuk mendapatkan pesan ACK dan *token*. Kemudian sistem akan melakukan redirect ke URL dari *PAYPAL_DG_URL* dengan memasukkan *token* yang didapatkan tersebut.

GetExpressCheckout

GetExpressCheckout sebenarnya merupakan langkah *optional*. *GetExpressCheckout* memberikan kesempatan bagi *user* untuk melakukan *review* transaksi pembayaran terhadap barang yang dibelinya. Jika penjual merasa tidak perlu ada proses *review*, maka penjual dapat langsung melakukan pembayaran dengan API *DoExpressCheckout*. Di dalam sistem ini, *GetExpressCheckout* tetap digunakan, karena sistem menyediakan fitur *review* sebelum membayar. Untuk melakukan *GetExpressCheckout*, sistem tinggal memanggil fungsi Get*ExpressCheckoutDetails*.

```
function GetExpressCheckoutDetails( $token )
{
    //'---------------------------------------------------------------
    //' At this point, the buyer has completed authorizing the payment
    //' at PayPal.  The function will call PayPal to obtain the details
    //' of the authorization, incuding any shipping information of the
    //' buyer.  Remember, the authorization is not a completed transaction
    //' at this state - the buyer still needs an additional step to finalize
    //' the transaction
    //'---------------------------------------------------------------

    //'---------------------------------------------------------------------------
    //' Build a second API request to PayPal, using the token as the
    //'  ID to get the details on the payment authorization
    //'---------------------------------------------------------------------------
    $nvpstr="&TOKEN=" . $token;
```

Gambar 3.19. Kode program dari fungsi Get*ExpressCheckout*Details

Di dalam fungsi tersebut, pesan *nvpstr* yang disusun sangat sederhana yaitu hanya berupa *token* dari pembeli. Pesan *nvpstr* tersebut kemudian akan dikirim dengan menggunakan CURL sama seperti pada saat *SetExpressCheckout*, hanya saja *METHOD* yang digunakan diganti dengan *GetExpressCheckout*.

Setelah pesan dikirim, sistem akan mendapat balasan dari PayPal. Apabila pesan ACK adalah *SUCCESS* maka sistem akan melanjutkan ke proses selanjutnya.

DoExpressCheckout.

Setelah melakukan *GetExpressCheckout* langkah berikutnya adalah melakukan transaksi pembayaran. Transaksi pembayaran dapat dilakukan dengan memanggil operasi API *DoExpressCheckout*. Pada saat melakukan *GetExpressCheckout*, sistem akan mendapatkan respone dari PayPal berupa PayerID dan *token*. *Token* yang didapatkan adalah *token* yang sama yang didapatkan pada saat *SetExpressCheckout*, sedangkan *PayerID* merupakan *ID* yang akan digunakan untuk melakukan finalisasi pembayaran. Untuk menggunakan fungsi *DoExpressCheckout* maka sistem memanggil fungsi *ConfirmPayment*.

```
    function ConfirmPayment( $token, $paymentType,
$currencyCodeType, $payerID, $FinalPaymentAmt )
    {
        /* Gather the information to make the final call to
           finalize the PayPal payment.  The variable nvpstr
           holds the name value pairs
           */
        $token              = urlencode($token);
        $paymentType        = urlencode($paymentType);
        $currencyCodeType   = urlencode($currencyCodeType);
        $payerID            = urlencode($payerID);
        $serverName         = urlencode($_SERVER['SERVER_NAME']);

        $nvpstr  = '&TOKEN=' . $token . '&PAYERID=' . $payerID .
'&PAYMENTREQUEST_0_PAYMENTACTION=' . $paymentType .
'&PAYMENTREQUEST_0_AMT=' . $FinalPaymentAmt;
        $nvpstr .= '&PAYMENTREQUEST_0_CURRENCYCODE=' .
$currencyCodeType . '&IPADDRESS=' . $serverName;
```

Gambar 3.20. Kode program dari fungsi *ConfirmPayment*

Pada gambar 3.20, pesan *nvpstr* yang disusun berisi informasi *token*, *paymentType*, *currencyCodeType*, *payerID*, dan *serverName*. Pesan tersebut disusun sesuai dengan Request Parameter dari *DoExpressCheckout*. Isi dari *field-field* yang ada pada parameter tersebut ada yang sama dengan parameter pada *SetExpressCheckout*, namun ada juga yang tidak sama. Jadi untuk menyusun

pesan *nvpstr* harus disesuaikan dengan *Method* yang digunakan. Setelah pesan disusun langkah berikutnya adalah mengirim pesan tersebut dengan menggunakan CURL sama seperti proses *SetExpressCheckout*, hanya saja *METHOD* yang digunakan diganti dengan *DoExpressCheckout*. Balasan dari PayPal akan diperiksa oleh sistem, jika pesan ACK adalah *SUCCESS* maka sistem akan menampilkan pemberitahuan bahwa transaksi berhasil kemudian sistem akan melanjutkan ke proses berikutnya. Bila terjadi kegagalan pada proses transaksi pembayaran, maka sistem akan menampilkan pesan *error*. Namun sebenarnya akan jarang sekali terjadi kegagalan pembayaran apabila sistem sudah pernah berjalan dengan baik.

Instant Payment Notification

Jika proses *DoExpressCheckout* telah berhasil maka PayPal akan memotong saldo dari pembeli dan akan menambah saldo dari penjual sebesar nilai transaksi yang dilakukan. Setelah itu, merupakan kewajiban penjual untuk memenuhi kebutuhan atau permintaan dari pembeli. Di dalam sistem ini maka sistem harus melakukan *update account* untuk *user* agar *account* bisa digunakan. Proses *update account* ini tidak dilakukan setelah sistem melakukan *DoExpressCheckout*, karena bisa saja setelah melakukan *DoExpressCheckout* ternyata pembayaran belum dilakukan oleh PayPal atau pembayaran mengalami penundaan. Oleh karena itu, lebih baik apabila proses *update account* dilakukan dengan cara menunggu konfirmasi dari PayPal bahwa transaksi pembayaran telah dilakukan. PayPal dapat mengirim informasi tersebut dengan menggunakan fitur *Instant Payment Notification* atau biasa disingkat dengan IPN.

Bila fitur IPN diaktifkan, maka apabila terjadi transaksi pembayaran maka PayPal akan mengirimkan pesan IPN ke alamat *server* sistem. Oleh karena itu, di dalam sistem harus disiapkan sebuah *file* yang akan memproses pesan IPN tersebut. Di dalam sistem ini, pesan IPN yang masuk akan diproses oleh *file ipnpaymentrespone.php*. Menurut cara kerja IPN yang telah dijelaskan pada Bab II, pesan yang diterima oleh sistem harus dikembalikan secara utuh ke PayPal baru setelah itu informasi transaksi pembayaran dapat diterima.

```
$request = "cmd=_notify-validate";
foreach ($_POST as $varname => $varvalue){
    $email .= "$varname: $varvalue\n";
    if(function_exists('get_magic_quotes_gpc') and get_magic_quotes_gpc()){
        $varvalue = urlencode(stripslashes($varvalue));
    }
    else {
        $value = urlencode($value);
    }
    $request .= "&$varname=$varvalue";
}
$ch = curl_init();
curl_setopt($ch,CURLOPT_URL,"https://www.sandbox.paypal.com/cgi-bin/webscr");
//curl_setopt($ch,CURLOPT_URL,"https://www.paypal.com");
curl_setopt($ch,CURLOPT_POST,true);
curl_setopt($ch,CURLOPT_POSTFIELDS,$request);
curl_setopt($ch,CURLOPT_FOLLOWLOCATION,false);
curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
$result = curl_exec($ch);
curl_close($ch);
```

Gambar 3.21. Kode program untuk mengirimkan pesan yang diterima dari PayPal

Pertama-tama sistem membuat variabel *request* dan memberi isi cmd=_notify-vali*date*. Artinya sistem sedang mengembalikan pesan yang diterima dari PayPal. Pesan yang diterima diambil nama *field*-nya sebagai *varname* dan isi dari *field* tersebut sebagai *varvalue*. Kemudian semua *field* dan isinya tersebut disimpan pada variabel request dengan format yang sama yaitu *&varname=varvalue*. Pesan kemudian dikirim dengan menggunakan CURL ke alamat *https://www.sandbox.paypal.com/cgi-bin/webscr* dengan memasukkan *POST field* yaitu isi dari variabel *request* tersebut. Hasil balasan dari PayPal akan berupa satu kata yaitu *VERIFIED* jika berhasil dan *INVALID* jika gagal.

Jika pesan balasan *VERIFIED* maka sistem akan memproses lebih lanjut dengan mengambil informasi yang dikirimkan oleh PayPal. Untuk memudahkan penggunaan maka informasi tersebut disimpan terlebih dahulu di dalam variabel. Variabel *custom* berisi *field* custom yang digunakan pada saat *SetExpressCheckout*, isinya juga sama yaitu *username*, item yang dibeli, *email*, dan *password*. Informasi tersebut berada di dalam satu variabel dan dipisahkan oleh tanda baca koma “,”. Oleh karena itu, perlu dilakukan pemisahan data dengan menggunakan perintah *explode*. Informasi lain yang penting ada *txn_id* yaitu *id* dari transaksi yang digunakan, *id* ini bersifat unik sehingga tidak mungkin

kembar. Selain itu ada juga *payment_status* yaitu berisi dari status pembayaran. Ada 3 status pembayaran yaitu *Completed* artinya transaksi berhasil, *Pending* artinya transaksi tertunda, dan *Failed* artinya transaksi gagal. Informasi lain yang didapatkan dari pesan IPN dapat dilihat pada gambar 3.22.

```
$custom = $_POST['custom'];
$data = explode(",", $custom);

$txn_id = $_POST['txn_id'];
$txn_type = $_POST['txn_type'];
$item_number = $data[1];
$item_name = $data[2];
$payment_status = $_POST['payment_status'];
$payment_amount = $_POST['mc_gross'];
$payment_currency = $_POST['mc_currency'];
$payment_date = $_POST['payment_date'];
$pending_reason = $_POST['pending_reason'];
$reason_code = $_POST['reason_code'];
//receipt_id = for credit card, login as guest
$receiver_email = $_POST['receiver_email'];
$payer_email = $_POST['payer_email'];
$username = $data[0];
$email = $data[3];
$pass = $data[4];
```

Gambar 3.22. Informasi yang diterima dari pesan IPN.

User membuat *account* berlangganan sesuai dengan tipe berlangganan yang dipilihnya. Sistem perlu memeriksa item yang dibeli oleh *user* sebelum melakukan *update* data, karena item yang dibeli akan menentukan *expiry_date* yang dimiliki oleh *user*. Apabila *user* membeli tipe berlangganan satu bulan maka *expiry_date* hanya berlaku untuk satu bulan, apabila berlangganan enam bulan maka *expiry_date* berlaku untuk enam bulan. Selain hal tersebut, hal lain yang perlu diperiksa adalah apakah *txn_id* valid atau tidak. *ID* dari transaksi bersifat unik sehingga perlu dilakukan pemeriksaan apakah di dalam *database* sudah terdapat *ID* transaksi yang sama. Bila tidak ada yang sama maka data transaksi dapat disimpan di dalam *database*. Perlu juga dilakukan pemeriksaan apakah terjadi kecurangan selama pembayaran. Kecurangan yang bisa terjadi adalah *user* mengubah nilai harga *item* menjadi lebih rendah sehingga *user* membeli dengan harga yang murah. Untuk mencegah hal tersebut maka sistem membandingkan harga yang digunakan selama transaksi dengan harga asli yang terdapat di dalam

database. Bila tidak sama maka terjadi kecurangan dan proses *update* data tidak akan dilakukan.

Setelah dilakukan proses pemeriksaan, bila txn_id bersifat valid dan harga tidak mengalami perubahan serta status pembayaran bersifat Completed, sistem akan memasukkan data transaksi ke dalam *database* dan melakukan *update account* dengan mengubah nilai *expiry_date* menjadi baru sesuai dengan tipe berlangganan yang dibeli. Kode program untuk memasukkan data dan melakukan *update* data terlihat pada gambar 3.23.

```
        // PAYMENT VALIDATED & VERIFIED!
        if($valid_txnid && $valid_price && $payment_status==
"Completed"){
            $insert = "INSERT INTO transaction_date (txn_id,
txn_type, item_number, item_name, payment_status, payment_amount,
payment_currency, payment_date, pending_reason, reason_code,
receiver_email, payer_email, username, expiry_date, created_date)
VALUES ('$txn_id', '$txn_type', '$item_number', '$item_name',
'$payment_status', '$payment_amount', '$payment_currency',
'$payment_date', '$pending_reason', '$reason_code',
'$receiver_email', '$payer_email', '$username', '$expiry_date',
'$created_date')";
            $result1 = mysql_query($insert);

            $update = "UPDATE user_account SET
expiry_date='$expiry_date' WHERE username='$username'";
            $result2 = mysql_query($update);
```

Gambar 3.23. Kode program untuk memasukkan data transaksi ke dalam *database* dan melakukan *update* data *account*.

Proses dilanjutkan dengan mengirim sebuah pesan *email* ke alamat *email user* yang memberitahukan bahwa proses pembuatan *account* sudah berhasil. Apabila di dalam status pembayaran bersifat *Pending* atau *Failed*, sistem juga akan mengirimkan pesan *email* yang memberitahukan hal tersebut.

3.6.3.2. Pembayaran dengan BCA

Tidak seperti pembayaran yang dilakukan di dalam PayPal, pembayaran melalui BCA menggunakan metode manual, artinya proses transfer pembayaran tidak dilakukan di dalam *website* melainkan berada di luar *website*. Pembeli harus melakukan proses transfer ke rekening tertentu secara manual, entah melalui ATM

ataupun melalui *website* BCA. Langkah-langkah yang dilakukan untuk melakukan pembayaran dengan BCA dijelaskan secara rinci pada sub bab berikutnya.

Proses transfer pembayaran

Setelah *user* memilih jenis *account* yang ingin digunakan, *account* untuk 1 bulan, *account* untuk 6 bulan, atau *account* untuk 1 tahun. *User* akan diminta untuk melakukan transfer ke dalam nomor rekening tertentu yang tertera dalam *website*. Ada beberapa informasi mengenai pembayaran yang tertera dalam *website* yaitu.

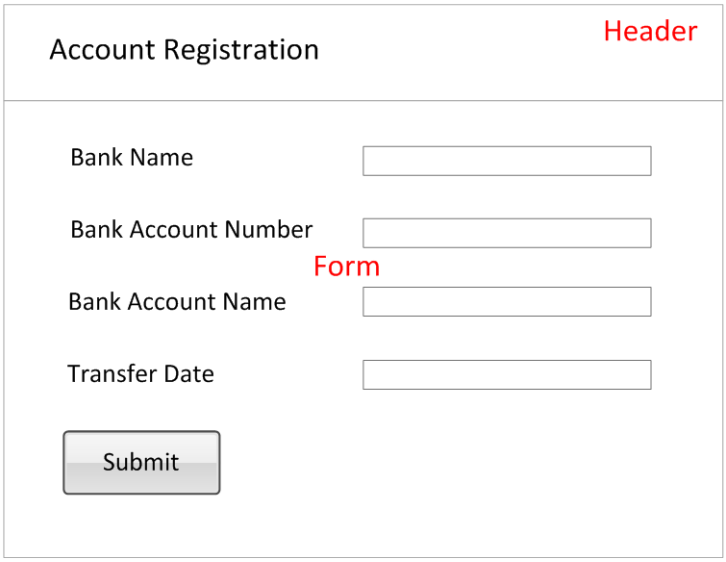
- *Bank Name* merupakan nama bank tujuan transfer. Bank tujuan transfer adalah bank dari penjual atau administrator.
- *Bank Account* merupakan nomor rekening dari bank tujuan transfer.
- *User Name* merupakan nama pemilik dari *account* bank tujuan transfer.
- *Total Payment* merupakan besarnya uang yang harus ditransfer atau dibayarkan. Besarnya uang ini tergantung dari jenis *account* yang dipilih *user* sebelumnya.

Setelah mengetahui informasi mengenai bank tujuan transfer, *user* melakukan pembayaran secara terpisah, di luar dari sistem yang disediakan dalam Tugas Akhir. Pembayaran bisa dilakukan dengan menggunakan ATM, transfer di dalam bank, atau melalui *website* yang disediakan oleh bank. Terlepas dari informasi pembayaran yang ditampilkan dalam *website*, di dalam proses ini juga, sistem telah mengirimkan sebuah *email* kepada *user*. *Email* tersebut berisi sebuah URL yang digunakan untuk melakukan proses konfirmasi pembayaran.

Proses konfirmasi pembayaran.

Sistem pembayaran dilakukan terpisah, berada di luar dari sistem. Sehingga sistem tidak bisa mengetahui apakah *user* sudah melakukan pembayaran atau belum. Untuk mengetahui hal tersebut diperlukan sebuah proses yang digunakan untuk konfirmasi pembayaran yang telah dilakukan oleh *user*. Pada proses sebelumnya, sistem telah mengirimkan sebuah *email* kepada *user* yang dilakukan untuk melakukan konfirmasi pembayaran. Jika *user* melakukan klik

terhadap URL yang ada di dalam *email*, maka *user* akan diarahkan oleh sistem menuju sebuah tampilan form di dalam *website* untuk konfirmasi. Layout untuk form tersebut terlihat pada gambar 3.24.



Gambar 3.24. Layout tampilan form konfirmasi pembayaran.

Di dalam form tersebut, *user* harus memasukkan beberapa informasi yang berkaitan dengan proses transfer pembayaran yang telah dilakukannya. Informasi tersebut adalah,

- *Bank Name* yaitu nama bank pembeli atau *user*, yang digunakan untuk melakukan proses transfer.
- *Bank Account Number* merupakan nomor rekening dari bank *user*, yang digunakan saat melakukan proses transfer.
- *Bank Account Name* adalah nama pemilik *account* bank *user*.
- *Transfer Date* merupakan tanggal transfer pembayaran tersebut dilakukan.

User akan mengisi form tersebut dengan informasi pembayaran yang telah dilakukannya, informasi yang telah diisi tersebut akan diproses oleh administrator pada bagian *Admin Center*. Informasi pembayaran yang valid atau benar akan dikonfirmasi oleh Admin dan *account* dari *user* akan dapat digunakan. Bila pada saat mengisi form konfirmasi, *user* memasukkan informasi yang palsu, maka *account* tidak akan diproses lebih lanjut.

Untuk menghindari *user* melakukan *junk post* secara berlebihan, maka di dalam sistem dilakukan pemeriksaan terhadap *username* dari *user* saat *user* membuka URL konfirmasi yang dikirim pada *email* mereka. Apabila *user* tersebut sudah melakukan konfirmasi pembayaran sebelumnya, maka *user* tidak bisa mengisi form tersebut untuk kedua kalinya.

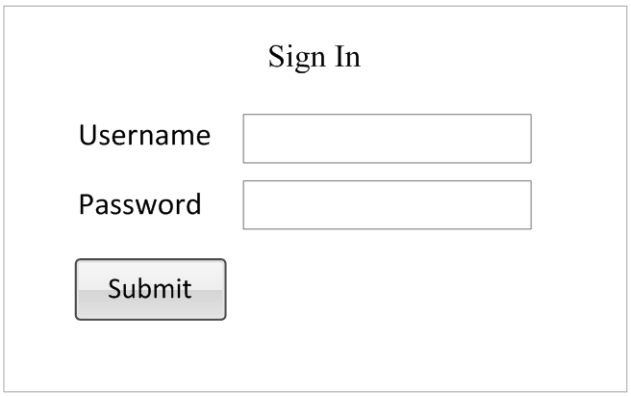
3.7. Fitur *Sign In*

Fitur *Sign In* atau biasa juga dikenal dengan *Log in* atau *Login*, adalah sebuah fitur yang digunakan untuk memberikan batasan bagi *user* dalam melakukan akses ke dalam bagian-bagian tertentu di dalam *website*. *Website* yang dibangun pada sistem ini merupakan *website* nilai prediksi saham yang memiliki informasi data yang sangat penting dan komersional. Oleh karena itu, diperlukan suatu batasan agar tidak semua *user* dapat mengetahui nilai prediksi saham tersebut.

Fitur *Sign In* yang dibuat didalam sistem ini memiliki beberapa spesifikasi atau persyaratan sebagai berikut,

- *User* bisa menyelesaikan proses *Sign In* bila *user* memasukkan kombinasi *username* dan *password* dengan benar. *Account* dari *user* juga tidak boleh expired atau kadaluarsa. *Account* yang kadaluarsa dapat diperpanjang pada fitur *Account*.
- Hanya ada satu *account user* yang boleh masuk ke dalam *website* dalam waktu yang relatif bersamaan. Contohnya, bila *user* telah masuk dengan menggunakan *account* A, lalu beberapa saat kemudian *user* lain masuk dengan menggunakan *account* A yang sama, maka *user* yang pertama kali masuk ke dalam *website* akan dikeluarkan dari *website*, sedangkan *user* yang terakhir masuk dapat mengakses informasi yang diinginkan.

Untuk memulai fitur *Sign In* maka sistem menyediakan sebuah form sederhana yang berisi *textfield* yang digunakan untuk mengisi *username* dan *password*. Layout tampilan dari *Sign In* terlihat pada gambar 3.25.



Gambar 3.25. Layout form dari proses *Sign In*.

User memasukkan informasi *username* dan *password* dari *account* yang dimiliki pada *textfield* yang disediakan, kemudian informasi tersebut dikirim pada saat *user* melakukan klik terhadap tombol *Submit*. Data yang dikirim ke dalam *server* akan diperiksa oleh sistem. Pemeriksaan yang dilakukan berhubungan dengan persyaratan *user* yang dapat mengakses *website* tersebut. Secara umum ada tiga buah pemeriksaan yaitu,

- Pemeriksaan *username* dan *password*
- Pemeriksaan aktivasi *email*
- Pemeriksaan masa aktif dari *account*

```
if (isset($_POST['submit'])){
        $username = $_POST['username'];
        $password = md5($_POST['password']);
        $mysql = mysql_query("SELECT * FROM user_account WHERE
username = '$username' AND password = '$password'");

        if (mysql_num_rows($mysql) < 1)
        {
            $msg = "Invalid Username / Password";
            //echo "<font color=#FF0000 size=3>Invalid Username /
Password</font>";
        }
```

Gambar 3.26. Kode program untuk memeriksa *username* dan *password*

Data dari form yang dimasukkan oleh *user* disimpan di dalam variabel *username* dan *password*. Kedua data tersebut kemudian dibandingkan dengan data yang ada di dalam tabel *user_account* di dalam *database*. Cara

membandingkannya adalah dengan melakukan *query* terhadap tabel dengan perintah *SELECT* pada mysql, disertai dengan informasi *username* dan *password* yang berangkutan. Bila data tidak ditemukan oleh sistem, atau *query* yang dilakukan sebelumnya tidak menunjukkan hasil, maka sistem akan menampilkan pesan *error* yaitu *invalid username or password*. Kode program untuk memeriksa *username* dan *password* terlihat pada gambar 3.27.

```
while($row = mysql_fetch_array($mysql))
{
$email_active = $row['email_active'];
$expiry_date = $row['expiry_date'];
$today = date("Y-m-d");
$exp = strtotime($expiry_date);
$etd = strtotime($today);

if($email_active == 0)
{
    $msg = "Activate Your Email";
    //echo "<font color=#FF0000 size=3>
>";
}
```

Gambar 3.27. Kode program untuk memeriksa verifikasi *email*.

Setelah melalui proses pemeriksaan *username* dan *password*, proses berikutnya yang harus dilakukan adalah memeriksa status *email* dari *account* tersebut. Pada proses registrasi yang dijelaskan pada Bab 3.6, ketika *user* melakukan registrasi di dalam *website*, ada salah satu bagian registrasi yang digunakan untuk melakukan verifikasi *email*. Jika *user* tidak melakukan verifikasi *email*, maka *account* tidak akan bisa digunakan. Untuk memeriksa apakah *email* dari *account* tersebut sudah diverifikasi, sistem melakukan *query* pada tabel *user_account* untuk mendapatkan informasi verifikasi *email*. Bila *email_active* bernilai 0 artinya *account* tersebut belum melakukan verifikasi *email* sebelumnya, maka sistem akan memunculkan pesan *error* yaitu activate your *email*. Kode program untuk memeriksa verifikasi *email* dapat dilihat pada gambar 3.27.

Pada gambar tersebut, sistem juga mengambil informasi *expiry_date* dari *account*. Informasi tersebut digunakan untuk langkah yang berikutnya yaitu memeriksa masa aktif dari *account*. Variabel *today* berisi tanggal saat *user* melakukan proses *Sign In* tersebut. Variabel tersebut kemudian dibandingkan

dengan informasi *expiry_date* dari *database*. Bila variabel *today* bernilai lebih besar artinya *account* tersebut bersifat expired atau masa aktifnya sudah habis. Kemudian sistem akan mengarahkan *user* tersebut ke form pilihan pembayaran seperti pada registrasi langkah ketiga.

```
                if($exp > $etd){
                    //successful login
                    $hash = md5( rand(0,1000) );
                    $update = mysql_query("UPDATE user_account SET
user_hash='$hash' WHERE username='$username'");
                    if ($update){
                        session_start();
                        $_SESSION['loggedin'] = "YES";
                        $_SESSION['name'] = $username;
                        $_SESSION['hash'] = $hash;
                        header("location:index.php");
                    }
                }else if($exp < $etd){
                    //redirect to expired user site
                    header(
"Location:register/manage.php?username=$username");
                }
```

Gambar 3.28. Kode program untuk memeriksa masa aktif *account*

Bila variabel *today* bernilai lebih kecil, berarti *account* dari *user* tersebut masih aktif sehingga sistem membuat *SESSION* untuk *username* tersebut dengan informasi nama yaitu *username* dan informasi *hash*. Kemudian sistem akan mengarahkan *user* ke menu *Home* dengan status *user* tersebut sudah melakukan proses *Sign In*. Selain membuat *SESSION* untuk *username* tersebut, sistem juga perlu melakukan *update* data *user_hash* di dalam tabel *user_account* dalam *database*. *user_hash* berisi 32 karater acak yang digunakan untuk memverifikasi *user* setiap *user* melakukan akses terhadap *website*. Kode program untuk memeriksa masa aktif dari *account* dapat dilihat pada gambar 3.28.

Ketika *user* sudah masuk ke dalam *website* melalui proses *Sign In*, sistem perlu memeriksa apakah ada *user* yang masuk dengan menggunakan *account* yang sama, karena sistem hanya boleh dimasuki oleh satu *account* saja untuk waktu yang bersamaan, tidak boleh ada *double account*. Untuk melakukan hal tersebut langkah yang dilakukan adalah sebagai berikut.

1. *User* 1 melakukan *Sign In* dalam *website* dengan menggunakan *account* A. Sistem akan membuat *hash* (32 buah karakter acak) dan memasukkan dalam *database*. Data *hash* tersebut juga dimasukkan dalam informasi *SESSION* untuk *User* 1 saat melakukan *Sign In*.
2. Setiap kali *User* 1 mengakses halaman-halaman yang ada di dalam *website*, sistem melakukan pemeriksaan terhadap *account* tersebut apakah *hash* yang ada pada *SESSION* sama dengan *hash* yang ada dalam *database*. Bila nilainya sama, maka *User* 1 masih bersifat *Sign In*.
3. *User* 2 melakukan *Sign In* dalam *website* dengan menggunakan *account* yang sama yaitu *account* A. Sistem akan membuat *hash* yang baru, kemudian memasukkan *hash* tersebut pada *SESSION* untuk *User* 2 dan meng*update* nilai *hash* di dalam *database*. (Saat ini berlaku, nilai *hash* pada *SESSION* untuk *User* 1 akan berbeda nilainya dengan nilai *hash* dalam *database*).
4. Ketika *User* 1 mengakses halaman *website* yang lain maka, sistem akan mendapati bahwa nilai *hash* berbeda sehingga sistem akan menutup *SESSION* untuk *User* 1. Sedangkan *User* 2 dapat melakukan akses ke halaman *website* tersebut. Gambar 3.29 menunjukkan kode program untuk memeriksa *hash* yang dimiliki oleh *user*.

```
session_start();
$check = mysql_query("SELECT * FROM user_account WHERE
username='{$_SESSION['name']}' AND user_hash='{$_SESSION['hash']}'"
);
if (mysql_num_rows($check) < 1){
    session_destroy();
}
```

Gambar 3.29. Kode program untuk memeriksa *user_hash*.

Di dalam fitur *Sign In* ini, juga disediakan sebuah fitur tambahan yaitu fitur *Forgot Password*. Fitur ini digunakan untuk melakukan *reset password* bagi user yang lupa akan *password* yang mereka gunakan. Jika user meminta untuk melakukan *reset password*, maka sistem akan meminta user untuk memasukkan *username* atau *email* mereka. Sistem akan melakukan pencarian ke dalam *database*, apakah *username* atau *email* tersebut ada di dalamnya.

Jika didapati sebuah *username* atau *email* yang cocok, maka sistem akan mengirimkan sebuah *email* konfirmasi ke email user yang bersangkutan. *Email* berupa sebuah URL yang digunakan untuk melakukan *reset password*. *User* yang bisa membuka *email* tersebut hanyalah *user* yang mempunyai *account* tersebut, sehingga proses *reset password* ini tidak bisa dilakukan oleh orang lain. Untuk melakukan *reset password*, sistem menyediakan sebuah form sederhana, yaitu berisi dua buah *field password*, dan *confirm password*. Sistem akan mengganti *password* dari *user* dengan data yang dikirimkan oleh *user* tersebut.

3.8. Fitur *Stock*

Fitur *Stock* atau fitur saham merupakan fitur utama di dalam sistem ini, di dalam fitur ini berisi inti dari informasi yang disediakan oleh *website*. Disinilah informasi mengenai data-data saham perusahaan akan disajikan kepada *user*. Penyajian informasi saham harus dibuat dengan mudah dan menarik bagi *user*, tetapi juga tidak menghilangkan detail dari informasi saham tersebut. Pada sistem ini akan disediakan 20 buah data saham dari perusahaan-perusahaan berikut ini,

- BBRI yaitu Bank Rakyat Indonesia (Persero)
- BBNI yaitu Bank Negara Indonesia (Persero)
- BBCA yaitu Bank Central Asia Tbk
- BDMN yaitu Bank Danamon Indonesia Tbk
- BMRI yaitu Bank Mandiri (Persero) Tbk
- BMTR yaitu Global Mediacom Tbk.
- BUMI yaitu Bumi Resources Tbk
- ELSA yaitu Elnusa Tbk.
- ELTY yaitu Bakrieland Development Tbk.
- ENRG yaitu Energi Mega Persada Tbk.
- EXCL yaitu XL Axiata Tbk.
- GJTL yaitu Gajah Tunggal Tbk.
- INCO yaitu Vale Indonesia Tbk.
- INDF yaitu Indofood Sukses Makmur Tbk.
- PGAS yaitu Perusahaan Gas Negara (Persero)
- SMGR yaitu Semen Indonesia (Persero) Tbk.

- TLKM yaitu Telekomunikasi Indonesia (Persero)
- TMPI yaitu AGIS Tbk
- TRAM yaitu Trada Maritime Tbk.
- UNVR yaitu Unilever Indonesia Tbk.

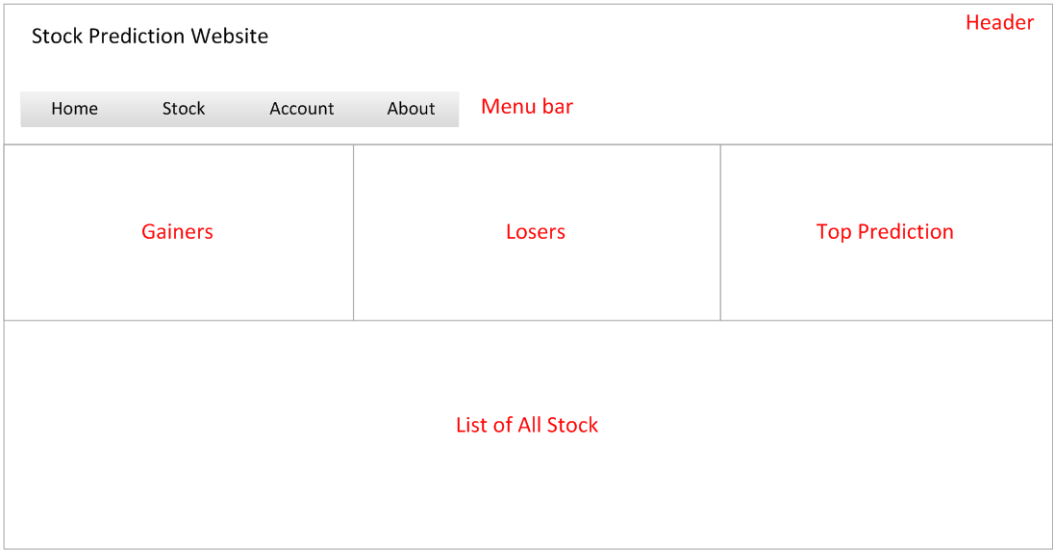
BBRI, BBNI, UNVR merupakan kode dari saham perusahaan tersebut, kode dari masing-masing saham perusahaan bersifat unik, sehingga tidak akan ada perusahaan yang memiliki kode saham yang saham. Kode saham yang digunakan di dalam sistem ini mengambil contoh dari kode saham yang digunakan oleh Yahoo Finance, hanya saja akhiran .JK pada Yahoo Finance dihilangkan. Contohnya di dalam sistem Yahoo Finance kode saham yang digunakan untuk Bank Central Asia adalah BBCA.JK, maka pada sistem ini kode saham yang digunakan adalah BBCA, tanpa menggunakan akhiran JK.

Penyajian informasi saham di dalam sistem ini secara umum dibagi menjadi dua bagian besar yaitu, informasi 20 saham secara keseluruhan dan informasi detail masing-masing saham. Informasi secara keseluruhan meliputi informasi saham mana sajakah yang mengalami kenaikan atau penurunan, juga meliputi saham mana sajakah yang mempunyai prediksi terbaik, dan juga akan ditampilkan data saham terakhir pada saat pasar saham tutup untuk 20 saham tersebut. Informasi detail yang disajikan akan meliputi, informasi data saham terakhir pada saat pasar saham tutup, informasi nilai prediksi saham untuk lima hari ke depan, informasi tentang *error rate* dari prediksi tersebut, serta tampilan grafik untuk saham tersebut.

3.8.1. Tampilan saham secara keseluruhan

Di dalam sistem akan disediakan sebuah menu yang bernama *Stock*. Ketika *user* melakukan klik terhadap menu tersebut, maka *user* akan diarahkan oleh sistem menuju sebuah tampilan yang menampilkan data saham dari 20 perusahaan secara keseluruhan. Data yang disediakan merupakan data terakhir pada saat pasar saham tutup, datanya yang disajikan merupakan data terkini dan *up-to-date* yang diambil dari Yahoo Finance. Setiap harinya sistem akan melakukan *update* data

seperti yang dijelaskan pada Fitur Auto *Update* Data. Tampilan Layout untuk menu *Stock* terlihat pada gambar 3.30.



Gambar 3.30. Layout tampilan menu *stock*

Seperti yang dijelaskan sebelumnya, pada tampilan ini akan dibagi menjadi beberapa bagian tampilan yang menunjukkan perkembangan dari saham-saham yang ada di dalam *database* sistem. Ada 4 bagian yang akan ditampilkan, yaitu *Gainers*, *Losers*, *Top prediction*, *List of all stock.*

3.8.1.1. *Gainers*

Di dalam bagian *Gainers* akan ditampilkan informasi mengenai saham-saham yang mengalami kenaikan pada hari perdagangan saham terakhir. Sistem akan menampilkan lima saham yang mengalami kenaikan tertinggi dan menampilkannya pada bagian ini. Cara menentukan kenaikan tertinggi adalah dengan melihat seberapa besar persentase kenaikan saham tersebut. Saham yang memiliki persentase kenaikan lebih besar akan berada di atas, dan yang memiliki persentase lebih kecil akan berada di bawah.

Untuk menerapkan informasi *Gainers* ini, maka pertama-tama sistem harus mengambil data saham dari *database* terlebih dahulu. Data yang diambil adalah dua data terakhir yang berada di dalam *database* berdasarkan tanggalnya. Sebelum mengambil data tersebut, sistem harus mengambil data dua tanggal terakhir yang

ada di dalam *database*. Hal ini diperlukan karena semua data saham dari 20 perusahaan tersebut berada dalam satu tabel. Oleh karena itu, diperlukan informasi tanggal untuk mengambil dua data saham.

```
    //collect last two date in database
    $mysql = mysql_query("SELECT * FROM real_stock_data WHERE
stock_id = 3 ORDER BY date DESC LIMIT 2");
    $numrows = mysql_num_rows($mysql);
    $y = 0;
    while($row = mysql_fetch_array($mysql))
    {
        $lastdate[$y] = $row['date'];
        $y = $y + 1;
    }
```

Gambar 3.31. Kode program untuk mengambil data tanggal

Data saham diambil dengan menggunakan perintah SQL *SELECT*. Data yang diambil diurutkan berdasarkan tanggal secara descending dengan menggunakan perintah *ORDER BY date DESC*. Untuk mengambil hanya dua data maka digunakan *LIMIT 2*. Data tanggal yang berhasil diambil tersebut akan dimasukkan ke dalam *array* yang bernama last*date*. Proses kemudian dilanjutkan dengan mengambil data saham untuk kedua tanggal tersebut.

```
    //collect data 1
    $mysql = mysql_query("SELECT * FROM real_stock_data WHERE date
= '$lastdate[0]'");
    $numrows = mysql_num_rows($mysql);
    while($row = mysql_fetch_array($mysql))
    {
        $key = $row['stock_id'];
        $value = $row['close'];
        $stock_array1[$key] = $value;
    }
    //collect data 2
    $mysql = mysql_query("SELECT * FROM real_stock_data WHERE date
= '$lastdate[1]'");
    $numrows = mysql_num_rows($mysql);
    while($row = mysql_fetch_array($mysql))
    {
        $key = $row['stock_id'];
        $value = $row['close'];
        $stock_array2[$key] = $value;
    }
```

Gambar 3.32. Kode program proses pengambilan data saham.

Sistem mengambil semua data saham pada tanggal pertama. Tanggal pertama adalah *lastdate[0]*, karena merupakan data pertama pada *array*. Data saham tersebut kemudian disimpan ke dalam *array* yang bernama *stock_array1* dengan menggunakan *stock*_id sebagai informasi *key* dan nilai dari *close* saham dimasukkan sebagai *value*. Data yang kedua diambil dengan cara yang sama, hanya saja tanggal yang digunakan adalah tanggal kedua yaitu *lastdate[1]*. Data tersebut kemudian disimpan sebagai *stock_array2* dengan konfigurasi *key* dan *value* yang sama seperti data pertama.

Setelah mendapatkan data saham pada kedua tanggal, langkah berikutnya adalah menghitung berapakah perubahan nilai *close* yang terjadi. Perubahan nilai close dihitung dnegan menggunakan rumus pada persamaan 3.1. Perubahan bisa bernilai positif atau negatif. Bernilai positif artinya saham tersebut mengalami kenaikan, bernilai negatif artinya saham tersebut mengalami penurunan. Data saham yang berada pada dua *array* yaitu *array_stock1* dan *array_stock2* dihitung dan dimasukkan ke dalam *array ret*. Proses perhitungan menggunakan informasi *key* sehingga perhitungan dilakukan untuk saham yang memiliki kode saham yang sama. Proses memasukkan ke *array ret* juga menggunakan informasi *key* yang sama yang digunakan saat perhitungan. Dengan demikian didapatkan perubahan yang di alami oleh masing-masing saham.

$$Change = \frac{Today - Previous}{Previous}\ x\ 100\ \% \tag{3.1}$$

Untuk menghitung persentase, di dalam sistem telah dibuat fungsi yang digunakan untuk menghitung persentase, sehingga untuk menghitungnya tinggal memanggil fungsi tersebut. Persentase yang sudah dihitung dengan menggunakan rumus pada persamaan 3.1, kemudin hasil perhitungan dimasukkan ke dalam *array per* dengan menggunakan informasi *key* yang sama dengan yang digunakan saat perhitungan. Kode program untuk menghitung nilai perubahan dan persentase terlihat pada gambar 3.33.

```
foreach ($stock_array1 as $key => $value) {
    $ret[$key] = $stock_array1[$key] - $stock_array2[$key];
    $per[$key] = percent($ret[$key],$stock_array1[$key]);
}

//sort top 5 gainers
arsort($per);
$gainers = array_slice($per,0,5,true);
```

Gambar 3.33. Kode program untuk menghitung nilai perubahan dan persentase.

Setelah didapatkan nilai perubahan dan persentase, sistem perlu mengurutkan nilai persentase tersebut, digunakan perintah *arsort* untuk mengurutkan secara *descending* sehingga nilai persentase terbesar berada di paling atas. Kemudian digunakan perintah *array_slice($per,0,5,true)* untuk memotong *array per* sehingga *array* kini berisi hanya 5 data. Hasil pemotongan *array* tersebut disimpan di dalam *array gainers*. Sekarang di dalam *array gainers* berisi data lima saham yang memiliki persentase kenaikan tertinggi.

Proses berikutnya adalah menampilkan kelima data saham tersebut ke *user*. Sistem mengambil nilai persentase di dalam *array* dengan menggunakan perintah *foreach* untuk melakukan perulangan di dalam *array* berdasarkan *key*-nya. Hal yang perlu diperhatikan sebelum menampilkan adalah memastikan bahwa isi dari *gainers* adalah benar data saham-saham yang mengalami kenaikan harga saham. Hal tersebut dapat dilakukan dengan memeriksa nilai persentase dari *gainers*. Apabila bernilai positif, maka saham naik. Bila negatif, berarti saham turun. Cara untuk memeriksa data *gainers* terlihat pada gambar 3.34. Proses ini perlu dilakukan karena ada kemungkian bahwa 20 saham mengalami penurunan nilai harga saham semua. Dalam kasus ini, seharusnya tidak ada data yang berada di dalam bagian *Gainers*.

```
foreach( $gainers as $key => $value ){
    if ($value > 0){
```

Gambar 3.34. Kode program untuk mengkonfirmasi nilai *gainers*.

Sistem menampilkan data setelah selesai melakukan konfirmasi data *gainers*. Data yang ditampilkan adalah kode saham, nilai *close* saham yang terbaru, nilai perubahan harga saham, dan persentase kenaikan harga saham. Data

ditampilkan dengan menggunakan tabel. Data dengan persentase kenaikan tertinggi akan berada pada baris teratas pada tabel. Baris berikutnya dilanjutkan dengan data saham untuk persentase kenaikan tertinggi kedua, dan seterusnya.

3.8.1.2. *Losers*

Selain menampilkan informasi mengenai saham-saham yang mengalami kenaikan, sistem juga menampilkan saham-saham yang mengalami penurunan nilai harga saham. Informasi tersebut ditampilkan pada bagaian *Losers*. Secara umum metode yang digunakan untuk menampilkan informasi ini hampir sama dengan yang digunakan pada bagian *Gainers*. Data-data saham yang ada di dalam *database* tidak perlu untuk dipanggil kembali, dan sistem juga tidak perlu melakukan perhitungan perubahan nilai harga saham kembali, karena kedua hal tersebut telah dilakukan pada bagian *Gainers*. Sehingga yang perlu dilakukan oleh sistem hanyalah mengolah data yang telah digunakan tersebut.

Data yang berisi perubahan nilai harga saham disimpan dalam *array* yang bernama *ret*. Data persentase kenaikan harga saham disimpan dalam *array* yang bernama *per*. Karena acuan yang digunakan untuk mengurutkan saham adalah data persentase, maka sistem mengurutkan *array per* secara ascending. Sehingga data yang mempunyai nilai minus terbesar akan berada di paling atas. Minus yang besar menandakan bahwa terjadi penurunan yang besar pula pada saham tersebut. Untuk mengurutkan data di dalam *array* digunakan perintah *asort*. Data yang telah diurutkan tersebut akan dipilih lima data dengan nilai terendah, untuk mendapatkan lima data saham yang memiliki prosentase penurunan terbesar. Cara untuk mengambil lima data tersebut adalah dengan memotong *array* tersebut dengan menggunakan perintah *array_slice*, dengan demikian data pada *array* sekarang hanya berisi lima data saja. Kode program untuk melakukan proses tersebut ditunjukkan pada gambar 3.35.

```
asort($per);
$losers = array_slice($per,0,5,true);
```

Gambar 3.35. Kode program untuk mengolah data *Losers*

Sama halnya dengan yang dilakukan pada bagian *Gainers*, sebelum menampilkan data yang telah dilakukan proses pengolahan data tersebut. Perlu diperiksa bahwa data yang ada di dalam *array losers* adalah benar data yang bernilai negatif. Karena data diurutkan secara *ascending*, bisa saja ada kemungkinan bahwa semua saham mengalami kenaikan sehingga tidak ada yang bernilai negatif. Data dalam *array* akan berisi lima data terendah namun bernilai positif. Untuk menghindari kesalahan dalam menampilkan data, maka proses pemeriksaan tersebut sangat perlu dilakukan. Satu hal lagi yang membedakan *Losers* dengan *Gainers* adalah kelima data yang telah dipisahkan tadi, setelah melalui pemeriksaan data negatif, perlu diubah bentuknya menjadi positif dengan menggunakan perintah abs yang menjadikan nilai tersebut bersifat absolut. Data yang bernilai negatif akan tidak menarik untuk disajikan, sebagai pengganti tanda negatif tersebut, digunakanlah tulisan dengan warna merah yang menandakan bahwa saham tersebut turun.

3.8.1.3. *Top prediction*

Top prediction merupakan bagian dari menu *Stock* yang menampilkan lima saham yang memiliki nilai prediksi terbaik. Penentuan saham yang memiliki nilai prediksi terbaik dilakukan dengan cara membandingkan nilai *high* dari saham saat ini dengan nilai *high* dari prediksi untuk hari berikutnya. Dari keseluruhan saham yang ada di dalam *database*, dicari lima yang memiliki persentase kenaikan tertinggi.

Cara untuk mencari lima saham tersebut sama dengan yang dijelaskan pada bagian *Gainers* dan *Losers*. Nilai *high* dari saham dan nilai *high* prediksi dimasukkan ke dalam *array* kemudian dibandingkan dan dihitung perbedaan dan persentasenya. Kemudian *array* diurutkan secara *descending* dengan perintah *arsort* dan diambil hanya 5 saham tertinggi dengan perintah *array_slice*. Data saham tersebut kemudian ditampilkan dalam tabel dengan urutan saham yang memiliki persentase kenaikan nilai *high* tertinggi berada di paling atas.

3.8.1.4. *List of all stock*

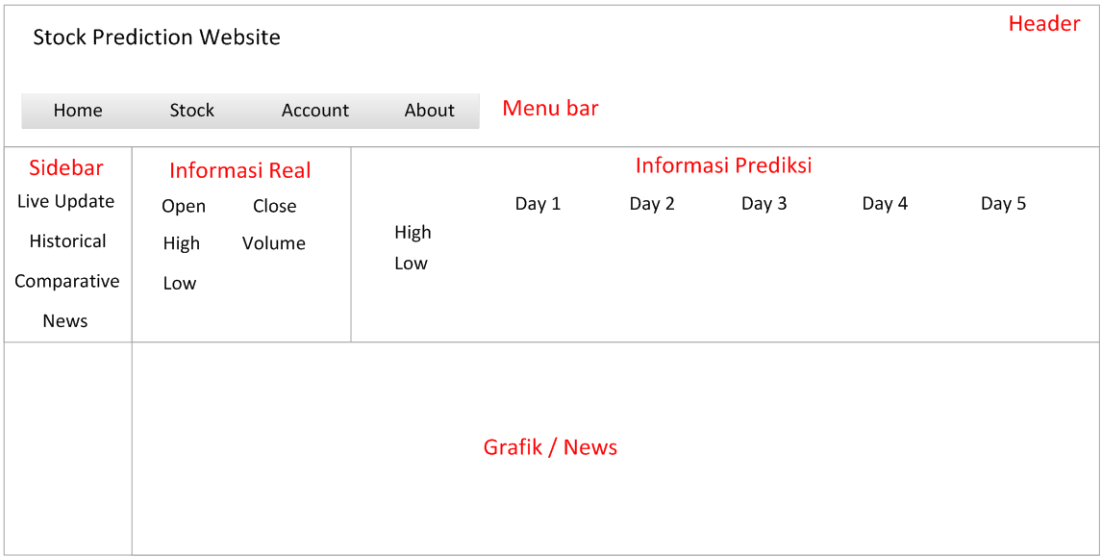
Pada bagian ini, sistem akan menampilkan data keseluruhan dari saham-saham perusahaan yang ada di dalam *database server*. Tujuan dibuatnya bagian ini untuk memberikan informasi singkat kepada *user* mengenai informasi perubahan nilai harga saham yang dialami oleh perusahaan. Tujuan lainnya adalah untuk memberikan informasi kepada *user* saham-saham apa sajakah yang dimiliki oleh sistem. Semua saham yang ada di dalam sistem pasti mempunyai data nilai prediksi, saham lain yang tidak dapat ditemukan pada bagian ini berarti, data saham tersebut tidak disediakan di dalam sistem. Data-data saham yang ditampilkan pada bagian ini adalah kode saham, nama perusahaan, nilai *open*, *high*, *low*, *close*, *volume*, serta informasi perubahan harga dan persentase perubahannya.

Metode yang digunakan untuk menampilkan data sangatlah sederhana. Informasi kode saham, nama perusahaan, nilai *open*, *high*, *low*, *close* dan *volume* diambil dari *database* sistem. Kemudian informasi perubahan data dan persentase diambil dari *array ret* dan *per* yang digunakan pada bagian *Gainers* dan *Losers*. Tidak ada proses pengolahan data yang dilakukan pada bagian ini. Sebagai gantinya, dilakukan proses pemeriksaan nilai perubahan harga saham untuk keperluan tampilan *user*. Saham yang mengalami kenaikan harga saham, nilai persentasenya akan berwarna hijau dengan tambahan icon anak panah ke atas. Saham yang mengalami penurunan, nilai persentasenya akan berwarna merah dengan icon anak panah ke bawah. Sedangkan saham yang tidak mengalami perubahan nilai harga saham akan ditampilkan dengan warna normal yaitu hitam.

Untuk menuju ke tampilan saham secara detail, *user* harus melakukan klik pada kode saham perusahaan. Kemudian sistem akan mengantarkan *user* menuju tampilan detail dari saham yang dipilihnya. Contohnya, bila *user* melakukan klik pada BBCA maka sistem akan mengarahkan *user* menuju tampilan detail dari saham BBCA. *User* dapat melakukan klik pada bagian kode saham yang ada di menu *Stock*, baik itu pada bagian *Gainers*, *Losers*, *Top prediction*, ataupun *List of all stock*.

3.8.2. Tampilan saham secara detail

Tampilan saham secara detail menampilkan informasi yang lebih detail mengenai saham yang bersangkutan. Di dalam tampilan ini akan ada beberapa informasi yang menarik yaitu informasi nilai asli saham dari Yahoo Finance, nilai prediksi saham, grafik, dan berita untuk saham yang bersangkutan. Secara umum layout dari tampilan saham secara detail terlihat pada gambar 3.36.



Gambar 3.36. Layout tampilan *stock* untuk satu saham.

Informasi *real*, merupakan informasi asli yang didapatkan dari Yahoo Finance. Di dalamnya terdapat kode saham, nama perusahaan, nilai saham yang terbaru, nilai perubahan data, persentase perubahan data, nilai *open*, *high*, *low*, *close*, dan *volume*. Di dalam sistem tidak lupa dicantumkan pula tanggal yang memberitahukan kapan data tersebut dilakukan proses *update*. Sehingga *user* bisa mengetahui bahwa data tersebut tidak *up to date* bila suatu saat sistem mengalami kegagalan, atau saat Yahoo Finance terlambat dalam menyediakan data yang baru.

Informasi nilai prediksi saham, memberikan informasi nilai prediksi saham kepada *user*. Nilai prediksi yang diberikan adalah nilai *high* dan nilai *low*. Dengan menggunakan prediksi nilai *high* dan nilai *low*, diharapkan *user* bisa mengetahui saat yang tepat untuk menjual atau membeli saham pada keesokan harinya saat pasar saham buka. Nilai prediksi yang disediakan hanya untuk lima

hari kedepan, sehingga prediksi tersebut akan lebih cocok untuk *user* yang memainkan jual beli saham untuk jangka pendek.

Di dalam nilai prediksi itu pula terdapat informasi *error* sistem. Error yang ditampilkan adalah *error* harian dan *error* mingguan. Error harian merupakan perbandingan perbedaan antara nilai prediksi dengan nilai yang sebenarnya, nilai perbandingan tersebut disajikan dalam bentuk persentase. Setiap hari sistem melakukan perhitungan *error* dan memasukkanya di dalam *database*. Error mingguan merupakan rata-rata dari lima *error* harian terakhir yang ada di dalam *database*. Sistem menghitung nilai rata-rata persentase *error* dan menampilkannya di dalam *website*.

Informasi grafik berisi grafik dari saham. Jenis grafik dan data yang ditampilkan pada grafik tersebut bervariasi tergantung dari menu yang dipilih oleh *user*. Di dalam tampilan detail ini, tampilan masih dibagi menjadi beberapa bagian dengan menggunakan menu *sidebar*, yaitu menu *Live Update*, *Historical*, *Comparative*, dan *News*. Menu *Live Update*, *Historical*, dan *Comparative* berisi grafik dari saham yang dijelaskan secara lengkap pada Fitur *Chart* pada Bab 3.9, sedangkan menu *News* berisi atikel yang berkaitan dengan saham tersebut, yang dijelaskan secara lengkap pada Fitur *News* pada Bab 3.10.

3.9. Fitur Chart

Seperti yang dijelaskan sebelumnya, bahwa di dalam sistem terdapat sebuah fitur untuk menampilkan data dari saham dalam bentuk grafik. Data yang ditampilkan dalam grafik tersebut bervariasi tergantung dari menu *sidebar* yang dipilih oleh *user*. Pada bagian ini, dijelaskan fungsi dari masing-masing menu *sidebar* beserta dengan metode yang digunakan untuk menampilkan grafik. Modul *javascript* yang digunakan di dalam sistem untuk menampilkan grafik adalah *Highstock*, penjelasan parameter atau *field* yang digunakan untuk mengoperasikan *Highstock* berada pada Bab 2.4.

3.9.1. *Live Update*

Menu *Live Update* memberikan informasi terkini dari nilai perubahan harga saham pada saat pasar saham buka, yang disajikan dalam sebuah grafik.

Pada saat sesi perdagangan bursa saham, nilai harga saham berubah-ubah nilainya. Perubahannya tergolong cepat karena dalam periode satu menit saja harga saham bisa berubah. Pada menu ini *user* dapat memantau perubahan nilai harga saham per-menit-nya. Sehingga *user* dapat membandingkan perubahan nilai harga saham saat ini dengan prediksi nilai *high* atau *low* pada hari itu, dan mendapatkan waktu yang tepat untuk menjual atau membeli saham.

Untuk menampilkan grafik, sistem memerlukan data. Oleh karena itu, hal pertama yang dilakukan oleh sistem adalah mengumpulkan data yang akan ditampilkan di dalam grafik. Data perubahan harga saham untuk *Live Update* didapatkan dari Yahoo Finance secara langsung, data tersebut tidak disimpan di dalam *database server*. Data yang disimpan di dalam *database* hanyalah data perubahan nilai harga saham *per* hari-nya. Untuk memperoleh data tersebut dibutuhkan koneksi dengan *server* Yahoo Finance pada URL tertentu, sehingga dibutuhkan bahasa pemrograman *server side* yatu PHP. *Highstock* disisi lain, menampilkan grafik secara client *side* dengan menggunakan Javascript. Untuk emnggabungkan kedua hal tersebut digunakan sebuah fungsi *jquery* yang bernama *getJSON*. *getJSON* adalah sebuah perintah di dalam *javascript* yang berfungsi untuk menjalankan program PHP tertentu, dan menyimpan hasilnya di dalam *javascript*. Sehingga *getJSON* dapat digunakan sebagai jembatan antara PHP dengan Javascript.

```
$sourceURL =
"http://chartapi.finance.yahoo.com/instrument/1.0/$stock_name.JK/ch
artdata;type=quote;range=1d/csv/";
$sourceData  = file_get_contents( $sourceURL );
```

Gambar 3.37. URL untuk mengambil data *Live* dari Yahoo Finance

Proses dimulai dengan membuat *file* PHP yang digunakan untuk mengambil data dari Yahoo Finance. Data dari Yahoo diambil dari URL pada gambar diatas. *stock*_name merupakan kode saham yang digunakan oleh Yahoo sehingga diberi akhiran .JK. Sistem mengambil data dengan menggunakan perintah *file_get_contents* dan menyimpannya dalam sourceData. Data di dalam sourceData kemudian dipisahkan perbaris-nya. Baris pertama hingga baris ke-15 merupakan informasi dasar serta informasi rangkuman dari perubahan harga

saham sehingga tidak perlu diambil oleh sistem. Mulai baris ke-16 dan seterusnya merupakan data perubahan nilai saham yang ditulis dengan format CSV.

```
$contents = str_getcsv( $line );

$date = $contents[0] + 25200;
$date_utc = $date * 1000;
$close = intval($contents[1]);
$high = intval($contents[2]);
$low = intval($contents[3]);
$open = intval($contents[4]);
$volume = intval($contents[5]);

$array[$i][0] = $date_utc;
$array[$i][1] = $open;
$array[$i][2] = $high;
$array[$i][3] = $low;
$array[$i][4] = $close;
$array[$i][5] = $volume;
```

Gambar 3.38. Proses pemisahan data *Live*.

Pada baris ke-16 dan seterusnya, sistem perlu memisahkan data setiap baris sehingga didapatkan informasi yang disimpan per-baris-nya. Dengan menggunakan perintah *str_getcsv* maka data per baris tersebut dipisahkan. Data pertama merupakan informasi tanggal yang disimpan dalam format UTC. Tanggal tersebut menggunakan timezone yang tidak tepat, sehingga harus disamakan dengan timezone Indonesia terlebih dahulu. Caranya adalah dengan menambahkan tanggal tersebut dengan angka 25200. *Highstock* hanya bisa membaca tanggal dengan format UTC *milisecond*, sehingga tanggal tersebut perlu dikali dengan angka 1000. Informasi berikutnya secara berurutan adalah nilai *close*, *high*, *low*, *open*, dan *volume*. Informasi tersebut masih berbentuk *string*. Oleh karena itu, data dirubah terlebih dahulu menjadi *integer* dengan menggunakan perintah *intval*.

Data-data yang telah berhasil dipisahkan tersebut, kemudian dimasukkan ke dalam sebuah *array*. *Array* tersebut kemudian diurutkan berdasarkan tanggal dengan menggunakan perintah *array_multisort*. Sehingga data yang berisi tanggal paling kecil berada pada posisi awal pada *array*, sedangkan tanggal paling besar berada di akhir. Hal ini perlu dilakukan agar data tersebut dapat ditampilkan di dalam grafik. *Highstock* tidak bisa menampilkan grafik bila data tersebut tidak

diurutkan terlebuh dahulu. Setelah itu data kemudian ditampilkan ke layar dengan menggunakan perintah *echo*. Data perlu ditampilkan agar bisa dibaca oleh *javascript* saat *javascript* memanggil perintah *getJSON*. Dengan ditampilkannya data tersebut maka, proses pengambilan data dengan menggunakan PHP sudah selesai. Sistem melanjutkan proses berikutnya dengan membuat kode program untuk menampilkan grafik.

```
var currentdate = new Date();
var hours = currentdate.getHours();
var minutes = currentdate.getMinutes();

// jam buka pasar saham jam 9 sampai jam 6
if (hours >= 9 && hours < 16){
    var Update = setInterval(function(){Chart()}, 60000);
}else if (hours == 16 && minutes <= 30){
    var Update = setInterval(function(){Chart()}, 60000);
}else{
    //clearInterval(Update);
}
```

Gambar 3.39. Fungsi *setInterval* untuk mengambil dan menampilkan data setiap menit.

Pada saat pasar saham buka, data perubahan nilai harga saham akan berubah-ubah setiap menitnya. Yahoo Finance akan menambahkan data yang baru setiap menit. Oleh karena itu, grafik di dalam sistem dibuat secara dinamis, artinya setiap menit-nya sistem akan mengambil data dari Yahoo Finance dan menampilkannya dalam grafik. Untuk membuat fungsi tersebut maka sistem memanfaatkan fungsi setInterval di dalam *javascript*. setInterval digunakan untuk menjalankan sebuah fungsi dalam waktu yang ditentukan. Di dalam kode program yang ditunjukkan pada gambar 3.39. Sistem akan menjalankan fungsi *Chart()* setiap 60000 *milisecond* atau satu menit. Fungsi tersebut dijalankan hanya pada saat jam buka saja yaitu pukul 9 pagi hingga pukul 4 sore. Sehingga *user* tidak perlu melakukan *refresh* terhadap *website* untuk menampilkan data yang baru, karena grafik akan secara otomatis berubah sesuai dengan data terkini yang ada di dalam Yahoo Finance.

```
$.getJSON(url, function (data1) {
    var ohlc = [],
        volume = [],
        closed = [],
        forecast = [],
        dataLength1 = data1.length;

    for (i = 0; i < dataLength1; i++) {
        ohlc.push([
            data1[i][0],
            data1[i][1],
            data1[i][2],
            data1[i][3],  // the date
            data1[i][4] // close
        ]);

        closed.push([
            data1[i][0], // the date
            data1[i][4] // close
        ]);
```

Gambar 3.40. Pengambilan data dengan menggunakan *JSON* dan pemisahan data.

Sebelum grafik bisa ditampilkan, sistem perlu mengambil data yang ada di dalam Yahoo Finance dengan menggunakan fungsi *JSON*. Perintah *getJSON* digunakan untuk menjalankan *file* PHP yang ada di dalam url. *File* PHP yang dipanggil adalah *file* PHP yang berisi data dari Yahoo Finance yang telah dibuat pada proses sebelumnya. Hasil dari pemanggilan *file* PHP tersebut disimpan dalam data1. Variabel data1 mempunyai banyak data yaitu tanggal, *open*, *high*, *low*, *close*, dan *volume*, sesuai dengan urutan *array* pada *file* PHP. Data tersebut dipisahkan terlebih dahulu sesuai dengan kebutuhan grafik. Pada grafik *Live Update* dibuat dua buah *yAxis* yaitu satu untuk menampilkan nilai harga saham, dan yang lain untuk menampilkan nilai *volume*. Nilai *close* juga perlu dipisahkan karena digunakan pada saat grafik menampilkan data dengan tipe grafik garis.

```
$('#container').highcharts('StockChart', {

        rangeSelector: {
            enabled: false
        },
        navigator: {
            enabled: false
        },
        scrollbar: {
            enabled: false
        },
        title: {
            text: title
        },

            series: [{
                type: 'ohlc',
                name: stockname,
                data: ohlc,
            }, {
                type: 'column',
                name: 'Volume',
                data: volume,
                yAxis: 1,
            }]
```

Gambar 3.41. Konfigurasi grafik pada *Live Update*.

Pada gambar 3.41, ditampilkan contoh konfigurasi grafik yang ingin ditampilkan. Grafik tersebut akan ditampilkan dalam sebuah bagian dalam HTML yang bernama *container*. Beberapa konfigurasi yang dilakukan antara lain yaitu, menon-aktifkan *rangeSelector*, *navigator*, dan *scrollbar*. Ketiga komponen tersebut tidak digunakan dalam tampilan grafik *Live Update* karena rentang waktu di dalam grafik tersebut sudah pasti yaitu antara pukul 9 pagi hingga pukul 4 sore. Konfigurasi lain yang dilakukan yaitu memilih data untuk ditampilkan dalam grafik. Konfigurasi tersebut dilakukan pada bagian *series*, data dipilih dengan memasukkan data tersebut ke dalam *field* yang bernama data. Data yang dimasukkan adalah data yang telah dipisahkan pada proses sebelumnya. Semua konfigurasi yang dilakukan pada bagian grafik mengikuti format dan *field Highstock* yang telah dijelaskan pada Bab 2.4.

Jenis grafik yang bisa digunakan oleh *user* untuk memantau perkembangan nilai saham ada tiga macam yaitu, grafik garis, grafik *OHLC* (*Open*, *High*, *Low*, *Close*), dan grafik *Candlestick*. Oleh karena itu, sistem harus

melakukan tiga konfigurasi untuk grafik yang berbeda. Konfigurasi yang dilakukan adalah mengganti jenis grafik pada parameter type yang ada di dalam variabel *series*. Pilihan *ohlc* untuk grafik *ohlc*, *candlestick* untuk grafik *candlestick* dan *line* untuk grafik garis.

3.9.2. *Historical*

Menu *Historical* memberikan informasi perubahan nilai harga saham *per* hari-nya. Disediakan data saham dari tahun 2007 hingga sekarang sehingga *user* dapat mengetahui sejarah perkembangan saham tersebut. Saham yang baik salah satunya ditentukan dari sejarah perkembangan sahamnya yang mengalami kenaikan setiap tahunnya. Untuk *user* yang memainkan jual beli saham dalam jangka panjang maka, informasi ini bisa menjadi salah satu acuan untuk menentukan saham mana yang baik untuk dibeli. Selain informasi nilai saham, *user* dapat mengetahui nilai prediksi saham *per* hari-nya. Data nilai prediksi di dalam sistem dimulai dari tahun 2013. *User* dapat melihat perkembangan nilai prediksi tersebut hari demi harinya, sehingga *user* bisa menentukan bagaimana perbandingan nilai prediksi dengan nilai asli secara keseluruhan, apakah memiliki nilai *error* yang stabil atau berubah-ubah. Semua informasi tersebut disajikan dalam bentuk grafik.

Metode yang digunakan untuk menampilkan grafik pada menu *Historical* hampir sama dengan yang dilakukan pada menu *Live Update*. Hal pertama yang dilakukan adalah membuat *file* PHP untuk mengambil data dari *server*. Data *Historical* dan prediksi di ambil dari *database* yang dimiliki oleh sistem, sehingga dengan menggunakan perintah SQL maka data saham asli dan prediksi akan dengan mudah didapatkan. Hal yang perlu diperhatikan adalah data tanggal di dalam *database* disimpan dalam format tanggal biasa sehingga perlu diubah menjadi UTC milisecond terlebih dahulu. Setelah data didapatkan, data tersebut dimasukkan ke dalam *array*. Seperti yang dilakukan pada *Live Update*, *array* tersebut kemudian diurutkan dengan perintah *array_multisort* dan ditampilkan dengan perintah *echo*.

```
$.getJSON(url, function (data1) {
    $.getJSON(urlprediction, function (data2) {
```

Gambar 3.42. Perintah *getJSON* untuk mengambil data.

Setelah data siap digunakan, maka langkah berikutnya adalah memanggil data tersebut dan melakukan konfigurasi grafik. Data dipanggil dengan menggunakan perintah *getJSON*. Terdapat dua perintah *getJSON* pada menu ini seperti yang terlihat pada gambar 3.42. Perintah pertama digunakan untuk mengambil data asli yang ada di dalam *database* melalui *file* PHP yang ada di dalam url. Sedangkan perintah kedua digunakan untuk mengambil data prediksi di dalam *database* melalui *file* PHP dalam urlprediction.

Dari dari *JSON* tersebut disimpan variabel *data1* dan *data2*, kemudian dilakukan proses pemisahan data seperti pada menu *Live Update*. Proses konfigurasi yang dilakukan secara umum sama, hanya saja pada menu *Historical* banyaknya *series* data yang ditampilkan ada empat, yaitu.

- *Series* 1 digunakan untuk menampilkan informasi harga saham, bisa berupa *OHLC*, *Candlestick*, ataupun garis.
- *Series* 2 digunakan untuk menampilkan informasi *volume* dari saham. *Volume* ditampilkan dengan menggunakan grafik batang.
- *Series* 3 digunakan untuk menampilkan informasi prediksi nilai *low*. Prediksi tersebut ditampilkan dengan grafik garis.
- *Series* 4 digunakan untuk menampilkan informasi prediksi nilai *high*. Prediksi ini juga ditampilkan dalam grafik garis.

3.9.3. *Comparative*

Selain menampilkan data untuk satu saham, data saham juga bisa ditampilkan secara komparatif, artinya dua saham dapat dibandingkan nilainya dan dilihat perbandingan grafiknya. Pada menu ini, terdapat sebuah form sederhana yang digunakan oleh *user* untuk menentukan saham mana yang ingin dibandingkan, di dalam menu ini *user* dapat membandingkan maksimal tiga buah saham, Setelah menentukan saham yang akan dibandingkan, data-data mengenai

saham terebut akan ditampilkan secara *side by side*, dan sistem akan menampilkan perbandingan grafik dari saham-saham yang dibandingkan.

Metode untuk menampilkan grafik sama dengan yang dilakukan sebelumnya, yaitu menyiapkan *file* PHP untuk mengambil data saham asli dari *database*. Hasilnya disimpan dalam *array* dan ditampilkan ke layar. Setelah itu *javascript* memanggil *file* PHP tersebut dengan menggunakan *getJSON*. Data yang didapatkan dari *JSON* tidak perlu melalui proses pemisahan data lagi karena data yang ditampilkan sudah pasti yaitu tanggal dan nilai *close* saja. Grafik pada menu *Comparative* berbeda dengan grafik pada *Live Update* dan *Historical*. Jika pada kedua menu tersebut nilai harga saham ditampilkan nilainya dalam grafik secara normal, artinya jika saham nilainya 500 maka grafik akan menunjukkan angka 500. Pada menu *Comparative* angka pada grafik ditunjukkan dalam persentase, artinya nilai harga saham ditampilkan dalam persentase kenaikan atau penurunan harga saham. Bila harga saham naik maka akan bernilai postif dan bila saham turun akan bernilai negatif. Untuk menerapkan hal tersebut, perlu dilakukan konfigurasi di dalam grafik seperti pada gambar 3.43. Konfigurasi yang dilakukan yaitu mengaktifkan *field compare* pada bagian *series*, dan memasukkan metode yang digunakan untuk membandingkan data tersebut, data bisa dibandingkan berdasarkan persentase dengan menggunakan percent, atau berdasarkan nilai dengan menggunakan perintah *value*.

```
plotOptions: {
    series: {
        compare: 'percent'
    }
},
```

Gambar 3.43. Konfigurasi grafik pada menu *Comparative*.

3.10. Fitur *News*

Di dalam *website* yang digunakan sebagai Tugas Akhir ini, tersedia layanan untuk menampilkan berita kepada *user*. Sumber dari berita tersebut berasal dari RSS Yahoo Finance. Dengan memanfaatkan RSS tersebut, sistem dapat menampilkan berita terkini mengenai perusahaan yang sahamnya ada di dalam *database server* sistem. Berita tersebut ditampilkan dalam salah satu menu

sidebar pada menu *Stock*. Sedangkan berita yang ditampilkan pada menu *Home* adalah berita umum mengenai bisnis dan keuangan di Indonesia. Fitur *News* ini diharapkan menjadi salah satu media bagi *user* untuk dapat mengetahui berita dan aktivitas dari perusahaan tersebut dan mengetahui perkembangan perekonomian di Indonesia. RSS ditulis dengan menggunakan format XML, sehingga untuk mendapatkan informasi berita yang ada di dalamnya perlu dilakukan pembongkaran atau pemisahan *tag-tag* yang ada di dalam *file* XML tersebut.

```
function parser($feedURL) {
    $rss = simplexml_load_file($feedURL);
    echo "<ul class='news'>";
    $i = 0;
    foreach ($rss->channel->item as $feedItem) {
        $i++;
        $myDate = ($feedItem->pubDate);
        $dateForm = explode(" ", $myDate);
```

Gambar 3.44. Kode program untuk memisahkan berita di dalam RSS.

Untuk memisahkan berita yang satu dengan berita yang lainnya, sistem membuat sebuah fungsi yang bernama *parser*, dimana di dalamnya terdapat kode program untuk memisahkan *tag-tag* XML. Hal pertama yang dilakukan adalah mengambil RSS tersebut dengan menggunakan perintah *simplexml_load_file*. Variabel *i* digunakan untuk menentukan banyaknya berita yang ingin ditampilkan ke dalam *website*. Setelah itu sistem harus masuk ke dalam *tag* <*item*> dimana berita tersebut disimpan. Sebelum masuk ke *tag* <*item*>, sistem harus masuk terlebih dahulu ke *tag* <*channel*> yang derajatnya berada di atas <*item*>. Proses masuk ke <*item*> tersebut disimpan sebagai *feedItem*. Kemudian sistem tinggal memanggil komponen *tag* berita yang dibutuhkan, misalnya sistem emnginginkan tanggal berita maka caranya adalah dengan kode *feedItem→pubDate*. Sistem secara otomatis akan masuk ke dalam *tag* <*pubDate*> untuk mengambil tanggal berita. Pada gambar 3.44, tanggal dipisahkan lagi dengan menggunakan fungsi explode untuk mendapatkan nilai tanggal, bulan, dan tahun. Hal ini dilakukan karena *tag* <*pubDate*> berisi informasi tanggal yang terlalu detail yang tidak menarik jika ditampilkan di dalam *website*.

```
        echo "<li class='list'>
            <font face='Tahoma,Geneva,sans-serif' color=#000066
class='news'><a href='$feedItem->link' title='$feedItem->title'>" .
 $feedItem->title . "</a></font>
            <font size=-1 class='date'>" . $dateForm[1] . ". " .
$dateForm[2] . ". " . $dateForm[3] . "." . "</font>
        </li>";

        if($i >= 20) break;
    }
```

Gambar 3.45. Kode program menampilkan berita dari RSS

Berita tersebut akan ditampilkan di dalam *website* sebagai list. Pada gambar di atas ditunjukkan bahwa yang ditampilkan hanyalah informasi judul berita dan tanggal berita tersebut. Untuk *tag* <*description*> tidak ditampilkan karena ridak semua berita perusahaan dalam RSS Yahoo Finance memiliki *tag* <*description*>. Sedangkan pada berita yang ada di dalam menu *Home*, *tag* <*description*> ditampilkan beserta dengan gambar yang tersedia dalam *tag* tersebut. Proses menampilkan berita ini akan terus berulang hingga berita yang ditampilkan sudah ada 20 berita. Proses juga akan berhenti bila tidak sampai 20 berita *file* RSS telah selesai dibaca. Jadi di dalam sistem, berita perusahaan yang ditampilkan maksimal 20 berita.

3.11. Fitur Auto *Update* Data

Bursa Saham Jakarta buka pada hari Senin sampai Jumat mulai pukul 9 pagi hingga 4 sore. Bursa tutup pada hari Sabtu dan Minggu, serta pada hari libur nasional. Selama saham buka maka data saham dari perusahaan tertentu akan berubah nilainya, dan pada pukul 4 sore ditentukan nilai harga *Close* saham yang baru beserta dengan nilai *High* dan *Low* pada hari tersebut. Setiap hari data tersebut akan berubah, sehingga sistem perlu melakukan *update* data setiap harinya agar data di dalam *website* selalu up-to-*date*.

3.11.1. *Update* Data Asli

Setiap hari Yahoo Finance menyediakan data baru berupa *Open*, *High*, *Low*, *Close*, *Volume* untuk hari tersebut, namun data tersebut tidak keluar pada pukul 4 sore saat Bursa Saham tutup, melainkan ada jeda waktu tertentu. Jeda

waktu munculnya data tersebut tidak pasti, terkadang bisa muncul dengan cepat terkadang lambat. Oleh karena itu, perlu ditentukan suatu waktu khusus dimana *server* pada sistem akan melakukan *update* data. Pada sistem ini ditentukan bahwa sistem akan melakukan *update* data pada pukul 3 pagi. Dipilih pukul 3 pagi dengan asumsi data pada Yahoo Finance sudah ada, sehingga *server* hanya perlu menjalankan *file* untuk melakukan *update* satu hari sekali, tidak perlu berulang-ulang kali. Cara melakukan *update* adalah dengan memanfaatkan sistem CRON pada *server* Ubuntu.

Untuk menggunakan *CRON* maka *file* crontab yang ada di dalam *server* harus dikonfigurasi terlebih dahulu. Cara membuka *file crontab* dilakukan dengan menggunakan perintah *crontab -e*, setelah itu *file crontab* akan terbuka dan perintah yang hendak dijalankan dimasukkan ke dalam *file* tersebut. Perintah yang dimasukkan seperti yang terlihat pada gambar berikut ini.

```
00 3 * * * /usr/bin/wget -q -O temp.txt http://203.189.123.200/~m23409017/ta/cr$
05 3 * * * /usr/bin/wget -q -O temp.txt http://203.189.123.200/~m23409017/ta/cr$
```

Gambar 3.46. Perintah yang dimasukkan untuk melakukan *update* data.

Dimasukkan dua buah perintah untuk melakukan *update* data asli. Perintah pertama merupakan perintah untuk melakukan *update* data saham dari perusahaan, sedangkan perintah kedua digunakan untuk melakukan *update* data indeks harga saham gabungan (IHSG) Jakarta. Pada pukul 03:00 dan pukul 03:05 pagi, sistem akan menjalankan perintah *wget* untuk menjalankan *file* php yang akan melakukan *update* data. Perintah *wget* merupakan perintah yang digunakan untuk membuka URL, *-q* merupakan *quite mode*, *-o temp.txt* artinya hasil dari URL yang dibuka akan disimpan di dalam *file temp.txt*. URL yang dibuka merupakan *file* php yang akan melakukan *update* data saham perusahaan dan data IHSG.

Data saham yang akan di-*update* berasal dari Yahoo Finance, di dalam *file* php tersebut terdapat perintah yang digunakan untuk mengambil data Yahoo Finance. Namun, sebelum mengambil data, ada beberapa hal yang perlu dilakukan yaitu menghitung banyaknya saham yang ada di dalam *database*. Informasi ini dapat diperoleh dengan menghitung banyaknya saham pada tabel *stock_list*. Hal

kedua yang perlu dilakukan adalah mengambil tanggal terakhir dari data saham yang ada di dalam *database*, hal ini perlu dilakukan agar sistem dapat melakukan *update* data dari tanggal terakhir di dalam *database* hingga tanggal saat ini. Jika kedua hal tersebut sudah dilakukan maka langkah berikutnya adalah mengambil data dari Yahoo Finance.

```
$mysql = mysql_query("SELECT * FROM real_stock_data WHERE date =
'$today'");
if (mysql_num_rows($mysql) < 1){
    //data belum ada, update data
    $sourceURL =
"http://ichart.finance.yahoo.com/table.csv?s=$array_stock[$x].JK&a=
$last_month&b=$last_day&c=$last_year&d=$today_month&e=$today_day&f=
$today_year&g=d&ignore=.csv";
    $sourceData  = file_get_contents( $sourceURL );
```

Gambar 3.47. Kode program untuk mengambil data dari Yahoo Finance.

Saat mengambil data perlu diperiksa sekali lagi apakah tanggal saat ini sudah ada datanya di dalam *database*, jika belum ada maka sistem akan mengambil data, bila sudah ada tidak perlu mengambil data supaya tidak ada tanggal yang sama di dalam *database*. URL untuk mengambil data adalah *ichart.finance.yahoo.com*. *s, a, b, c, d, e, f, g* adalah parameter yang digunakan seperti yang sudah dijelaskan pada Bab II. Dengan mengganti parameter *a, b, c* sebagai tanggal terakhir di dalam *database*, dan parameter *d, e, f* sebagai tanggal saat ini maka akan didapatkan data nilai saham terbaru yang belum ada di dalam *database*. Parameter *s* diisi dengan nama *stock* yang ada di dalam *database* dengan ditambahi indeks .JK. Parameter *g* selalu bernilai d karena data yang dibutuhkan oleh sistem adalah data harian. Data yang berhasil didapatkan disimpan di dalam variabel sourceData. Data tersebut berupa *file* CSV sehingga perlu dipisah-pisah terlebih dahulu sebelum dimasukkan ke dalam *database*.

```
// separate into lines
$sourceLines = str_getcsv($sourceData, "\n");
foreach( $sourceLines as $line ) {

    if ($line == 0){
        //header tidak perlu dimasukkan ke database
    }
    else{
        $contents = str_getcsv( $line );
```

Gambar 3.48. Kode program untuk memisahkan komponen pada *file* csv

File CSV di dalam variabel sourceData dipisahkan barisnya terlebih dahulu. Karena setiap barisnya ditandai dengan ganti baris maka digunakanlah "\n" ketika memisahkan baris dengan perintah *str_getcsv*. Perintah tersebut berfungsi untuk mengubah *file* csv menjadi *string* dan memisahkannya. Setelah dipisahkan perbaris maka selanjutnya adalah mengambil komponen perbaris. Data dari Yahoo Finance mempunyai urutan komponen sebagai berikut Tanggal, *Open*, *High*, *Low*, *Close*, *Volume*, *Adj.Close*. Komponen tersebut dipisahkan dan disimpan dalam *array* pada variabel *contents*. Baris pertama dari data adalah *header* jadi tidak perlu dimasukkan di dalam *database*.

Setelah data dipisahkan, maka data siap untuk dimasukkan ke dalam *database*. Sebagai pengaman tambahan, maka sebelum dimasukkan ke dalam *database* tanggal harus diperiksa sekali lagi apakah sudah ada di dalam *database*. Data tanggal yang ganda untuk saham yang sama di dalam *database* dapat membuat *error* sistem. Oleh karena itu, tidak boleh salah memasukkan tanggal. Setelah tanggal diperiksa barulah data tersebut dimasukkan ke dalam *database*. Untuk data saham IHSG cara melakukan *update*nya sama dengan data saham perusahaam hanya saja URL parameter s yang digunakan diganti dengan JKSE.SE.

3.11.2. *Update* Data Prediksi

Selain data saham dan data IHSG, data nilai prediksi juga perlu dilakukan proses *update*. Setiap hari Administartor akan menambahkan *file* CSV ke dalam direktori yang bernama csv. Sistem akan memeriksa direktori tersebut dan mengambil data baru yang ada di dalam *file* tersebut. Format penulisan dari data prediksi yang ada di dalam *file* CSV terlihat pada gambar berikut.

```
1, p_high_1, p_low_1
2, p_high_2, p_low_2
3, p_high_3, p_low_3
4, p_high_4, p_low_4
5, p_high_5, p_low_5
```

Gambar 3.49. Format penulisan data prediksi dalam *file* CSV

p_high_1 dan *p_low_1* merupakan data prediksi *high* dan *low* untuk hari ke-1, *p_high_2* dan *p_low_2* merupakan data prediksi *high* dan *low* untuk hari ke-2, demikian seterusnya sampai hari ke-5. Jadi di dalam sistem akan ada data prediksi untuk lima hari ke depan yang selalu di-*update*. Di dalam *file* CSV tersebut tidak ada informasi mengenai tanggal sehingga sistem harus menambahkan informasi tanggal untuk nilai prediksi yang bersangkutan karena nilai prediksi di dalam *database* harus disimpan dengan mencantumkan informasi tanggal.

Untuk menambah tanggal untuk lima hari kedepan, diperlukan informasi tanggal saat ini. Oleh karena itu, dibuat variabel *today* yang berisi informasi tanggal saat ini. Kemudian untuk mendapatkan informasi tanggal lima hari kedepan, sistem menambahkan satu hari dari tanggal hari ini hingga lima hari ke depan. Bursa saham akan tutup pada hari Sabtu dan Minggu, serta pada hari libur. Oleh karena itu, perlu dilakukan pemeriksaan terhadap hari yang telah ditambahkan tersebut untuk mengetahui apakah hari tutup Bursa atau tidak. Fungsi *check_valid* yang ada pada gambar 3.50, merupakan fungsi yang digunakan untuk memeriksa hal tersebut.

```
function check_valid($date){
    $isValid = true;
    $timestamp = strtotime($date);
    $day = date('N', $timestamp);
    if ($day == 6 || $day == 7){
        $isValid = false;
    }
    $mysql= mysql_query("SELECT * FROM holiday_data WHERE date = '$date'");
    if (mysql_num_rows($mysql) >= 1){
        $isValid = false;
    }
    return $isValid;
}
```

Gambar 3.50. Fungsi check_valid untuk memeriksa hari tutup bursa.

Tanggal yang dimasukkan dalam fungsi tersebut disimpan dalam variabel *date*. Tanggal tersebut kemudian diperiksa apakah hari Sabtu atau Minggu, angka 6 pada gambar merupakan hari Sabtu dan angka 7 adalah hari Minggu. Bila tanggal yang dimasukkan adalah hari Sabtu atau Minggu maka fungsi akan mengembalikan nilai false, artinya tanggal tersebut merupakan hari tutup bursa. Di dalam *database* sistem juga terdapat data mengenai hari libur dimana bursa akan tutup. Data hari libur yang dimaksud adalah hari libur yang ada di Indoensia dimana saham akan tutup, seperti hari Natal, Idul Fitri, dsb. Selain itu hari libur yang dimaksud juga merupakan hari Cuti Bersama dimana bursa juga akan tutup. Apabila tanggal yang dimasukkan ada pada *database* hari libur, maka fungsi juga akan mengembalikan nilai false.

Untuk mendapatkan hari pertama langkahnya adalah sebagai berikut, Tanggal hari ini akan ditambahkan ditambahkan satu hari dan disimpan dalam variabel *tommorow_one*. Variabel tersebut kemudian diperiksa apakah hari libur atau tidak dengan menggunakan fungsi *check_valid* dan hasilnya disimpan dalam variabel valid. Apabila variabel valid bernilai false maka data *tommotow_one* berisi hari tutup bursa, sehingga tanggal tersebut perlu ditambah satu hari lagi dan demikian seterusnya hingga variabel valid bernilai *true*. Variabel valid bernilai *true* artinya tanggal tersebut sudah bukan merupakan hari tutup bursa.

```
$tommorow_one = date('Y-m-d', strtotime($today . ' + 1 day'));
$valid = check_valid($tommorow_one);
while (!$valid){
    $tommorow_one = date('Y-m-d', strtotime($tommorow_one . ' + 1
day'));
    $valid = check_valid($tommorow_one);
    if($valid){
        break;
    }
}
```

Gambar 3.51. Kode program untuk mendapatkan hari pertama.

Hari kedua, ketiga, keempat, dan kelima didapatkan dengan metode yang sama dengan hari pertama. Ketika didapatkan lima buah tanggal, sistem melanjutkan ke proses berikutnya yaitu mengambil data dari *file* CSV. *File* CSV dibuka dengan menggunakan perintah *fopen*. *File* CSV di dalam *server* disimpan dengan menggunakan nama sesuai dengan kode saham yang ada di dalam *database*, misalnya nilai prediksi untuk saham bank BCA disimpan dengan nama BBCA.csv. *File* CSV yang berhasil dibuka kemudian diambil data-data yang ada di dalamnya dengan melakukan *parsing file* CSV. Parsing tersebut dapat dilakukan dengan menggunakan perintah PHP yaitu *fgetcsv*. Hasil dari parsing disimpan dalam variabel data. Kemudian data nilai prediksi *high* dapat diambil dalam *data[0]*, sedangkan data nilai prediksi *low* dapat diambil dalam *data[1]*.

```
    if (($handle = fopen("../csv/$array_stock[$y].csv", "r")) !==
FALSE) {
        while (($data = fgetcsv($handle, 1000, ",")) !== FALSE) {
            if ($row == 1){
                $double = mysql_query("SELECT date FROM
prediction_stock_data WHERE stock_id = '$array_id[$y]' AND date =
'$today'");
                if (mysql_num_rows($double) < 1){
                    $insert = mysql_query("INSERT INTO
prediction_stock_data (stock_id, date, p_high, p_low) VALUES
('$array_id[$y]', '$today', '$data[0]', '$data[1]')");
                }else if (mysql_num_rows($double) >= 1){
                    $update = mysql_query("UPDATE
prediction_stock_data SET p_high = '$data[0]', p_low = '$data[1]'
WHERE stock_id = '$array_id[$y]' AND date = '$today'");
                }
```

Gambar 3.52. Proses membuka *file* CSV, parsing, dan menyimpan ke *database*

Sebelum memasukkan nilai tersebut di dalam *database*, perlu diperiksa apakah data prediksi untuk tanggal yang akan dimasukkan sudah ada di dalam *database* atau belum. Hal ini dilakukan karena di dalam *database* tidak boleh ada data tanggal yang sama untuk *stock*_id yang sama. Apabila tanggal tersebut sudah ada di dalam *database*, *query* yang dilakukan adalah *query UPDATE*. Sedangkan bila data belum ada, maka *query* yang digunakan adalah *query INSERT*. Kode program untuk proses ini dapat dilihat pada gambar 3.52.

Kode program untuk melakukan *update* ini dijalankan dengan menggunakan CRON seperti yang dilakukan pada saat melakukan *update* data *stock* dan data IHSG. Hal yang berbeda adalah proses konfigurasi untuk menjalankan CRON. kode program ini tidak perlu dijalankan pada saat hari Sabtu dan Minggu, karena pada kedua hari tersebut bursa saham tutup dan Administrator tidak mengeluarkan *file* CSV yang baru. Terlepas dari CRON, di dalam kode program untuk melakukan *update* data prediksi, sistem melakukan pemeriksaan tanggal saat ini, sehingga proses *update* tidak akan dijalankan bila hari ini adalah hari libur. Hal ini dilakukan karena Administrator juga tidak melakukan *update* data prediksi bila hari libur.

Setelah file CSV berhasil di-*update*, sistem menghapus file CSV tersebut untuk menghindari sistem melakukan proses *update* file CSV yang lama, bila pada hari berikutnya admin lupa untuk memasukkan file CSV yang baru. Jika proses *update* gagal maka sistem akan memindahkan file CSV menuju direktori yang lain untuk keperluan *back up*.

3.11.3. *Update* Data Error

Setelah data asli dari Yahoo Finance dilakukan proses *update*, beberapa menit kemudian data prediksi dilakukan proses *update*. Setelah kedua data tersebut berhasil dilakukan proses *update*, bagian berikutnya adalah melakukan perhitungan nilai *error* untuk nilai prediksi. Setiap hari sistem akan melakukan perhitungan *error* untuk nilai prediksi. Caranya adalah dengan membandingkan data asli dan data prediksi.

Sistem mengambil dua buah data yaitu *high* dan *low*. Data *high* dan *low* yang diambil merupakan data asli dan data prediksi untuk tanggal yang sama. Jadi

total diambil empat buah data, Kemudian data asli dan prediksi tersebut dihitung persentase *error* nya dengan menggunakan rumus berikut.

$$DE = \frac{Prediction - Real}{Real} \; x \; 100 \; \% \tag{3.2}$$

DE adalah *Daily Error* atau *error* harian. Data prediksi dan data asli yang digunakan dalam rumus di atas adalah untuk masing-masing nilai *high* dan *low*. Sehingga akan didapatkan dua buah *error* yaitu *error* dari nilai *high* dan *error* dari nilai *low*. Nilai persentase *error high* dan *error low* tersebut kemudian akan dimasukkan ke dalam *database*. Proses perhitungaan ini dilakukan dengan menggunakan *query* MySQL. Sistem akan melakukan *query* perhitungan terhadap data prediksi yang nilai *error* nya masih NULL atau kosong, kemudian melakukan *update* di dalam tabel prediction_*stock*_data. Kode program untuk melakukan *query* dapat dilihat pada gambar berikut.

```
$updatehigh = mysql_query("
update prediction_stock_data
join real_stock_data
on real_stock_data.date = prediction_stock_data.date and
real_stock_data.stock_id = prediction_stock_data.stock_id
and prediction_stock_data.error_high IS NULL
set error_high = ((p_high - high)/high*100);
");
```

Gambar 3.53. Kode program untuk melakukan *query* perhitungan *error high* dan *low*.

WE merupakan *Weekly Error* atau *error* mingguan. Data *error* harian yang telah disimpan di dalam *database*, dapat digunakan untuk menghitung *error* mingguan dari nilai prediksi saham. Sistem mengambil lima buah data *error* harian (DE). Data error tersebut merupakan lima data *error high* dan *low* terakhir dari *database*, kemudian dihitung RMS-nya untuk mendapatkan nilai *error* mingguan. Rumusnya sebagai berikut.

$$WE = \sqrt{\frac{1}{5}\sum_{i=1}^{5} DE_i^2} \quad (3.3)$$

ME merupakan *Monthly Error* yaitu error bulanan. Cara menghitung error bulanan hampir sama dengan cara menghitung error mingguan. Hanya saja sistem mengambil 30 data *error* harian (DE) terakhir dari database, untuk mendapatkan nilai error bulanan. 30 data *error high* dan *error low* tersebut dihitung RMS-nya dan didapatkanlah *error* bulanan. Rumus untuk menghitung *error* bulanan adalah sebagai berikut.

$$ME = \sqrt{\frac{1}{30}\sum_{i=1}^{30} DE_i^2} \quad (3.4)$$

3.12. Fitur *Account*

User yang telah melakukan registrasi akan mendapatkan *account* di dalam *website*. Sehingga di dalam *website* diperlukan sebuah menu untuk memfasilitasi *user* dalam mengelola *account* yang dimilikinya. Di dalam fitur *Account* ini, *user* dapat mengelola *account* sesuai dengan kebutuhan *user*. Ada tiga bagian sub menu yang terdapat di dalam fitur ini. Masing-masing sub menu dijelaskan pada sub bab yang berbeda.

3.12.1. *Account* Information

Sub menu ini berisi informasi mengenai *account* yang dimiliki oleh *user*. Di dalam menu ini terdapat informasi dasar yaitu.

- *Username*, yaitu *username* dari *account* yang dimiliki oleh *user*
- *Created Date* merupakan informasi tanggal *account* tersebut dibuat. Tanggal ini dibuat pada saat proses registrasi, yaitu pada saat proses pembayaran berhasil divalidasi.
- *Expiry Date* adalah informasi masa aktif dari *account*.

Di dalam menu ini, juga terdapat sebuah tombol yang digunakan mengelola status langganan dari *account*. Tombol manage *account* digunakan untuk mengelola status langganan dari *account*. *User* dapat menambah masa aktif dari *account* yang dimiliki dengan melakukan klik terhadap tombol ini. Ketika *user* melakukan klik, maka sistem akan mengarahkan *user* menuju form pembayaran seperti pada saat registrasi langkah ketiga. Pada form terebut, *user* akan memilih jenis pembayaran. Setelah pembayaran berhasil divalidasi, maka nilai dari *Expiry Date* akan bertambah sesuai dengan pembayaran yang dilakukan.

3.12.2. Personal Information

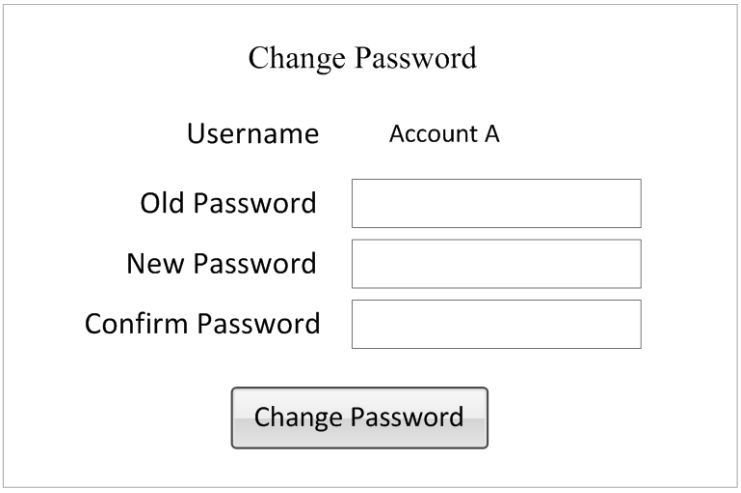
Sub menu ini berisi informasi mengenai data pribadi dari *user*. Data tersebut diisi oleh *user* ketika *user* melakukan registrasi *account*. Pada bagian ini disajikan kembali data tersebut. Informasi yang disajikan adalah,

- *Username*, yaitu *username* dari *account* yang digunakan oleh *user*.
- *Email* merupakan *email* yang dimiliki oleh *user*. *Email* ini adalah *email* aktif dimana segala informasi yang dibutuhkan oleh sistem akan dikirimkan ke *email* ini.
- *Gender* adalah informasi jenis kelamin dari *user*.
- *Job Role* yaitu informasi pekerjaan dari *user*.
- *City* merupakan informasi kota tempat *user* tinggal.

Username dan *email* merupakan informasi penting yang dibutuhkan oleh sistem. Sedangkan informasi *Gender*, *Job Role*, dan *City* merupakan informasi tambahan yang digunakan sebagai data *mining* oleh sistem. Di dalam sub menu ini, *user* juga dapat mengubah informasi personal dari *user* dengan melakukan klik terhadap tombol *Edit*. Ketika tombol tersebut diklik, maka akan muncul *textfield* untuk melakukan edit terhadap informasi personal tersebut. Data yang di-*edit* oleh *user* akan disimpan ke dalam *database*.

3.12.3. Change *Password*

User bisa mengganti *password* dari *account* yang digunakan ketika masuk ke sub menu *Change Password*. Untuk mengganti *password user* harus mengisi form yang disediakan. Layout dari form terlihat pada gambar berikut.



Gambar 3.54. Layout form untuk mengganti *password*

Dari form ini terdapat empat informasi yang dimunculkan yaitu,

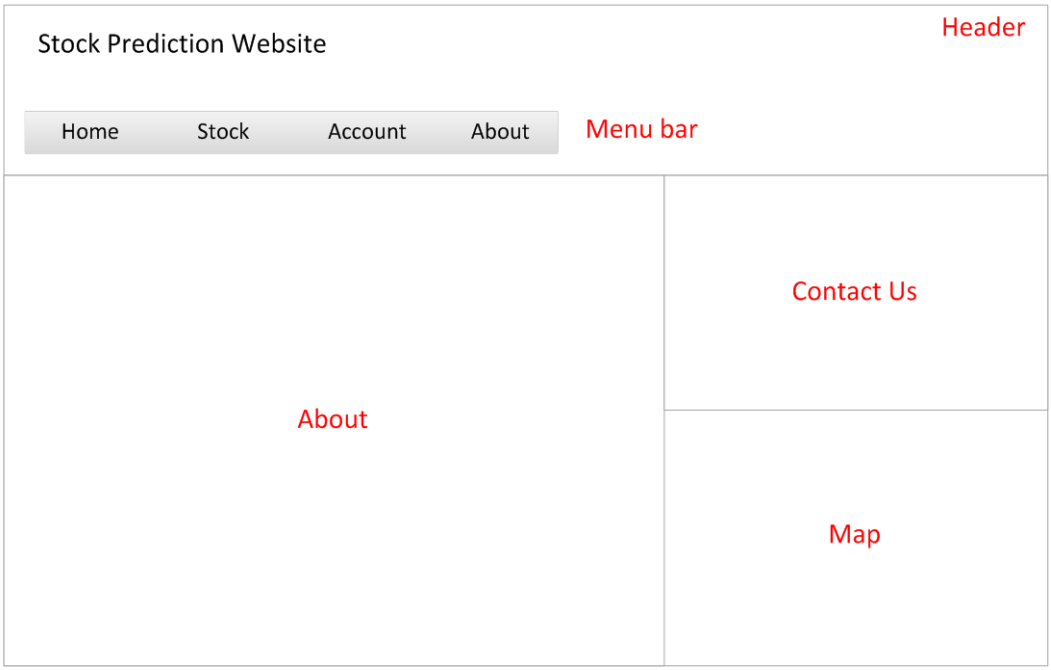
- *Username*, yaitu *username* dari *account* yang digunakan oleh *user*.
- *Old Password* merupakan *textfield* yang digunakan untuk memasukkan *password* lama yang dimiliki oleh *account*. Data yang dimasukkan ke dalam *textfield* ini harus sama dengan data *password* yang ada di dalam *database*. Apabila *user* salah memasukkan *password*, maka akan muncul pesan *error* dan proses penggantian *password* tidak dijalankan.
- *New Password* adalah *password* baru yang ingin digunakan oleh *user*.
- *Confirm* New *Password* berisi *password* baru yang digunakan oleh *user*. Informasi ini digunakan sebagai konfirmasi dari *password* yang dimasukkan oleh *user*. Hal ini dilakukan untuk memastikan bahwa *user* tidak melakukan salah ketik ketika memasukkan *password*.

Ketika tombol change *password* ditekan maka sistem akan memeriksa isi dari form yang dimasukkan oleh *user*. Proses pergantian *password* akan berhasil bila data *Old Password* sesuai dengan data *password* di dalam *database*, serta data *New Password* dan *Confirm New Password* berisi nilai yang sama.

3.13. Fitur *About*

Fitur *About* merupakan sebuah fitur yang memberikan informasi mengenai *website* tersebut. Di dalam sistem ini, maka fitur *about* akan memberikan penjelasan kepada *user* mengenai tujuan dan fungsi *website* ini. Tujuan dibuatnya

fitur *about* ini adalah agar *user* baru dapat mengerti informasi penting tentang *website*, seperti hal apa saja yang ada di dalam *website*, sekaligus mengajak *user* baru tersebut untuk membuat *account* di dalam *website*. Tujuan lainnya adalah memberikan informasi *email*, alamat, nomor telepon dari kantor perusahaan sehingga apabila *user* ingin bertanya mengenai *website* dapat melalui menghubungi media yang disediakan. Layout tampilan *About* terlihat pada gambar berikut.



Gambar 3.55. Layout tampilan fitur *about*.

Di dalam *About*, terdapat dua menu yaitu menu *About* sendiri dan menu *Contact us*. Menu *About* menjelaskan mengenai apa saja yang ada di dalam *website*, yaitu *website* menyediakan dua buah data *stock* yaitu data asli dan data prediksi. Pada menu *Contact us* berisi informasi *email*, alamat, kota, negara, nomor telepon, dan peta yang menunjukkan lokasi dari perusahaan. Selain itu juga terdapat *link* untuk mengirim *email* dan *link* untuk menuju ke facebook dan twitter. Untuk peta menggunakan Google Maps API yang disediakan oleh Google. Untuk menggunakan API tersebut, pertama-tama yang harus dilakukan adalah mendaftarkan diri ke dalam *developer* google. Setelah itu, sebuah *key* akan

didapatkan. *Key* bersifat unik dan akan selalu dipakai untuk memanggil *javascript* dari Google Maps API. Kode program untuk peta terlihat pada gambar berikut.

```
<script>
var myCenter=new google.maps.LatLng(-7.33949,112.73754);

function initialize()
{
var mapProp = {
  center:myCenter,
  zoom:15,
  mapTypeId:google.maps.MapTypeId.ROADMAP
  };

var map=new google.maps.Map(document.getElementById("googleMap"),
mapProp);

var marker=new google.maps.Marker({
  position:myCenter,
  });

marker.setMap(map);
```

Gambar 3.56. Kode program membuat peta dengan Google Maps API

myCenter merupakan data lokasi dari perusahaan berupa *Longitude* dan *Latitude*. Di sistem ini dimasukkan *Longitude* dan *Latitude* dari Universitas Kristen Petra. Konfigurasi peta lainnya yang dilakukan adalah mengatur titik tengah dari peta pada *field center*, *field* ini diberi nilai dari *myCenter*. Kemudian mengatur kedalaman dari peta pada *field zoom*, semakin dalam peta maka peta akan semakin terlihat dengan jelas. Field *mapTypeID* merupakan jenis dari peta yang akan ditampilkan, ada 2 jenis peta yaitu *RoadMap* atau peta biasa dan *SatelliteMap* yaitu peta yang berupa gambar asli yang didapatkan dari satelit. Variabel *marker* digunakan untuk memberikan marker yaitu tanda merah pada peta, *field position* digunakan untuk mengatur posisi dari marker tersebut. Marker kemudian dicetak dengan menggunakan perintah *setMap*. Variabel *map* digunakan untuk mencetak peta.

3.14. Fitur Admin Center

Beberapa bagian dari sistem yang dibuat dalam Tugas Akhir ini tidak semuanya dapat dilakukan secara otomatis. Ada beberapa bagian yang

memerlukan campur tangan dari manusia untuk melakukan pekerjaan di dalam sistem. Contohnya adalah pembayaran dengan menggunakan BCA, menambahkan *stock* baru, dan menambahkan data tanggal hari libur saham. Manusia yang melakukan pekerjaan tersebut disebut dengan istilah Administrator. Untuk melakukan pekerjaannya sebagai Administrator, maka di dalam sistem dibuatlah menu khusus yang terpisah dari *website*. Menu tersebut diberi nama Admin Center.

Admin Center dapat diakses dengan menggunakan URL yang sama dengan URL *website*. Hanya saja pada akhir URL diberi akhiran /admin. Dalam Tugas Akhir ini maka URL yang digunakan adalah *http://203.189.123.200/~m23409017/ta/admin*. Ketika URL tersebut dibuka halaman pertama yang muncul adalah menu untuk melakukan *Sign In*. Pada sistem dibuatlah sebuah form sederhana untuk melakukan *Sign In*, yaitu berupa dua buah *textfield username* dan *password*, serta sebuah tombol untuk mengirim data tersebut. Proses melakukan *Sign In* sama seperti yang dijelaskan pada Fitur *Sign In* pada Bab 3.7, hanya saja pada kasus ini tidak perlu dilakukan pemeriksaan *email_active* dan *user_expired*. Setelah berhasil masuk maka admin atau administrator dapat mengakses menu-menu untuk melaksanakan tugasnya.

3.14.1. BCA Validation

Bagian pertama dari tugas seorang admin adalah melakukan verifikasi pembayaran BCA yang telah dilakukan oleh *user*. Proses untuk melakukan verifikasi pembayaran terdapat di dalam *file validate.php*. Pertama-tama sistem menampilkan semua data dari tabel *bca_data* di dalam *database*, dimana nilai dari *valid_transfer* adalah 0. Nilai 0 berarti data pembayaran BCA tersebut belum dilakukan validasi. Data dalam tabel_bca ditambahkan ketika *user* mengisi form konfirmasi pembayaran BCA pada saat registrasi. Data dalam tabel juga ditambahkan ketika *user* mengisi form konfirmasi pembayaran BCA, pada saat *user* hendak memperpanjang masa aktif dari *account* melalui Fitur *Account*.

Dari beberapa data yang ada di dalam tabel *bca_data*, admin hanya memvalidasi proses transaksi yang benar-benar terjadi sesuai dengan data yang dikirimkan oleh *user*. Pada bagian ini admin harus mencocokkan antara data yang

dikirim oleh *user* dengan transaksi yang terjadi pada *account* BCA admin. Untuk melakukan validasi, admin melakukan klik terhadap tombol Vali*date*. Sistem akan memproses data yang dipilih oleh admin. Hal-hal yang dilakukan oleh sistem untuk melakukan proses validasi adalah,

- Melakukan *update* data *user*_expired dalam tabel *user_account*. Data *user*_expired yang ditambahkan disesuaikan dengan kondisi dari *account user*. Bila *account* dari *user* adalah *account* baru, atau pun *account* yang sudah habis masa aktifnya, maka *user_expired* ditambahkan dari tanggal saat proses validasi dilakukan hingga masa aktif berlangganan yang dipilih oleh *user* (1 bulan, 6 bulan, atau 12 bulan). Bila *account* dari *user* masih belum habis masa aktifnya (*user* melakukan perpanjangan) maka data *user_expired* ditambahkan dari tanggal terakhir dalam *database*.
- Melakukan *update* data pada tabel *bca_data*. Data yang dipilih oleh admin dengan melakukan klik terhadap tombol vali*date* merupakan data yang valid sehingga nilai dari *valid_transfer* harus diubah menjadi 1, agar data tersebut tidak muncul lagi pada tabel validasi, karena data tersebut telah divalidasi.
- Mengirim *email* kepada *user* untuk memberitahukan bahwa *account* yang dipesan oleh *user* sudah bisa digunakan.

3.14.2. Menambahkan saham baru

Untuk saat ini, data saham yang ada di dalam *database* adalah 20 saham. Seiring dengan berkembangnya *website* maka isi dari *website* akan bertambah pula. Pada masa yang akan datang, *user* akan menuntut *website* untuk menambah data saham sehingga *user* memiliki lebih banyak pilihan dalam mengetahui prediksi dari saham-saham yang ada. Oleh karena itu, di dalam sistem perlu ditambahkan sebuah menu untuk menambahkan data saham yang baru. Proses penambahan saham ini tidak bisa dilakukan secara otomatis karena proses ini memerlukan input dari seorang admin untuk menentukan jenis saham apa yang akan ditambahkan.

Sistem menyediakan sebuah form sederhana untuk menambahkan saham, yaitu berupa dua buah *textfield* yang berisi,

- *Stock Code*, merupakan informasi kode saham untuk saham yang baru. Kode saham ini harus sesuai dengan kode saham yang ada di dalam Yahoo Finance.
- *Stock Name*, merupakan informasi dari nama perusahan saham tersebut. Misalnya Bank BCA, PT. Unilever, dsb.

```
    $double = mysql_query("SELECT * FROM stock_list WHERE
stock_code='$add_code'");
    if (mysql_num_rows($double) >= 1){
        $msg = "Stock is already in the database";
    }
```

Gambar 3.57. Kode program untuk memeriksa kode saham yang sama.

Ketika tombol submit diklik oleh *user* maka sistem akan memproses data yang dikirim oleh admin. Data kode saham disimpan dalam variabel *add_code*, sedangkan data nama perusahaan disimpan dalam add_name. Proses pertama yang dilakukan adalah memeriksa apakah kode saham tersebut sudah ada di dalam *database*. Pada umumnya kode saham bersifat unik, sehingga di dalam satu bursa tidak ada perusahaan yang memiliki kode saham yang sama. Sistem melakukan *query* dengan informasi kode saham di dalam *add_code*. *Query* dilakukan pada tabel *stock*_list yang berisi daftar dari saham yang ada di dalam *database*. Bila *query* tersebut mendapatkan hasil berarti saham tersebut sudah ada di dalam *database*. Kode program untuk melakukan hal tersebut ada pada gambar 3.57.

Bila kode saham yang dimasukkan tidak ada di dalam *database*, maka proses berikutnya adalah memeriksa apakah kode saham tersebut terdapat di dalam Yahoo Finance atau tidak. Sistem mengumpulkan data *real* dengan memanfaatkan data saham yang disajikan di dalam Yahoo Finance, sehingga perlu dipastikan apakah kode saham untuk saham yang baru ada di dalam Yahoo Finance. Untuk memeriksa hal tersebut, kode saham dimasukkan ke dalam URL yang digunakan untuk mendapatkan data saham dari Yahoo Finance. Parameter yang digunakan adalah sebagai berikut,

- Parameter s diisi dengan kode saham dengan imbuhan .JK
- Parameter a, b, c diisi informasi tanggal awal dari saham. Semua data saham di dalam *database* sistem dimulai dari tanggal 1 Februari 2007

hingga sekarang. Oleh karena itu, nilai a, b, dan c diisi untuk tanggal 1 Februari 2007.

- Parameter d, e, dan f diisi dengan tanggal saat ini.
- Parameter g diisi dengan d karena data yang disimpan dalam *database* adalah data harian.

```
        $sourceURL =
"http://ichart.finance.yahoo.com/table.csv?s=$add_code.JK&a=1&b=1&c
=2007&d=$today_month&e=$today_day&f=$today_year&g=d&ignore=.csv";
        $sourceData  = file_get_contents( $sourceURL );

        if(empty($sourceData)){
            $msg = "Stock not found in Yahoo Finance";
        }else if(!empty($sourceData)){
            $insert1 = mysql_query("INSERT INTO stock_list
(stock_code, stock_name) VALUES ('$add_code', '$add_name')");
```

Gambar 3.58. Proses pemeriksaan kode saham di dalam Yahoo Finance

URL tersebut kemudian dijalankan dengan perintah *file_get_contents*. Hasilnya disimpan dalam variabel *sourceData*. Apabila variabel tersebut bernilai kosong atau *empty*, berarti sistem tidak bisa mendapatkan data dari Yahoo Finance. Maka diasumsikan bahwa kode saham yang tersebut tidak ada di dalam Yahoo Finance. Bila variabel tersebut tidak kosong berarti kode saham ada di dalam Yahoo Finance. Data dalam variabel tersebut kemudian dilakukan proses parsing CSV untuk mendapatkan data-data saham di dalamnya seperti *date*, *open*, *high*, *low*, *close*, dan *volume*. Proses berikutnya adalah memasukkan data-data tersebut di dalam *database*. Kode program untuk mendapatkan data dari Yahoo Finance untuk kode saham yang baru, dapat dilihat pada gambar 3.58.

3.14.3. Menambahkan data hari libur bursa

Format penulisan *file* CSV yang berisi data prediksi tidak mencantumkan informasi tanggal di dalamnya. Sehingga sistem harus menambahkan informasi tanggal tersebut terlebih dahulu, sebelum memasukkan data ke dalam *database* sistem. Namun, sistem tidak mengetahui pada tanggal berapa saja bursa saham akan tutup. Pada saat bursa saham tutup, maka proses jual beli saham akan

berhenti, sehingga pada saat itu tidak ada data *real* maupun data prediksi yang masuk ke dalam *database*.

KALENDER LIBUR BURSA TAHUN 2013

Bulan	Hari	Tgl	Keterangan	Hari Bursa
Januari	Selasa	1	Tahun Baru 2013	21
	Kamis	24	Maulid Nabi Muhammad SAW	
Februari	*Tidak ada hari libur kecuali Sabtu dan Minggu*			20
Maret	Selasa	12	Hari Raya Nyepi Tahun Baru Saka 1935	19
	Jumat	29	Wafat Yesus Kristus	
April	*Tidak ada hari libur kecuali Sabtu dan Minggu*			22
Mei	Kamis	9	Kenaikan Yesus Kristus	22
Juni	Kamis	6	Isra Mi'raj Nabi Muhammad SAW	19
Juli	*Tidak ada hari libur kecuali Sabtu dan Minggu*			23
Agustus	Senin	5	Cuti Bersama Idul Fitri 1434 Hijriyah	17
	Selasa	6	Cuti Bersama Idul Fitri 1434 Hijriyah	
	Rabu	7	Cuti Bersama Idul Fitri 1434 Hijriyah	

Gambar 3.59. Data tanggal hari libur bursa tahun 2013.

Sumber : PT. Bursa Efek Indonesia (2013, para.1)

Disinilah peran seorang admin dibutuhkan. Admin akan menambahkan data tanggal ke dalam *database* yaitu tanggal untuk hari libur bursa. Hari libur bursa pada umumnya sama seperti data hari libur yang ada di Indonesia. Tetapi ada sedikit perbedaan, yaitu ada penambahan hari Cuti Bersama yang penentuan tanggalnya dilakukan oleh Pemerintah. Bursa juga akan tutup apabila kegiatan kliring ditiadakan oleh Bank Indonesia, dan apabila ada pengumuman dari Pemerintah tentang peniadaan hari kerja pada tanggal tertentu. Contoh hari libur bursa tahun 2013 dapat dilihat pada gambar di atas. Jika melihat pada ketentuan hari libur ini, maka sistem tidak bisa mencari sendiri data tanggal hari libur bursa. Sistem membutuhkan seorang admin untuk melakukan entry data hari libur tersebut.

Pada menu ini, sistem menyediakan sebuah form yang dapat digunakan untuk melakukan entry data hari libur bursa. Form berupa dua buah *textfield* yaitu,

- Date merupakan informasi tanggal hari libur yang ingin dimasukkan ke dalam *database*
- Description merupakan penjelasan dari informasi tanggal tersebut. Penjelasan berupa informasi hari libur apa yang terjadi pada tanggal tersebut.

Ketika admin melakukan klik terhadap tombol Submit maka sistem akan memasukkan data yang dikirim oleh admin. Sebelum memasukkan sistem akan memeriksa apakah ada kemungkinan tanggal libur yang dimasukkan sudah ada di dalam *database*. Jika data belum ada di dalam *database* maka sistem akan memasukkan tanggal tersebut.

3.14.4. Menampilkan daftar saham yang ada di dalam *database*

Dalam fitur Admin Center ini, juga terdapat menu untuk menampilkan daftar seluruh saham yang ada di dalam *database*. Informasi ini hanya digunakan untuk melakukan proses *review* saham-saham apa saja yang sudah ada di dalam *database*. Data-data tersebut akan disajikan dalam sebuah tabel. Data yang ditampilkan adalah *stock_id*, kode saham, dan nama perusahaan dari saham.