BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Di SISPIK, departemen Teknologi Informasi (IT) memiliki struktur organisasi yang terdiri dari beberapa bagian untuk mendukung operasional teknologi sekolah. Struktur ini dirancang untuk memastikan setiap aspek teknologi, mulai dari perangkat keras hingga pengembangan perangkat lunak, dapat berjalan dengan optimal dan terkoordinasi. Struktur organisasi IT dapat dilihat pada Gambar

Departemen IT dipimpin oleh seorang IT Department Manager yang bertanggung jawab untuk mengelola seluruh aktivitas dan proyek IT di sekolah. Posisi ini dipegang oleh Bapak Junaedy. Beliau dibantu oleh empat staf yang memiliki spesialisasi di bidang masing-masing:

Web Development Staff: Bapak Adi Saputra bertugas dalam pengembangan dan pemeliharaan aplikasi berbasis web, termasuk sistem informasi sekolah. Database Staff: Bapak Iqbal bertanggung jawab untuk pengelolaan basis data. Hardware and Network Staff: Bapak Thomas memastikan perangkat keras dan jaringan sekolah berfungsi dengan baik serta mendukung kebutuhan operasional, siswa dan guru. Web Design: Bapak Edo bertugas untuk merancang spanduk, poster, sertifikat, pengeditan video dan foto untuk poster sekolah.

Struktur organisasi ini menunjukkan pembagian tugas yang jelas dan spesifik di setiap posisi, sehingga setiap individu dapat fokus pada tanggung jawabnya masing-masing. Hal ini juga memudahkan koordinasi antaranggota tim IT dalam mengelola infrastruktur teknologi di SISPIK. Dalam magang kali ini, penulis menjadi seorang *Web developer* dibawah bimbingan Pak Adi Saputra.

3.2 Tugas yang Dilakukan

Selama 4 bulan praktik kerja magang adapun tugas-tugas yang diberikan yaitu *Web Developer* yaitu

1. Belajar dan berkenalan dengan dotnet
2. Membuat *database*
3. Belajar menggunakan SQL Server *Management Studio*

4. Belajar menghubungkan SQL Server ke Microsoft Visual Studio
5. Belajar menggunakan parameter agar mempermudah kodingan.
6. Membuat *sign in, login, edit, delete* di dalam sebuah *web*.
7. Membuat *transaction history* pada *database* yang pernah dibuat.
8. Membuat *edit user* dan *edit list* untuk admin
9. Belajar mengenal *Backend* menggunakan bahasa C#

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1.

Tabel 3.1. Uraian pelaksanaan praktik kerja magang

Minggu ke	Pekerjaan yang dilakukan
1	• Memahami dan mempelajari seputar dotnet. • Membuat halaman *login* dan *sign up* serta *backend*-nya. • Membuat *database* dan menghubungkan ke *login* dan *sign up*. • Membuat *masterpage* untuk *register*, *admin*, dan *login*.
2	• Membuat *edit user*, *edit admin*, dan *edit list*. • Membuat *backend* untuk *edit user*, *edit admin*, dan *edit list*. • Menghubungkan *edit user* dan *edit admin* ke *database*. • Menghubungkan *masterpage* ke *user list*, *user edit*, dan *edit admin*.
Lanjut pada halaman berikutnya	

Tabel 3.1: Lanjutan Uraian pelaksanaan praktik kerja magang

Minggu ke	Pekerjaan yang dilakukan
3	• Membuat *chart* dan menghubungkannya ke *masterpage*. • Membuat *register*, *masterpage*, dan *database* untuk karyawan. • Membuat *login* untuk karyawan. • Membuat konten untuk karyawan. • Membuat *edit* dan *list* untuk karyawan. • Menerapkan *Bootstrap* pada *web*.
4	• Menerapkan sistem *role* pada *database*. • Membuat *stock list* dan *stock edit*. • Menerapkan *Bootstrap* pada *web* untuk mempercantik tampilan. • Membuat *database* untuk barang.
5	• Menghubungkan *stock list* dan *stock edit* dengan *database* barang yang sudah dibuat. • Membuat *shopping* dan *cart* untuk perbelanjaan. • Membuat *backend* untuk jual beli barang. • Membuat tampilan ”terimakasih” setelah pembelian. • Membuat tampilan *transaction history*.
	Lanjut pada halaman berikutnya

Tabel 3.1: Lanjutan Uraian pelaksanaan praktik kerja magang

Minggu ke	Pekerjaan yang dilakukan
6	• Membuat fitur untuk menghapus *history* dan memastikan pembelian muncul di *transaction history*. • Mempercantik tampilan *transaction history*. • Meningkatkan tampilan pada web agar lebih menarik.
7	• Membuat tampilan baru untuk *login* pada proyek baru. • Membuat *sidebar* dan *navbar*. • Mempercantik *sidebar* dan *navbar*. • Membuat *backend* untuk *login* dan *register*.
8	• Membuat *masterpage* dan menyambungkannya ke konten. • Belajar menghubungkan SQL Server *Management Studio* ke Microsoft Visual Studio. • Mengerjakan *login* dan *backend*-nya menggunakan SQL Server Management Studio. • Membuat template untuk *sidebar* dan *navbar*. • Belajar menggunakan Telerik. • Membuat *logic* untuk *login* dan *register*.
	Lanjut pada halaman berikutnya

Tabel 3.1: Lanjutan Uraian pelaksanaan praktik kerja magang

Minggu ke	Pekerjaan yang dilakukan
9	• Membuat *login* dan *register* untuk proyek baru. • Membuat *logic* untuk *login* serta template *sidebar* dan *navbar*. • Mempelajari penggunaan *parameter* dan *database*, lalu belajar menghubungkannya ke dotnet.
10	• Membuat *database* untuk *holding*, *school*, *users*, dan *school year*. • Membuat tampilan untuk masing-masing *database*. • Membuat tampilan untuk *edit*. • Membuat *backend* untuk masing-masing *database*.
11	• Memperbaiki kesalahan pada kode. • Menerapkan *primary key* dan *foreign key* pada database. • Membuat pilihan tambahan pada web.
12	• Memperbaiki tampilan dari masing-masing database pada *web*. • Memastikan apakah sudah berjalan dengan baik semua. • Merapikan sedikit *navbar* dan *sidebar*.
Lanjut pada halaman berikutnya	

Tabel 3.1: Lanjutan Uraian pelaksanaan praktik kerja magang

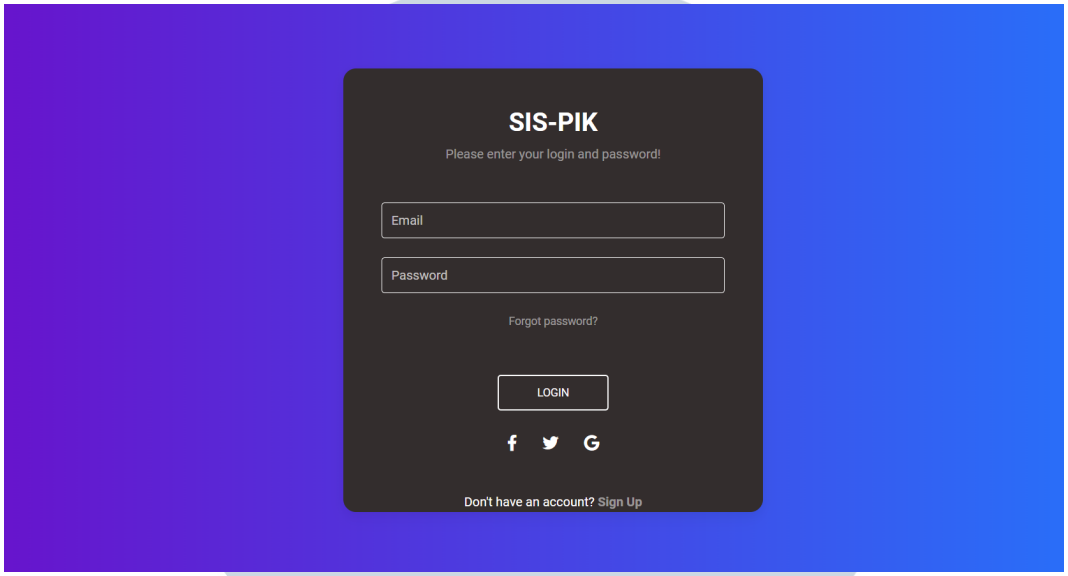
Minggu ke	Pekerjaan yang dilakukan
13	• Membuat tampilan *level* dan *levelgroup*. • Membuat *backend* untuk *level* dan *level group*. • Memperbaiki *error* dan melanjutkan *backend* untuk *level* dan *level group*.
14	• Membuat tampilan untuk *parent portal*. • Membuat *backend* untuk *student infomation* di *parent portal*. • Menambahkan beberapa informasi di *student infomation*. • Membuat tampilan dan *backend* untuk *student's grade*

3.4 Hasil Pengerjaan

Dalam pengerjaan proyek ini, bahasa pemrograman yang digunakan adalah C#, sedangkan untuk pengelolaan *database* menggunakan MySQL. Selain itu, proyek ini juga memanfaatkan *stored procedure*. Secara singkat, *stored procedure* merupakan sekumpulan perintah SQL yang disimpan di dalam *database* dan dapat dijalankan secara berulang.

Penggunaan *stored procedure* pada pembuatan *web* ini bertujuan untuk mempermudah tim IT dalam memasukkan data murid ke dalam sistem. Oleh karena itu, dibuatlah *project* ini sebagai langkah awal sebelum mengembangkan fitur *parent portal*. Fitur ini memungkinkan tim IT untuk mengelola data murid secara sistematis sebelum diintegrasikan ke dalam *web* utama.

3.4.1 *Login Page*



Gambar 3.1. *login page*

Gambar 3.1 tampilan dari *login page* yang dirancang untuk digunakan oleh para *admin*. Halaman ini dirancang agar dapat diakses dengan mudah dan mendukung autentikasi yang aman. Untuk melihat implementasi kode yang digunakan dalam pembuatan halaman ini. Untuk bagian

*backend*nya bisa diliat pada kode 3.1

```
using System;
using System.Data;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using SISDataAccess;

namespace sispik
{
    public partial class Login : System.Web.UI.Page
    {
        SchoolData pay = new SchoolData();

        protected void Page_Load(object sender, EventArgs e)
        {
            if (!Page.IsPostBack)
            {
```

```
18                  // Jika pengguna sudah diautentikasi, langsung
    arahkan ke halaman tujuan
19                  if (Request.IsAuthenticated)
20                  {
21
22                  }
23              }
24          }
25      }
```

Kode 3.1: Kode *Backend Login*

Kode 3.1 adalah bagian dari implementasi halaman login pada aplikasi berbasis ASP.NET *Web Forms*. Kelas *Login* yang digunakan mewarisi dari System.Web.UI.Page, menunjukkan bahwa kelas ini merupakan representasi dari sebuah halaman web. Sebuah objek bernama pay yang merupakan *instance* dari kelas *SchoolData* dideklarasikan untuk menangani berbagai operasi terkait data sekolah. Pada metode `Page_Load`, terdapat logika yang memastikan bahwa kode hanya akan dijalankan saat halaman pertama kali dimuat, yaitu ketika properti *Page.IsPostBack* bernilai *false*. Hal ini untuk mencegah pengolahan ulang data ketika terjadi *postback* akibat aksi pengguna seperti klik tombol atau pengiriman formulir. Selain itu, terdapat pengecekan terhadap status autentikasi pengguna menggunakan *Request.IsAuthenticated*. Jika pengguna sudah terautentikasi, maka blok kode tersebut dapat digunakan untuk mengarahkan pengguna secara langsung ke halaman tujuan tanpa harus kembali melalui proses *login*. Penggunaan fitur ini bertujuan untuk meningkatkan efisiensi dan kenyamanan pengguna dalam mengakses aplikasi.

Untuk bisa mengakses data dari SISPIK, harus diketik sintaks `using SISDataAccess;` baru kodingan itu akan jalan. `Page_Load` itu sendiri digunakan ketika pengguna atau *user* baru masuk atau halaman tersebut *refresh*.

if (!Page.IsPostBack) berguna untuk mengecek apakah halaman dimuat untuk pertama kali atau akibat *postback*. *postback* adalah mekanisme dimana data dikirim kembali ke server. Sedangkan if (Request.IsAuthenticated) berguna untuk mengecek apakah pengguna sudah *login* atau diautentikasi.

```
1  protected void Button_login_Click(object sender, EventArgs e)
2          {
3              string email = txtEmail.Text.Trim();
4              string password = txtPassword.Text.Trim();
5
```

```
6               if (string.IsNullOrEmpty(email) || string.
    IsNullOrEmpty(password))
7               {
8                   Response.Write("<script>alert('Diisi dulu ya')
    ;</script>");
9                   return;
10              }
11
12              DataTable dataaccount = (DataTable)pay.
    getdatabyparam2(1, email, password, 1);
13
14              if (dataaccount.Rows.Count == 1)
15              {
16                  FormsAuthentication.SetAuthCookie(email, true);
17                  Response.Cookies.Add(new HttpCookie("USERID",
    email));
18                  Response.Redirect("/test/users.aspx");
19              }
20              else
21              {
22                  Response.Write("<script>alert('Hayoo... salah
    password atau usernamenya hayoo');</script>");
23              }
24          }
```

Kode 3.2: Kode *Backend Login*

Pada saat tombol *login* ditekan, terdapat dua sintaks utama yang berfungsi untuk membaca dan memastikan input dari pengguna. Sintaks pertama, `string email = txtEmail.Text.Trim();` dan `string password = txtPassword.Text.Trim();`, digunakan untuk mengambil nilai yang dimasukkan oleh pengguna pada *textbox* yang ada di halaman *frontend*. Metode `Trim()` digunakan untuk menghapus spasi ekstra yang tidak diinginkan pada awal atau akhir input, sehingga memastikan data yang diambil lebih bersih dan siap untuk divalidasi.

Selanjutnya, sintaks `string.IsNullOrEmpty` digunakan untuk memeriksa apakah salah satu atau kedua input tersebut kosong. Langkah ini bertujuan untuk menghindari pemrosesan data yang tidak *valid* jika pengguna belum mengisi salah satu atau kedua kolom input. Dengan metode ini, sistem dapat mencegah kesalahan dan memastikan hanya data yang valid yang akan diproses lebih lanjut.

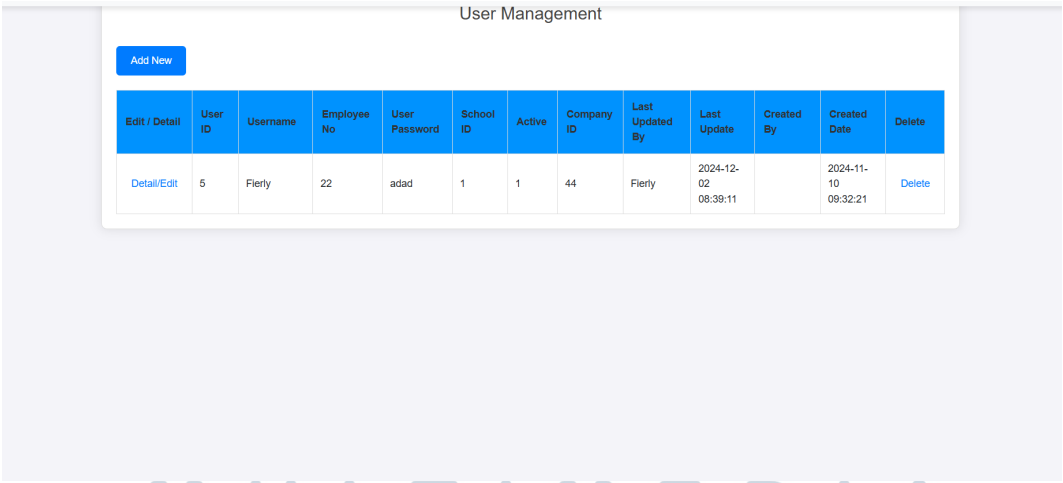
Jika ditemukan input yang kosong atau tidak *valid*, maka sintaks

`Response.Write("<script>alert(...)")` digunakan untuk menampilkan pesan *error* kepada pengguna melalui *JavaScript alert*. Pesan ini memberikan notifikasi secara langsung bahwa ada kesalahan pada input yang diberikan, sehingga pengguna dapat memperbaiki datanya sebelum mencoba kembali.

Selain itu, sintaks `DataTable dataaccount = (DataTable)pay.getdatabyparam2(1, email, password, 1);` digunakan untuk melakukan validasi lebih lanjut terhadap data pengguna. Sintaks ini memanfaatkan metode *stored procedure*, di mana:

- Angka `1` pada parameter pertama menunjukkan jenis perintah SQL yang digunakan.
- *email* dan *password* adalah nilai input yang dikirimkan untuk dicocokkan dengan data yang ada di *database*.
- Angka `1` pada parameter terakhir menunjukkan lokasi *database* di perangkat lokal.

3.4.2 *User Page*



Gambar 3.2. tampilan *user*

Gambar 3.2 adalah tampilan *user* yang berguna untuk menyimpan *admin* yang bisa masuk kedalam *web* tersebut. Untuk membuat tabel itu didalam dotnet menggunakan salah satu *tools* bernama *Gridview*.

```
<div class="container">
      <h1>User Management</h1>
      <asp:Button
          ID="btnNew"
          runat="server"
          Text="Add New"
          CssClass="btn-new"
          OnClick="btnNew_Click" />

      <asp:GridView
          ID="GridView1"
          runat="server"
          AutoGenerateColumns="False"
          AllowPaging="True"
          HeaderStyle-BackColor="#f0f0f0"
          HeaderStyle-ForeColor="black"
          ShowHeaderWhenEmpty="True"
          CssClass="gridview"
          OnPageIndexChanging="GridView1_PageIndexChanging" >
          <Columns>
              <asp:TemplateField HeaderText="Edit / Detail">
                  <ItemTemplate>
                      <asp:Label
                          ID="lblID" runat="server" Visible="
  false" Text='<%# Bind("userid") %>'/>
                      <asp:LinkButton
                          ID="lnkview"
                          runat="server"
                          Text="Detail/Edit"
                          CssClass="link-button"
                          OnClick="lnkview_Click"/>
                  </ItemTemplate>
                  <ItemStyle HorizontalAlign="Center" />
              </asp:TemplateField>

              <asp:BoundField DataField="userid" HeaderText="
  User ID" ItemStyle-Width="100px" />
              <asp:BoundField DataField="username" HeaderText
  ="Username" ItemStyle-Width="150px" />
              <asp:BoundField DataField="Employeeno"
  HeaderText="Employee No" ItemStyle-Width="120px" />
              <asp:BoundField DataField="userpassword"
  HeaderText="User Password" ItemStyle-Width="120px" />
```

```
39                 <asp:BoundField DataField="schoolid" HeaderText
   ="School ID" ItemStyle-Width="100px" />
40                 <asp:BoundField DataField="active" HeaderText="
   Active" ItemStyle-HorizontalAlign="Center" ItemStyle-Width="
   80px" />
41                 <asp:BoundField DataField="compid" HeaderText="
   Company ID" ItemStyle-Width="100px" />
42                 <asp:BoundField DataField="lastupdateby"
   HeaderText="Last Updated By" ItemStyle-Width="150px" />
43                 <asp:BoundField DataField="lastupdate"
   HeaderText="Last Update" DataFormatString="{0:yyyy-MM-dd HH:
   mm:ss}" ItemStyle-HorizontalAlign="Center" ItemStyle-Width="
   150px" />
44                 <asp:BoundField DataField="createby" HeaderText
   ="Created By" ItemStyle-Width="150px" />
45                 <asp:BoundField DataField="createdate"
   HeaderText="Created Date" DataFormatString="{0:yyyy-MM-dd HH
   :mm:ss}" ItemStyle-HorizontalAlign="Center" ItemStyle-Width=
   "150px" />
46
47                 <asp:TemplateField HeaderText="Delete">
48                     <ItemTemplate>
49                         <asp:LinkButton
50                             ID="lnkdelete"
51                             runat="server"
52                             Text="Delete"
53                             CssClass="delete-link"
54                             OnClientClick="return confirm('This
    action will remove the selected item. Are you sure?');"
55                             OnClick="lnkdelete_Click" />
56                     </ItemTemplate>
57                     <ItemStyle HorizontalAlign="Center" />
58                 </asp:TemplateField>
59             </Columns>
60             <HeaderStyle BackColor="#f0f0f0" ForeColor="Black"
   />
61         </asp:GridView>
62     </div>
```

Kode 3.3: Kode *User*

Kode 3.3 merupakan implementasi halaman manajemen pengguna yang dirancang menggunakan ASP.NET *Web Forms*. Tampilan utama diatur dalam

sebuah kontainer dengan judul "*User Management*," yang berfungsi memberikan identitas halaman. Terdapat tombol "*Add New*" yang memungkinkan pengguna menambahkan data baru dengan menangani klik melalui metode *server-side* bernama *btnNew_Click*. Komponen inti halaman ini adalah *GridView* yang digunakan untuk menampilkan daftar pengguna dalam bentuk tabel. Properti seperti *AutoGenerateColumns* diatur ke *False* untuk memungkinkan kustomisasi kolom, dan *AllowPaging* diaktifkan untuk mendukung *pagination*, sehingga memudahkan navigasi data dalam jumlah besar. Setiap kolom pada tabel mencakup berbagai informasi seperti *User ID*, *Username*, dan *Created Date*, dengan format tanggal yang diformat untuk keterbacaan yang lebih baik. Selain itu, terdapat tombol "*Detail/Edit*" pada setiap baris untuk melihat atau mengedit data, yang diimplementasikan menggunakan *LinkButton* dengan metode *handler lnkview_Click*. Fitur penghapusan data juga disediakan melalui tombol "*Delete*" dengan konfirmasi klien untuk mencegah penghapusan yang tidak disengaja, yang diproses oleh metode *lnkdelete_Click* di *server-side*. Elemen-elemen ini didukung oleh kelas CSS untuk menjaga konsistensi tampilan dan pengalaman pengguna yang responsif, menciptakan antarmuka manajemen data yang interaktif dan mudah digunakan.

BoundField DataField digunakan untuk menentukan nama kolom dari sumber data yang ingin ditampilkan dalam kolom *GridView*. Properti ini berfungsi untuk mengikat data secara langsung dari sumber data ke kolom yang bersangkutan. Sementara itu, properti *HeaderText* digunakan untuk menampilkan nama kolom pada *GridView*, sehingga pengguna dapat memahami isi dari setiap kolom berdasarkan nama yang tertera.

Selain *BoundField*, terdapat *TemplateField*, yaitu tipe kolom dalam kontrol data seperti *GridView* di ASP.NET yang memberikan fleksibilitas lebih dalam menyesuaikan tampilan atau interaksi data. Berbeda dengan *BoundField*, yang secara langsung mengikat data dari sumber data, *TemplateField* memungkinkan pengembang untuk menyisipkan kontrol server, seperti *TextBox*, *DropDownList*, atau *Button*, serta elemen HTML kustom. Hal ini memungkinkan pengembang untuk menciptakan antarmuka pengguna yang lebih dinamis dan interaktif sesuai kebutuhan aplikasi.

```
1  public partial class users : System.Web.UI.Page
2      {
3          SchoolData pay = new SchoolData();
4          protected void Page_Load(object sender, EventArgs e)
```

```
5           {
6               if (!Page.IsPostBack)
7               {
8                   bindgrid();
9               }
10
11          }
12          private void bindgrid()
13          {
14              GridView1.DataSource = pay.getdatabyparam2(2,
    string.Empty, string.Empty, 1);
15              GridView1.DataBind();
16          }
17          protected void GridView1_PageIndexChanging(object
    sender, GridViewPageEventArgs e)
18          {
19              GridView1.PageIndex = e.NewPageIndex;
20              bindgrid();
21          }
22          protected void btnNew_Click(object sender, EventArgs e)
23          {
24              Response.Redirect("/test/editUsers.aspx?ID=0");
25          }
26
27          protected void lnkview_Click(object sender, EventArgs e
    )
28          {
29              LinkButton btn = sender as LinkButton;
30              GridViewRow gRow = btn.NamingContainer as
    GridViewRow;
31              Label ids = (Label)gRow.FindControl("lblID");
32              Response.Redirect("/test/editUsers.aspx?ID="+ ids.
    Text);
33          }
34
35          protected void lnkdelete_Click(object sender, EventArgs
     e)
36          {
37              LinkButton btn = sender as LinkButton;
38              GridViewRow gRow = btn.NamingContainer as
    GridViewRow;
39              Label ids = (Label)gRow.FindControl("lblID");
40              pay.DeleteData(2, string.Empty, ids.Text, string.
```

```
        Empty, 1);
41              ScriptManager.RegisterStartupScript(this, this.
        GetType(), "Alert", "alert('Delete data successful')", true)
        ;
42              bindgrid();
43          }
44      }
```

Kode 3.4: Kode *Backend User*

Kode 3.4 di atas merupakan bagian dari *backend* yang berfungsi untuk mengelola *logic user*. Pada metode `Page_Load`, digunakan perintah `bindgrid();` yang berguna untuk menampilkan tabel yang telah dibuat pada aplikasi SQL Server *Management Studio*. Sintaks ini memungkinkan data yang tersimpan di *database* ditampilkan dalam format tabel di antarmuka pengguna.

Selanjutnya, pada sintaks `GridView1.DataSource`, komponen ini dapat diuraikan menjadi dua bagian, yaitu `GridView1` dan `DataSource`. `GridView1` merujuk pada *ID* dari *GridView* yang digunakan dalam halaman, sedangkan *DataSource* adalah properti yang menentukan sumber data yang akan digunakan oleh *GridView*. Dalam hal ini, sumber datanya berasal dari metode `pay.getdatabyparam`, yang mengakses data dari *stored procedure* atau perintah SQL.

Untuk menampilkan data dalam *GridView*, diperlukan sintaks tambahan `GridView1.DataBind();`. Setelah `DataSource` ditentukan, metode `DataBind()` harus dipanggil untuk mengikat data dari sumbernya ke dalam *GridView*. Proses ini mengisi baris-baris di *GridView* dengan data yang diambil dari sumber data, sehingga tabel pada antarmuka pengguna dapat ditampilkan dengan data yang relevan.

Selanjutnya ada `GridView1_PageIndexChanging`, yang berfungsi untuk menangani *event PageIndexChanging* pada kontrol *GridView*. *Event* ini dipicu ketika pengguna melakukan navigasi ke halaman berikutnya atau sebelumnya pada *GridView* dengan fitur *pagination*.

Setelah itu, ada `btnNew_Click`, yaitu perintah *OnClick* yang dipanggil ketika tombol tertentu diklik. Sintaks `Response.Redirect` digunakan untuk mengarahkan *user* ke halaman tujuan. Pada kasus ini, halaman yang dituju adalah `/test/editUsers.aspx`. Terakhir, *query* string `?ID=0` digunakan untuk menandakan bahwa tidak ada data spesifik yang sedang di*edit*. Nilai `ID=0` menunjukkan operasi ”Tambah baru”.

Untuk sintaks `lnkview_Click`. Sintaks `LinkButton btn = sender as LinkButton;` digunakan untuk membuat variabel `btn` dengan mengambil objek yang memicu *event* (dalam hal ini, objek yang merujuk pada *LinkButton*). `sender` berisi referensi ke elemen yang mengaktifkan *event*, dan dengan meng-*cast*-nya ke `LinkButton`, sistem dapat memastikan bahwa objek tersebut memiliki tipe yang sesuai.

Kemudian, `GridViewRow gRow = btn.NamingContainer as GridViewRow;` berfungsi untuk mendapatkan baris *GridView* tempat `LinkButton` berada. `btn.NamingContainer` merujuk pada kontainer dari `LinkButton`, yang dalam kasus ini adalah baris *GridView*. Variabel `gRow` menyimpan referensi ke baris tersebut.

Selanjutnya, `Label ids = (Label)gRow.FindControl("lblID");` digunakan untuk menemukan kontrol bernama `lblID` di dalam baris *GridView* (`gRow`). Kontrol ini adalah sebuah *Label* yang biasanya digunakan untuk menampilkan informasi seperti ID atau data relevan lainnya

dari baris yang dimaksud.

Terakhir, sintaks `Response.Redirect("/test/editUsers.aspx?ID=" + ids.Text);` digunakan untuk mengarahkan pengguna ke halaman `editUsers.aspx`. Nilai `ID` pada *query string* diisi dengan teks dari `Label lblID`. Nilai ini biasanya adalah ID pengguna yang diambil dari baris *GridView* yang berisi *LinkButton* yang diklik.

Sintaks `lnkdelete_Click`, yang membedakan dengan `lnkview_Click` adalah di *delete* ini menggunakan *param* lagi dan menggunakan *SCriptManager* untuk memberikan pesan atau notifikasi berupa *alert JavaScript*. Setelah sintaks itu sudah dijalankan ditutup kembali dengan `bindgrid();` untuk tetap menampilkan data pada *GridView*.

```
1 else IF (@Param = 2)
2 BEGIN
3   DELETE users WHERE userid = @ID2
4 END
```

Kode 3.5: MYSQL

Kode 3.5 merupakan sintaks di SQL untuk membuang satu kolom yang ada di *GridView*. Karena pada sintaks di SQL itu, memerintahkan untuk membuang ID tersebut, jadi yang di ID itu akan terbuang semua datanya.

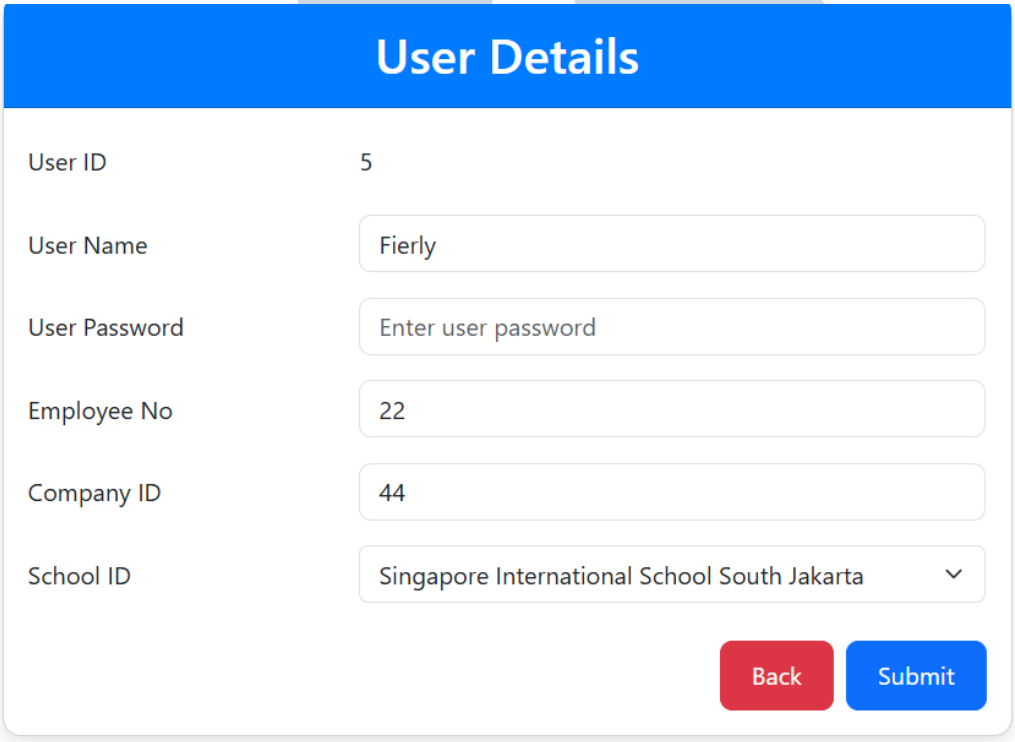
3.4.3 *Edit User Page*

```
1       <div class="mb-3 row">
2         <label for="DDLSchool" class="col-sm-4 col-form-label">
      School ID</label>
3           <div class="col-sm-8">
4             <asp:DropDownList runat="server" ID="DDLSchool"
      CssClass="form-select" DataTextField="schoolName"
      DataValueField="schoolID" />
5           /div>
6       </div>
```

Kode 3.6: Kode *Dropdown Edit User*

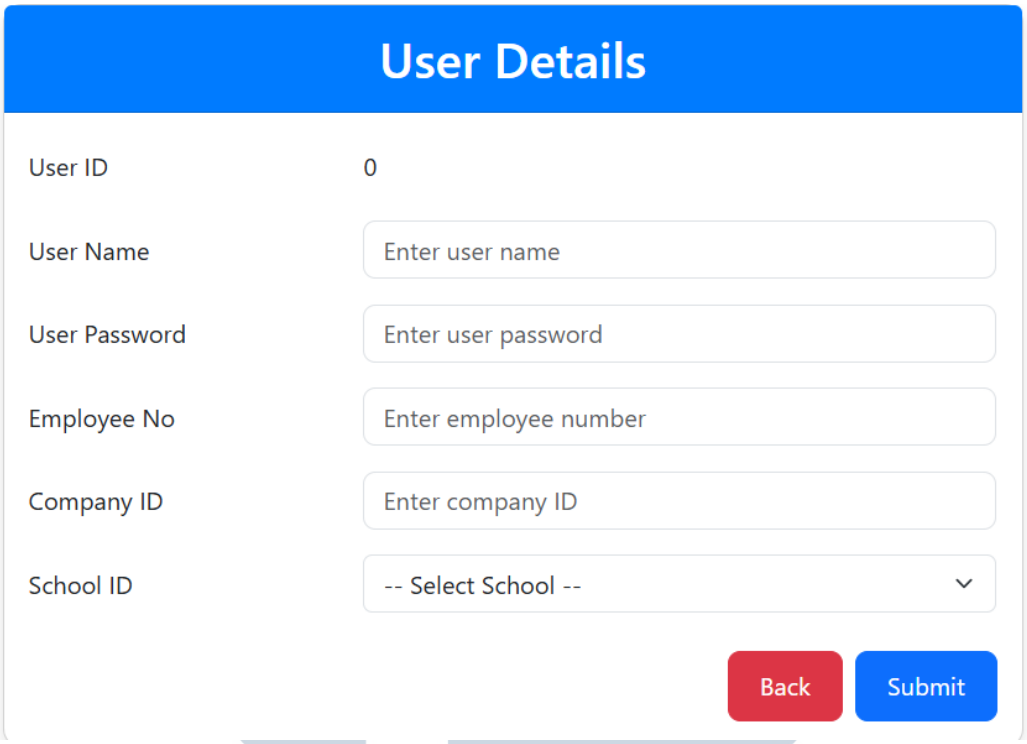
Kode 3.6 adalah bagian dari formulir HTML yang dirancang menggunakan ASP.NET untuk menampilkan *dropdown list (combo box)* guna memilih *School ID*. Elemen ini menggunakan *DropDownList server-side* ASP.NET dengan atribut *ID* bernama *DDLSchool*. *Dropdown* ini mengambil data dari sumber yang ditentukan di *backend*, dengan properti *DataTextField* yang diatur untuk menampilkan nama sekolah (*schoolName*) dan *DataValueField* untuk menyimpan nilai unik dari setiap sekolah (*schoolID*). *Label "School ID"* ditampilkan di sebelah kiri *dropdown* sebagai petunjuk, diatur dengan menggunakan elemen label dan dikelompokkan dalam tata letak bootstrap row untuk memastikan struktur yang rapi dan responsif. *Class form-select* diterapkan pada *DropDownList* untuk mengadopsi gaya standar Bootstrap, memberikan tampilan yang

modern dan konsisten dengan antarmuka pengguna berbasis *web*. Kode 3.6 di atas digunakan untuk menampilkan komponen *dropdown*, dengan perbedaan utama bahwa dalam fitur *edit user*, komponen *DropDown List* digunakan untuk menampilkan beberapa pilihan dari *School ID*. *School ID* ini akan muncul setelah data diambil dari *database*, memudahkan pengguna untuk memilih *ID* sekolah yang sesuai.



Gambar 3.3. Tampilan *Edit User*

Gambar 3.3 adalah tampilan ketika ingin *edit* atau melihat *detail* dari *user*. Jika membuat baru, didalam *textbox* itu akan kosong, tidak ada isinya. Bisa diliat pada gambar 3.4.



Gambar 3.4. tampilan *add new*

Sintaks yang membedakan antara *edit* atau *detail* dengan menambah baru adalah di dalam bagian `btnNew_Click` menggunakan `?ID=0` yang menandakan tidak ada yang sedang di *edit*.

```
public partial class editUsers : System.Web.UI.Page
    {
        SchoolData pay = new SchoolData();
        SchoolData1 pay1 = new SchoolData1();
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!Page.IsPostBack)
            {
                string users = Request.Params["ID"].ToString();
                lblID.Text = users;
                DDLSchool.DataSource = pay.getdatabyparam2(15,
    string.Empty, string.Empty, 1);
                DDLSchool.DataBind();
                DDLSchool.Items.Insert(0, new ListItem("--
    Select School --", ""));
                foreach (DataRow dr in pay.getdatabyparam2(7,
    lblID.Text, string.Empty, 1).Rows)
                {
                    txtusername.Text = dr["username"].ToString
    ();
```

```
17                      txtpassword.Text = dr["userpassword"].
      ToString();
18                      EmployeeNo.Text = dr["EmployeeNo"].ToString
      ();
19                      CompID.Text = dr["compid"].ToString();
20                      DDLSchool.SelectedValue = dr["schoolID"].
      ToString();
21                  }
22              }
23          }
```

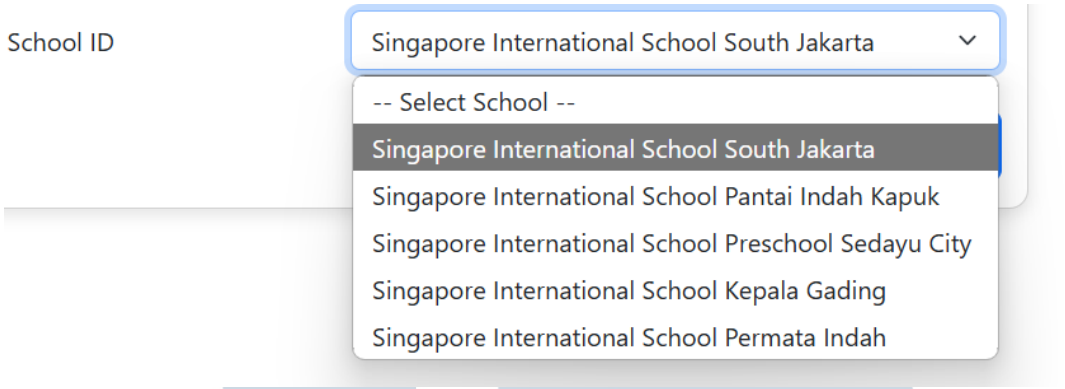
Kode 3.7: Kode *Edit User*

Kode 3.7 ini memuat data pengguna dan dropdown sekolah saat halaman diakses pertama kali. Kondisi `if (!Page.IsPostBack)` memastikan proses ini hanya berjalan pada load pertama, bukan saat postback. Parameter `ID` diambil dari URL dan ditampilkan pada label `lblID`. Data sekolah diambil menggunakan fungsi `pay.getdatabyparam2` dengan parameter tipe data `15`, kemudian diikat ke dropdown list `DDLSchool`, serta ditambahkan opsi default `"-- Select School --"`. Selanjutnya, data pengguna dengan `ID` tertentu diambil menggunakan fungsi `pay.getdatabyparam2` dengan parameter tipe data `7`. Setiap baris data diisi ke kontrol form, seperti `txtusername` untuk nama pengguna, `txtpassword` untuk kata sandi, `EmployeeNo` untuk nomor pegawai, `CompID` untuk ID perusahaan, dan `DDLSchool.SelectedValue` untuk menentukan sekolah yang dipilih. Proses ini memungkinkan halaman untuk menampilkan data pengguna yang dapat diedit dengan mudah. Sintaks *foreach* digunakan untuk membaca setiap baris data dari hasil tersebut.

```
1 else IF (@param1 = 15)
2 BEGIN
3   SELECT * FROM school order BY schoolid
4 END
```

Kode 3.8: MySQL

Jadi, nama nya diurutkan berdasar ID yang sudah ada di dalam *database*. Agar lebih jelas, bisa dilihat pada gambar 3.5



Gambar 3.5. Tampilan *Dropdown*

Sekolah-sekolah dalam daftar tersebut diurutkan berdasarkan `ID` yang telah ditentukan sebelumnya. Untuk mendukung pengurutan ini, *dropdown list* menggunakan properti `DataTextField` dan `DataValueField`. Kedua properti ini memiliki fungsi yang berbeda namun saling melengkapi.

`DataTextField` digunakan untuk menentukan teks yang akan ditampilkan kepada pengguna dalam daftar. Sebaliknya, `DataValueField` berfungsi untuk menentukan nilai unik yang terkait dengan setiap item dalam daftar tersebut. Sebagai contoh, seperti yang ditunjukkan pada kode 3.6, properti `DataTextField` di*set* ke `schoolName`, sehingga nama sekolah akan muncul sebagai opsi dalam *dropdown*. Sementara itu, properti `DataValueField` diatur ke `schoolID`, sehingga setiap pilihan dalam *dropdown* memiliki nilai unik yang merepresentasikan ID sekolah yang sesuai.

Penggunaan kedua properti ini memastikan bahwa pengguna dapat melihat nama sekolah dengan jelas dalam *dropdown list*, sementara sistem tetap dapat mengelola data secara akurat berdasarkan *ID* unik dari masing-masing sekolah.

```
1         protected void btnsubmit_Click(object sender, EventArgs
      e)
2         {
3             pay1.addnUpdateObject8(2, txtusername.Text,
      txtpassword.Text, DDLSchool.SelectedValue, EmployeeNo.Text,
      CompID.Text, string.Empty, lblID.Text, User.Identity.Name,1)
      ;
4             Response.Redirect("/test/users.aspx");
5         }
6
7         protected void btnback_Click(object sender, EventArgs e
      )
8         {
9             Response.Redirect("/test/users.aspx");
10        }
11    }
```

Kode 3.9: Kode *Edit User*

Pada kode 3.9 yang diberikan, terdapat dua metode *event handler* yang terkait dengan tombol di halaman *web*. Metode pertama adalah btnsubmit_Click, yang dijalankan ketika tombol *submit* diklik. Dalam metode ini, fungsi *addnUpdateObject*8 dari objek *pay1* dipanggil untuk menambahkan atau memperbarui entri di database dengan sejumlah parameter yang diambil dari elemen-elemen form di halaman tersebut, seperti nama pengguna (*txtusername.Text*), kata sandi (*txtpassword.Text*), *ID* sekolah yang dipilih dari *dropdown (DDLSchool.SelectedValue*), serta beberapa data lain yang diperlukan. Setelah operasi tersebut selesai, pengguna akan diarahkan ke halaman users.aspx. Metode kedua, yaitu *btnback_Click,* dijalankan ketika tombol kembali diklik. Metode ini hanya akan mengarahkan pengguna kembali ke halaman *users*.aspx tanpa melakukan perubahan data apapun. Secara keseluruhan, kode ini digunakan untuk mengelola *form* pengisian data pengguna, termasuk untuk memperbarui informasi dan navigasi kembali ke daftar pengguna.

```
1 else IF (@Param = 2)
2 BEGIN
3   UPDATE users SET username = @Value1, userpassword = @Value2,
     schoolid = @value3, Employeeno=@Value4, compid=@Value5,
     lastupdateby = @Value8, lastupdate = GETDATE()
4   WHERE userid = @Value7
5   IF (@@ROWCOUNT = 0)
6   BEGIN
7     INSERT users (username, userpassword, schoolid, EmployeeNo,
      compid, createby, createdate)
8     VALUES(@Value1, @Value2, @value3, @value4, @value5, @Value8
     ,GETDATE())
9   END
```

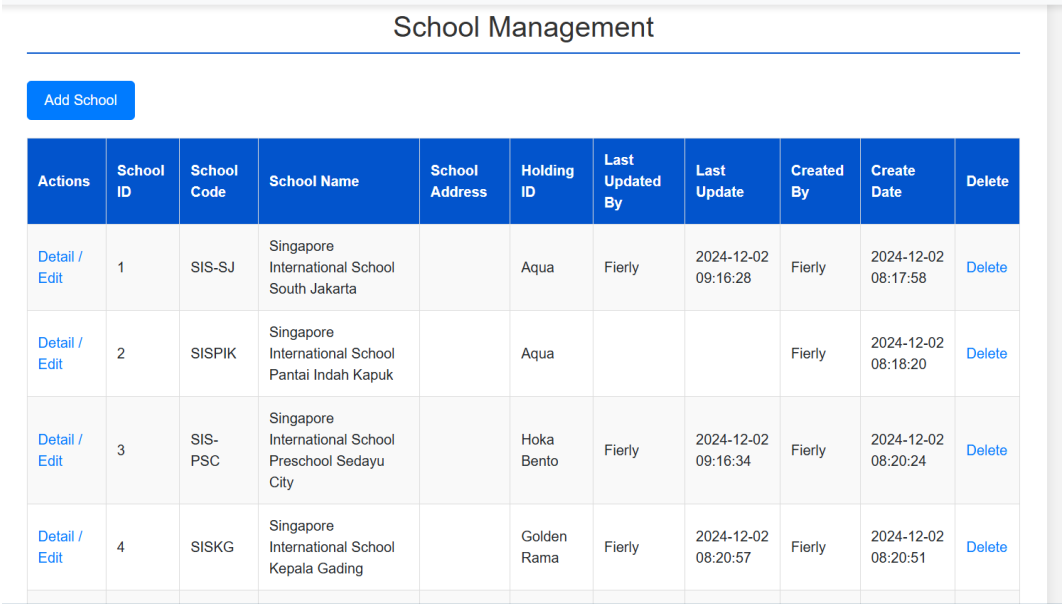
Kode 3.10: SQL *Edit User*

Kode 3.10 adalah sebuah logika untuk mengelola pembaruan data pengguna dalam sebuah sistem. Pertama, kode memeriksa apakah parameter yang diberikan adalah 2 melalui kondisi *IF* (@Param = 2). Jika ya, maka sistem akan mencoba memperbarui data pengguna yang ada dengan nilai-nilai baru yang diterima melalui parameter seperti @Value1 untuk username, *@Value2* untuk *userpassword*, dan seterusnya. Pembaruan ini terjadi pada pengguna yang memiliki userid sesuai dengan nilai *@Value7*. Proses ini juga mengubah kolom *lastupdate* dengan tanggal dan waktu saat ini. Setelah itu, sistem memeriksa apakah ada baris yang terpengaruh oleh pembaruan tersebut menggunakan *@@ROWCOUNT*. Jika tidak ada baris yang terpengaruh, ini berarti pengguna dengan userid tersebut tidak ditemukan, dan sistem kemudian melakukan penambahan data pengguna baru dengan perintah *INSERT*. Data yang dimasukkan mencakup informasi seperti *username*, *userpassword*, *schoolid*, dan lain-lain, dengan *createdate* diisi dengan tanggal dan waktu saat ini. Dengan demikian, logika ini memastikan bahwa data pengguna akan diperbarui jika ditemukan, dan jika tidak ditemukan, data baru akan ditambahkan ke dalam sistem.

Proses pengelolaan data ini dilakukan dengan menginputkan data yang ingin diubah atau ditambahkan ke dalam parameter `VALUES` dalam perintah SQL. Dengan demikian, semua perubahan data akan langsung tercatat di *database*.

Keuntungan dari pendekatan ini adalah sintaks di *backend* menjadi lebih sederhana karena cukup memanggil fungsi atau perintah SQL yang telah dibuat. Hal ini tidak hanya menghemat waktu, tetapi juga meminimalkan risiko kesalahan pada penulisan kode.

3.4.4 *School Page*



Gambar 3.6. tampilan *School Page*

Pada kodingan untuk membuat tampilan untuk *School Page* tidak jauh beda dengan kodingan sebelumnya. Masih menggunakan *Gridview*, *BoundField*, dan *TemplateField*. Kode 3.11 adalah *backend* untuk *School Page*.

```
1         public partial class School : System.Web.UI.Page
2     {
3           SchoolData pay = new SchoolData();
4           protected void Page_Load(object sender, EventArgs e)
5           {
6               bindgrid();
7           }
8           private void bindgrid()
9           {
10              GridView1.DataSource = pay.getdatabyparam2(3,string
      .Empty, string.Empty,1);
11              GridView1.DataBind();
12          }
13
14          protected void btnAdd_Click(object sender, EventArgs e)
```

```
15          {
16              Response.Redirect("editSchool.aspx?ID=0");
17          }
18
19          protected void lnkview_Click(object sender, EventArgs e
    )
20          {
21              LinkButton btn = sender as LinkButton;
22              GridViewRow gRow = btn.NamingContainer as
    GridViewRow;
23              Label ids = (Label)gRow.FindControl("lblID");
24              Response.Redirect("/test/editSchool.aspx?ID=" + ids
    .Text);
25          }
26
27          protected void lnkdelete_Click(object sender, EventArgs
     e)
28          {
29              LinkButton btn = sender as LinkButton;
30              GridViewRow gRow = btn.NamingContainer as
    GridViewRow;
31              Label ids = (Label)gRow.FindControl("lblID");
32              pay.DeleteData(3, string.Empty, string.Empty, ids.
    Text, 1);
33              ScriptManager.RegisterStartupScript(this, this.
    GetType(), "Alert", "alert('Delete data successful')", true)
    ;
34              bindgrid();
35          }
36          protected void GridView1_PageIndexChanging(object
    sender, GridViewPageEventArgs e)
37          {
38              GridView1.PageIndex = e.NewPageIndex;
39              bindgrid();
40          }
41      }
```

Kode 3.11: Kode *School Page*

Untuk kodingan *backend* juga kurang lebih sama, tapi yang membedakan pada `bindgrid();` karena dia menggunakan angka 3.

```
1 else IF (@Param1 = 3)
2 BEGIN
```

```
3    SELECT h.holdingname,a.* FROM school a LEFT OUTER JOIN
       Holding h ON a.holdingid = h.holdingid
4  END
```

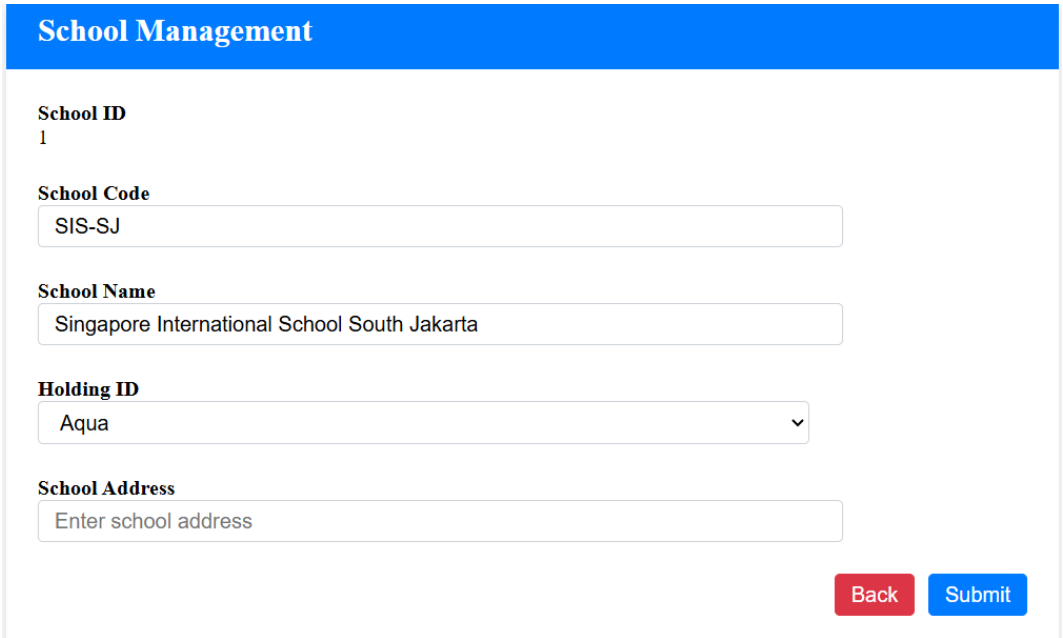
Kode 3.12: SQL School

Maksud dari sintaks pada 3.12, pada *query* ini, a adalah alias untuk tabel *users*, dan s adalah alias untuk tabel *school*. *Query* ini memilih kolom *schoolid* dari tabel *users*, semua kolom dari `users` (`a.`), dan semua kolom dari *school* (`s.`).

Dengan menggunakan *LEFT OUTER JOIN*, semua data dari tabel *users* akan ditampilkan, bahkan jika tidak ada kecocokan di tabel *school* berdasarkan kolom *schoolid*. Jika terdapat kecocokan antara kolom *schoolid* di kedua tabel, data dari kedua tabel akan digabungkan. Namun, jika tidak ada kecocokan, data dari tabel *school* akan berisi nilai *NULL*.

Sebagai contoh, jika tabel *users* memiliki *schoolid* yang tidak ditemukan dalam tabel *school*, hasil *query* tetap akan menyertakan data dari tabel *users*, tetapi kolom-kolom dari tabel *school* akan bernilai *NULL*. *Query* ini sangat berguna untuk menampilkan data yang terhubung dengan relasi antar-tabel, sambil tetap menyertakan data dari tabel utama meskipun tidak memiliki pasangan di tabel lainnya.

3.4.5 *Edit School Page*



Gambar 3.7. tampilan *Edit School Page*

```
1    public partial class editSchool : System.Web.UI.Page
2    {
```

```
3         SchoolData pay = new SchoolData();
4         SchoolData1 pay1 = new SchoolData1();
5         protected void Page_Load(object sender, EventArgs e)
6         {
7             if (!Page.IsPostBack)
8             {
9                 string school = Request.Params["ID"].ToString()
      ;
10                lblID.Text = school;
11
12                DDLHolding.DataSource = pay.getdatabyparam2(14,
       string.Empty, string.Empty, 1);
13                DDLHolding.DataBind();
14
15                DDLHolding.Items.Insert(0, new ListItem("--
      Select Holding --", ""));
16
17                foreach (DataRow dr in pay.getdatabyparam2(8,
      lblID.Text, string.Empty, 1).Rows)
18                {
19                    txtschoolcode.Text = dr["schoolcode"].
      ToString();
20                    txtschoolname.Text = dr["schoolname"].
      ToString();
21                    DDLHolding.SelectedValue = dr["HoldingID"].
      ToString();
22                    txtschooladdress.Text = dr["schooladdress"
      ].ToString();
23                }
24            }
25        }
26
27        protected void btnback_Click(object sender, EventArgs e
      )
28        {
29            Response.Redirect("/test/School.aspx");
30        }
31
32        protected void btnsubmit_Click(object sender, EventArgs
       e)
33        {
34            pay1.addnUpdateObject8(3, txtschoolcode.Text,
      txtschoolname.Text, DDLHolding.SelectedValue,
```

```
        txtschooladdress.Text, string.Empty, string.Empty, lblID.
        Text, User.Identity.Name, 1);
35              Response.Redirect("/test/School.aspx");
36          }
37      }
```

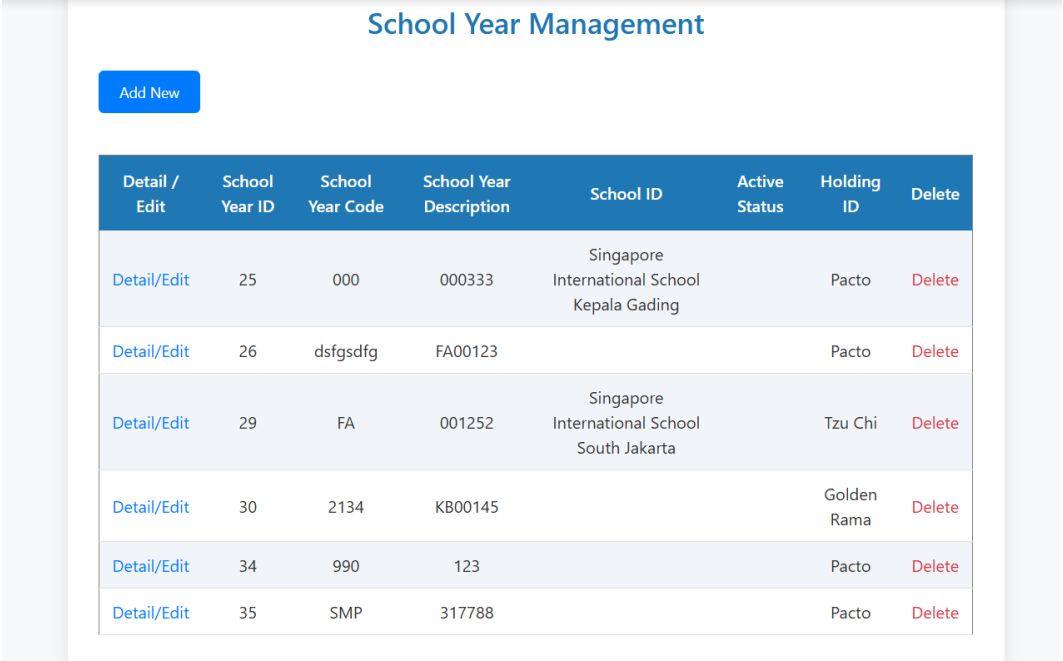
Kode 3.13: Kode *Edit School*

Kode 3.13 merupakan bagian dari halaman *web* yang menggunakan ASP.NET *Web Forms* untuk menangani operasi terkait data sekolah. Pada bagian *if (!Page.IsPostBack)*, halaman ini memeriksa apakah permintaan (*request*) pertama kali atau tidak. Jika pertama kali, nilai parameter *ID* yang ada di *URL* diambil dan ditampilkan di label *lblID*. Kemudian, data untuk dropdown *DDLHolding* diisi dengan hasil dari pemanggilan metode *getdatabyparam2*, yang mengambil data berdasarkan parameter tertentu. Item pertama pada *dropdown* tersebut diset menjadi opsi ”– *Select Holding* –”. Selanjutnya, data sekolah yang terkait dengan *lblID.Text* akan dimuat dan ditampilkan di berbagai kontrol input, seperti *txtschoolcode*, *txtschoolname*, *DDLHolding*, dan *txtschooladdress*.

Untuk tombol *btnback_Click*, fungsinya adalah untuk mengarahkan kembali pengguna ke halaman ”School.aspx” ketika tombol diklik. Sedangkan untuk tombol *btnsubmit_Click*, ketika diklik, data yang telah diisi dalam form akan diproses melalui metode *addnUpdateObject8*, yang kemungkinan besar akan menyimpan atau memperbarui informasi sekolah ke dalam database. Setelah itu, pengguna diarahkan kembali ke halaman ”*School*.aspx”

3.4.6 *School Year Page*

Gambar 3.8 adalah tampilan untuk *School Year*. Untuk kodingannya masih sama, menggunakan *label* dan *textbox*.



Gambar 3.8. Tampilan *School Year*

```
        <div class="container">
        <h1>School Year Management</h1>
        <div class="button-container">
            <asp:Button ID="btnAdd" runat="server" Text="Add New" CssClass="btn-add" OnClick="btnAdd_Click" />
        </div>

        <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" CssClass="gridview">
            <Columns>
                <asp:TemplateField HeaderText="Detail / Edit">
                    <ItemTemplate>
                        <asp:Label ID="lblID" Visible="false" runat="server" Text='<%# Bind("SchoolYearID") %>' />
                        <asp:LinkButton ID="lnkview" runat="server" Text="Detail/Edit" CssClass="link-button" OnClick="lnkview_Click" />
                    </ItemTemplate>
                </asp:TemplateField>
                <asp:BoundField DataField="SchoolYearID" HeaderText="School Year ID" />
                <asp:BoundField DataField="SchoolYearCode" HeaderText="School Year Code" />
```

```
17                 <asp:BoundField DataField="SchoolyearDesc"
    HeaderText="School Year Description" />
18                 <asp:BoundField DataField="SchoolName"
    HeaderText="School ID" />
19                 <asp:BoundField DataField="IsActive" HeaderText
    ="Active Status" />
20                 <asp:BoundField DataField="HoldingName"
    HeaderText="Holding ID" />
21                 <asp:TemplateField HeaderText="Delete">
22                     <ItemTemplate>
23                         <asp:LinkButton ID="lnkdelete" runat="
    server" Text="Delete" CssClass="delete-link" OnClientClick="
    return confirm('Are you sure you want to delete this school
    year?');" OnClick="lnkdelete_Click" />
24                     </ItemTemplate>
25                 </asp:TemplateField>
26             </Columns>
27             <HeaderStyle BackColor="#1f78b4" ForeColor="White"
    />
28         </asp:GridView>
29     </div>
```

Kode 3.14: Kode *School Year*

Kodingan tersebut adalah *front end* dari tampilan *School Year*. Dari kodingan tersebut, tidak ada yang membedakan dari kodingan sebelum-sebelumnya. Yang membedakan hanya pada kodingan *style*nya.

Backend ini juga tidak membedakan dari kode-kode yang sebelumnya. Yang membedakan hanya `bindgrid();` dan `lnkdelete_Click` karena perbedaan perintah yang digunakan ketika membuang dan mengambil datanya.

```
1 ELSE IF (@Param1 = 6)
2 BEGIN
3   SELECT a.*, h.HoldingName, s.SchoolName FROM Schoolyear a
     LEFT OUTER JOIN Holding h ON a.HoldingID = h.HoldingID LEFT
     OUTER JOIN School s ON a.SchoolID = s.SchoolID;
4 END
```

Kode 3.15: SQL Untuk Memunculkan Data

```
1 ELSE IF (@Param = 4)
2 BEGIN
3   DELETE schoolyear WHERE schoolyearid = @ID1
4 END
```

```
5 END
```

Kode 3.16: SQL untuk membuang data

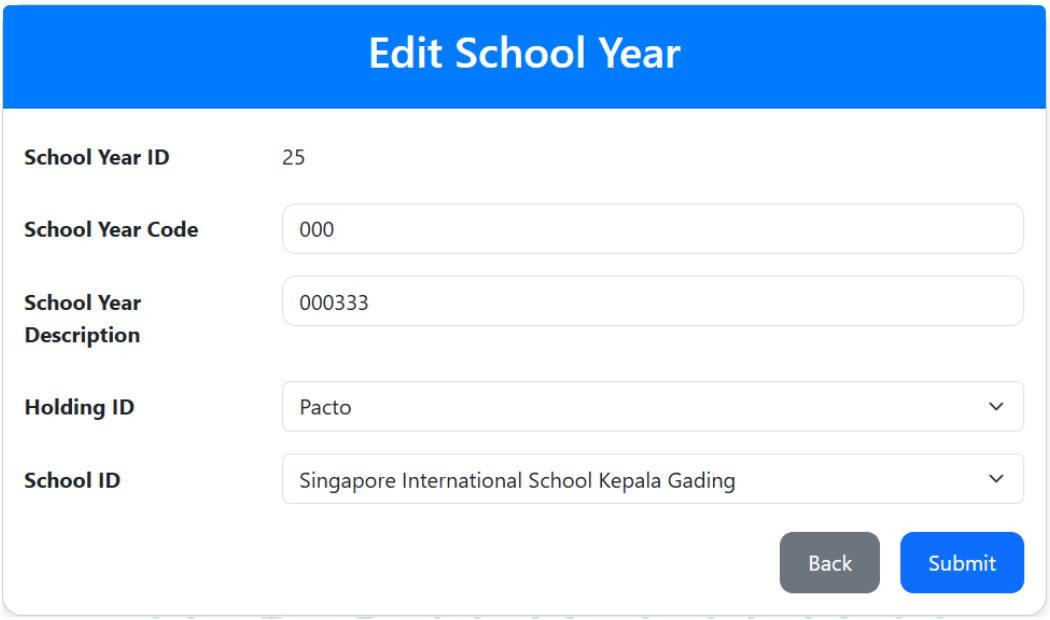
Gambar 3.15 menunjukkan *query* yang digunakan untuk mengambil data dari tabel *Schoolyear*, *Holding*, dan *School*. *Query* ini menghasilkan kombinasi data dari ketiga tabel tersebut. Bagian a.* berfungsi untuk mengambil seluruh kolom dari tabel *Schoolyear* yang diberi alias a. Sementara itu, kolom *HoldingName* diambil dari tabel *Holding* yang diberi alias h, dan kolom *SchoolName* diambil dari tabel *School* yang diberi alias s.

Proses pengambilan data dimulai dari tabel *Schoolyear* yang diberi alias a. Penggabungan dilakukan dengan menggunakan perintah *LEFT OUTER JOIN* antara tabel *Schoolyear* dan tabel *Holding*, yang dihubungkan melalui kolom *HoldingID*. Dengan metode ini, seluruh data dari tabel *Schoolyear* tetap ditampilkan, meskipun tidak terdapat data yang cocok di tabel *Holding*. Dalam kasus di mana tidak ada kecocokan, kolom-kolom dari tabel *Holding* akan bernilai NULL.

Selanjutnya, penggabungan dilakukan antara tabel *Schoolyear* dan tabel *School* menggunakan perintah *LEFT OUTER JOIN*, dengan relasi melalui kolom *SchoolID*. Sama seperti proses sebelumnya, seluruh data dari tabel *Schoolyear* akan tetap ditampilkan meskipun tidak ada kecocokan di tabel *School*. Apabila tidak ditemukan kecocokan, kolom-kolom dari tabel *School* akan bernilai *NULL*.

3.4.7 *Edit School Year Page*

Gambar 3.9 merupakan tampilan dari *Edit School Year*.



Gambar 3.9. Tampilan *Edit School Year*

```
1 namespace sispik.test
```

```
{
    public partial class EditSchoolYear : System.Web.UI.Page
    {
        SchoolData pay = new SchoolData();
        SchoolData1 pay1 = new SchoolData1();
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!Page.IsPostBack)
            {
                string schoolyear = Request.Params["ID"].
    ToString();
                lblID.Text = schoolyear;
                DDLHolding.DataSource = pay.getdatabyparam2(14,
     string.Empty, string.Empty, 1);
                DDLSchool.DataSource = pay.getdatabyparam2(15,
    string.Empty, string.Empty, 1);
                DDLHolding.DataBind();
                DDLHolding.Items.Insert(0, new ListItem("--
    Select Holding --", ""));
                DDLSchool.DataBind();
                DDLSchool.Items.Insert(0, new ListItem("--
    Select School --", ""));

                foreach (DataRow dr in pay.getdatabyparam2(9,
    lblID.Text, string.Empty, 1).Rows)
                {
                    txtschoolyearcode.Text = dr["schoolyearcode
    "].ToString();
                    txtschoolyeardesc.Text = dr["schoolyeardesc
    "].ToString();
                    DDLHolding.SelectedValue = dr["holdingID"].
    ToString();
                    DDLSchool.SelectedValue = dr["schoolID"].
    ToString();
                }
            }
        }

        protected void btnsubmit_Click(object sender, EventArgs
     e)
        {
            pay1.addnUpdateObject8(4, txtschoolyearcode.Text,
    txtschoolyeardesc.Text, DDLSchool.SelectedValue, DDLHolding.
```

```
      SelectedValue, string.Empty, string.Empty, lblID.Text, User.
      Identity.Name, 1);
33              Response.Redirect("/test/SchoolYear.aspx");
34          }
35
36          protected void btnback_Click(object sender, EventArgs e
      )
37          {
38              Response.Redirect("/test/SchoolYear.aspx");
39          }
40      }
41  }
```

Kode 3.17: Kode *Edit School Year*

Kode 3.17 adalah gambar *backend* dari *edit School Year*, yang membedakan dari kodingan ini adalah dalam *school year* ini memiliki dua *ID* jadinya menggunakan 2 *dropdown*. Jadi pada kode *backend*nya menggunakan 2 *DataSource* dan 2 *DataBind*.

```
1 else IF (@Param1 = 14) --editSchoolYear,editSchool
2 BEGIN
3   SELECT * FROM Holding ORDER BY holdingID
4 END
5
6 else IF (@param1 = 15) --editSchoolYear,editSchool,editusers,
      editLevelGroup
7 BEGIN
8   SELECT * FROM school order BY schoolid
9 END
```

Kode 3.18: Kode SQL *Edit School Year*

2 SQL pada kode 3.18 adalah perintah untuk mengurutkan *ID* dari *Holding* dan *School* agar muncul di *dropdown*.

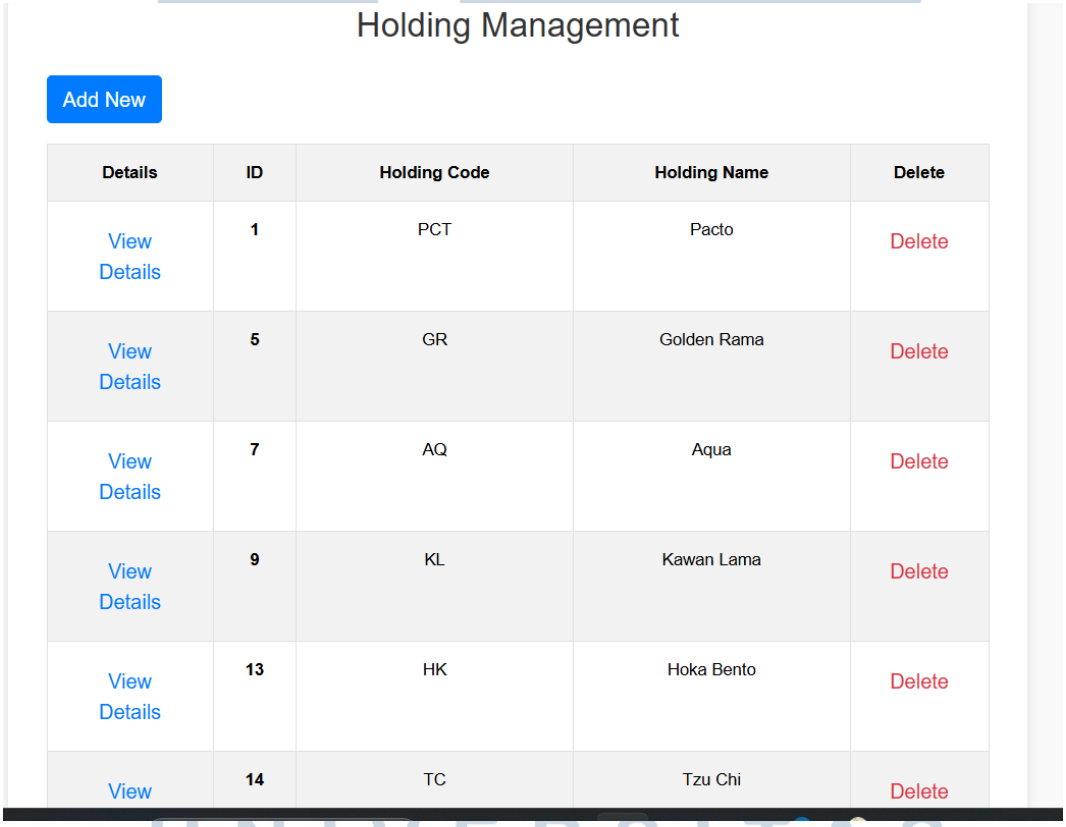
```
1 else IF (@Param = 4)
2 BEGIN
3   update schoolyear SET schoolyearcode= @Value1, schoolyeardesc
      = @Value2, schoolid=@value3, holdingid=@Value4, lastupdateby
      = @Value8, lastupdate=GETDATE()
4   where schoolyearid= @value7
5   IF(@@rowcount=0)
6   BEGIN
7       insert Schoolyear(schoolyearcode, schoolyeardesc,
      schoolid, holdingid, createby, createdate)
```

```
8       VALUES(@Value1, @Value2, @value3, @Value4, @Value8, GETDATE
        ())
9       END
10  END
```

Kode 3.19: Kode SQL *Edit School Year*

3.4.8 *Holding Page*

Gambar 3.20 adalah tampilan dari *Holding*. Holding itu sendiri memiliki arti nama perusahaan jika diartikan menurut SIS-PIK.



Gambar 3.10. Tampilan *Holding*

.

Pada kodingan *frontend* dan *backend* kurang lebih sama seperti yang sebelumnya. *Frontend* masih menggunakan *label*, *textbox*, dan *linkbutton*. Sedangkan *backend* masih sama dengan yang sebelumnya, hanya menyesuaikan isinya.

```
1   public partial class WebForm1 : System.Web.UI.Page
2       {
3
```

```
4           SchoolData pay = new SchoolData();
5           SchoolData1 pay1 = new SchoolData1();
6           protected void Page_Load(object sender, EventArgs e)
7           {
8
9               if (!Page.IsPostBack)
10              {
11                  BindGrid();
12              }
13          }
14
15          private void BindGrid()
16          {
17              GridView1.DataSource = pay.getdatabyparam2(4, "",
    string.Empty,1);
18              GridView1.DataBind();
19          }
20
21          protected void lnkview_Click(object sender, EventArgs e
    )
22          {
23              LinkButton btn = sender as LinkButton;
24              GridViewRow gRow = btn.NamingContainer as
    GridViewRow;
25              Label ids = (Label)gRow.FindControl("lblID");
26              Response.Redirect("WebForm1detail.aspx?ID="+ ids.
    Text);
27          }
28
29          protected void lnkdelete_Click(object sender, EventArgs
     e)
30          {
31              LinkButton btn = sender as LinkButton;
32              GridViewRow gRow = btn.NamingContainer as
    GridViewRow;
33              Label ids = (Label)gRow.FindControl("lblID");
34              pay.DeleteData(1, ids.Text, string.Empty, string.
    Empty, 1);
35              ScriptManager.RegisterStartupScript(this, this.
    GetType(), "Alert", "alert('Delete data successful')", true)
    ;
36              BindGrid();
37          }
```

```
38
39          protected void btnNew_Click(object sender, EventArgs e)
40          {
41              Response.Redirect("WebForm1detail.aspx?ID=0");
42          }
43      }
```

Kode 3.20: Kode *Holding Page*

Selanjutkan ada SQL untuk menghapus data dari *holding* yaitu pada gambar 3.21.

```
1 IF (@Param = 1)
2 BEGIN
3   DELETE Holding WHERE holdingid = @ID1
4 END
```

Kode 3.21: Kode SQL *Holding*

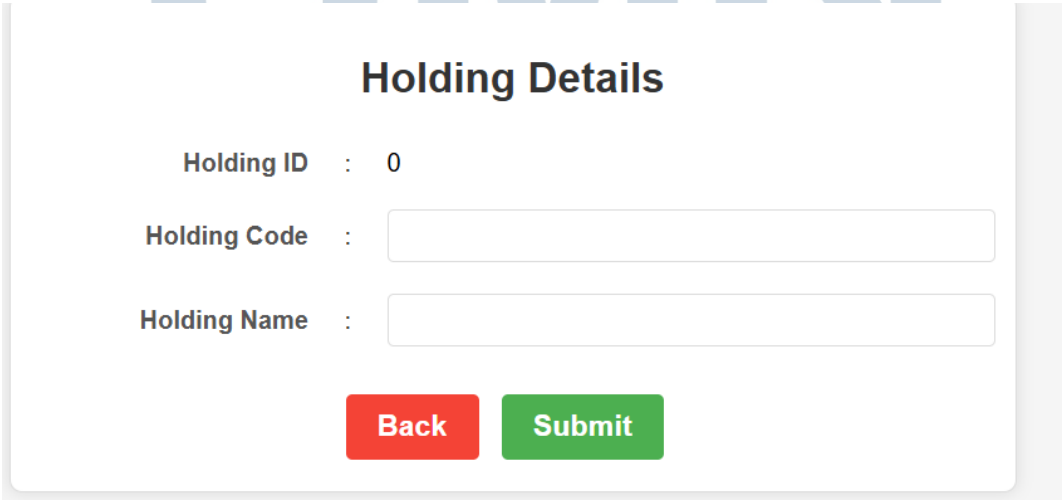
Terakhir, pada gambar 3.22, ada SQl yang menampilkan data agar muncul di *gridview*.

```
1 ELSE IF (@Param1 = 4) --holding
2 BEGIN
3   SELECT * FROM Holding
4 END
```

Kode 3.22: Kode SQL *Holding*

3.4.9 *Edit Holding Page*

Gambar 3.11 adalah tampilan dari *edit Holding*.



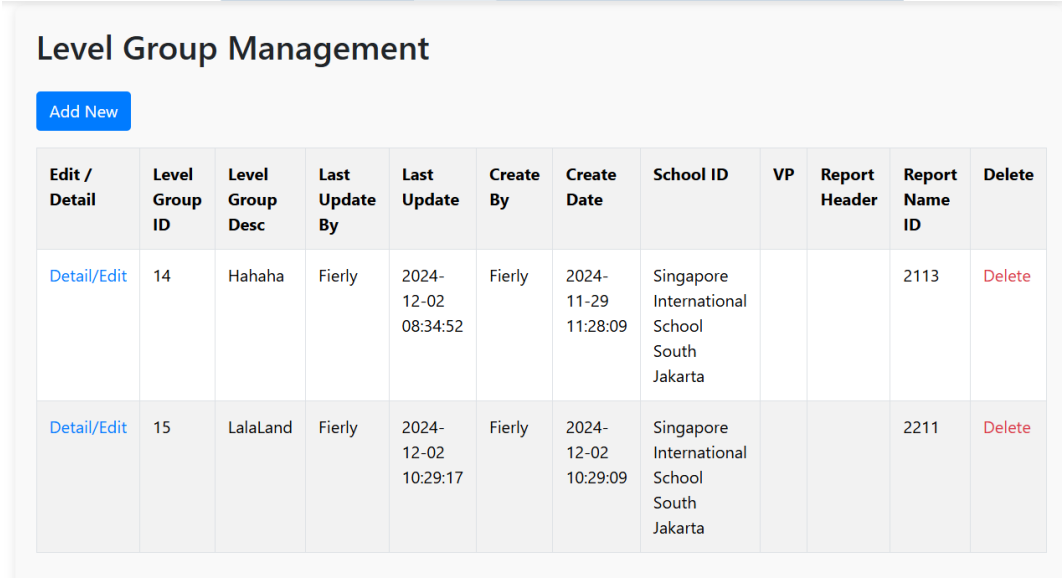
Gambar 3.11. *Edit Holding*

.

Untuk kodingannya sendiri mau *frontend*, *backend*, maupun SQL itu kurang lebih sama dari sebelum-sebelumnya. Hanya menyesuaikan saja.

3.4.10 *Level Group Page*

Tampilan dari *Level Group* bisa dilihat di gambar 3.12. *Level Group* ini adalah grup per kategori kelas seperti SD, SMP, SMA.



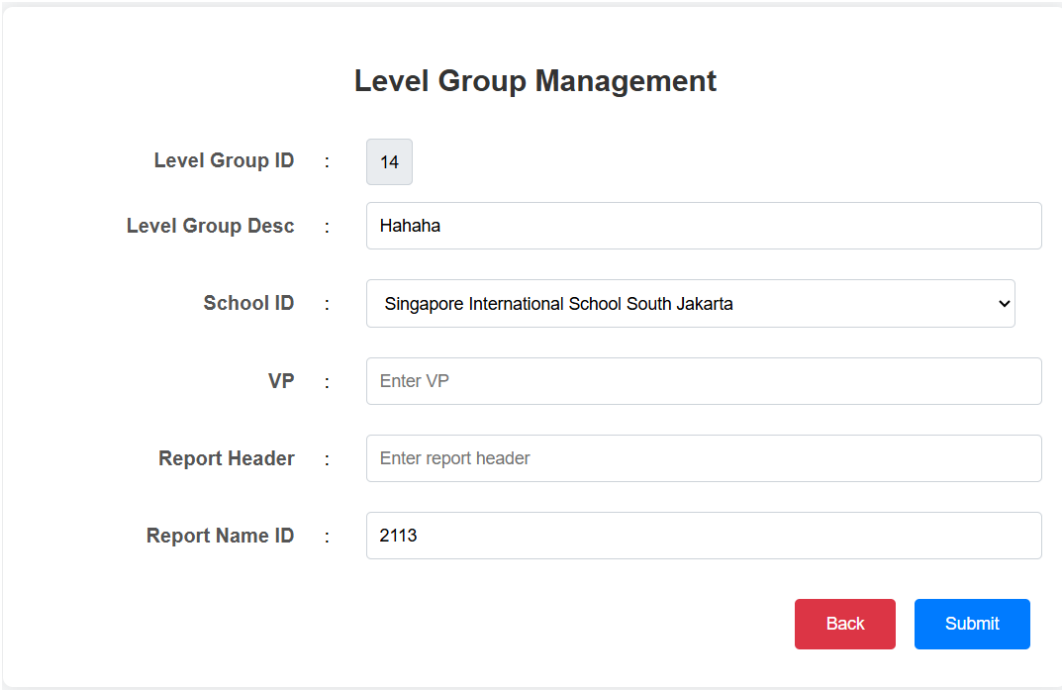
Gambar 3.12. Level Group

.

Untuk kodingan *frontend* dan *backend* kurang lebih sama dengan sebelumnya.

3.4.11 *Edit Level Group Page*

Tampilan dari *edit Level Group* bisa dilihat di gambar 3.13.



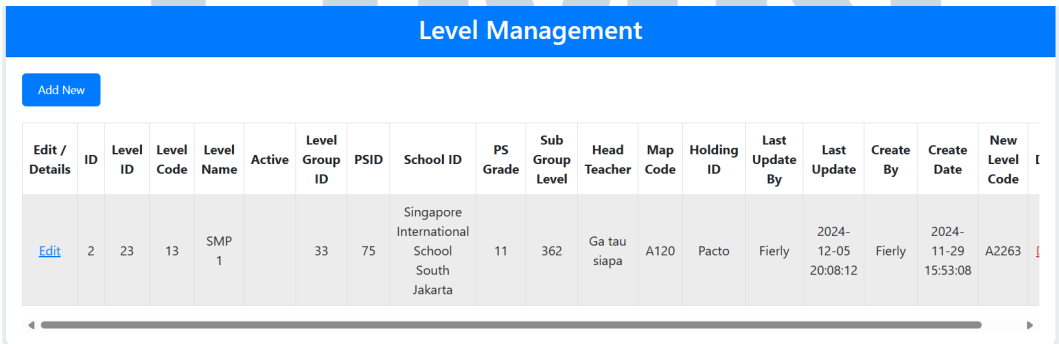
Gambar 3.13. *edit Level Group*

.

Untuk kodingan *frontend* dan *backend* kurang lebih sama seperti kodingan sebelumnya atau pada *page-page* sebelumnya.

3.4.12 *Level Page*

Tampilan *Level* dapat dilihat pada gambar 3.14



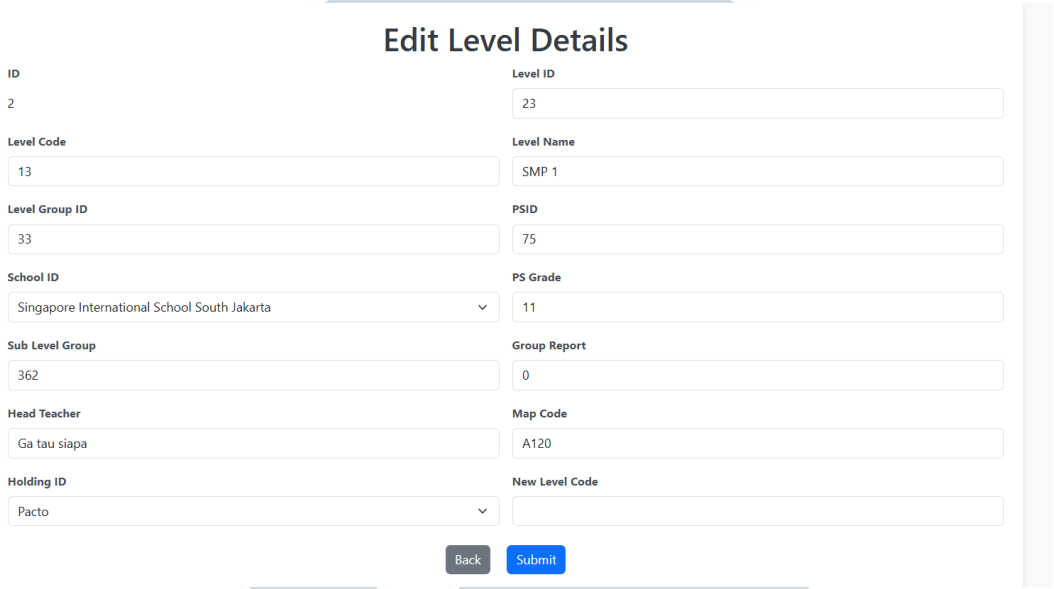
Gambar 3.14. backend

.

Untuk kodingan *frontend* dan *backend* kurang lebih sama seperti kodingan sebelumnya atau pada *page-page* sebelumnya.

3.4.13 *Edit Level Page*

Tampilan *edit level* dapat dilihat pada gambar 3.15.



Gambar 3.15. tampilan *edit level*

.

Untuk kodingan *frontend* dan *backend* kurang lebih sama seperti kodingan sebelumnya atau pada *page-page* sebelumnya.

Setelah ini, akan pembuatan *parent portal*. Gambar setelah ini adalah gambar dari tampilan dan *backend* dari *parent portal*.

3.4.14 *Login Parent Portal*

Gambar 3.16 adalah gambar dari *login parent portal*.

Gambar 3.16. Tampilan *Login Parent Portal*

```
public partial class _default : System.Web.UI.Page
{
    SchoolData pay = new SchoolData();
    SchoolData1 pay1 = new SchoolData1();
    SISCryptography.Crypto cr = new SISCryptography.Crypto
();
    //string passwordencr = "ptlcipayroll12345@R4h4514";
    //string APPID = "11";

    public enum MessageType { Success, Error, Info, Warning
 };

    protected void ShowMessage(string message, MessageType
type)
    {
        ScriptManager.RegisterStartupScript(this, GetType()
, Guid.NewGuid().ToString(),
            $"ShowMessage('{message}', '{type}');", true);
    }

    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            lblCP.Text = DateTime.Now.Year.ToString();
            txtPassword.Text = string.Empty;
```

```
            }

            if (Request.IsAuthenticated)
            {

            }
        }

        protected void btnLogin_Click(object sender, EventArgs e)
        {
            DataTable drs = pay.getdatabyparam3(2, txtUserName.Text, txtPassword.Text, 1);
            if (drs.Rows.Count.ToString() == "1")
            {
                FormsAuthentication.SetAuthCookie(drs.Rows[0]["parentid"].ToString(), true);
                Response.Cookies.Add(new HttpCookie("ParentsID", drs.Rows[0]["parentid"].ToString()));
                Response.Redirect("/secure/students.aspx", false);
            }
            else
            {
                ScriptManager.RegisterStartupScript(this, this.GetType(), "message", "alert('username dan password yang dimasukan salah');", true);
            }
        }
        protected void lnkclose_Click(object sender, EventArgs e)
        {
            lblalert.Visible = false;
        }
```

Kode 3.23: Kode *Frontend Login*

Kode 3.23 adalah kodingan *parent portal*. Untuk komponen yang digunakan kurang lebih sama dengan kodingan yang sebelumnya. Yang membedakan tentu ada di *backend*nya.

Enumerasi *MessageType* digunakan untuk menentukan empat jenis pesan utama yang dapat ditampilkan kepada pengguna, yaitu: *Success* yang menunjukkan operasi berhasil, *Error* yang menunjukkan kesalahan, Info yang memberikan informasi tambahan, dan Warning yang menyampaikan peringatan. Penggunaan enumerasi ini menggantikan string atau angka dengan nama-nama yang lebih deskriptif, sehingga membuat kode menjadi lebih terstruktur, mudah

dipahami, dan kecil kemungkinan terjadi kesalahan.

Metode *ShowMessage* dirancang untuk menampilkan pesan kepada pengguna menggunakan skrip JavaScript. Metode ini menerima dua parameter: pertama adalah message, yaitu konten pesan yang akan ditampilkan, dan kedua adalah *type*, yang menentukan jenis pesan berdasarkan enumerasi *MessageType*. Dengan pendekatan ini, sistem dapat menyampaikan pesan kepada pengguna sesuai konteks operasi yang dilakukan, baik itu berupa keberhasilan, kesalahan, informasi, maupun peringatan.

Untuk menjalankan skrip di sisi klien (*browser*), digunakan *ScriptManager*, khususnya melalui metode *ScriptManager.RegisterStartupScript*. Metode ini membutuhkan beberapa parameter, termasuk referensi ke halaman saat ini, *GetType()* untuk menentukan tipe objek halaman, dan *Guid.NewGuid().ToString()* untuk menghasilkan pengidentifikasi unik guna menghindari konflik dengan skrip lain. Skrip JavaScript seperti ShowMessage('message', 'type'); akan dijalankan di browser dengan parameter message dan type yang diteruskan ke fungsi ShowMessage. Pembuatan skrip dilakukan secara dinamis menggunakan interpolasi string di kode $"ShowMessage('message', 'type');", sehingga fungsi *ShowMessage* di sisi klien dapat menerima dan menampilkan pesan yang relevan sesuai jenisnya. Hal ini memberikan pengalaman interaktif yang lebih baik bagi pengguna.

Pada proses login, sistem diawali dengan pengambilan *username* dan *password* dari *input* pengguna melalui komponen *txtUserName.Text* dan *txtPassword.Text*. Data ini kemudian diverifikasi menggunakan fungsi *pay.getdatabyparam3*, yang memeriksa kesesuaian kombinasi *username* dan *password* dengan data di dalam *database*. Apabila fungsi ini menghasilkan tepat satu baris data *(Rows.Count == 1)*, *login* dianggap berhasil.

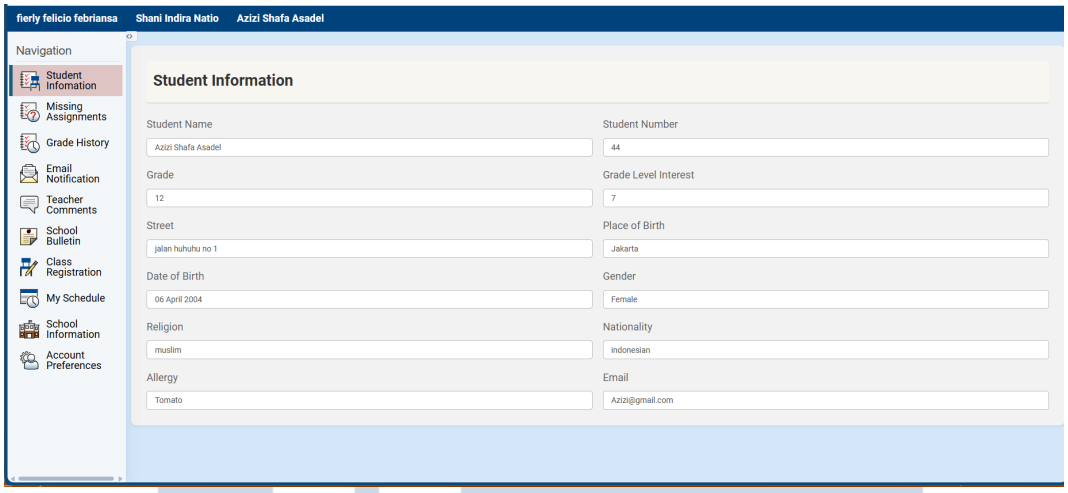
Setelah proses verifikasi berhasil, cookie autentikasi dibuat menggunakan *FormsAuthentication.SetAuthCooki*e berdasarkan nilai *parentid* dari hasil *query*. Selain itu, *cookie* tambahan dengan nama *ParentsID* dibuat untuk menyimpan informasi *login* pengguna. Pengguna yang berhasil *login* akan diarahkan ke halaman */secure/students.aspx*.

Sebaliknya, jika proses *login* gagal karena data yang dimasukkan tidak cocok dengan data di *database*, maka sistem akan menampilkan pesan kesalahan kepada pengguna dalam bentuk alert berbasis JavaScript. Pesan kesalahan ini dirancang agar mudah dipahami oleh pengguna dan memberi tahu bahwa *login* tidak berhasil.

Dalam proses ini, terdapat dua fungsi tambahan. Fungsi `lnkclose_Click` digunakan untuk menyembunyikan elemen *alert* dengan mengatur properti Visible menjadi *false*. Sementara itu, fungsi *ShowAlert* dirancang sebagai fungsi yang dapat digunakan kembali untuk menampilkan pesan alert dinamis menggunakan JavaScript. Dengan fungsi ini, sistem dapat menampilkan pesan sesuai kebutuhan secara fleksibel.

3.4.15 *Student Information*

Dalam SIS-PIK, metode yang mereka gunakan adalah *siblings*. Jadi setiap orang tua akan bisa melihat nilai, kehadiran, dan alamat. Seperti contoh pada gambar 3.17.



Gambar 3.17. Tampilan *Student Information*

.

Kode 3.24 adalah untuk *backend* dari *student information*.

```
        protected void Page_Load(object sender, EventArgs e)
        {

            if (!Page.IsPostBack)
            {
                string parentid = Request.Cookies["ParentsID"].
Value;
                dlstudent.DataSource = pay.getdatabyparam3(3,
parentid, string.Empty, 1);
                dlstudent.DataBind();
                foreach(DataListItem dli in dlstudent.Items)
                {
                    Label stdid = ((Label)dli.FindControl("
lblID"));
                    LinkButton lnkstd = ((LinkButton)dli.
FindControl("lnkstd"));
                    double number = 1;
                    if(number == 1)
                    {
                        lnkstd.CssClass = "actives";
                        lblstudentklik.Text = stdid.Text;
                    }
                    number += 1;
                }
                fillstudent();
            }
```

```
23          }
```

Kode 3.24: Kode *Student Infomation*

Pada kode ini, kondisi *Postback* diperiksa dengan menggunakan `if (!Page.IsPostBack)` untuk memastikan logika berikut hanya dijalankan saat halaman dimuat pertama kali, bukan saat terjadi *postback*, seperti setelah klik tombol atau pengiriman formulir. Setelah itu, nilai *ParentsID* diambil dari *cookie browser* dengan menggunakan `Request.Cookies["ParentsID"].Value`, yang digunakan untuk mengidentifikasi pengguna yang sedang *login*.

Selanjutnya, fungsi `pay.getdatabyparam3(3, parentid, string.Empty, 1)` digunakan untuk mengambil data siswa yang terkait dengan *parentid* dari *database*. Data yang diperoleh kemudian diset sebagai sumber data untuk sebuah *DataList* bernama *dlstudent* dengan mengatur *DataSource*, dan data tersebut diikat ke *DataList* melalui *dlstudent.DataBind()*, yang menampilkan daftar siswa di halaman.

Dalam proses ini, dilakukan iterasi terhadap setiap *item* dalam *DataList* dengan menggunakan *foreach (DataListItem dli in dlstudent.Items)*. Setiap item berisi *label lblID* yang menampilkan *ID* siswa dan sebuah *LinkButton lnkstd* untuk memungkinkan tindakan lebih lanjut terhadap siswa tertentu. Jika kondisi tertentu terpenuhi (misalnya, number == 1), maka siswa pertama akan diberi CSS class "*actives*" melalui `lnkstd.CssClass = "actives"`, menandakan bahwa siswa tersebut aktif. *ID* siswa yang aktif kemudian disimpan ke dalam `lblstudentklik.Text` untuk referensi selanjutnya.

Terakhir, fungsi *fillstudent()* dipanggil untuk memuat atau memperbarui *detail* siswa yang terkait berdasarkan siswa yang telah dipilih, sehingga informasi lebih lanjut dapat ditampilkan atau diperbarui sesuai kebutuhan.

```
1  protected void lblstdid_Click(object sender, EventArgs e)
2          {
3              LinkButton myButton = sender as LinkButton;
4
5              Label lbl = (Label)myButton.FindControl("lblID");
6              lblstudentklik.Text = lbl.Text;
7              fillstudent();
8              myButton.CssClass = "actives";
9              foreach (DataListItem dli in dlstudent.Items)
10             {
11                 Label stdid = ((Label)dli.FindControl("lblID"))
       ;
12                 LinkButton lnkstd = ((LinkButton)dli.
       FindControl("lnkstd"));
13                 if(lbl.Text != stdid.Text)
14                 {
15                     lnkstd.CssClass = "";
16                 }
17             }
```

```
18              }
```

Kode 3.25: Kode *Student Infomation*

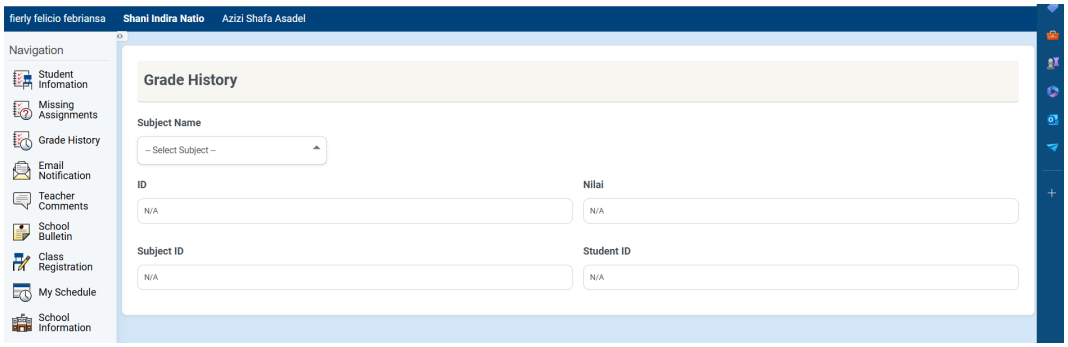
Pada kode 3.25, ketika sebuah *LinkButton* diklik, objek *sender*, yang merupakan elemen yang memicu *event*, di-*casting* menjadi tipe *LinkButton* dan disimpan dalam variabel *myButton*. Kemudian, *FindControl("lblID")* digunakan untuk menemukan kontrol *Label* dengan *ID lblID* yang ada di dalam *LinkButton* yang diklik. Label ini kemungkinan berisi ID siswa yang dipilih dan disimpan dalam variabel lbl. Nilai teks dari label tersebut kemudian ditampilkan pada kontrol lain, lblstudentklik.Text, untuk menampilkan *ID* siswa yang dipilih.

Setelah itu, metode *fillstudent()* dipanggil untuk memperbarui atau menampilkan informasi siswa berdasarkan *ID* yang dipilih. Untuk memberikan indikasi bahwa *LinkButton* yang diklik adalah yang aktif, CSS *class* dari *myButton* diubah menjadi "*actives*", memberikan gaya tampilan aktif pada tombol tersebut.

Kode kemudian melanjutkan untuk melakukan iterasi pada setiap *item* dalam *DataList dlstudent* dengan menggunakan `foreach (DataListItem dli in dlstudent.Items)`. Dalam proses ini, jika *ID* siswa yang ada di dalam *label stdid* tidak sama dengan *ID* yang dipilih, CSS *class* dari *LinkButton* lain dihapus dengan `lnkstd.CssClass = ""`, memastikan bahwa hanya tombol yang dipilih yang terlihat aktif.

3.4.16 *Student Grade*

Tampilan untuk *student grade* bisa dilihat pada gambar 3.18.



Gambar 3.18. Tampilan *Student grade*

Untuk kodingan *frontend* kurang lebih sama, hanya menyesuaikan sesuai yang dibutuhkan. Yang berbeda yaitu *backend*.

Sedangkan untuk *backend*nya bisa dilihat pada kode 3.26

```
1  protected void Page_Load(object sender, EventArgs e)
2          {
3              if (!Page.IsPostBack)
4              {
5                  string subject = Request.Params["ID"];
```

```
                if (!string.IsNullOrEmpty(subject))
                {
                    lblstudentklik.Text = subject;
                }
                else
                {
                    lblstudentklik.Text = ""; // Default jika tidak ada parameter
                }

                // Bind dropdown subject
                BindSubjectDropdown();

                // Bind student list
                string parentId = Request.Cookies["ParentsID"]?.Value;
                if (!string.IsNullOrEmpty(parentId))
                {
                    BindStudentList(parentId);
                }

                // Fill grade details
                fillgrade();
            }
        }
```

Kode 3.26: Kode *Student Grade*

Metode `Page_Load` dijalankan saat halaman pertama kali dimuat. Jika terdapat parameter *ID* dalam URL, nilai parameter tersebut akan ditampilkan pada *label lblstudentklik*. *Dropdown* subjek akan diisi dengan data yang relevan menggunakan metode *BindSubjectDropdown*(). Daftar siswa yang terkait dengan orangtua yang sedang *login* akan diisi jika *ParentsID* ditemukan dalam *cookie*. Selain itu, *detail* nilai atau *grade* siswa akan diisi melalui pemanggilan metode *fillgrade*().

```
  private void BindSubjectDropdown()
        {
            try
            {
                DDLSubject.DataSource = pay.getdatabyparam3(5, string.Empty, string.Empty, 1);
                DDLSubject.DataTextField = "name";
                DDLSubject.DataValueField = "subjectID";
                DDLSubject.DataBind();
                DDLSubject.Items.Insert(0, new ListItem("--
```

```
    Select Subject --", ""));
10              }
11              catch (Exception)
12              {
13                  // Log error
14                  // Example: LogError(ex);
15              }
16          }
```

Kode 3.27: Kode *Student Grade*

kode 3.27 ini, blok *try* digunakan untuk mengeksekusi serangkaian perintah yang dapat berisiko menyebabkan kesalahan (*exception*). Jika terjadi kesalahan selama eksekusi, eksekusi akan beralih ke bagian *catch* untuk menangani kesalahan tersebut.

Pertama, data untuk *dropdown DDLSubject* diambil melalui pemanggilan fungsi pay.getdatabyparam3(5, string.Empty, string.Empty, 1). Fungsi ini kemungkinan mengembalikan sebuah DataTable atau kumpulan data yang berisi daftar subjek. Data yang diperoleh kemudian diatur sebagai sumber data (*DataSource*) untuk *dropdown DDLSubject*.

Selanjutnya, kolom name dari sumber data dipilih sebagai teks yang akan ditampilkan di dropdown dengan properti *DataTextField*, sementara kolom *subjectID* digunakan sebagai nilai (*DataValueField*) untuk item yang dipilih. Dengan memanggil *DDLSubject.DataBind()*, data diikatkan ke kontrol *dropdown*, yang memungkinkan dropdown menampilkan nama subjek dengan nilai ID yang sesuai.

Kemudian, item pertama ditambahkan ke dropdown menggunakan `DDLSubject.Items.Insert(0, new ListItem("-- Select Subject --", ""))`. Ini akan menampilkan pilihan default bagi pengguna untuk memilih subjek dari daftar. Jika terjadi kesalahan selama eksekusi kode dalam blok *try*, blok *catch* akan menangkapnya. Namun, pada kode ini, tidak ada penanganan khusus yang dilakukan di dalam blok *catch* (seperti mencatat kesalahan atau menampilkan pesan kesalahan kepada pengguna).

```
1 private void BindStudentList(string parentId)
2         {
3             try
4             {
5                 dlstudent.DataSource = pay.getdatabyparam3(3,
    parentId, string.Empty, 1);
6                 dlstudent.DataBind();
7 
8                 foreach (DataListItem dli in dlstudent.Items)
9                 {
10                    Label stdid = dli.FindControl("lblID") as
    Label;
11                    LinkButton lnkstd = dli.FindControl("lnkstd
    ") as LinkButton;
```

```
12
13                          if (stdid != null && lnkstd != null)
14                          {
15                              if (stdid.Text == lblstudentklik.Text)
16                              {
17                                  lnkstd.CssClass = "actives";
18                              }
19                          }
20                      }
21                  }
22                  catch (Exception)
23                  {
24
25                  }
26              }
```

Kode 3.28: Kode *Student Grade*

Pada kode 3.28, blok *try* memastikan bahwa kode di dalamnya dapat berjalan dengan aman. Jika terjadi kesalahan, eksekusi akan dialihkan ke blok *catch*. Pertama, data siswa diambil menggunakan metode `pay.getdatabyparam3` dengan parameter tertentu, yaitu angka 3 untuk menunjukkan tipe data yang diambil, *parentId* sebagai *ID* orang tua untuk mendapatkan daftar siswa terkait, *string.Empty* sebagai parameter kosong, dan 1 untuk menandai status data yang aktif. Data yang diperoleh diatur sebagai sumber data (*DataSource*) untuk kontrol *DataList* bernama *dlstudent*, kemudian diikat menggunakan `dlstudent.DataBind()` agar data siswa dapat ditampilkan di halaman.

Selanjutnya, dilakukan iterasi pada setiap *item* dalam kontrol *DataList* menggunakan perulangan *foreach*. Pada setiap *item*, kontrol *lblID* yang berisi *ID* siswa ditemukan menggunakan FindControl dan diubah menjadi tipe *Label*, lalu kontrol *lnkstd* yang merupakan *LinkButton* ditemukan dan diubah menjadi tipe yang sesuai. Validasi dilakukan untuk memastikan bahwa kedua kontrol tersebut tidak bernilai *null*. Jika *ID* siswa pada *lblID* sesuai dengan teks pada `lblstudentklik.Text`, artinya siswa tersebut sedang dipilih atau diaktifkan. Dalam kasus ini, kelas CSS dari tombol lnkstd diubah menjadi "*actives*" untuk memberikan gaya visual yang berbeda pada siswa aktif. Apabila terjadi kesalahan selama proses ini, blok *catch* akan menangkapnya, meskipun tidak ada *log* atau penanganan khusus yang dilakukan dalam blok tersebut.

```
1  private void UpdateActiveLink(string activeStudentId)
2          {
3              foreach (DataListItem dli in dlstudent.Items)
4              {
5                  Label stdid = dli.FindControl("lblID") as Label
       ;
6                  LinkButton lnkstd = dli.FindControl("lnkstd")
       as LinkButton;
```

```
7
8                   if (stdid != null && lnkstd != null)
9                   {
10                      lnkstd.CssClass = (stdid.Text ==
    activeStudentId) ? "actives" : "";
11                  }
12              }
13          }
```

Kode 3.29: Kode *Student Grade*

Kode 3.29, perulangan *foreach (DataListItem dli in dlstudent.Items)* digunakan untuk memproses setiap item dalam *DataList* bernama *dlstudent*. Pada setiap iterasi, item tersebut diperlakukan sebagai objek *DataListItem*. Kemudian, kontrol *Label* dengan *ID lblID* dicari di dalam *item* menggunakan *dli.FindControl("lblID")* untuk mendapatkan ID siswa yang terkait. Kontrol lainnya, yaitu LinkButton dengan ID lnkstd, juga ditemukan menggunakan metode yang sama. Kontrol ini nantinya akan digunakan untuk memperbarui kelas CSS pada tampilan.

Selanjutnya, validasi dilakukan dengan memastikan bahwa *stdid (Label)* dan *lnkstd (LinkButton)* tidak bernilai *null*. Jika salah satu dari kontrol tersebut tidak ditemukan, kode berikutnya tidak akan dieksekusi. Setelah itu, teks dari kontrol *stdid* dibandingkan dengan variabel activeStudentId. Jika nilainya cocok, properti *CssClass* dari lnkstd diatur ke "*actives*", sehingga tombol menampilkan gaya visual yang menandakan status aktif. Sebaliknya, jika nilainya tidak cocok, properti *CssClass* diatur ke *string* kosong (""). Hal ini mengembalikan tampilan tombol ke gaya *default* tanpa penandaan khusus.

3.5 Kendala dan Solusi yang Ditemukan

Ada juga kendala yang dialami dalam pelaksanaan magang kali ini yaitu:

1. Kebingungan Memahami dan Menulis Kode
   - Terjadi karena keterbatasan pengetahuan dan pengalaman terhadap bahasa pemrograman yang baru dipelajari.
2. Kesulitan Memahami Logika dan Proses Pengkodean
   - Meskipun sudah mencari informasi melalui sumber daring atau bertanya kepada mentor, terkadang logika dan struktur kode masih sulit dipahami.
3. Keterbatasan Pemahaman terhadap Struktur dan Sintaks
   - Ketidaktahuan tentang cara kerja sintaks spesifik bahasa pemrograman dapat menyebabkan kebingungan saat menulis atau membaca kode.

Solusi yang bisa dilakukan adalah:

1. Mencari Bantuan dari Mentor atau Sumber Daring

- Mengajukan pertanyaan kepada mentor atau mencari referensi melalui sumber-sumber terpercaya seperti Google atau forum diskusi pemrograman.

2. Memperbanyak Latihan dan Pengalaman
    - Dengan terus mencoba menulis kode dan mempraktikkan konsep pemrograman, pemahaman terhadap logika dan proses akan meningkat seiring waktu.
3. Mempelajari Struktur dan Sintaks Secara Mendalam
    - Menggunakan dokumentasi resmi atau tutorial berkualitas untuk memperdalam pemahaman mengenai bahasa pemrograman yang digunakan.