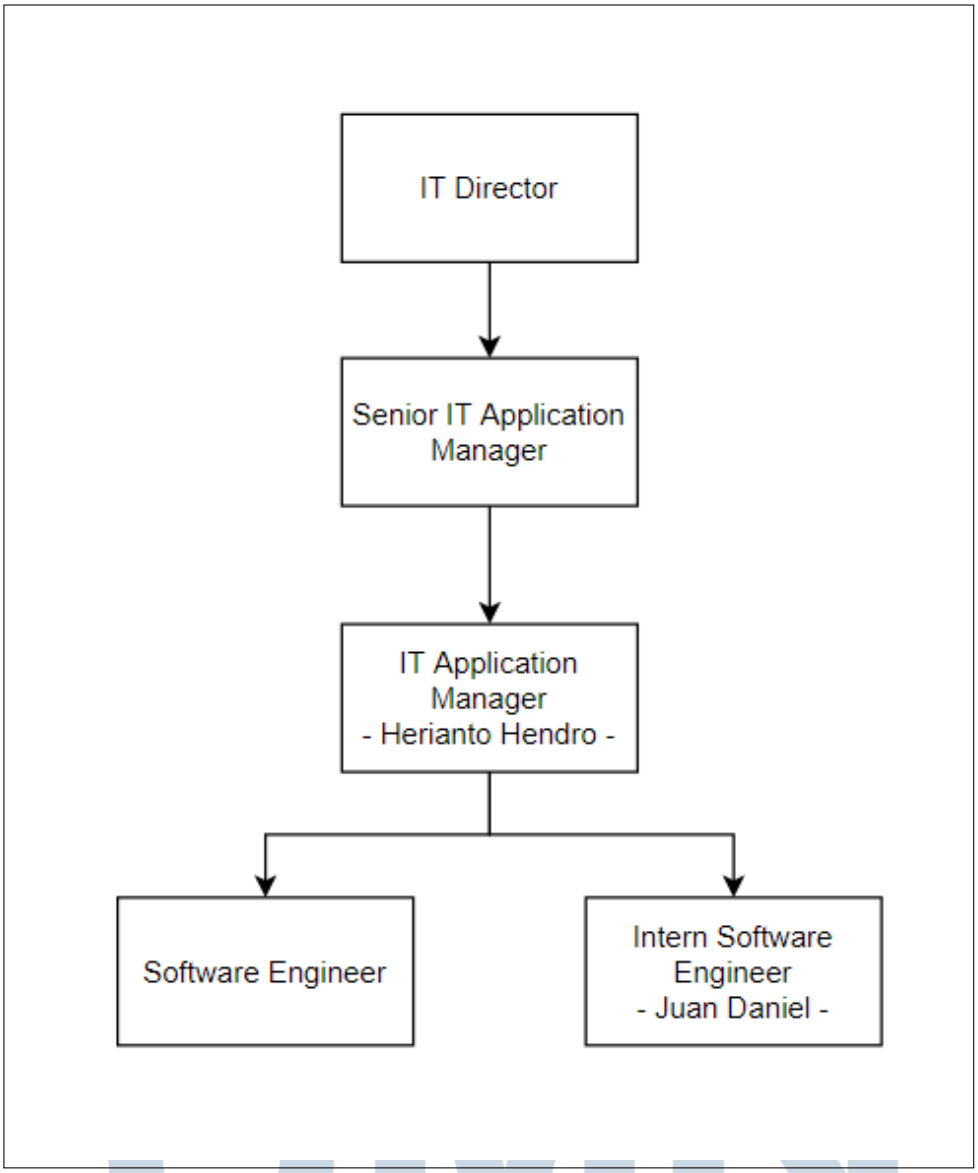
BAB 3
PELAKSANAAN KERJA MAGANG

Pelaksanaan kerja magang ini dilaksanakan selama 6 bulan atau satu semester sesuai dengan permendikbud nomor 30 tahun 2020 yang memberikan hak istimewa kepada mahasiswa untuk belajar diluar kampus. Selama diluar lingkungan kampus, mahasiswa diharapkan dapat mengembangkan wawasan serta kompetensinya dalam mengejar impian dengan bebas, tidak terbatas hanya di lingkungan kampus namun juga tempat pengabdian lain seperti tempat kerja, riset, desa dan lainnya.

3.1 Kedudukan dan Organisasi

Kedudukan dalam divisi IT selama menjalani kegiatan magang di Kawan Lama Group berada setara dengan *software engineer* yaitu sebagai intern software engineer yang terdapat di bawah IT Application Engineer. Pembimbing selama kegiatan magang ini adalah Bapak Herianto Hendro yang bekerja sebagai IT Application Engineer dan juga Integration Lead. Untuk lebih jelas mengenai kedudukan selama magang dalam struktur organisasi dapat dilihat pada Gambar 3.1. Setiap pekerjaan atau *application request* (AR) yang diberikan diharapkan untuk bertanggung jawab kepada IT Application Manager untuk menyelesaikannya dalam waktu yang telah ditentukan atau meminta perpanjangan waktu saat dibutuhkan.

Pekerjaan dilakukan dalam tim yang terdiri dari dua orang software developer yang saling berkoordinasi dalam mengerjakan AR yang diberikan yaitu bagian API dan front-end sekaligus back-end . Setiap AR memiliki *system analyst* (SA) yang mengoordinasi dan memonitor pengerjaan AR agar sesuai dengan permintaan serta menetapkan batas waktu dari pengerjaan AR tersebut dan diserahkan pada IT Application Manager. Koordinasi dilakukan dengan mengikuti rapat singkat yang dilaksanakan beberapa kali selama proses berjalannya suatu AR. AR yang dikerjakan dimasukan ke dalam penyimpanan bersama agar dapat dilihat pihak yang bersangkutan dan dilakukan pengetesan, jika sudah selesai maka akan dilanjutkan ke tahap produksi.



Gambar 3.1. Kedudukan dalam struktur organisasi departemen IT di Kawan Lama Group

3.2 Tugas yang Dilakukan

Selama magang di Kawan Lama Group menjadi Intern Software Engineer akan diberikan pekerjaan untuk membuat modul dalam situs web internal perusahaan. Dalam situs tersebut terdapat 4 modul Checklist Dashboard yaitu Issue, Site, Trend dan User. Salah satu AR yang diberikan untuk tugas magang merupakan rancang bangun modul User. Checklist Dashboard User merupakan modul dalam situs web yang digunakan sebagai alat monitor dari perkerjaan yang dilakukan oleh

PIC atau toko cabang. Modul akan menampilkan data dari *Application Programming Interface* (API) yang dimasukkan ke dalam tabel untuk memberikan informasi mengenai jumlah keseluruhan jadwal, jumlah jadwal yang sudah terpenuhi, jumlah jadwal yang gagal dipenuhi, jumlah jadwal yang *pending* dan persentase jadwal yang terpenuhi dari jumlah keseluruhan jadwal yang diberikan pada PIC atau toko cabang. Data yang dapat ditampilkan merupakan data yang ada dalam periode tertentu dengan maksimal rentang waktu 31 hari namun secara bawaan, data yang ditampilkan merupakan data dalam periode 1 minggu sebelumnya dari hari modul tersebut dibuka dan dapat disaring sesuai dengan kebutuhan pengguna. Selain itu pengguna juga dapat mengunduh data pada tabel dalam bentuk Excel dengan menekan tombol *download*.

Modul Checklist Dashboard User juga dapat memberikan data yang lebih detail dengan membawa pengguna ke halaman Checklist Dashboard User Detail yang menampilkan data yang hanya dimiliki oleh PIC atau toko cabang yang dipilih dari tabel. Pada halaman tersebut akan memiliki dua tampilan yaitu dalam berupa lis dan kalender, kedua tampilan tersebut menampilkan data yang dimiliki PIC atau toko cabang di setiap hari selama sebulan (tanggal satu hingga akhir bulan). Jika ada jadwal yang telah dikerjakan maka akan muncul fitur untuk mengunduh laporan yang diunggah oleh PIC atau toko cabang dan juga dapat mengunduh data keseluruhan dalam bentuk Excel.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang untuk setiap minggunya diuraikan seperti pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke -	Pekerjaan yang dilakukan
1	Persiapan perangkat keras dan lunak
2	Belajar dasar C#
3	Belajar dasar C# dan latihan
4	Belajar tingkat lanjut C# dan latihan
5	Membuat program sederhana dengan C#
6	Membuat program dengan C# dan menerapkan class, object, list, regex dan enumerable
7	Belajar mengenai API menggunakan .NET framework
8	Latihan membuat dasar API dan menerapkan MVC dan Session
9	Latihan membuat API untuk CRUD
10	Latihan membuat ”ToDo” API dengan basis data dan mempelajari project Checklist Dashboard User
11	Membuat child row dan action button
12	Menambah interaksi dan mengubah pengambilan data dari database menjadi menggunakan API
13	Menambahkan fitur pencarian dan kontrol pada tabel serta fitur download untuk seluruh data tabel
14	Membuat modul Checklist Dashboard User Detail dan menghubungkannya dengan modul yang sebelumya dibuat
15	Menampilkan data detail user dalam bentuk list dan membuat fitur download
16	Membuat button toggle untuk berpindah tampilan antara list dan kalender serta menampilkan data dalam bentuk kalender
17	Membuat fitur pergantian bulan menggunakan action button, fitur detail per hari dalam kalender serta download pada Checklist Dashboard User Detail dan fitur filter pada Checklist Dashboard User
18	Merapikan tampilan pada Checklist Dashboard User Detail dan *deployment* untuk pengetesan
19	*Deployment* untuk produksi, *review* modul dan rapat kerja untuk modul Checklist Dashboard selanjutnya
20	Membuat UI Checklist Dashboard Site dan Issue
21	Membuat back-end Checklist Dashboard Site
22	Membuat back-end Checklist Dashboard Issue

Pada minggu ke-1 hingga ke-10 diluangkan untuk menyiapkan berbagai hal yang diperlukan untuk dapat bekerja di Kawan Lama Group. Pihak perusahaan memberikan laptop untuk bekerja sehingga hanya perlu melakukan instalasi perangkat lunak di dalam laptop tersebut. Selama waktu tersebut juga digunakan untuk mempelajari banyak hal mulai dari dasar hingga melakukan beberapa latihan mandiri untuk menyiapkan diri dan terbiasa dengan perangkat lunak yang digunakan untuk bekerja. Beberapa perangkat lunak tersebut meliputi Postman, Visual Studio dan berbagai situs web internal perusahaan untuk melakukan absensi serta pemonitoran jalannya pekerjaan.

Pada minggu ke-11 yaitu tepat 1 September, diberikan AR untuk rancang bangun Checklist Dashboard User. Pengerjaan AR dilakukan dengan berkoordinasi bersama beberapa rekan kerja untuk mendapatkan berkas situs web dari Git internal perusahaan agar dapat mulai bekerja. Selama proses AR berjalan akan dibimbing sehingga dapat menemukan inspirasi mengenai cara-cara yang digunakan untuk membuat modul tersebut. Pengerjaan modul User menggunakan *framework* ASP.NET dari Microsoft yang berbasis bahasa pemrograman C#. Pada minggu tersebut data model untuk modul tersebut belum tersedia sehingga hanya melakukan pembuatan model *user interface* (UI) modul tersebut.

Pada minggu ke-12 terdapat perubahan dimana pengambilan data yang sebelumnya dilakukan menggunakan *Stored Procedure* menjadi menggunakan API yang telah dibuat. Perubahan tersebut dilakukan agar data yang sama dapat diperoleh walaupun menggunakan perangkat keras yang berbeda. Terdapat beberapa penyesuaian yang harus dilakukan seperti cara mengirim *request* ke API menggunakan *custom header* dan menerima *response* dari API serta memasukkan data ke dalam tabel pada modul menggunakan data dalam bentuk JSON yang dikembalikan API.

Pada minggu ke-13, tabel yang ditampilkan pada halaman Checklist Dashboard User diminta untuk memiliki fitur pencarian untuk data yang ada di dalamnya sehingga harus melakukan penyesuaian terhadap kompatibilitas dari Bootstrap dan fungsi *library* .NET yang digunakan pada tabel tersebut. Terdapat cara lain yang dapat digunakan yaitu membuat secara mandiri menggunakan .NET namun terlalu rumit dan harus disatukan dengan CSS untuk terlihat lebih baik sehingga tidak digunakan.

Pada minggu ke-14 diminta untuk membuat halaman web baru yang digunakan untuk menampilkan data detail dari PIC atau toko cabang yang dipilih pengguna pada halaman Checklist Dashboard User, halaman tersebut diberi judul Check-

list Dashboard User Detail. Agar PIC atau toko cabang yang dipilih pengguna dapat dialihkan ke halaman lain terdapat metode ”Session” dari .NET untuk menyimpan variabel yang diambil menggunakan JavaScript saat data nama dalam tabel ditekan.

Pada minggu ke-15 pengerjaan dialihkan ke halaman web yang baru dibuat yaitu untuk menampilkan data selama satu bulan yang dimiliki oleh PIC atau toko cabang yang telah dipilih. Untuk minggu ini, pengerjaan difokuskan pada penampilan data dalam bentuk lis dan penambahan fitur untuk mengunduh keseluruhan data yang ada dalam lis tersebut.

Pada minggu ke-16, data detail yang dimiliki oleh PIC atau toko cabang diminta agar dapat ditampilkan dalam bentuk kalender. Dokumentasi yang terbatas menyebabkan pengerjaan fitur ini menjadi sulit dan cukup memakan waktu karena harus membaca dan melakukan uji coba berdasarkan buku manual yang diberikan oleh Microsoft. Interaksi pergantian dari tampilan lis dan kalender dibuat menggunakan cara *show* dan *hide* dari panel yang memuat lis dan kalender tersebut berdasarkan *post back* yang dipicu oleh tombol dan mengembalikan teks untuk pemenuhan syarat interaksi penggantian tampilan.

Pada minggu ke-17 membuat fitur pergantian bulan dengan tombol yang melakukan *post back* berisikan data bulan ini untuk kemudian berjalan maju ke bulan berikutnya atau mundur ke bulan sebelumnya. Pada minggu ini juga melakukan penambahan penampilan data detail per hari dalam bentuk lis jika tanggal pada kalender ditekan oleh pengguna dan menambahkan fitur untuk memfilter data yang ditampilkan pada halaman Checklist Dashboard User.

Untuk minggu ke-18 dan ke-19 hanya merapikan sedikit UI untuk halaman Checklist Dashboard User Detail dan kemudian melakukan *deployment* untuk pengetesan dan setelah itu ke tahap produksi.

AR untuk halaman Checklist Dashboard User dan halaman Checklist Dashboard User Detail telah selesai dan mempersiapkan untuk modul Checklist Dashboard berikutnya pada minggu ke-20 dan mengerjakan AR tersebut di minggu-minggu berikutnya sesuai waktu yang ditentukan.

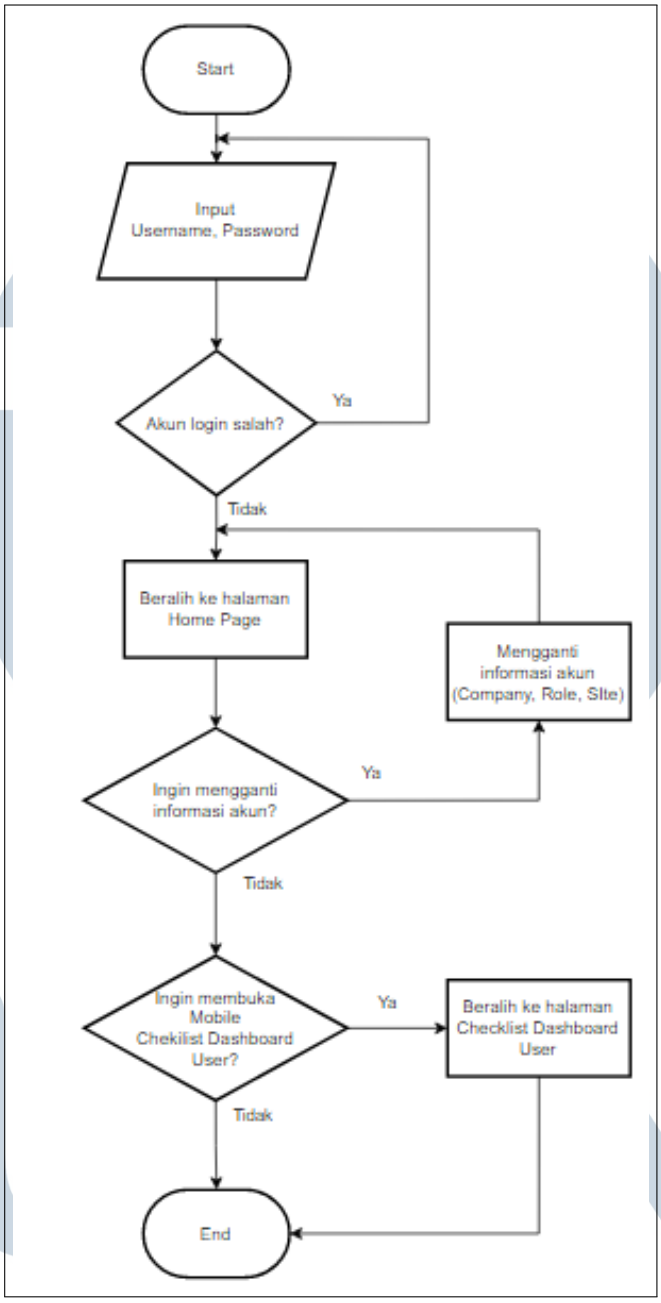
3.3.1 Perancangan

Setiap data disimpan dalam basis data dan dapat diambil menggunakan API yang telah dibuat khusus untuk internal perusahaan. Halaman situs web akan mengambil data dan menampilkannya serta membuat fitur untuk mengunduh data tersebut, baik data secara menyeluruh atau terperinci yang digunakan sebagai alat untuk memonitor pekerjaan. Checklist Dashboard User dan User Detail menggunakan *endpoint* API yang berbeda karena memiliki *request header* dan *body* yang berbeda untuk mengambil data.

A. Flowchart

A.1 Flowchart Halaman Utama

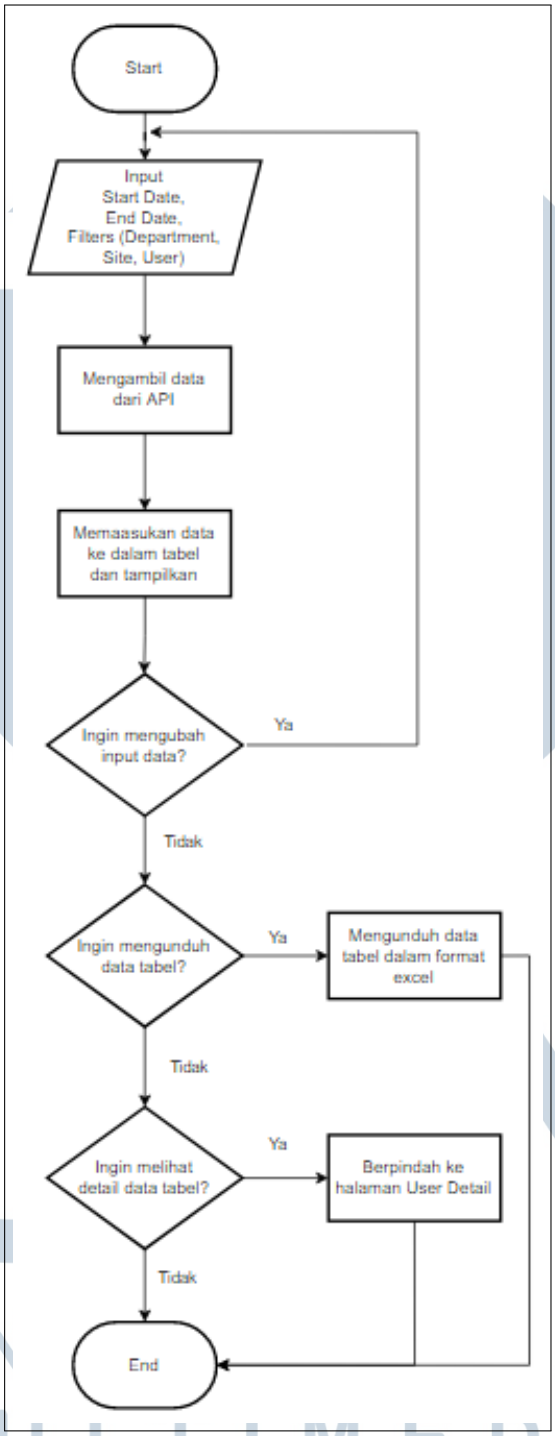
Sebelum memasuki halaman Checklist Dashboard User harus melewati sistem autentikasi dengan *login* menggunakan *username* dan *password* yang telah terdaftar. Jika lolos autentikasi maka akan dialihkan ke halaman utama dari situs web. Alur data dan interaksi pada halaman utama ditunjukkan pada Gambar 3.2.



Gambar 3.2. Flowchart Halaman Utama

A.2 Flowchart Halaman Checklist Dashboard User

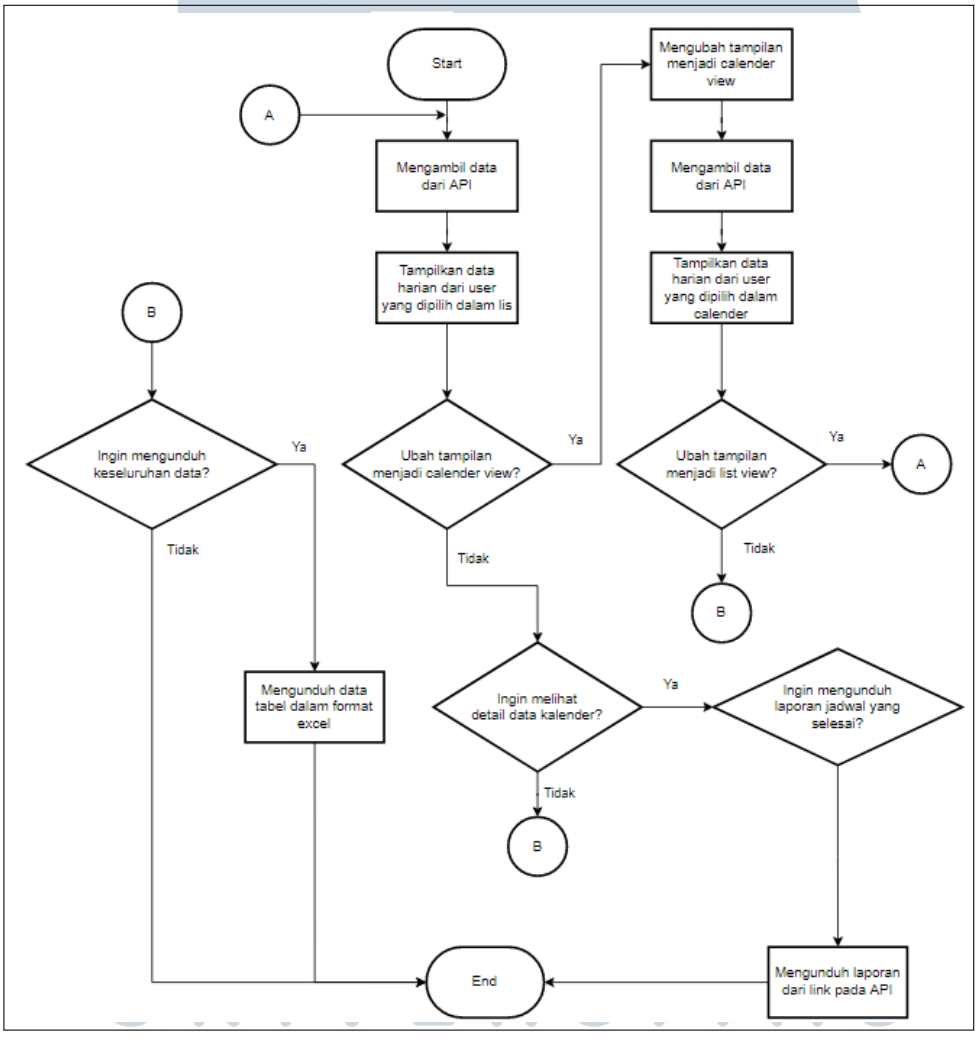
Alur data dan interaksi yang dapat dilakukan pengguna saat di halaman Checklist Dashboard User dijelaskan pada Gambar 3.3.



Gambar 3.3. Flowchart Checklist Dashboard User

A.3 Flowchart Halaman Checklist Dashboard User Detail

Pada halaman Checklist Dashboard User Detail terdapat berbagai hal yang dapat dilakukan oleh pengguna yang dijelaskan melalui Gambar 3.4 beserta dengan proses alur data pada halaman tersebut.

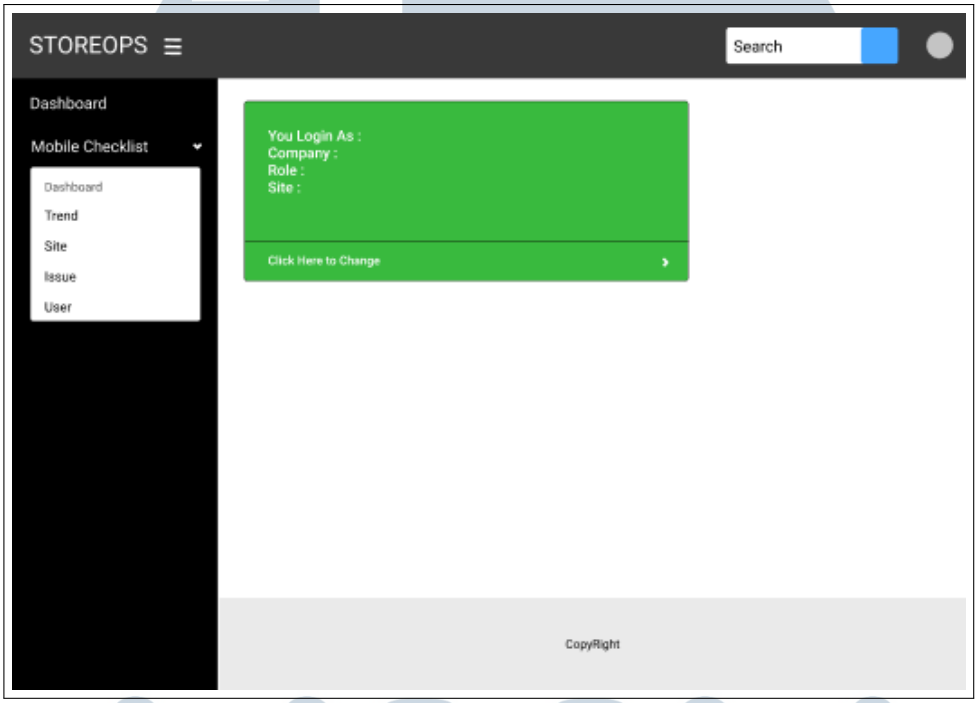


Gambar 3.4. Flowchart Checklist Dashboard User Detail

B. Mockup

B.1 Mockup Halaman Utama

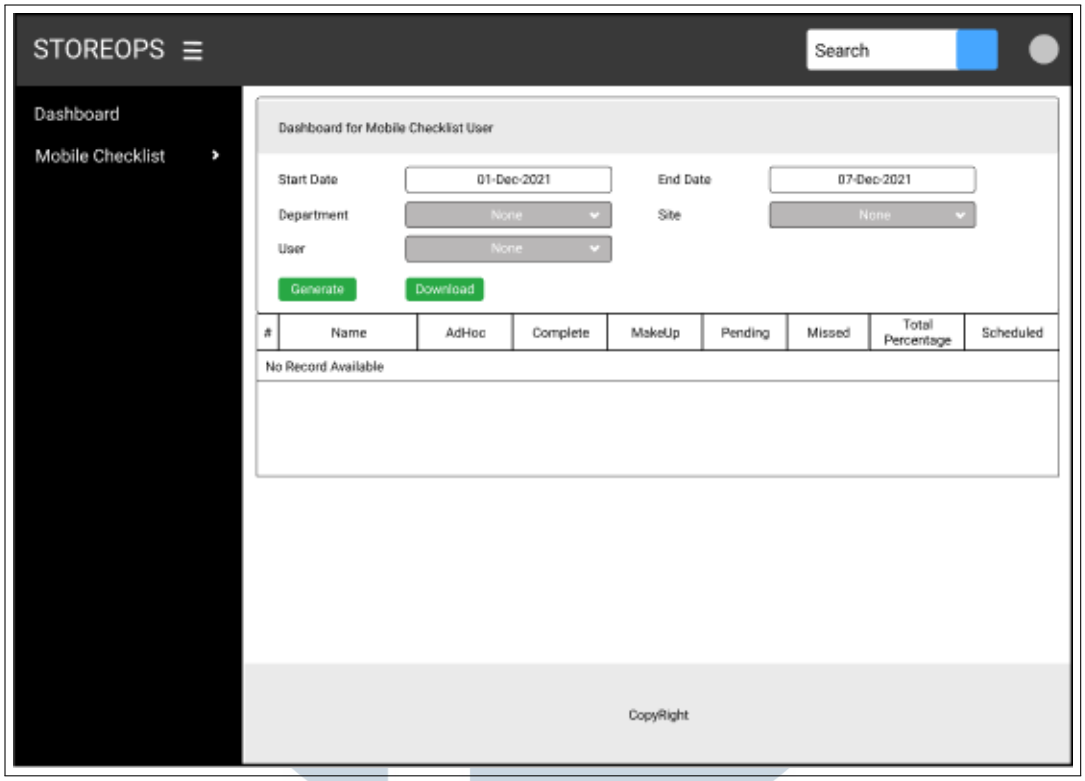
Gambar 3.5 merupakan rancangan dari tampilan yang ada pada halaman utama.



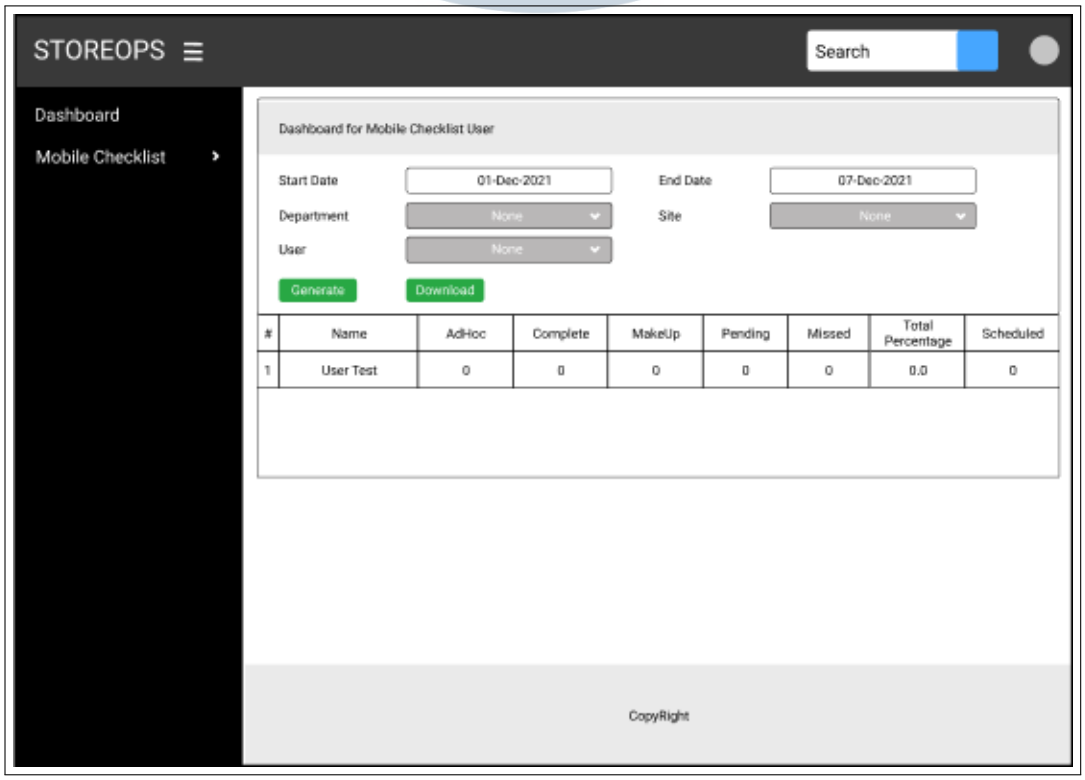
Gambar 3.5. MockUp Halaman Utama

B.2 Mockup Halaman Checklist Dashboard User

Perancangan untuk tampilan dari data untuk filter dan data pada tabel ditunjukkan pada Gambar 3.6 saat tidak ada data dan Gambar 3.7 saat ada data.



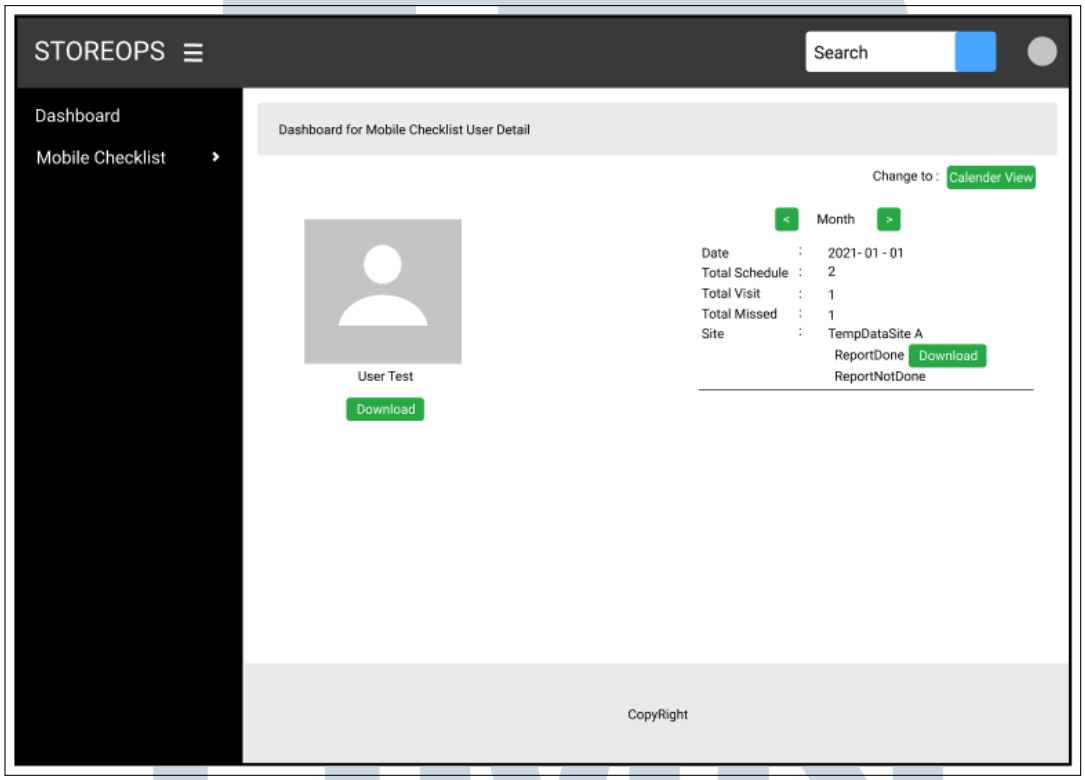
Gambar 3.6. MockUp Halaman Checklist Dashboard User Tanpa Data



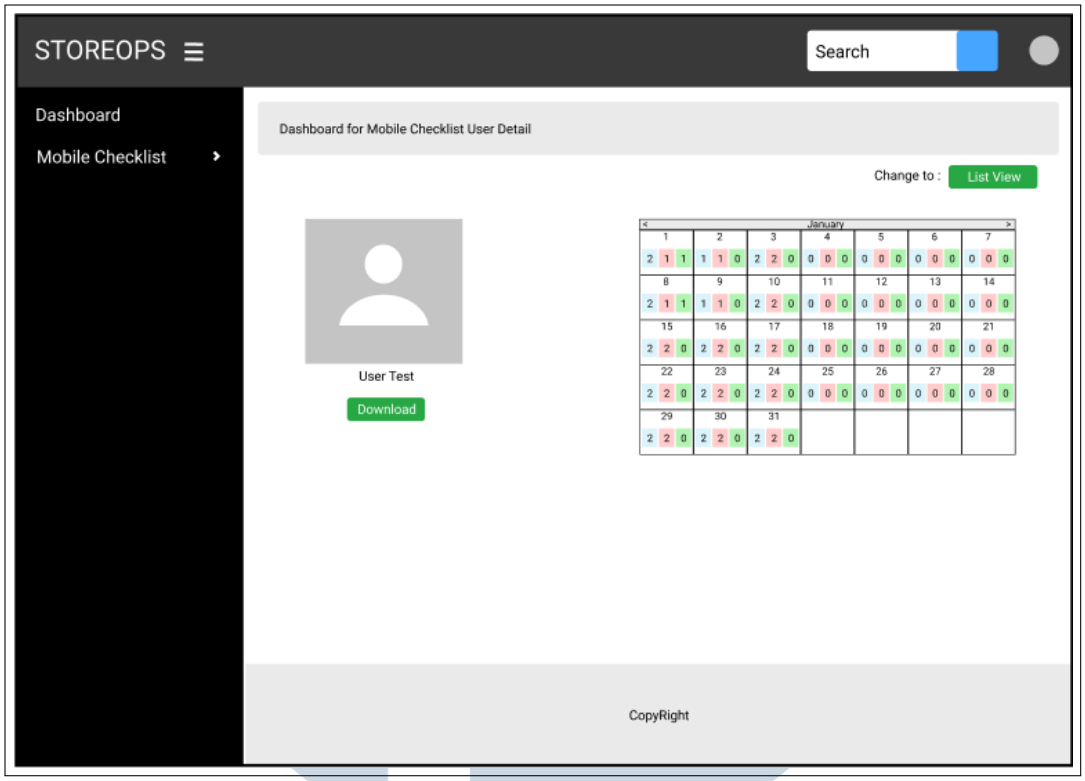
Gambar 3.7. MockUp Halaman Checklist Dashboard User Dengan Data

B.3 Mockup Halaman Checklist Dashboard User Detail

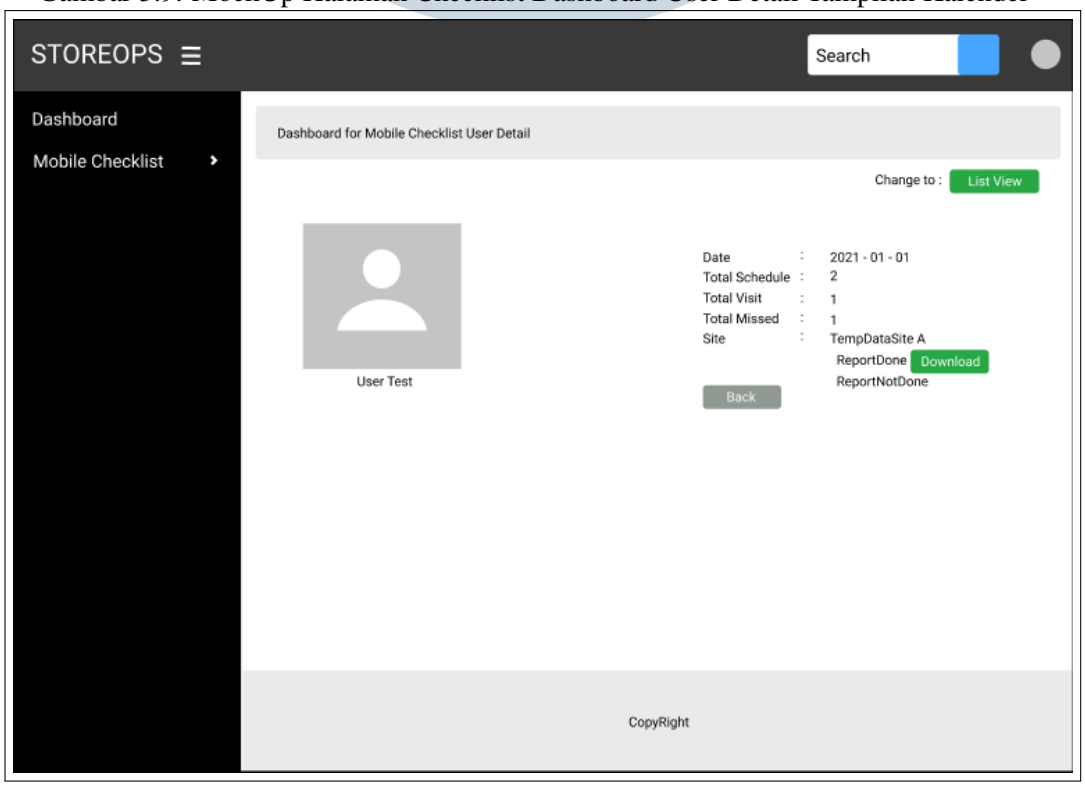
Rancangan tampilan akan dibuat berdasarkan Gambar 3.8 untuk lis dan Gambar 3.9 untuk kalender. Penampilan data harian pada kalender akan dibuat seperti pada Gambar 3.10.



Gambar 3.8. MockUp Halaman Checklist Dashboard User Detail Tampilan Lis



Gambar 3.9. MockUp Halaman Checklist Dashboard User Detail Tampilan Kalender



Gambar 3.10. MockUp Halaman Checklist Dashboard User Detail Tampilan Data Harian Kalender

3.3.2 Implementasi

A. Implementasi Halaman Checklist Dashboard User

Data yang akan digunakan untuk halaman ini tersedia pada API dengan *endpoint* "RequestReportDashboard_Header" untuk data tabel dan "RequestReportDashboardAllDeptSiteUser" untuk data *dropdown* filter. Berikut adalah proses pengambilan data dengan *request* API serta JSON yang dikembalikan dan proses menampilkan data ke halaman situs web.

```
private string GetAPIUrl(string endpoint)
{
    string url = "";
    sql.Length = 0;
    sql.Append("select PARAMETER_VALUE from MASTER_PARAMETER ");
    sql.Append("where PARAMETER_NAME = 'API_BASE_URL' ");
    sql.Append("and APLICATION_NAME = 'MobileCheklist' and BU_CODE ='" + Session["userCompany"].ToString() +"'");
    DataTable dt_url = cKres.OpenDataTable(Session["strMainServer"].ToString(), Session["strMainDB"].ToString(),
                                Session["strMainUser"].ToString(), Session["strMainPassword"].ToString(), sql.ToString());
    url = dt_url.Rows[0].ItemArray[0].ToString();
    url = url + endpoint;
    return url;
}
```

Gambar 3.11. Metode untuk mendapatkan URL API

Request dilakukan dengan menggunakan properti dari *library* System.Net.WebRequest yang diisi masukan sebagai parameter dan juga *header* yang digunakan untuk *query* oleh API ke basis data. *Request* dilakukan dalam bentuk JSON sehingga diperlukan *library* System.Web.Script.Serialization untuk "*Serialize*" agar data dapat dikirim oleh situs web.

```
string _URL = GetAPIUrl("/RequestReportDashboard_Header");
var webRequest  = System.Net.WebRequest.Create(_URL);
var inputJson = new System.Web.Script.Serialization.JavaScriptSerializer();
string dept = ddlDept.SelectedValue;
string site = ddlSite.SelectedValue;
string user = ddlUser.SelectedValue;

ReportRequest request = new ReportRequest();
request.DATA_RANGE_START = DateTime.Parse(txtStartDate.Text).ToString("yyyy-MM-dd");
request.DATA_RANGE_END = DateTime.Parse(txtEndDate.Text).ToString("yyyy-MM-dd");
request.COMPANY_CODE = site;
request.MASTER_DEPARTEMEN_ID = dept;
request.USER_AUDITOR = user;
string inputJSONConversion = inputJson.Serialize(request);

webRequest.Method = "POST";
webRequest.Timeout = 60000;
webRequest.ContentType = "application/json";
webRequest.Headers.Add("BU", Session["userCompany"].ToString());
webRequest.Headers.Add("Intranet", "true");
```

Gambar 3.12. Metode *request* ke API untuk data tabel

```
public class ReportRequest
{
    1 reference
    public string DATA_RANGE_START { get; set; }
    1 reference
    public string DATA_RANGE_END { get; set; }
    1 reference
    public string MASTER_DEPARTEMEN_ID { get; set; }
    1 reference
    public string COMPANY_CODE { get; set; }
    1 reference
    public string USER_AUDITOR { get; set; }
}
```

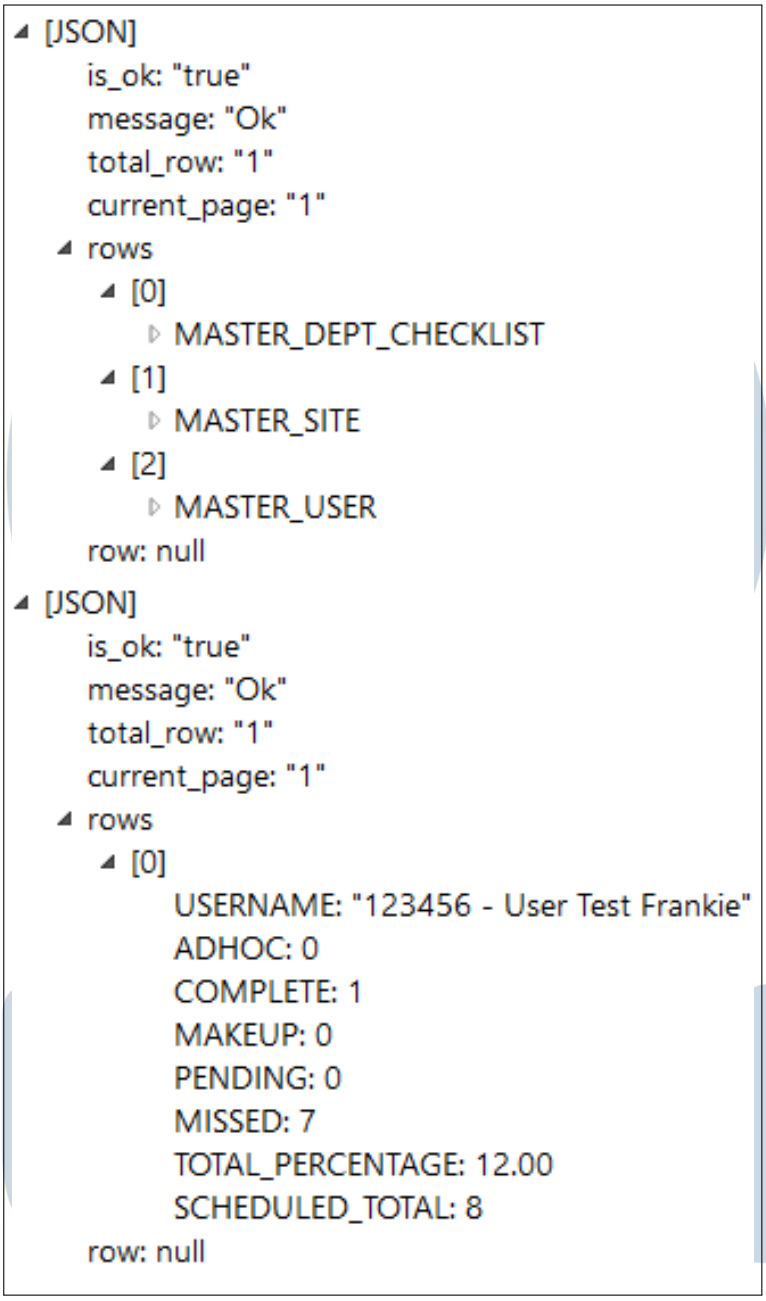
Gambar 3.13. *Class* untuk *request* ke API untuk data tabel

Class dibuat untuk menampung parameter *request* yang akan dikirim ke API dalam bentuk JSON untuk menjadi *query* dalam pengambilan data dari basis data. Jika tidak dibutuhkan *query* parameter lain maka hanya perlu mengirimkan *header* dari JSON sebagai *request* seperti yang digunakan untuk data filter pada Gambar 3.14.

```
string _URL = GetAPIUrl("/RequestReportDashboardAllDeptSiteUser");
var webRequest = System.Net.WebRequest.Create(_URL);

webRequest.Method = "POST";
webRequest.Timeout = 60000;
webRequest.ContentType = "application/json";
webRequest.ContentLength = 0;
webRequest.Headers.Add("BU", Session["userCompany"].ToString());
webRequest.Headers.Add("Intranet", "true");
```

Gambar 3.14. Metode *request* ke API untuk data filter

Response yang dikembalikan dalam bentuk JSON memerlukan *library* Newtonsoft.Json.JsonConvert untuk "*Deserialize*" agar data dapat digunakan oleh situs web. Data yang telah diterima akan dimasukan ke dalam *class* yang disesuaikan dengan judul JSON yang diterima agar dapat digunakan dalam situs web.



Gambar 3.15. *Response* dari API untuk data filter (atas) dan data tabel (bawah)

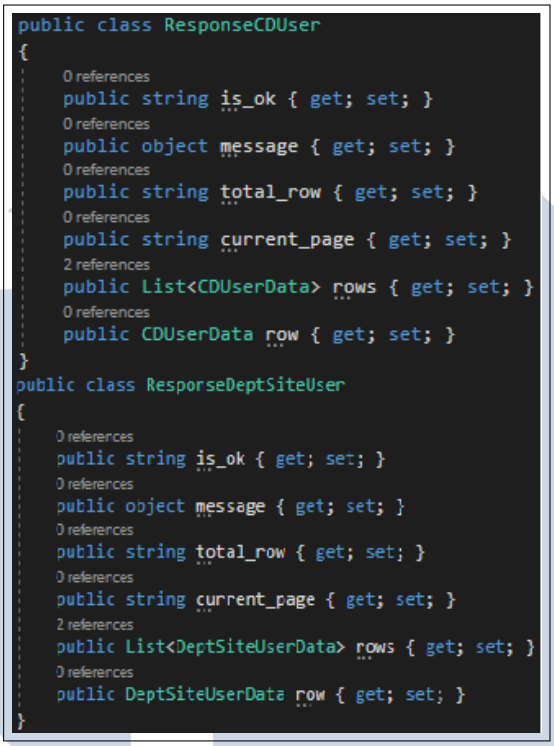
```
try
{
    using (Stream st = webRequest.GetResponse().GetResponseStream())
    {
        using (StreamReader sr = new StreamReader(st))
        {
            var jsonResponse = sr.ReadToEnd();
            response = JsonConvert.DeserializeObject<ResponseDeptSiteUser>(jsonResponse);
        }
    }
    return response;
}
catch (Exception ex)
{
    ScriptManager.RegisterStartupScript(this, GetType(), "ServerControlScript", "alert("+ ex +");", true);
    return null;
}
```

Gambar 3.16. Metode yang menerima *response* dari API untuk data filter

```
try
{
    using (Stream s = webRequest.GetRequestStream())
    {
        using (StreamWriter sw = new StreamWriter(s))
            sw.Write(inputJSONConversion);
    }

    using (Stream st = webRequest.GetResponse().GetResponseStream())
    {
        using (StreamReader sr = new StreamReader(st))
        {
            var jsonResponse = sr.ReadToEnd();
            response = JsonConvert.DeserializeObject<ResponseCDUser>(jsonResponse);
        }
    }
    return response;
}
catch(Exception ex)
{
    ScriptManager.RegisterStartupScript(this, GetType(), "ServerControlScript", "alert(" + ex + ");", true);
    return null;
}
```

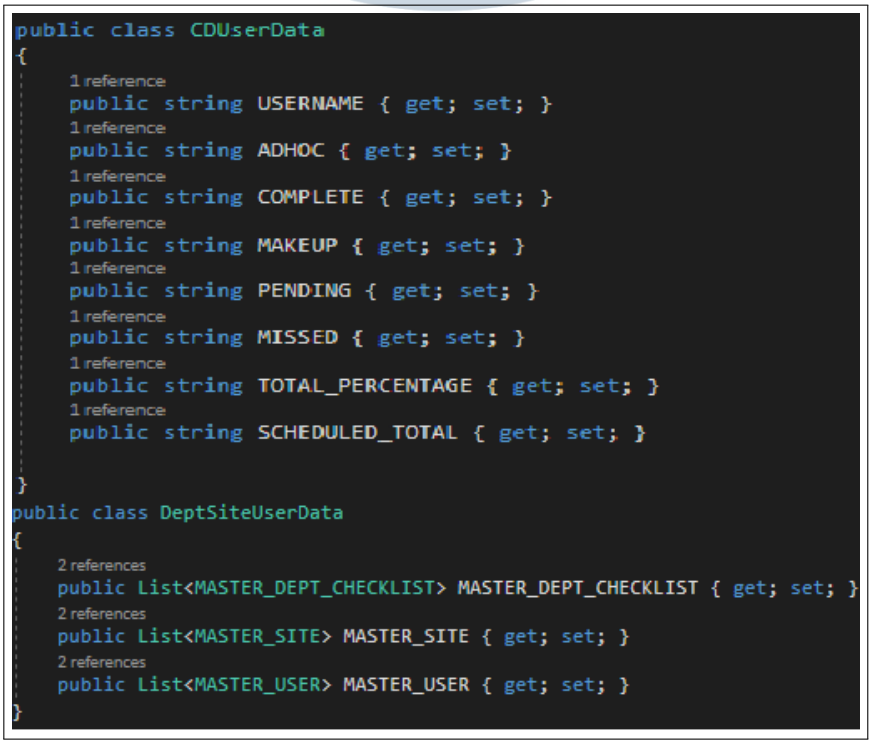
Gambar 3.17. Metode yang menerima *response* dari API untuk data tabel

Setiap data JSON yang telah di *Deserialize* akan ditampung ke dalam sebuah *class* sehingga kemudian dapat digunakan dengan cara memanggil objek dan variabel dari *class* tersebut untuk mendapatkan nilai yang ditampung oleh variabelnya.



```
public class ResponseCDUser
{
    0 references
    public string is_ok { get; set; }
    0 references
    public object message { get; set; }
    0 references
    public string total_row { get; set; }
    0 references
    public string current_page { get; set; }
    2 references
    public List<CDUserData> rows { get; set; }
    0 references
    public CDUserData row { get; set; }
}
public class ResporseDeptSiteUser
{
    0 references
    public string is_ok { get; set; }
    0 references
    public object message { get; set; }
    0 references
    public string total_row { get; set; }
    0 references
    public string current_page { get; set; }
    2 references
    public List<DeptSiteUserData> rows { get; set; }
    0 references
    public DeptSiteUserData row { get; set; }
}
```

Gambar 3.18. *Class* yang menampung header data untuk tabel (atas) dan untuk filter (bawah)



```
public class CDUserData
{
    1 reference
    public string USERNAME { get; set; }
    1 reference
    public string ADHOC { get; set; }
    1 reference
    public string COMPLETE { get; set; }
    1 reference
    public string MAKEUP { get; set; }
    1 reference
    public string PENDING { get; set; }
    1 reference
    public string MISSED { get; set; }
    1 reference
    public string TOTAL_PERCENTAGE { get; set; }
    1 reference
    public string SCHEDULED_TOTAL { get; set; }

}
public class DeptSiteUserData
{
    2 references
    public List<MASTER_DEPT_CHECKLIST> MASTER_DEPT_CHECKLIST { get; set; }
    2 references
    public List<MASTER_SITE> MASTER_SITE { get; set; }
    2 references
    public List<MASTER_USER> MASTER_USER { get; set; }
}
```

Gambar 3.19. *Class* yang menampung data untuk tabel (atas) dan untuk filter (bawah)

Setiap nilai dari data yang telah ditampung dalam *class* akan dimasukan ke dalam tabel yang disediakan oleh *library* System.Data agar kemudian dapat ditampilkan dengan mengikatnya pada tag HTML pada ASP GridView. Contoh penggunaannya sebagai berikut pada Gambar 3.20 dan Gambar 3.21.

```
DataTable dataTabel = new DataTable();
DataColumn dc = new DataColumn();
dc.ColumnName = "namaKolom";
dc.DataType = typeof(string);
dataTabel.Columns.Add(dc);

Response response = LoadWithAPI();

if (response.rows.Count > 0)
{
    foreach (CDUserData result in response.rows)
    {
        DataRow dr = dataTabel.NewRow();
        dr["namaKolom"] = result.data;
        dataTabel.Rows.Add(dr);
    }
}
```

Gambar 3.20. Contoh pengisian data ke dalam tabel

```
<asp:GridView runat="server" ID="GridView1" UseAccessibleHeader="true"
    AutoGenerateColumns="false" ShowHeaderWhenEmpty="true">
    <Columns>
        <asp:BoundField DataField="namaKolom" HeaderText="Nama Kolom" />
    </Columns>
    <EmptyDataTemplate>
        No Record Available
    </EmptyDataTemplate>
</asp:GridView>
```

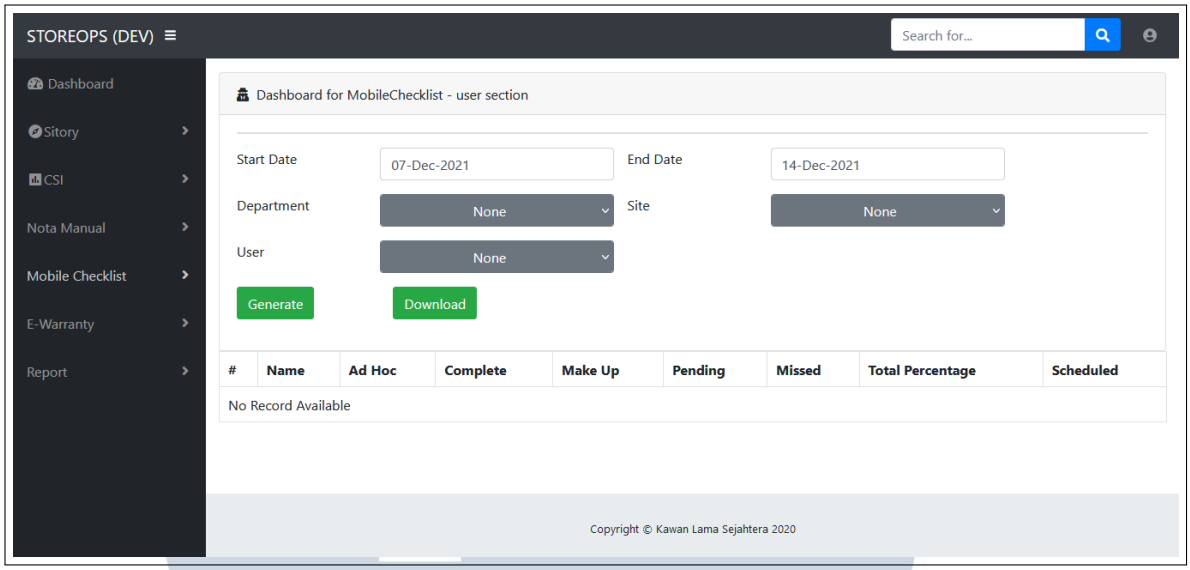
Gambar 3.21. Contoh pengikatan data tabel ke dalam grid

Fitur yang terdapat pada halaman ini adalah tombol untuk mengunduh data yang terdapat pada tabel dengan format Excel dan berpindah ke halaman detail dari PIC atau toko cabang yang telah dipilih oleh pengguna dengan menekan nama yang terdapat pada tabel. Fungsi unduh bekerja dengan cara menempatkan setiap data dari API dalam tabel dan memasukannya dalam Excel menggunakan *library* NPOI.HSSF.UserModel.HSSFWorkbook untuk kemudian di ekspor menggunakan *library* HttpResponse dengan objek Page.Response seperti pada Gambar 3.22.

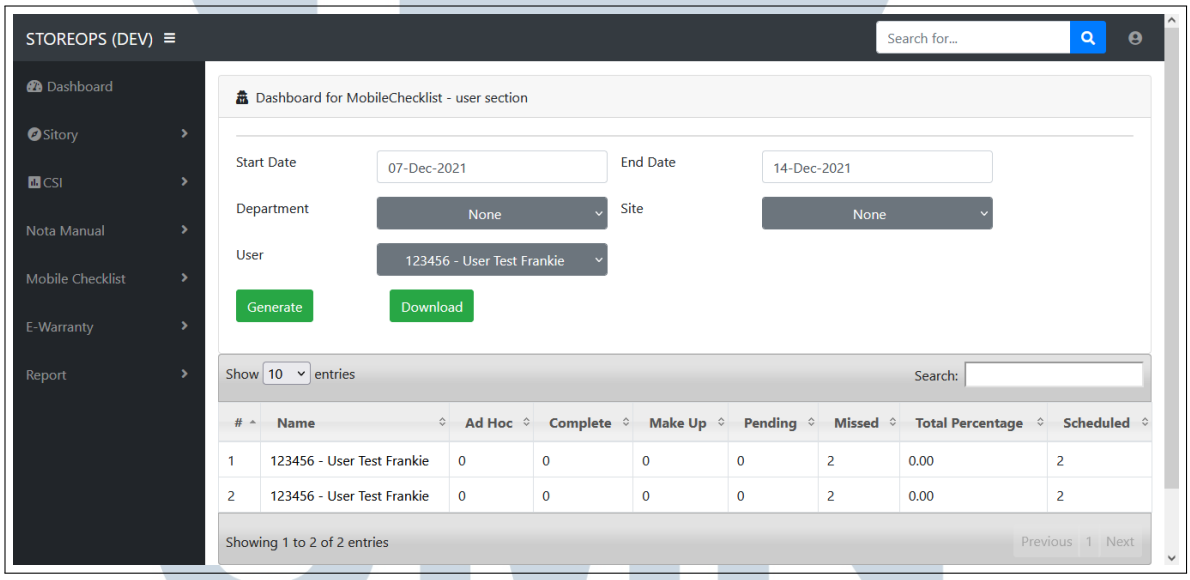
```
protected void btnDownload_Click(object sender, EventArgs e)
{
    LoadToTable();
    if(dt.Rows.Count > 0)
    {
        HSSFWorkbook workbook = new HSSFWorkbook();
        var sheet = workbook.CreateSheet("Sheet1");

        var row = sheet.CreateRow(0);
        int columnIndex = 0;

        foreach (DataColumn column in dt.Columns)
        {
            row.CreateCell(columnIndex).SetCellValue(column.ColumnName);
            columnIndex++;
        }
        for (int i = 0; i < dt.Rows.Count; i++)
        {
            row = sheet.CreateRow(i + 1);
            columnIndex = 0;
            foreach (DataColumn column in dt.Columns)
            {
                row.CreateCell(columnIndex).SetCellValue(dt.Rows[i][column.ColumnName].ToString());
                columnIndex++;
            }
        }

        using (var exportData = new MemoryStream())
        {
            workbook.Write(exportData);
            string saveAsFileName = "ChecklistDashboardUser.xls";
            Response.ContentType = "application/vnd.ms-excel";
            Response.AddHeader("Content-Disposition", string.Format("attachment;filename={0}", saveAsFileName));
            Response.Clear();
            Response.BinaryWrite(exportData.GetBuffer());
            Response.End();
        }
```

Gambar 3.22. Fungsi untuk mengunduh data tabel

Hasil kompilasi dari program yang telah dibuat untuk halaman Checklist Dashboard User ditunjukkan pada Gambar 3.23 ketika tanpa data dan Gambar 3.24 ketika dengan adanya data.



Gambar 3.23. Tampilan pada Checklist Dashboard User



Gambar 3.24. Tampilan pada Checklist Dashboard User dengan Data

B. Implementasi Halaman Checklist Dashboard User Detail

Data detail dari PIC atau toko cabang yang dipilih oleh pengguna pada halaman Checklist Dashboard User tersedia pada API dengan *endpoint* ”RequestReportDashboard_Detail”. Beberapa metode yang digunakan pada halaman ini memiliki kemiripan dengan metode pada halaman Checklist Dashboard User sehingga tidak ditampilkan lengkap pada sub-bagian ini.

Fitur yang terdapat pada halaman ini adalah fungsi unduh untuk data keseluruhan, fungsi unduh ketika ada laporan yang telah selesai dikerjakan dan fungsi untuk berganti bulan dari data yang ditampilkan. Penampilan data pada halaman ini memiliki 2 rupa yaitu dalam bentuk lis dan kalender, keduanya menampilkan data dalam periode 1 bulan yaitu dari tanggal 1 hingga tanggal terakhir pada bulan tersebut.

Pengambilan data dari API untuk data detail dari PIC atau toko cabang yang dipilih adalah dengan mengirimkan *request* nama PIC atau toko cabang dalam Session yang telah diisi sebelumnya di halaman Checklist Dashboard User seperti pada Gambar 3.25.

Checklist Dashboard User

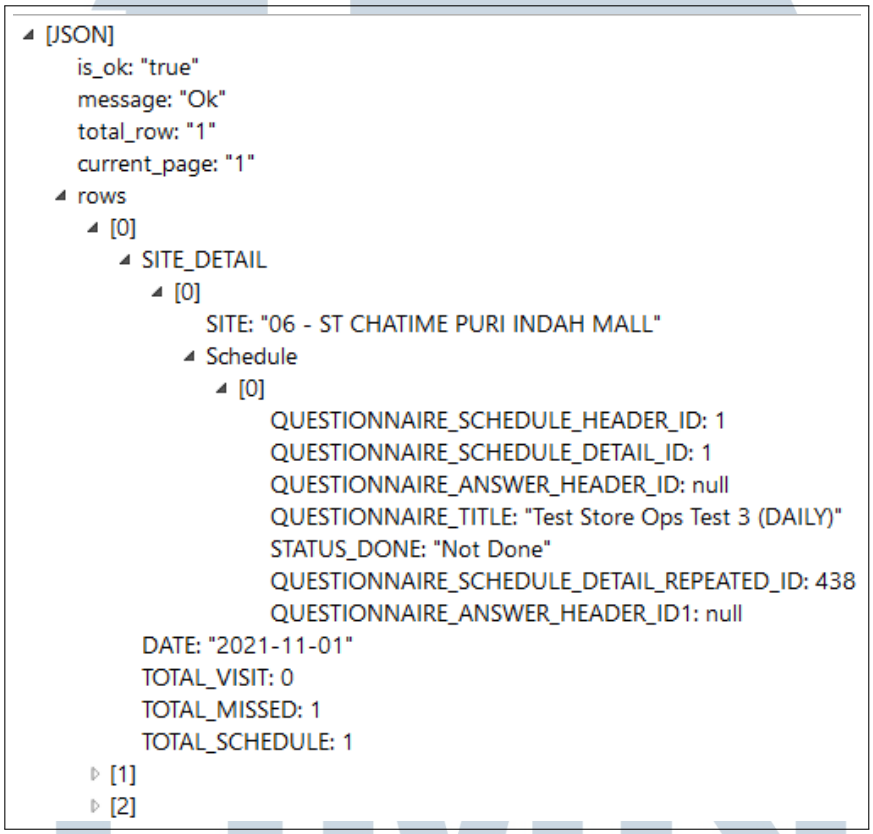
```
protected void btn_user_Click(object sender, EventArgs e)
{
    Session["userDetails"] = userSession.Value;
    Response.Redirect("~/ChecklistDashboardUserDetail.aspx");
}
```

Checklist Dashboard User Detail

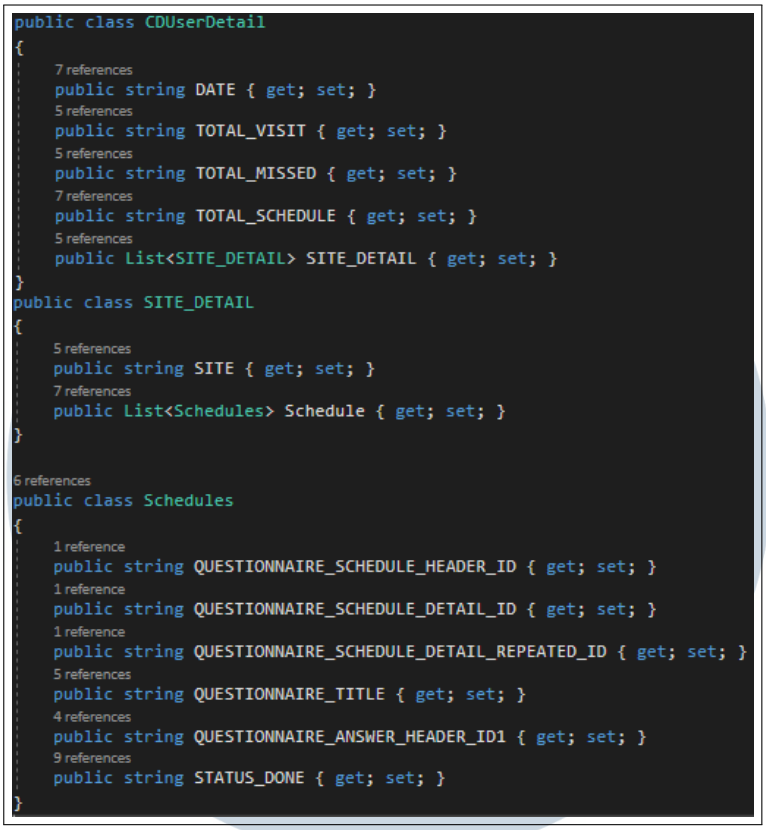
```
int charLocation = Session["userDetails"].ToString().IndexOf(" -", StringComparison.Ordinal);
string userAuditor = Session["userDetails"].ToString().Substring(0, charLocation);
request.USER_AUDITOR = userAuditor;
```

Gambar 3.25. Mendapatkan Nama PIC atau toko cabang

Metode penerimaan data JSON dari API dibuat mirip dengan metode pada Checklist Dashboard User yaitu menggunakan *class* penampung dengan properti yang sama seperti *key* pada JSON yang dikembalikan. Setiap JSON akan dimasukkan ke dalam *class* mirip seperti yang ditunjukkan pada Gambar 3.18 namun dengan nama *class* yang berbeda.



Gambar 3.26. *Response* dari API



```
public class CDUserDetail
{
    7 references
    public string DATE { get; set; }
    5 references
    public string TOTAL_VISIT { get; set; }
    5 references
    public string TOTAL_MISSED { get; set; }
    7 references
    public string TOTAL_SCHEDULE { get; set; }
    5 references
    public List<SITE_DETAIL> SITE_DETAIL { get; set; }
}
public class SITE_DETAIL
{
    5 references
    public string SITE { get; set; }
    7 references
    public List<Schedules> Schedule { get; set; }
}

6 references
public class Schedules
{
    1 reference
    public string QUESTIONNAIRE_SCHEDULE_HEADER_ID { get; set; }
    1 reference
    public string QUESTIONNAIRE_SCHEDULE_DETAIL_ID { get; set; }
    1 reference
    public string QUESTIONNAIRE_SCHEDULE_DETAIL_REPEATED_ID { get; set; }
    5 references
    public string QUESTIONNAIRE_TITLE { get; set; }
    4 references
    public string QUESTIONNAIRE_ANSWER_HEADER_ID1 { get; set; }
    9 references
    public string STATUS_DONE { get; set; }
}
```

Gambar 3.27. *Class* untuk menampung data detail

Penampilan data untuk tampilan lis menggunakan Repeater Class dari System.Web.UI.WebControls sehingga data tidak perlu dimasukan dalam sebuah grid namun seperti tabel yang mengulang bagian badan tabel tersebut hingga selesai yang dipisahkan oleh bagian pemisah yang dapat dibuat mandiri. Berikut contoh penggunaan dari Repeater pada Gambar 3.28 dengan mengikat data seperti contoh pada Gambar 3.20.

```
<asp:Repeater ID="Repeater1" runat="server">
    <HeaderTemplate>
        <table>
    </HeaderTemplate>
    <SeparatorTemplate>
        <tr>
            <td>
                <hr />
            </td>
        </tr>
    </SeparatorTemplate>
    <ItemTemplate>
        <tbody>
            <tr>
                <td>
                    <asp:Label runat="server" Text='<%#Eval("namaKolom")%>'></asp:Label>
                </td>
            </tr>
        </tbody>
    </ItemTemplate>
    <FooterTemplate>
        </table>
    </FooterTemplate>
</asp:Repeater>
```

Gambar 3.28. Contoh penggunaan Repeater

Penampilan data untuk tampilan kalender menggunakan Calendar Class dari System.Web.UI.WebControls yang menampilkan data dalam blok-blok harian seperti dalam kalender pada umumnya. Pada kalender ini data dimasukan melalui metode yang *event-triggered* yaitu berjalan saat kondisi tertentu terpenuhi, dalam hal ini ketika blok hari dalam kalender akan ditampilkan. Contoh pada Gambar 3.29 menggunakan variabel initRender untuk menentukan apakah blok hari pertama pada kalender ingin ditampilkan, fungsinya adalah mempercepat waktu setiap kali blok hari lain ingin ditampilkan dengan menyimpan data yang dikembalikan dari API dalam sebuah List. List data dalam program lebih cepat dieksekusi dibandingkan dengan mengambil data dari API terus-menerus setiap kali blok hari ingin ditampilkan. Fokus penampilan data pada kalender ini adalah hanya tanggal 1 sampai akhir bulan tersebut sehingga blok hari yang termasuk bulan lainnya akan dikosongkan.

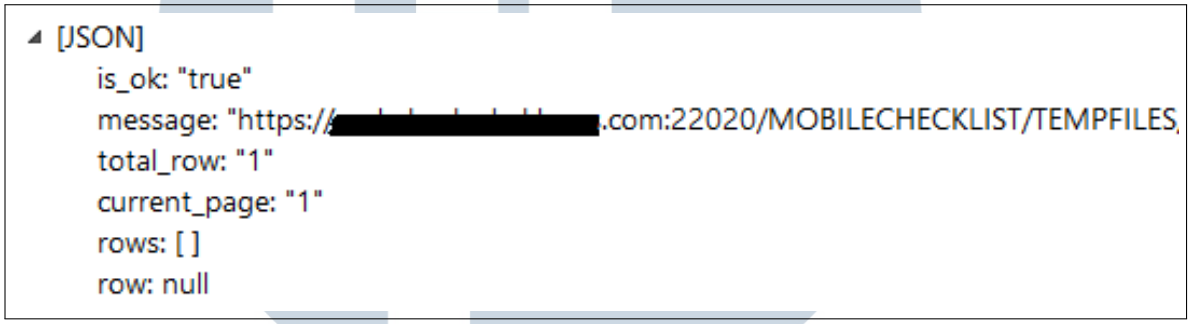
```
protected void dataCalender_DayRender(object sender, DayRenderEventArgs e)
{
    if (initRender)
    {
        ResponseCDUserDetail response = LoadWithAPI();
        listDetail = new List<CDUserDetail>();
        foreach (CDUserDetail result in response.rows)
        {
            listDetail.Add(new CDUserDetail
            {
                DATE = result.DATE.ToString(),
                TOTAL_MISSED = result.TOTAL_MISSED.ToString(),
                TOTAL_SCHEDULE = result.TOTAL_SCHEDULE.ToString(),
                TOTAL_VISIT = result.TOTAL_VISIT.ToString()
            });
        }
        initRender = false;
    }
    if (e.Day.IsOtherMonth)
    {
        e.Cell.Text = string.Empty;
    }
    else
    {
        DateTime date = e.Day.Date;
        string day = date.ToString("yyyy-MM-dd");

        if (listDetail.Count > 0)
        {
            foreach (CDUserDetail result in listDetail)
            {
                if (result.DATE == day)
                {
                    e.Cell.Controls.Add(new LiteralControl(
                        "<br/> <div class='container-fluid'>" +
                        "<div style='background-color:#DBF3FA'><p font-size:large'>" + result.TOTAL_SCHEDULE + "</p></div>" +
                        "<div style='background-color:#FFCCCB'><p font-size:large'>" + result.TOTAL_MISSED + "</p></div>" +
                        "<div style='background-color:#B1F3B1'><p font-size:large'>" + result.TOTAL_VISIT + "</p></div>" +
                        "</div>"
                        ));
                }
            }
        }
```

Gambar 3.29. Metode penampilan data dalam kalender

Fitur unduh pada halaman ini terdapat 2 bagian yaitu untuk data detail setiap hari dalam sebulan dan untuk laporan dari pekerjaan yang telah selesai. Data detail dalam sebulan yang ingin diunduh akan menggunakan metode yang sama seperti pada Gambar 3.22. Ketika ingin mengunduh laporan untuk pekerjaan yang telah selesai maka akan menggunakan API dengan *endpoint* ”RequestReport” yang akan mengembalikan pesan berisikan sebuah pranala (Gambar 3.31) yang mengunduh laporan secara otomatis saat halaman yang ditunjuk pranala tersebut dibuka. *Request* ke API tersebut menggunakan metode yang sama seperti pada Gambar 3.12 dengan menggunakan *class* pada Gambar 3.30.

```
public class PDFRequest
{
    1 reference
    public List<string> QUESTIONNAIRE_ANSWER_HEADER_ID { get; set; }
    1 reference
    public string FILE_TYPE { get; set; }
    1 reference
    public string REQUEST_TYPE { get; set; }
}
```

Gambar 3.30. *Class* untuk mengunduh laporan



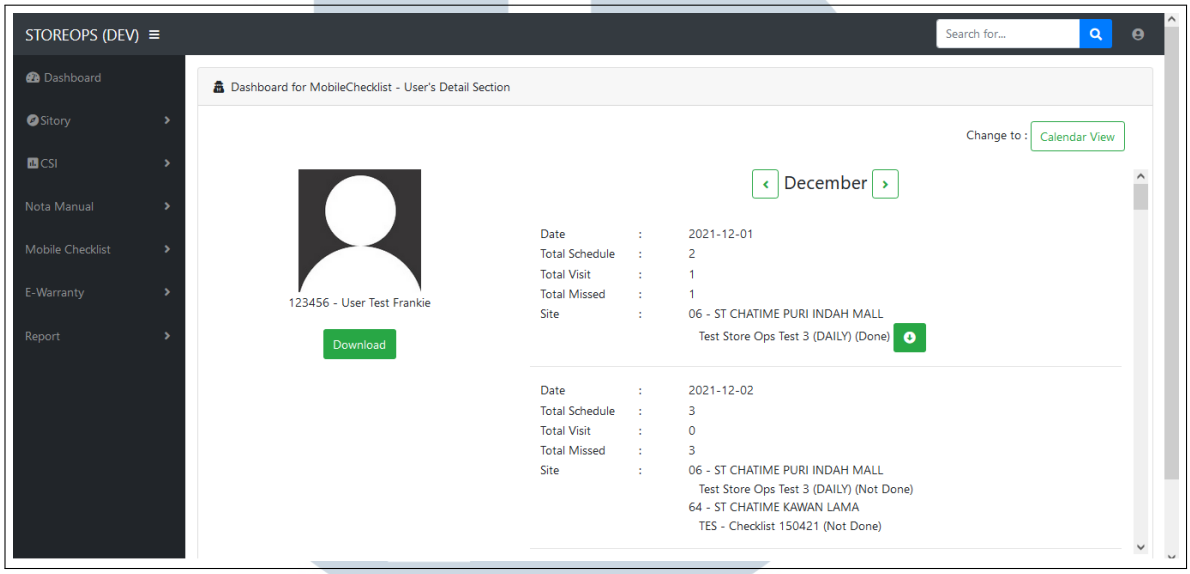
Gambar 3.31. JSON yang dikembalikan untuk mengunduh laporan

Fitur pergantian bulan menggunakan metode yang sudah disediakan oleh Calender Class dengan beberapa penyesuaian seperti pada Gambar 3.32. Jika data untuk 1 bulan pada kalender berganti maka data pada lis juga mengikuti.

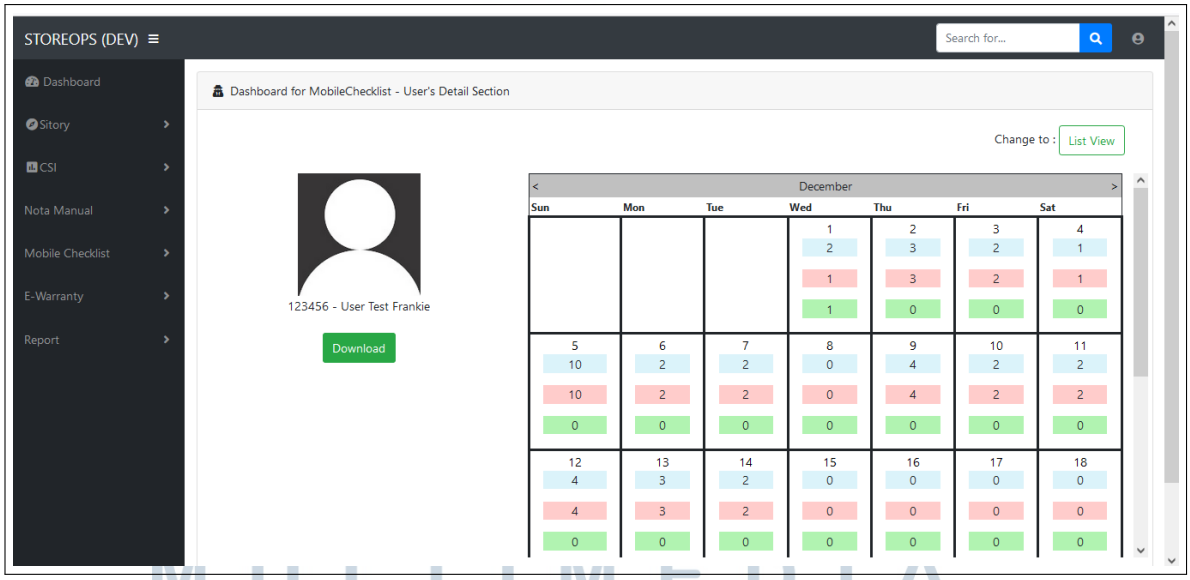
```
protected void dataCalender_VisibleMonthChanged(object sender, MonthChangedEventArgs e)
{
    now = e.NewDate;
    Session["startDate"] = now;
    txtStartDate = new DateTime(now.Year, now.Month, 1);
    txtEndDate = txtStartDate.AddMonths(1).AddDays(-1);

    currentMonth.Text = now.ToString("MMMM");

    CreateTable();
    LoadToTable();
}
```

Gambar 3.32. Metode pergantian bulan

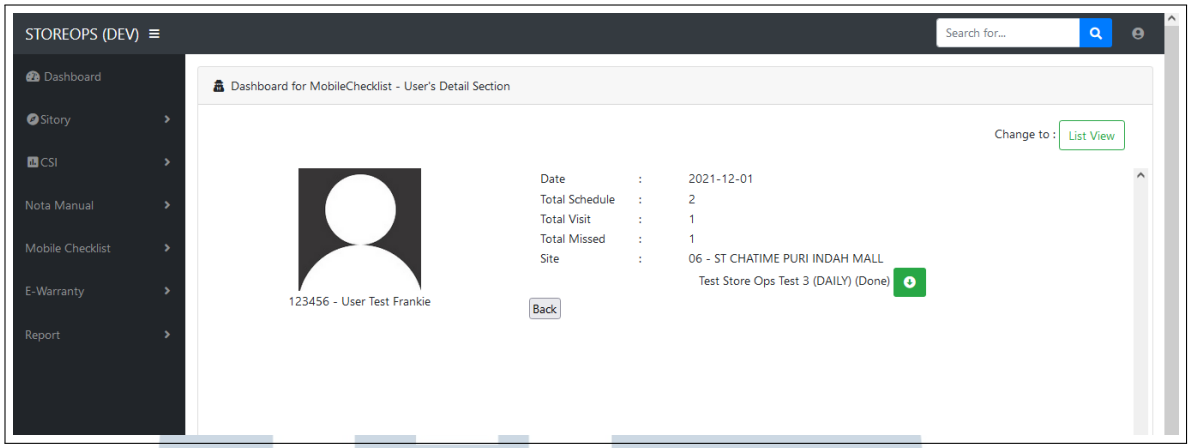
Hasil kompilasi dari program yang telah dibuat untuk halaman Checklist Dashboard User Detail ditunjukkan pada Gambar 3.33 untuk tampilan lis, Gambar 3.34 untuk tampilan kalender dan Gambar 3.35 untuk data detail harian pada kalender.



Gambar 3.33. Tampilan Checklist Dashboard User Detail untuk lis



Gambar 3.34. Tampilan Checklist Dashboard User Detail untuk kalender



Gambar 3.35. Tampilan Checklist Dashboard User Detail untuk data detail harian kalender

3.4 Kendala dan Solusi yang Ditemukan

3.4.1 Kendala

Dalam pengerjaan Checklist Dashboard User dan Checklist Dashboard User Detail terdapat berbagai macam kendala yang dihadapi seperti yang diuraikan berikut ini :

- Terdapat perbedaan kompatibilitas pada ASP.NET saat menggunakan Bootstrap untuk menampilkan data.
- Inisiasi yang tidak tepat untuk metode dan variabel ketika melewati siklus *post-back* dapat mengakibatkan eror atau waktu respon melambat.
- Fitur yang ingin dibuat memiliki sedikit dokumentasi mulai dari parameter yang digunakan hingga contoh penggunaan.

3.4.2 Solusi

Cara yang ditempuh untuk menyelesaikan setiap masing-masing kendala yang telah disebutkan adalah sebagai berikut :

- Melakukan debugging menggunakan breakpoint untuk server-side dan *inspect element* dan konsol untuk client-side dengan JavaScript.
- Menginisiasi ulang setiap metode dan variabel yang dibutuhkan serta menampilkannya kembali sesuai kebutuhan.
- Membaca dokumentasi yang diberikan oleh Microsoft dan mencari contoh penggunaan masing-masing *syntax* dan metode dari program yang pernah dibuat atau dibahas oleh orang lain.