# BAB 3
# PELAKSANAAN KERJA MAGANG

## 3.1 Kedudukan dan Koordinasi

Praktik kerja magang ini dilaksanakan sebagai *Full Stack Developer Intern*. Posisi ini bertanggung jawab membangun aplikasi berbasis *website*, berdiskusi tentang proyek yang sedang dikerjakan bersama *Business Analyst* untuk memastikan proyek yang dikerjakan sesuai dengan kebutuhan klien, serta melakukan *bug fixing* apabila dibutuhkan perbaikan atau penyesuaian pada hasil pekerjaan sebelumnya. Di awal masa praktik kerja magang, dilaksanakan pelatihan terkait teknologi yang digunakan di perusahaan selama 2 minggu secara daring melalui aplikasi *Google Meet* atau *Discord*.

Pelaksanaan praktik kerja magang dibimbing oleh Adrian Christanto, Eagar Satya, Roberto Wu selaku *IT Architect* dan disupervisi oleh Gina Akmalia selaku *Vice President Human Resources and Finance*. Setelah pelaksanaan pelatihan, *Full Stack Developer Intern* akan ikut serta dalam pengerjaan proyek klien, baik internal maupun eksternal.

Proyek pertama yang dikerjakan merupakan proyek eksternal yang dipimpin oleh Adityo Bayu Pranoto selaku *Project Manager* dan Eagar Satya selaku *IT Architect*. Koordinasi antar tim dilakukan melalui *group chat Whatsapp* atau pertemuan daring melalui aplikasi *Google Meet* atau *Discord*. Dalam rangka meningkatkan organisasi dan memudahkan *tracking progress task*, tim menggunakan aplikasi *Trello* dan *Clockify*.

## 3.2 Tugas yang Dilakukan

Dalam pelaksanaan kerja magang, tugas yang dikerjakan adalah sebagai berikut.

1. Pelatihan *Full Stack Developer Intern* mencakup 3 topik utama yaitu, *front-end, backend,* dan *database system*. Materi *front-end* meliputi penggunaan *AngularJS, Axios, Bootstrap, CSS, HTML,* dan *JavaScript*. Materi *back-end* meliputi penggunaan bahasa *C#, .NET,* dan *Linq*. Materi *database system* meliputi *SQL* melalui aplikasi *Microsoft SQL Server Management Studio*. Di akhir periode pelatihan dilaksanakan ujian dalam bentuk implementasi sistem

*Create, Read, Update, Delete (CRUD)* pada sistem *Enterprise Resource Planning (ERP)* yang telah dikembangkan sebelumnya.

2. Pengembangan fitur *Create, Read, Update, Delete (CRUD)* dan *pop-up Announcement* pada proyek *Announcement System*.

## 3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1.

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

| **Minggu Ke -** | **Pekerjaan yang dilakukan** |
|---|---|
| 1 | Mengikut program *Developer Intern Training* yang terdiri dari materi pembelajaran *front-end, back-end, dan database system* menggunakan *C#, Git, SQL, ASP .NET,* dan *Linq* |
| 2 | Mengikuti program *Developer Intern Training* yang mempelajari struktur *project* yang akan dikerjakan selama praktik kerja magang |
| 3 | Mengikuti program *Developer Intern Training* dengan mencoba mengimplementasikan fitur baru pada *project* yang dipelajari dan mempersiapkan diri untuk pengerjaan *external project (Announcement System)* dengan mempelajari struktur dan perubahan yang akan dikembangkan |
| 4 | Mempersiapkan diri untuk pengerjaan *external project (Announcement System)* selanjutnya dengan mempelajari struktur dan perubahan yang akan dikembangkan |
| 5 | Mempersiapkan diri untuk pengerjaan *external project (Announcement System)* dengan mempelajari struktur dan perubahan yang akan dikembangkan |
| 6 | Mengembangkan fitur *Master Announcement, Create Announcement* pada *external project (Announcement System)* |
| 7 | Mengembangkan fitur *Create Announcement, Edit Announcement* pada *external project (Announcement System)* |
| Lanjut pada halaman berikutnya | |

Tabel 3.1 Pekerjaan yang dilakukan setiap minggu selama pelaksanaan kerja magang (lanjutan)

| **Minggu Ke -** | **Pekerjaan yang dilakukan** |
|---|---|
| 8 | Mengembangkan fitur *Pop-up Announcement, Update Status Announcement, Delete Announcement* serta melakukan optimisasi kode pada fitur yang sudah dikembangkan dalam *external project (Announcement System)* |
| 9 | Mempersiapkan dan melaksanakan *external project (Announcement System) System Integrity Testing* dengan melakukan *bug-fixing* dan *improvement* jika dibutuhkan |
| 10 | Mempersiapkan dan melaksanakan *external project (Announcement System) System Integrity Testing* dan *User Acceptance Testing* dengan melakukan *bug-fixing* dan *improvement* jika dibutuhkan |
| 11 | Melaksanakan *external project (Announcement System) User Acceptance Testing* dengan melakukan *bug-fixing* dan *improvement* jika dibutuhkan |
| 12 | Melaksanakan *external project (Announcement System) Unit Testing* dengan mempersiapkan, menjalankan, dan mendokumentasikan *script unit test* |
| 13 | Mempersiapkan dan melaksanakan *external project (Announcement System) Go Live* |
| 14 | Mempersiapkan *template project* untuk pelaksanaan *unit test internal project* |
| 15 | Mengembangkan *improvement* yang dibutuhkan untuk *external project* |
| 16 | Mengembangkan *improvement* yang dibutuhkan untuk *external project* |

### 3.3.1 *Announcement System*

*Change Request (CR)* ini bertujuan untuk memudahkan klien dalam menyebarkan informasi secara *real-time* kepada pengguna aplikasi. Fitur yang dibutuhkan oleh klien adalah *administrator panel* untuk manajemen *announcement*

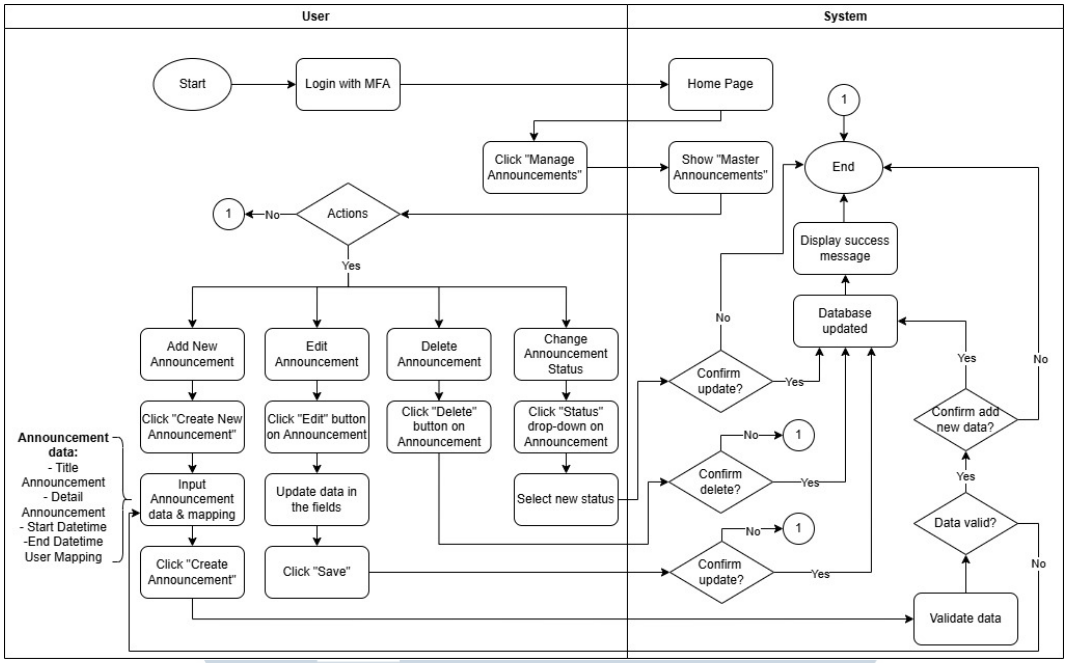
yang terdiri dari operasi *Create, Read, Update, Delete (CRUD)* serta *announcement pop-up* yang muncul di halaman utama sistem. Berikut merupakan tahapan yang dilalui untuk mengerjakan CR ini.

## A. Perancangan Sistem

Tahap ini merupakan tahap paling awal dalam pengerjaan suatu *Change Request (CR)*. Tujuan tahap ini adalah mengumpulkan seluruh informasi mengenai CR ini sehingga dapat dirancang desain sistem yang sesuai dengan kebutuhan klien. Langkah pertama yang dilakukan adalah melakukan analisis dan mencatat poin-poin penting yang menjadi fokus utama pada CR ini bersama dengan *Business Analyst* terkait. Berdasarkan analisis, beberapa poin penting dalam CR ini adalah sebagai berikut.

1. Dibutuhkan 4 buah komponen/halaman yang terdiri dari *Master Announcement* yang terdiri dari *list announcement* dimana user dapat melakukan *filter* data, merubah *status announcement* dan menghapus *announcement*; *Create Announcement* dan *Edit Announcement* yang terdiri dari *form input* mengenai data-data yang diperlukan, *Pop-up Announcement* yang terdiri dari *pop-up modal* yang menampung *announcement* yang sedang aktif.
2. *Form* pada halaman *Create Announcement* serta *Edit Announcement* wajib memiliki validasi input untuk memastikan integritas data.
3. *Announcement* memiliki 3 tipe *visibility*, *"All Users", "Specific Apps",* dan *"Specific Users"*.

Berdasarkan hasil analisis sebelumnya, *Business Analyst* merancang *flowchart* yang merepresentasikan alur kerja fungsi yang sudah dirancang. Gambar 3.1 merupakan hasil perancangan *flowchart*.

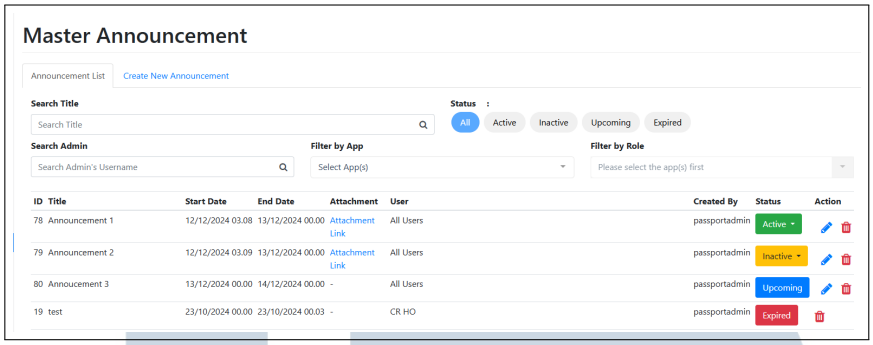



Gambar 3.1. Flowchart *Announcement System*

## B. Implementasi

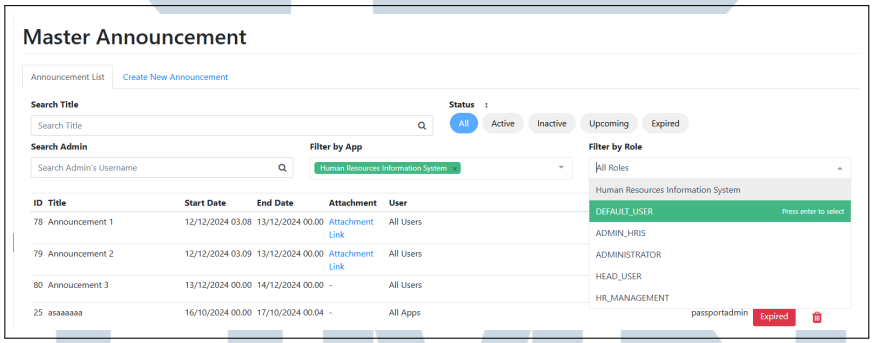
Tahap ini merupakan pengembangan fitur berdasarkan hasil analisis di tahap sebelumnya. Dalam tahap ini pengembangan yang dilakukan adalah dari sisi *front-end* dan *back-end*, sedangkan untuk sisi *database system* sudah disediakan oleh *IT Architect*. Pengembangan yang dilakukan mengikuti *software design pattern* yang sudah diimplementasi pada aplikasi sebelumnya yaitu *Model-View-Controller (MVC)*. *Model* bertanggung jawab menangani struktur data, *Service* bertanggung jawab menangani *business logic*, mengakses data dari *database*, mengolah data, dan menyediakan data untuk *View* dan *Controller*. *View* bertanggung jawab menampilkan data (UI) dan mengirimkan input *user* ke *Controller*. *Controller* bertanggung jawab mengontrol alur aplikasi dan bertindak sebagai perantara antara *Service* dan *View*. Kode yang terlampir pada lampiran 6 merupakan contoh potongan kode dari *Announcement Service*.

Selain sisi *back-end*, sisi *front-end* dikembangkan dengan menggunakan *VueJS*. Halaman/komponen yang dikembangkan terdiri dari *Master Announcement, Create Announcement, Edit Announcement,* dan *Pop-up Announcement*. Gambar 3.2 merupakan contoh bentuk halaman *Master Announcement* yang terdiri dari *filter, list announcements*, dan opsi untuk *update status* serta *delete announcement*.



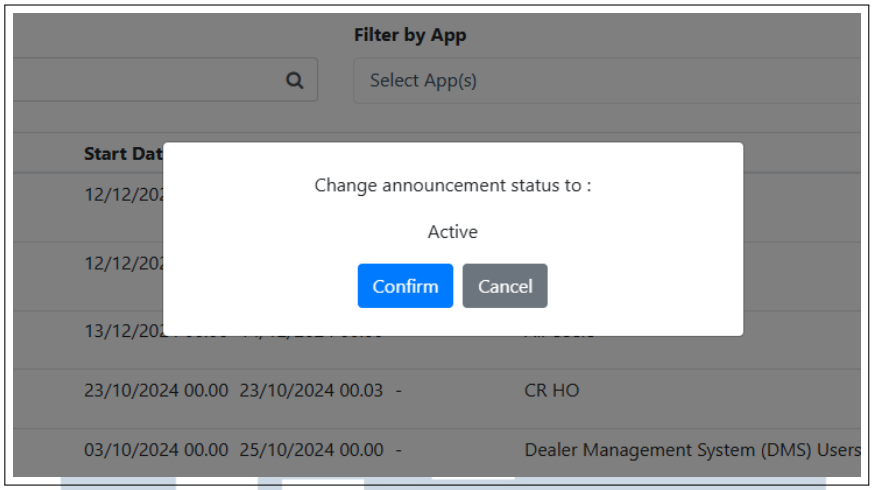

Gambar 3.2. Contoh halaman *Master Announcement*

Disediakan opsi *filter announcement* berdasarkan *title, status, username creator, application mapping*, dan *role mapping*. Untuk *filter role mapping* hanya dapat diisi ketika *filter application mapping* sudah diisi, karena opsi *role* yang tersedia menyesuaikan dengan *role* yang sudah ter-*mapping* pada aplikasi. Contoh bentuk opsi *filter announcement* dapat dilihat pada gambar 3.3.



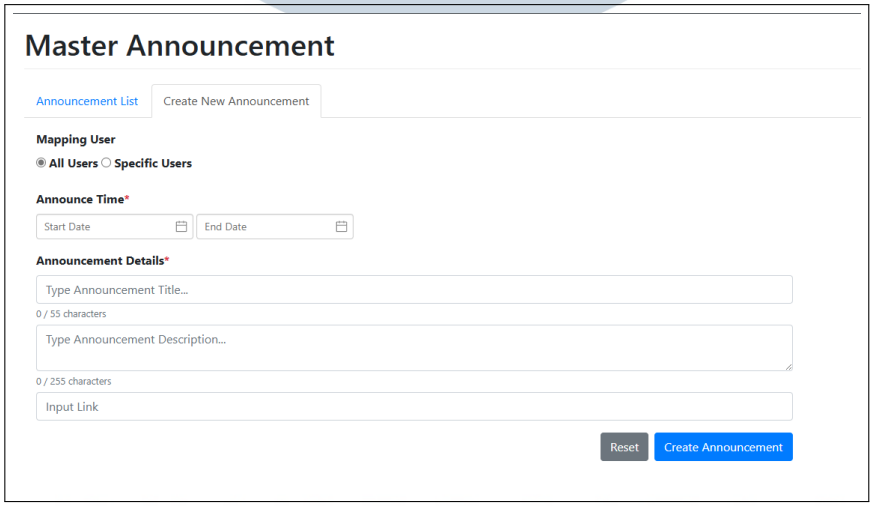

Gambar 3.3. Contoh opsi *filter* pada halaman *Master Announcement*

Ketika *user* melakukan operasi *update* maupun *delete* akan muncul *confirmation pop-up* seperti di contoh gambar 3.4.



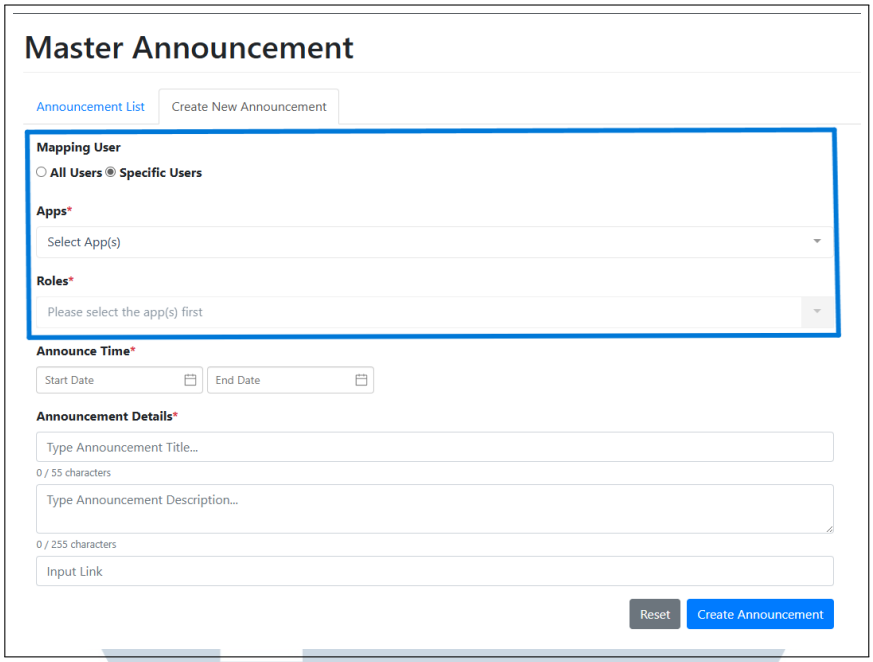

Gambar 3.4. Contoh *confirmation pop-up* pada halaman *Master Announcement*

Gambar 3.5 merupakan contoh bentuk halaman *Create Announcement* dan *Edit Announcement* yang terdiri dari *form input*. Sebelum *user* dapat men-submit form, setiap input akan divalidasi kembali sesuai dengan *rules* yang ditetapkan oleh klien.



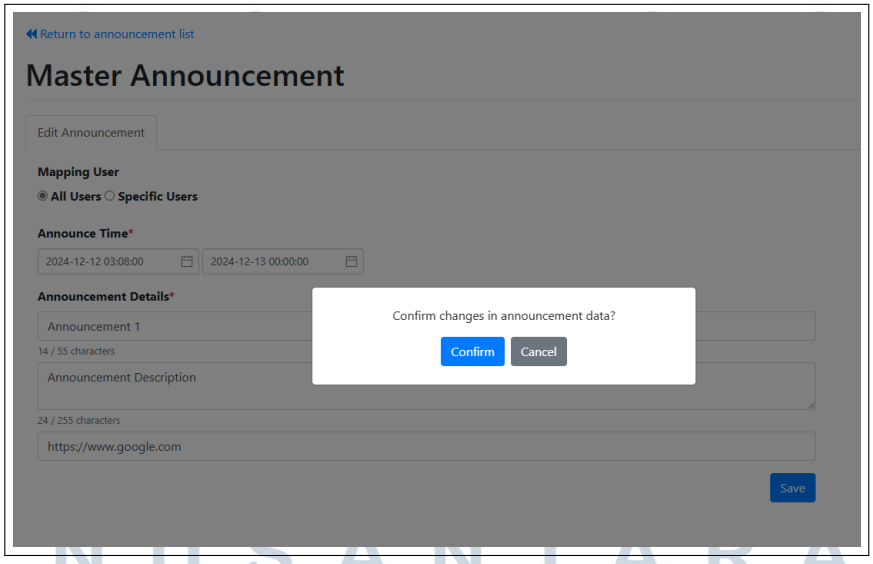

Gambar 3.5. Contoh halaman *Create Announcement* dan *Edit Announcement*

Kedua halaman ini memiliki *input* tambahan pada bagian *"Mapping Users"* ketika memilih opsi *"Specific Users"* yang dapat dilihat pada gambar 3.6.




Gambar 3.6. Contoh opsi *"Specific Users"* pada halaman *Create Announcement* dan *Edit Announcement*

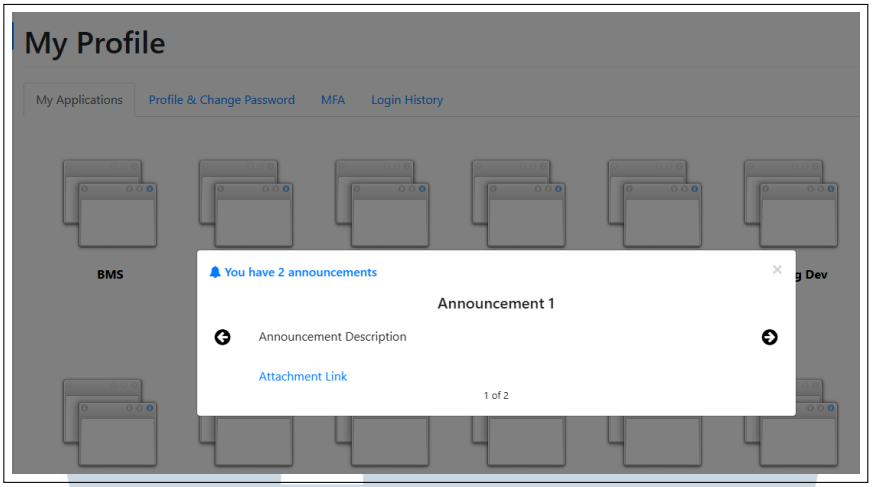
Namun, khusus untuk halaman *Edit Announcement*, ketika *user* ingin menyimpan hasil yang telah diubah maka akan muncul *confirmation pop-up* yang dapat dilihat contohnya pada gambar 3.7.




Gambar 3.7. Contoh *confirmation pop-up* pada halaman *Edit Announcement*

Ketika *user* sedang berada di halaman utama sistem dan terdapat *active announcement* yang ditujukan kepada *user* tersebut maka akan dimunculkan *pop-up*

seperti di contoh gambar 3.8.




Gambar 3.8. Contoh *Pop-up Announcement*

## C. *Unit Testing*

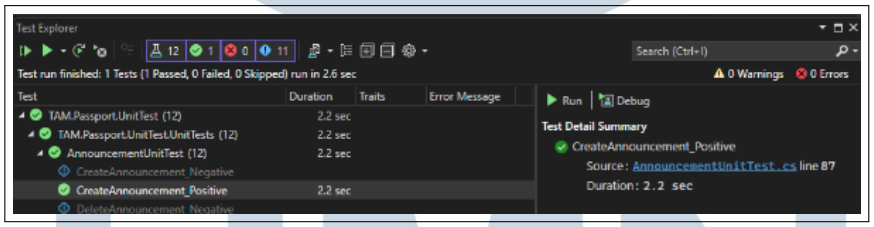
Tahap ini merupakan tahap terakhir dimana seluruh *unit* yang sudah dikembangkan dalam proyek ini akan diujicoba untuk memastikan *unit* sudah berfungsi dengan benar. Dalam proyek ini tahap *Unit Testing* dilakukan dengan menggunakan *library NUnit* untuk membantu memudahkan pengecekan. Pengerjaan *unit testing* disiapkan *case positive* yang berarti mencoba skenario alur/data sesuai dengan ekspektasi sistem dan *case negative* yang berarti mencoba skenario alur/data tidak sesuai dengan ekspektasi sistem. Kode 3.1 merupakan contoh potongan kode dari *unit test case positive* untuk operasi *create announcement*. Potongan kode *unit test* ini disebut sebagai *case positive* karena *test* ini mencoba mengirimkan data yang valid kepada rute *API create announcement*.

```
1  [Test]
2  public async Task CreateAnnouncement_Positive()
3  {
4      var newAnnouncement = new AnnouncementCreateModel
5      {
6          AnnouncementDetail = "Unit Test",
7          AnnouncementMappingTypeId = 2,
8          AnnouncementTitle = "Announcement Test" + Guid.NewGuid().
       ToString(),
9          AppId = new List<Guid?> { new Guid("eca95eee-4143-4dfc-
       b595-d232b98437c2") },
10         StartDate = DateTimeOffset.Now.AddDays(1),
```

```
            EndDate = DateTimeOffset.Now.AddDays(30),
            Mapping = "App A Users",
            AppRoleMap = new List<AnnouncementMappingModel?>()
        };

        var request = new StringContent(JsonConvert.SerializeObject(
    newAnnouncement), Encoding.UTF8, "application/json");

        var response = await _client.PostAsync("api/v1/announcement/
    create", request);

        Assert.That(response.IsSuccessStatusCode, Is.True);
}
```

Kode 3.1: Contoh potongan kode *Create Announcement Positive Case Unit Testing*

Setelah menjalankan *unit test* tersebut, didapatkan hasil *test* seperti pada contoh gambar 3.9. Hasil *test* ini membuktikan bahwa operasi *create announcement* menggunakan data yang valid berhasil dilakukan karena *response code* yang didapatkan menunjukan status sukses (200).




Gambar 3.9. Contoh Hasil *Unit Testing*

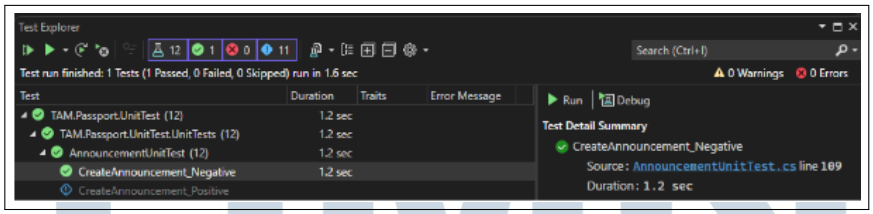
Kode 3.2 merupakan contoh potongan kode dari *unit test case negative* untuk operasi *create announcement*. Potongan kode *unit test* ini disebut sebagai *case negative* karena *test* ini mencoba mengirimkan data yang tidak valid kepada rute *API create announcement*.

```
[Test]
public async Task CreateAnnouncement_Negative()
{
    var newAnnouncement = new AnnouncementCreateModel
    {
        AnnouncementDetail = "Unit Test",
        AnnouncementMappingTypeId = 2,
        AnnouncementTitle = "Announcement Negative Test" + Guid.
    NewGuid().ToString(),
```

```
9           AppId = new List<Guid?> { new Guid("eca95eee-4143-4dfc-
    b595-d232b98437c2") },
10          StartDate = DateTimeOffset.Now.AddDays(1),
11          EndDate = DateTimeOffset.Now.AddDays(30),
12          Mapping = "App A Users",
13          AppRoleMap = new List<AnnouncementMappingModel?>()
14      };
15
16      var request = new StringContent(JsonConvert.SerializeObject(
    newAnnouncement), Encoding.UTF8, "application/json");
17
18      var response = await _client.PostAsync("api/v1/announcement/
    create", request);
19
20      Assert.That(response.IsSuccessStatusCode, Is.False);
21  }
```

Kode 3.2: Contoh potongan kode *Create Announcement Negative Case Unit Testing*

Setelah menjalankan *unit test* tersebut, didapatkan hasil *test* seperti pada contoh gambar 3.10. Hasil *test* ini membuktikan bahwa operasi *create announcement* menggunakan data yang tidak valid gagal dilakukan karena *response code* yang didapatkan tidak menunjukan status sukses (200).




Gambar 3.10. Contoh Hasil *Unit Testing*

## 3.4 Kendala yang Ditemukan

Selama pelaksanaan kerja magang ini ditemukan beberapa kendala ketika mengerjakan berbagai *project* yang sudah dilampirkan di atas. Beberapa kendala yang ditemukan adalah sebagai berikut.

a. Limitasi operasi *group join* menggunakan *Linq*

Pada halaman *Master Announcement* terdapat kolom *user* yang berisikan informasi mengenai tipe *visibility announcement*. Jika tipe *visibility* yang sudah di-*assign* merupakan tipe *"Specific Apps"* atau *"Specific Users"* maka

wajib dilampirkan nama aplikasi atau *role* yang dituju. Hal ini dilakukan untuk memberikan klien untuk melakukan filter terhadap *List Announcement* berdasarkan aplikasi maupun *role* tertentu. Namun, dalam merealisasikan kebutuhan tersebut dibutuhkan operasi *group join* dari *table Announcement* ke lebih dari dua *table*. Ketika mencoba melakukan *group join* menggunakan *function GroupJoin()* pada *Linq*, ditemukan *error* dikarenakan *Linq* tidak dapat memproses *function* tersebut.

b. *Database load concern*

Pada halaman *Master Announcement* terdapat kolom *user* yang berisikan informasi mengenai tipe *visibility announcement*. Jika tipe *visibility* yang sudah di-*assign* merupakan tipe *"Specific Apps"* atau *"Specific Users"* maka wajib dilampirkan nama aplikasi atau *role* yang dituju. Hal ini dilakukan untuk memberikan klien untuk melakukan filter terhadap *List Announcement* berdasarkan aplikasi maupun *role* tertentu. Namun, dalam merealisasikan kebutuhan tersebut dibutuhkan operasi *group join* dari *table Announcement* ke lebih dari dua *table*. Muncul kekhawatiran terhadap *database load* pada halaman ini dikarenakan banyaknya jumlah data yang diproses, khususnya jika harus melakukan operasi *group join* terhadap lebih dari dua *table*.

## 3.5 Solusi atas Kendala yang Ditemukan

Selama pelaksanaan kerja magang ini ditemukan beberapa kendala ketika mengerjakan berbagai *project* yang sudah dilampirkan di atas. Namun, setiap kendala yang ditemukan harus disertai dengan sebuah solusi. Beberapa solusi atas kendala yang ditemukan tersebut adalah.

a. Limitasi operasi *group join* menggunakan *Linq*

Solusi dari kendala limitasi *Linq* adalah berdiskusi dengan *IT Architect* dan melakukan riset mengenai opsi selain menggunakan Linq. Dari kedua metode tersebut akhirnya ditemukan solusi dalam bentuk menggunakan *query builder* sehingga operasi *group join* tersebut dapat dijalankan pada sisi *back-end*.

b. *Database load concern*

Solusi dari kendala yang ditemukan adalah berdiskusi dengan *IT Architect* mengenai ide-ide untuk mengurangi waktu eksekusi *query*. Dari hasil *brainstorming* dengan *IT Architect* ditemukan solusi dengan cara

menyimpan *partial value* dalam bentuk *string* pada kolom *mapping* di *table Announcement* yang dapat diakses sebagai *placeholder* agar tidak melakukan operasi *group join* terhadap lebih dari dua *table* jika tidak dibutuhkan. *Placeholder* ini digunakan jika opsi *filter List Announcement* berdasarkan aplikasi maupun *role* tertentu tidak diaplikasikan.