TUGAS AKHIR - KI091391

Perancangan Pembuatan Perangkat Lunak Digital Signage untuk Layanan Informasi Seputar Kampus Teknik Informatika ITS dengan Menggunakan Media Kontroler Kinect

Ramadhani Tegar Perkasa
NRP 5110100 220

Dosen Pembimbing
Dwi Sunaryono, S.Kom, M.Kom
Ridho Rahman Hariadi, S.Kom, M.Sc

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2014

FINAL PROJECT – KI091391

Digital Signage *Software Development and Build Design for Information Service using Kinect Media Controller Case Studies: Informatics Department ITS*

Ramadhani Tegar Perkasa
NRP 5110100 220

Dosen Pembimbing
Dwi Sunaryono, S.Kom, M.Kom
Ridho Rahman Hariadi, S.Kom, M.Sc

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2014

Perancangan Pembuatan Perangkat Lunak Digital Signage untuk Layanan Informasi Seputar Kampus Teknik Informatika ITS dengan Menggunakan Media Kontroler Kinect

Nama Mahasiswa	: Ramadhani Tegar Perkasa
NRP	: 5110 100 220
Jurusan	: Teknik Informatika FTIf-ITS
Dosen Pembimbing 1	: Dwi Sunaryono, S.Kom, M.Kom
Dosen Pembimbing 2	: Ridho Rahman Hariadi, S.Kom, M.Sc.

ABSTRAKSI

Digital Singage adalah sebuah perangkat lunak yang dibangun guna memenuhi kebutuhan pelayanan informasi beserta pengelolaanya secara terintegrasi. Pada umumnya aplikasi ini diterapkan pada perusahan-perusahan besar di dunia. Seiring berjalanya waktu aplikasi ini semakin dikembangkan pada lingkungan-lingkungan non-bisnis seperti kepentingan sosial, maupun pusat pendidikan.Kinect adalah sebuah perangkat teknologi modern pengembangan dari motion sensing input devices yang dikembangan oleh Microsoft untuk penggunaan game konsol/pc, aplikasi, maupun riset. Dengan perpaduan Digital Signage dan Kinect sebagai media kontroler diharapkan penyampaian informasi dapat berjalan lebih inovatif, interaktif, dan lebih menarik.

.

Kata kunci: Digital Signage, *Interactive* Display, *Online* Digital Signage, ASP.Net, Vodigi *Services*.

Digital Signage Software Development and Build Design for Information Service Using Kinect Media Controller Case Studies: Informatics Department ITS

Student Name	: Ramadhani Tegar Perkasa
NRP	: 5110 100 220
Major	: Teknik Informatika FTIf-ITS
Advisor 1	: Dwi Sunaryono, S.Kom, M.Kom
Advisor 2	: Ridho Rahman Hariadi, S.Kom, M.Sc.

ABSTRACTION

Digital Singage is a software application for satisfiying the need of information services as well as integrated information management. This kind of application is used by most of big corporation. The growth of this type of aplication has reached more areas, such as social business or education sector. Kinect is a modern development device form motion sensing input devices, which is developed by Microsoft for gaming or research purposes.The combination between Digital Singage and kinect as motion sensing media is predicted to create more interesting, innovative, and interactive of information service.

Keyword : Digital Signage, *Interactive* Display, *Online* Digital Signage, ASP.Net, Vodigi *Services*

LEMBAR PENGESAHAN

Perancangan Pembuatan Perangkat Lunak Digital Signage untuk Layanan Informasi Seputar Kampus Teknik Informatika ITS dengan Menggunakan Media Kontroler Kinect

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Rekayasa Perangkat Lunak
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
Ramadhani Tegar Perkasa
NRP : 5110 100 220

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Dwi Sunaryono, S.Kom, M.Kom. .....
NIP: 19720528 199702 1 001 (pembimbing 1)

Ridho Rahman Hariadi, S.Kom, M.Sc. .....
NIP: 051100123 (pembimbing 2)

SURABAYA
JUNI 2014

KATA PENGANTAR

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul :

"*Perancangan Pembuatan Perangkat Lunak Digital Signage untuk Layanan Informasi Seputar Kampus Teknik Informatika ITS dengan Menggunakan Media Kontroler Kinect*"

Melalui lembar ini, penulis hanya ingin menyampaikan ucapan terima kasih dan penghormatan yang sebesar-besarnya kepada:

1. Bapak, Ibu, adik, kakak dan keluarga yang selalu memberikan dukungan penuh untuk menyelesaikan Tugas Akhir ini.
2. Bapak Dwi dan Bapak Ridho selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberikan petunjuk selama proses pengerjaan Tugas Akhir ini.
3. Bapak, Ibu dosen Jurusan Teknik Informatika ITS yang telah banyak memberikan ilmu dan bimbingan yang tak ternilai harganya bagi penulis.
4. Seluruh staf dan karyawan FTIf ITS yang banyak memberikan kelancaran administrasi akademik kepada penulis.
5. Teman-teman administrator Laboratorium Pemrograman 2 yaitu Fernandes, Agus Tri, Suliadi, Wawang, Anno, Dhea, Deva, Febri yang selalu mencairkan kejenuhan ketika penulis menyusun Tugas Akhir ini.
6. Teman-teman seperjuangan sejak menjalani masa-masa perkuliahan yaitu Galang, Fernandes, Almas, Azi, Dadang, Bagus, Hanif yang selalu berjuang bersama dan saling menyemangati.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1. Latar Belakang

Seiring berkembangnya zaman kebutuhan masyarakat terhadap sarana pelayanan informasi semakin meningkat. Bertambahnya nilai kebutuhan dalam mendapatkan pelayanan informasi ini dipicu oleh berkembangnya nilai produktivitas setiap orang di dunia. Bukan hanya dalam hal kemanfaatan dan keakuratan informasi namun juga pada hal keefisienan dan penggunaan yang lebih praktis adalah hal yang sangat perlu diperhatikan.

Minimnya waktu dan tingkat kesibukan yang tinggi di lingkungan kampus membuat seseorang cenderung kurang memperhatikan informasi yang bahkan terkadang bersifat penting. Ini semua tidak terlepas disebabkan oleh kurangnnya media layanan informasi yang dapat menyajikan semua informasi menjadi lebih cepat diakses dan lebih praktis digunakan.

Dengan hadirnya sebuah inovasi baru berupa layanan informasi Digital Signage yang didukung dengan Kinect sebagai media kontroler akan membawa revolusi layanan media informasi yang baru khususnya di kampus teknik informatika ITS.

Digital Signage adalah teknologi layanan informasi modern yang telah digunakan oleh instansi dengan kualitas menengah keatas, dengan paduan teknologi Kinect sebagai media kontrol yang dilengkapi dengan sensor gerakan untuk berinteraksi yang dinilai lebih praktis dan cepat, maka bukan tidak mungkin bagi semua warga kampus Teknik Informatika ITS untuk tidak lagi kehilangan informasi yang penting walau disela kesibukan dan rutinitas yang padat.

1.2. Tujuan

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

1. Membangun perangkat lunak Digital Signage yang disesuaikan dengan fitur-fitur layanan informasi..
2. Pengintegrasian kontrol sensor gerakan dengan menggunakan Kinect SDK..
3. Membangun layanan informasi sederhana yang terintegrasi serta disesuaikan dengan kebutuhan informasi seputar kampus Teknik Informatika ITS.

1.3. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana membuat perancangan perangkat lunak terintegrasi yang disesuaikan dengan display Digital Signage pada sisi *Client-Desktop*.
2. Bagaimana membuat perancangan perangkat lunak yang disesuaikan dengan Kinect sebagai media kontroler pada sisi *Client-Desktop*.
3. Bagaimana membangun layanan informasi ASP.Net dengan metode MVC yang disesusaikan dengan kebutuhan informasi seputar kampus Teknik Informatika ITS yang dapat diakses secara *online*.

1.4. Batasan Permasalahan

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, diantaranya sebagai berikut:

1. Berjalan pada sistem operasi windows 8 dan *Framework* .NET 4.5.

2. Aplikasi ini dicoba dengan perangkat keras Microsoft Kinect dan *Framework* Kinect SDK 1.8.
3. Menggunakan Bahasa C#, WPF, dan ASP.NET serta SQL*Server* sebagai pengolahan *database*.
4. *Server* yang digunakan adalah *server* lokal yang hanya bisa diakses di lingkungan kampus Teknik Informatika ITS.

1.5. Tujuan

Berikut dijelaskan tujuan dari tugas akhir ini sebagai berikut :

a) Membangun perangkat lunak Digital Signage yang disesuaikan dengan fitur-fitur layanan informasi berbasis *online*.
b) Membangun layanan informasi yang disesuaikan dengan kebutuhan informasi seputar Teknik Informatika ITS Surabaya.
c) Pengintegrasian kontrol sensor gerakan dengan menggunakan Kinect.

1.6. Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini yaitu:

1. Studi literatur

Dalam pembuatan tugas akhir ini telah dipelajari tentang hal-hal yang dibutuhkan sebagai ilmu, komponen, serta referensi penunjang dalam penyelesaiannya. Berikut subjek literatur yang dipelajari:

d) Merancang dan membangun perangkat lunak yang disesuaikan dengan layanan informasi Digital Signage.
e) Eksplorasi Microsoft Kinect SDK sebagai media

kontroler.

f) Pengintegrasian layanan informasi dengan *server* serta pengolahan *database*

2. Analisis dan Desain Perangkat Lunak

Pada tahap ini dilakukan analisa awal dan pendefinisian kebutuhan sistem untuk mengetahui permasalahan yang sedang dihadapi. Selanjutnya, dirumuskan rancangan *sistem* yang dapat memberi solusi terhadap permasalahan tersebut. Proses analisis dari perangkat lunak yang akan dibuat ini adalah dengan pemecahan masalah masalah yang telah dirumuskan dalam bab rumusan masalah, seperti:

a) Perancangan antarmuka serta alur pemakaian layanan informasi Digital Signage pada *Client*.
b) Perancangan integrasi pada *server* serta manajemen pengolahan data pada *database*.
c) Analisis dan perancangan pola gerakan dan *gesture* pada Microsoft Kinect SDK sebagai media kontroler.
d) Analisis kebutuhan *non*-fungsional (kemanan jaringan, *hardware*, serta instalasi dan *deploying* perangkat lunak)

.

3. Implementasi

Berikut beberapa hal yang diperlukan dalam implementasi perancangan pembangunan perangkat lunak:

a) Plat*form* pengembangan: *Desktop*.
b) Sistem Operasi*:* Windows 8
c) *IDE :* Visual Studio 2013, Blend for Visual Studio 2013
d) Bahasa Pemrograman *C#,* ASP.NET.
e) .Net *Framework* 4.5 dan Microsoft Kinect SDK 1.8
f) *Database :* SQL *Server* 2013

4. **Pengujian dan evaluasi**

   Pengujian aplikasi ini akan menggunakan *black box testing*, dimana *input* berupa data yang disuaikan oleh *form* atau bentuk masukkan dari perangkat lunak Digital Signage, dan akan dianalisis apakah hasil atau *output* yang dihasilkan memiliki hasil sesuai dengan alur program yang diharapkan. Dan juga mendeteksi apakah program dapat berjalan dengan seharusnya.

5. **Penyusunan buku Tugas Akhir**

   Pada tahap ini dilakukan pendokumentasian dan pelaporan dari seluruh konsep, dasar teori, implementasi, proses yang telah dilakukan, dan hasil-hasil yang telah didapatkan selama pengerjaan Tugas Akhir.

1.7. Sistematika Penulisan

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Selain itu, diharapkan dapat berguna bagi pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan Tugas Akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan Tugas Akhir.

Bab II Dasar Teori

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan Tugas Akhir ini.

Bab III Analisis dan Perancangan Sistem

Bab ini membahas mengenai perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan data, arsitektur, proses dan perancangan antarmuka pada Perangkat Lunak Digital Signage.

Bab IV Implementasi

Bab ini berisi implementasi dari perancangan perangkat lunak Digital Signage dan implementasi fitur-fitur penunjang Digital Signage.

Bab V Pengujian dan Evaluasi

Bab ini membahas pengujian dengan metode pengujian subjektif untuk mengetahui penilaian aspek kegunaan (*usability*) dari perangkat lunak dan pengujian fungsionalitas yang dibuat dengan memperhatikan fungsi program, kesesuaian kebutuhan, dan kemudahan pengoperasian serta kebutuhan *non*-fungsional berupa kecepatan per*form*a dan kemanan aplikasi setelah itu evaluasi terhadap hasil pengujian pada perangkat lunak Digital Signage.

Bab VI Kesimpulan

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan Tugas Akhir.

Lampiran

Merupakan bab tambahan yang berisi daftar istilah yang penting pada aplikasi ini.

BAB II
DASAR TEORI

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan Tugas Akhir. Teori-teori tersebut meliputi Digital Signage, MVC *Design Pattern*, MVVM *Design Pattern*, ASP.NET, IIS (*Internet Information Services*), SQL *Server*, dan Kinect SDK.

2.1. Digital Signage

Digital Signage adalah sebuah *electronic display* yang pada umumnya digunakan untuk menampilkan program televisi, informasi, layanan iklan, ataupun sebuah pesan. Teknologi Digital Signage dapat dijumpai di tempat-tempat umum atau fasilitas swasta (perusahaan, hotel, rumah makan, maupun pemerintahan). Pada dasarnya aplikasi Digital Signage yang baik dan modern adalah menggunakan konsep *client-server* dalam pembuatanya, sehingga dalam hal ini akan lebih efisien dan efektif dalam proses berjalanya aplikasi. Contoh Digital Signage secara umum dapat dilihat pada Gambar 2.1

Gambar 2.1 Rancangan Digital Signage *System Deployment*

Semenjak Digital Signage semakin sering digunakan dalam aspek teknologi informasi, maka seiring berjalanya waktu dikembangkan pula fitur-fitur yang ada pada aplikasi Digital Signage, diantaranya: *movement detection, image capture device, speech recognition,* dsb.

Digital Signage digunakan untuk berbagai tujuan dan dapat digunakan di banyak bidang, dibawah ini terdapat contoh jenis-jenis penggunaan secara umum Digital Signage :

1. *Public Information*

   Jenis penggunan aplikasi Digital Signage paling umum, dalam penggunaanya ditujukan dan dapat diakses oleh semua orang dan biasanya memberikan informasi-informasi seputar keadaan lingkungan sekitar. Contoh: berita harian, ramalan cuaca, info suhu, *tourism information*, dsb.

2. *Advertising*

   Jenis aplikasi Digital Signage yang bersifat komersial, dalam penggunaanya ditujukan dan dapat diakses oleh semua orang dan biasanya memb erikan informasi-informasi persuasif yang bertujuan untuk mengiklankan atau mempromosikan sebuah produk, jasa, perusahaan, maupun perorangan. Contoh: *space* iklan.

3. *IT Support*

   Digunakan untuk salah satu keperluan yang bersifat fungsionaliti dalam sebuah organisasi perusahaan, penggunaanya ditujukan pada orang-orang yang memiliki kebutuhan tertentu. Informasi yang ditampilkan biasanya dalam bentuk informasi yang disesuaikan dengan kebutuhan. Contoh: informasi tutorial penggunaan sebuah benda, informasi nomor antrian pada bank atau rumah sakit, dsb.

2.2. MVC *(Model-View-Controller) Design Pattern*

Design pattern Model-View-*Controller* (MVC) adalah basis arsitektur dari beberapa *framework web* application pada umumnya, seperti ASP.NET, Rails, STRUTS, dsb. *Pattern* pada MVC secara original pertama kali telah di implementasikan pada Smalltalk-80 yang dikembangkan pada IDE XEROX-PARC (Goldberg and Robinson, 1985). Seiring berjalanya waktu *Pattern* pada MVC semakin dikembangkan dengan konsep yang berbeda-beda.

2.2.1. MVC in Smalltalk

MVC *in Smalltalk* adalah konsep MVC klasik yang termasuk pertama kali diimplementasikan dalam pembangunan sebuah software dalam project.

Secara implementasi *design pattern* pada MVC *in Smalltalk* terbagi menjadi 3 komponen fungsionalitas terpenting, yaitu:

1. *Model component*

   Bertugas untuk mengenkasulapsi struktur dan fungsionalitas yang terspesifikasi. Pada dasarnya berisi *state* dari sebuah aplikasi atau operasi yang dapat mempengaruhi perubahan *state*. Juga untuk menjaga dependensi dari komponen *view* dan *controller*.

2. *View component*

   Komponen yang merepresentasikan sebuah informasi yang mana akan ditujukan pada pengguna. Pada umumnya ditampilkan pada *Graphical User Interface* (GUI). Tidak menutup kemungkinan terdapat banyak *view* dengan operasi yang berbeda dalam sebuah aplikasi. *View* juga dapat bersifat *Hierarchical*, dibangun dari elemen terkecil (*subview*). Setiap data yang terdapat pada *view*

berasal dari komponen *model* yang mana *view* dapat melakukan *query* secara langsung atau melalui *controller* untuk mendapatkan informasi yang akan ditampilkan.

3. *Controller component*

   Bertugas untuk merespon setiap *action* dari pengguna *via user interface*. Bertanggung jawab atas *passing transactions* dan *execution* atau *invoke* pada *model*. *Controller* berada pada *one-to-one correspondence* dengan *views*. Ketika *controller* menerima sebuah input maka akan mengaktifkan *subcontroller* terlebih dahulu, sehingga proses input tersebut akan ditangani oleh level terendah dari hierarchy terlebih dahulu.

2.2.2. MVC in *Web Framework*

ASP.Net MVC adalah salah satu *web framework* yang menggunakan konsep MVC *design architecture* yang berbasis *web form*. ASP.Net MVC *Framework* menggunakan *single handler HTTP request* yang akan menentukan dan menginstansiasi sebuah *controller* yang tepat untuk setiap *request. Controller* akan bertanggung jawab penuh dalam penanganan *request* yang didapat, mengatur transaksi dan meprosesnya melalui *model*, dan menyiapkan data untuk *subsequent* elemen *view*, lalu mengaktifasi elemen *view* untuk menghasilkan sebuah respon.

MVC pada *Web Framework* pada umumnya memiliki sebuah konsistensi dimana *Model – View – Controller* benar-benar berdiri secara terpisah. Dibawah ini perbedaan paling terlihat antara MVC Smalltalk dengan MVC *Web Framework* saat ini:

1. Tidak adanya *inherit view* ⟹ *model* dependensi (*Observer Pattern*)

   Komponen *model* pada *web* aplikasi tidak perlu melakukan *notify* pada elemen *view* untuk setiap perubahan *model*, namun *controller* menentukan *view*

behavior berdasarkan keluar masuknya data dari proses transaksi komponen *model*.

2. Tidak adanya 1-1 *view-controller correspondence*

   Pada original MVC sebelumnya setiap elemen *view* memiliki elemen *controller* yang unik yang mana secara tidak langsung mendefiniskan elemen *view* itu sendiri.

3. Menggunakan *Front Controller Pattern*

   Setiap elemen *controller* bertanggung jawab atas masuknya setiap HTTP *requests*, yang berdasarkan *requested* URL dan konfigurasi data.

4. *Model* tidak benar-benar didefiniskan secara jelas

   Hampir semua *web framework* yang ada pada saat ini memiliki definisi yang kurang jelas pada komponen *model*.

MVC *Web* yang ada saat ini adalah salah satu contoh yang menggambarkan perubahan evolusi dari sebuah konsep MVC *design pattern* yang mana telah di implementasikan pada *web-web framework* saat ini. Dibawah ini adalah karakteristik dari *model*, *view*, dan kontroler pada MVC *in web framework*:

1. MVC-*Web model component* secara umum menangani *application state*. Hal yang ditangani terdiri dari:
   - Data *persistence*: menangani *database* beserta *abstract database interface.*
   - *Transaction processing*: proses transaksi.
   - *External interface*: manajemen interaksi dengan agen eksternal, seperti *web service* dsb.
   - *Query handling*: melakukan aktivitas *query*.

2. MVC-*Web view component* merepresentasikan sebuah *user interface*, termasuk *data presentation* dan *input devices*. Hal yang ditangani terdiri dari:
    - Pengambilan dan penampilan informasi
    - *User Input* : penanganan setiap *action* atau *input* pada *user*.
    - *Client-side dynamic behavior*: menangani tampilan yang lebih bersifat dinamis, seperti JavaScript, Ajax, dsb)

3. MVC-*Web controller component* memiliki 3 hal pokok yang akan ditangani yaitu:
    - *Front controller*: menangani proses menerima request.
    - *Action handler*: menangani proses hasil request
    - *Control flow*: menangani prosesrespon hasil request. [1]

2.3. MVVM *(Model-View-ViewModel) Design Pattern*

MVVM adalah salah satu *design pattern* modern yang saat ini juga dikembangkan oleh Microsoft yang dikenal pada aplikasi WPF (*Windows Presentation Foundation*) dan Silverlight, namun juga masih bisa dijumpai pada *framework-framework* lain seperti: HTML5, AngularJS, KnockoutJS, VueJS, dan di java dikenal dengan ZK *Framework* yang memiliki konsep pattern tidak jauh beda yakni (*Model-View*-Binder).

Konsep MVVM membuat *developer* dapat membangun lapisan antarmuka melalui halaman atau *user control* atau keduanya dan bahkan juga dapat menuliskan semua *logical code* (*Event Handling, Initialization, Data Model*, dsb) pada kelas yang menempel pada komponen antarmuka tersebut. Sehingga secara tidak langsung dapat dilihat bahwa MVVM memiliki dependensi yang kuat antara UI (*User* Interface) dengan *Data Binding Logic* maupun *Business Operations*.

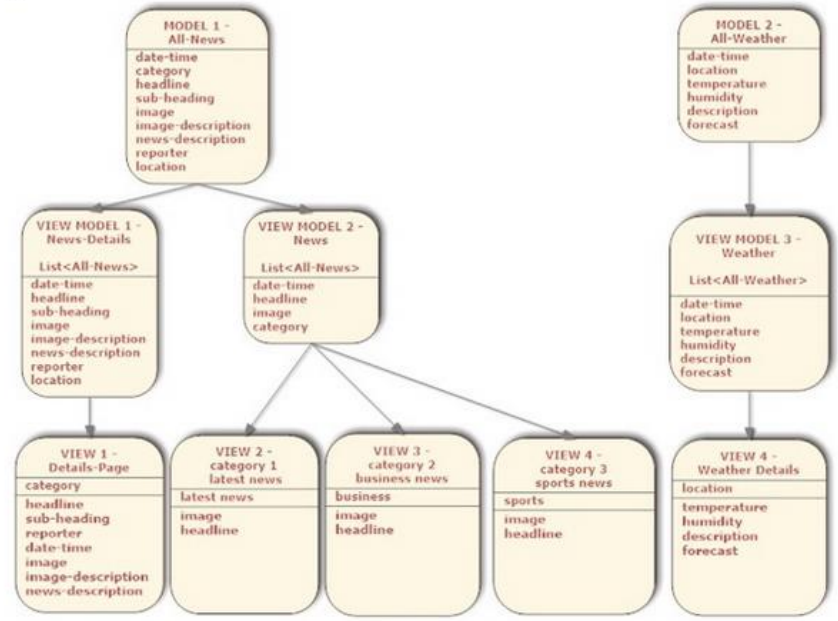
MVVM memiliki 3 pokok komponen utama dalam penerapanya diantaranya: [2]

- *View* (*User* Interface berupa XAML)
  View pada MVVM adalah *view* pada umumnya, tidak jauh berbeda dengan MVC. *View* pada MVVM bertugas untuk menangani *User* Interface seperti: tombol, label, dan elemen-elemen lainya.
- *Model*
  Juga tidak jauh pada konsep pada umumnya *Model* pada MVVM bertugas untuk menangani *Business Rule, Data Access, Model Class*, dsb.
- *ViewModel*
  ViewModel disebut juga dengan "*Model* of *View*" yang artinya *view* yang bersifat abstrak yang juga bertugas untuk menengani antara *view* dan *model* dengan menggunakan *view data binding.* Secara tidak langsung *ViewModel* juga berperan sebagai *controller*. [3]

2.3.1. MVVM in WPF (*Windows Presentation Foundation*)

WPF adalah subset dari .NET *framework* yang mana sebagian besar *library* nya berada dibawah System.Windows namespace. WPF dinilai sebagai salah satu *framework* yang *user*-friendly dan memiliki tatanan UX (*User* eXperience) yang baik. WPF dapat dibangun dengan Bahasa pemrograman C# atau Visual Basic.

Pada Gambar 2.2 menjelaskan salah satu penggunaan *viewModel* yang sederhana yakni menerjemahkan data *model* 1 dan data *model* 2 yang akan ditampilkan ke lapisan antarmuka. [4]



Gambar 2.2 Representasi MVVM *Design Pattern* pada aplikasi *News Reader*

Pada dasarnya sturktur pada MVVM Design Pattern tidak jauh berbeda dengan MVC namun dalam lapisan kontrolnya sedikit memiliki sifat yang berbeda. Pada MVVM kontrol dengan jelas akan menemple pada antarmuka dalam hal ini kontrol atau yang disebut *ViewModel* akan secara otomatis dibangun ketika adanya sebuah antarmuka yang dibuat.

Dibawah ini pada Kode Sumber 2.1, Kode Sumber 2.2, Kode Sumber 2.3 yang akan menjelaskan implementasi dasar dan sederhana dalam penerapan sebuah konsep MVVM Design Pattern dengan menggunakan WPF: [4]

```
public class Book
{
        public string Title { get; set; }
        public string Author { get; set; }
        public string Category { get; set; }
        public string Language { get; set; }
}
```

Kode Sumber 2.1 *Model*

```
public class MainPageView*Model* : BindableBase
    {
       private List<Book> books;
       public List<Book> Books
      {
           get
           {
            return books;
           }
           set
           {
            SetProperty(ref books, value);
           }
       public MainPageView*Model*()
       {
           Books = new List<Book>();
           Books.Add(new Book
               {
                   Title = "Harry Potter",
                   Author = "J. K. Rowling",
                   Category = "Young-adult fiction",
                   Language = "English"
               });
           Books.Add(new Book
           {
                   Title = "Written Lives",
                   Author = "Javier Marias",
                   Category = "Biography",
                   Language = "Spanish"
           });
    }
}
```

Kode Sumber 2.2 View*Model*

```
<TextBlock x:Name="bookTitle”
HorizontalAlignment="Left" TextWrapping="Wrap"
Text="{Binding Title}"
VerticalAlignment="Top"/>

<TextBlock x:Name="bookAuthor”
HorizontalAlignment="Left" TextWrapping="Wrap"
Text="{Binding Author}"
VerticalAlignment="Top" Margin="0,142,0,0"/>

<TextBlock x:Name="bookCategory”
HorizontalAlignment="Left" TextWrapping="Wrap"
Text="{Binding Category}"
VerticalAlignment="Top" Margin="0,242,0,0"/>

<TextBlock x:Name="bookLanguage”
HorizontalAlignment="Left" TextWrapping="Wrap"
Text="{Binding                    Language}"
VerticalAlignment="Top" Margin="0,342,0,0"/>
```

Kode Sumber 2.3 *View*

2.4. ASP.NET

ASP.NET adalah *Server-Side Web Application Framework* yang didesain untuk pembuatan atau pengembangan *web* yang memiliki halaman yang lebih dinamis. ASP.NET adalah salah satu *framework* yang dikembangan lanjut dari *framework* .NET oleh Microsoft.

Aplikasi ASP.NET dihostingkan ke sebuah *web server* yang aksesnya secara *default* menggunakan *stateless* HTTP *protocol*, jika aplikasi membutuhkan *stateful interaction* maka perlu mengimplementasikan *state management*. Secara konsep penanganan sebuah *state* pada ASP.NET sama seperti *state* pada GUI (*Graphical User Interface*). Pengaturan state pada ASP.NET dengan authentication membuat *Web* Scraping menjadi lebih sulit atau bahkan tidak mungkin dilakukan. *Session state* pada

ASP.NET dapat digunakan untuk menyimpan maupun mendapatkan *values* untuk *user* yang disesuaikan dengan navigasi pada setiap halaman dari *user* tersebut. Karena secara default ASP.NET menggunakan *stateless protocol* maka *web server* menangani tiap HTTP *reques*t pada setiap halaman secara independen.

Gambar 2.3 Arsitektur Dasar ASP.NET

Untuk menunjang pengembangan pada pembuatan aplikasi ASP.NET maka terdapat *file extension* yang *compatible* bahkan sering kali diasosiasikan dengan ASP.NET. Dibawah ini beberapa contoh *file extensions* yang akan paling sering digunakan dalam pembangunan aplikasi ASP.NET: [5]

- ASAX
  Global application file, biasanya digunakan untuk mendefinisikan variable-variabel yang bersifat sangat umum dan dapat diakses secara global.
- ASMX
  Default web services dari ASP.NET berisi tentang semua service yang dibutuhkan dalam project.

- SVC
  File web services jika menggunakan WCF (Windows Communication Foundation)
- ASPX
  Halaman *web form* pada ASP.NET yang berisi *web control, presentation*, dan *business logic*.
- Config
  Sebuah *file* XML yang berisi tentang konfigurasi mendasar pada ASP.NET ataupun tambahan seperti *connection string*, dsb.
- CSHTML
  View layer pada aplikasi ASP.NET. Adopsi dari penggabungan C# dan HTML yang menggunakan razor syntax.

2.4.1. ASMX *Web Services*

Web Service adalah metode untuk komunikasi antara dua buah alat elektronik. *World Wide Web Consortium* (W3C) mendeskripsikan *web service* sebagai suatu sistem perangkat lunak yang didesain untuk mendukung pengoperasian interaksi antar mesin di dalam sebuah jaringan.

ASMX adalah salah satu *web services* keluaran Microsoft yang cukup sering digunakan selain WCF (Windows Communicartion Foundation). ASMX pada umumnya menempel pada aplikasi *web* berbasis ASP.NET dan akan dibuat jika aplikasi tersebut membutuhkan *web services* atau dalam hal distribusi *function, method*, ataupun API. Sedangkan WCF adalah *web services framework* yang bersifat *service-oriented application* dan dapat berdiri sendiri. [6]

Berikut dapat dilihat pada Tabel 2.1 Perbedaan ASMX *Web Services* dan WCF *Web Services* beberapa perbedaan antara ASMX dan WCF:

Tabel 2.1 Perbedaan ASMX *Web Services* dan WCF *Web Services*

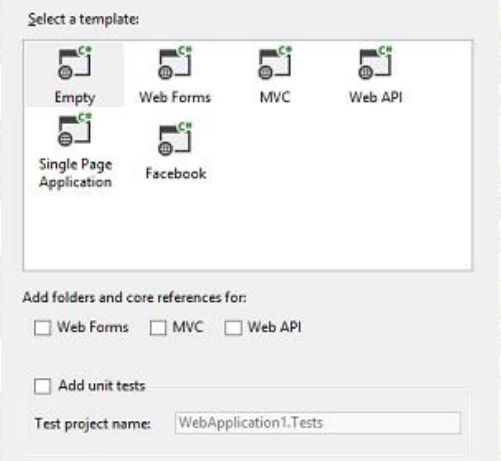
Fitur	ASMX	WCF
Hosting	IIS atau ASP.NET (.asmx) SoapReceivers	IIS atau ASP.NET (.svc) ServiceHost<T> WAS
Programming	[*Web*Service], [*Web*Method], and so on (supports interfaces, generics, and the like)	[ServiceContract], [OperationContract], and so on (supports interfaces, generics, and so on)
Behaviors (enabled via attributes or configuration)	Local DTC transactions	Concurrency
	HTTP buffering	Instancing
	HTTP caching	Thread-binding
	HTTP sessions	Session management
	Custom (via SoapExtensions, WSE filters)	Exception handling and faults

2.4.2. MVC 5 Project *Template*

Salah satu *project template* untuk pembangunan aplikasi berbasis *web* dari ASP.NET yang dirancang sesuai dengan kebutuhan MVC Design Pattern. Secara otomatis *template* tersebut akan menyiapkan keperluan-keperluan pembangunan aplikasi berbasis *web* yang menggunakan metode MVC.

MVC 5 adalah produk *project template* terbaru dari Visual Studio untuk saat ini. MVC 5 juga memiliki fitur-fitur baru tambahan atau fitur-fitur lama yang telah di-*update*. Dibawah ini fitur-fitur MVC 5 yang telah di-*update* Microsoft, yaitu: [7]

- ASP.Net *Identity*
  Developer dapat melakukan kustomisasi dari MVC Project untuk melakukan konfigurasi autentikasi. Seperti contoh dapat menggunakan autentikasi dari Facebook, Google, dan beberapa macam API lainya.
- Bootstrap
  Dilengkapi dengan fitur tambahan Bootstrap untuk pengembangan tampilan yang lebih baik.
- Aunthentication *Filters*
  Dapat melakukan filter autentikasi contoh seperti melakukan spesifikasi autentikasi *logic* per-*action*, per-*controller*, atau global untuk semua *controller*.
- *Attribute Routing*
  Support dengan fitur *attribute routing*, dengan adanya *attribute routing developer* dapat menspesifikasikan *route* yang diingingkan.



Gambar 2.4 Project *Template* yang tersedia pada ASP.NET Visual Studio 2013

2.5. IIS (Internet In*form*ation *Services*)

IIS adalah sebuah kumpulan atau wadah untuk sebuah *internet server* (*Web*, HTPS, FTPS, dsb) keluaran microsoft. IIS juga digunakan untuk membangun dan sebagai tempat pengaturan maupun konfigurasi *internet server*. IIS dikenal sebagai salah satu layanan *services* (*Web Server*) yang handal, mudah penggunaanya, *high performance*, dan juga aman. Dibawah ini adalah fitur-fitur pada IIS pada umumnya: [8]

- Terintegrasi dengan Windows NT secara penuh (system keamanan, *auditing*, dan izin akses NTFS)
- Mendukung penuh protokol HTTP, FTP, NNTP, SSL, namun terbatas untuk protokol SMTP.
- Dapat digunakan sebagai *platform* dimana aplikasi *web* berjalan seperti: Active *Server* Pages (ASP), ASP.Net, Internet *Server* API (ISAPI), Common Gateway Interface (CGI), beberapa Bahasa *script* yang dapat diinstalasikan seperti PERL atau PHP, dan sebagainya.
- Mengizinkan aplikasi *web* untuk dijalankan sebagai proses yang terisolasi dalam ruangan memori yang terpisah untuk mencegah satu apliaksi yang dapat mengakibatkan *crash* aplikasi lain.
- Dapat diatur dengan beberapa aplikasi penunjang seperti: Microsoft Management Console, via *Web browser*. Ataupun juga dapat menggunakan *script* yakni Windows Scripting Host, dan sebagainya.
- Bandwidth Throttling yakni dapat mencegah sebuah situs *web* memonopoli *bandwidth* yang tersedia. [9]

2.6. SQL *Server* 2014

SQL *server* adalah relasional RDBMS (*Relational Database Management System*) keluaran micosoft yang juga dapat

digunakan sebagai analisis sistem dalam bidang *e-commerce, line-of-business, warehousi*ng solutions, dsb. SQL *Server* juga dapat digunakan dalam komunikasi jaringan menggunakan protokol TDS (*Tabular Data Stream*) selain itu juga mendukung ODBC (*Open Database Connectivity*). Salah satu kelebihan dari SQL *Server* adalah dapat membuat basis data *mirroring* dan *clustering*. [10]

2.7. Kinect SDK

Kinect adalah sebuah perangkat teknologi modern pengembangan dari motion *sensing input devices* yang dikembangan oleh Microsoft untuk penggunaan game konsol atau pc, aplikasi, maupun riset.

Kinect memiliki sensor batang horizontal yang terhubung dengan alas kecil yang memiliki poros yang dapat berputar. Perangkat ini memiliki kamera RGB, sensor kedalaman infrared laser yang dikombinasikan dengan *monochrome* CMOS sensor, dan mikrofon. Sehingga teknologi Kinect ini mampu menyediakan fitur *full body 3D motion capture, facial recognition,* dan *voice recognition.* Namun untuk saat ini *voice recognition* hanya mampu menerjemahkan dalam beberapa Bahasa saja (UK , US , Japan).

Kinect *Software Development Kid* (SDK) adalah *library* paling penting dalam pembuatan tugas akhir kali ini khususnya untuk elemen *view* pada sisi *client*. SDK ini salah satu *tools* terpenting dalam pembuatan aplikasi yang memerlukan penggunaan Kinect. [11]

2.8. Vodigi *Services*

Vodigi adalah sebuah *framework* yang didesain untuk pembangunan Digital Signage. Dalam aplikasi ini Vodigi *Services* berperan sebagai modul eksternal pada yang digunakan sebagai *template* pada perancangan maupun desain pengiriman data.

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas tahap analisis permasalahan dan perancangan Tugas Akhir. Analisis permasalahan membahas permasalahan yang yang diangkat dalam pengerjaan Tugas Akhir. Solusi yang ditawarkan oleh penulis juga dicantumkan pada tahap permasalahan analisis ini. Analisis kebutuhan mencantumkan kebutuhan-kebutuhan yang diperlukan perangkat lunak. Selanjutnya dibahas mengenai perancangan sistem yang dibuat. Pendekatan yang dibuat dalam perancangan ini adalah pendekatan berorientasi objek. Perancangan direpresentasikan dengan diagram UML (*Unified Modelling Language*).

3.1. Analisis

Tahap analisis dibagi menjadi beberapa bagian antara lain cakupan permasalahan, deskripsi umum sistem, kasus penggunaan sistem, dan kebutuhan perangkat lunak.

3.1.1. Cakupan Permasalahan

Permasalahan utama yang diangkat dalam pembuatan Tugas Akhir ini adalah bagaimana membuat perancangan perangkat lunak yang disesuaikan dengan display Digital Signage sebagai *view layer*. Permasalahan yang kedua bagaimana membuat perancangan perangkat lunak berbasis *desktop* dengan metode MVVM *Design Pattern* yang disesuaikan dengan dengan Kinect sebagai media kontroler. Permasalahan yang ketiga bagaimana membuat dan menggunakan *gesture* di dalam fitur Kinect dalam mengembangkan perangkat lunak. Dan permasalahan keempat adalah membangun layanan informasi dalam bentuk aplikasi *web* menggunakan ASP.Net dengan metode MVC *Design Pattern* yang disesusaikan dengan kebutuhan informasi seputar kampus Teknik Informatika ITS yang dapat diakses secara *online*.

3.1.2. Deskripsi Umum Sistem

Digital Signage adalah sebuah *electronic display* yang pada umumnya digunakan untuk menampilkan program televisi, informasi, layanan iklan, ataupun sebuah pesan. Sedangkan aplikasi Digital Signage pada tugas akhir kali ini adalah sebuah perangkat lunak yang memiliki sistem lebih terpadu dan terintegrasi yang mana memiliki konsep *server-client* sehingga dapat digunakan maupun dioperasikan secara *online*. Untuk memodifikasi aplikasi ini agar lebih interaktif dan menarik maka pada *view layer* di sisi *display client* dilengkapi dengan fitur Kinect.

Aplikasi ini didesain agar aplikasi terintegrasi sehingga dalam proses pembangunanya menggunakan *services* yang mana *services* tersebut harus dapat diakses dan digunakan oleh 2 aplikasi *client* yang ada yakni untuk aplikasi layanan informasi *web* dan aplikasi *client-desktop*. Untuk mempermudah dalam hal *compatibility* perangkat lunak pada tugas akhir kali ini menggunakan *framework* maupun *library* di bawah naungan Microsoft.

Aplikasi layanan informasi *web* pada sistem yang dibangun ini menggunakan *framework* ASP.NET memiliki konsep MVC *Design pattern* dan *Object Oriented Programming* yang lebih bagus dan rapi. Secara fungsionalitas disesuaikan dengan kebutuhan administrasi pada perangkat lunak Digital Signage yang disesuaikan dengan kebutuhan pada kampus Teknik Informatika ITS Surabaya.

Aplikasi *client-desktop* pada sistem yang dibangun ini menggunakan *framework* WPF (*Windows Presentation Foundation*) memiliki konsep MVVM *Design Pattern.* Secara implementasi aplikasi ini menggunakan `userControl` sebagai *sub-view* untuk setiap halaman yang akan ditampilkan beserta penanganan *controller*nya. Pada aplikasi *client-desktop* ini juga telah terintegrasi dengan Kinect sebagai penunjang interaksi *user*.

3.1.3. Spesifikasi Kebutuhan Perangkat Lunak

Bagian ini berisi semua kebutuhan perangkat lunak yang diuraikan secara rinci dalam bentuk diagram kasus, diagram urutan, dan diagram aktivitas. Masing-masing diagram menjelaskan perilaku atau sifat dari sistem ini. Kebutuhan perangkat lunak dalam sistem ini mencakup kebutuhan fungsional saja. Pada bab ini juga dijelaskan tentang spesifikasi terperinci pada masing-masing kebutuhan fungsional. Rincian spesifikasi dari kasus penggunaan disajikan dalam bentuk tabel.

3.1.3.1. Kebutuhan Fungsional

Kebutuhan fungsional berisi kebutuhan utama yang harus dipenuhi oleh sistem agar dapat bekerja dengan baik. Kebutuhan fungsional mendefinisikan layanan yang harus disediakan oleh sistem, bagaimana reaksi terhadap masukan, dan apa yang harus dilakukan sistem pada situasi khusus. Daftar kebutuhan fungsional dapat dilihat pada **Tabel 3.1**.

Tabel 3.1 Daftar Kebutuhan Fungsional Perangkat Lunak

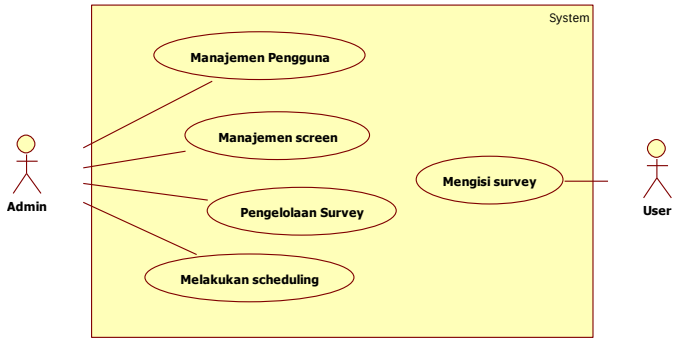
Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
TA-F0001	Manajemen akun	Admin dapat manajemen akun pada halaman aplikasi *web.*
TA-F0002	Manajemen *screen*	Admin dapat melakukan manajemen pada tampilan *screen* yang akan ditayangkan pada halaman aplikasi *web.*
TA-F0003	Membuat *survey*	Admin dapat membuat *survey* untuk diberikan pada *user* pada halaman aplikasi *web*.
TA-F0004	Melakukan *scheduling*	Admin dapat melakukan *scheduling* untuk menentukan waktu *screen* yang akan ditampilkan pada halaman aplikasi *web*
TA-F0005	Mengisi *survey*	*User* dapat mengisi *survey* yang terdapat aplikasi *Client-Desktop*

3.1.4. Aktor

Aktor mendefinisikan entitas-entitas yang terlibat dan berinteraksi langsung dengan sistem. Entitas ini bisa berupa manusia maupun sistem atau perangkat lunak yang lain. Penulis mendefinisikan aktor untuk sistem ini yaitu perancang perangkat lunak, dan pengembang perangkat lunak yang menggunakan *Web* dan aplikasi *desktop*. Karena aplikasi ini terdiri dari 2 aplikasi *client* maka setiap aplikasi disesuaikan dengan kegunaanya masing-masing, yakni aplikasi *web* untuk admin dan aplikasi *desktop* untuk *user* umum.

3.1.5. Kasus Penggunaan

Kasus-kasus penggunaan dalam sistem ini akan dijelaskan secara rinci pada subbab ini. Kasus penggunaan secara umum akan digambarkan oleh salah satu *model* UML, yaitu diagram kasus penggunaan. Rincian kasus penggunaan berisi spesifikasi kasus penggunaan, diagram aktivitas, dan diagram urutan untuk masing-masing kasus penggunaan. Diagram kasus penggunaan dapat dilihat pada Gambar 3.1. Daftar kode diagram kasus penggunaan sistem dapat dilihat pada Gambar 3.1**Error! Reference source not found.**.



Gambar 3.1 Diagram Kasus Penggunaan Sistem

Tabel 3.2 Daftar Kode Diagram Kasus Penggunaan

Kode Kasus Penggunaan	Nama
TA-UC0001	Manajemen akun
TA-UC0002	Manajemen *screen*
TA-UC0003	Pengelolaan *survey*
TA-UC0004	Melakukan *scheduling*
TA-UC0005	Mengisi *survey*

3.1.5.1. Manajemen Akun

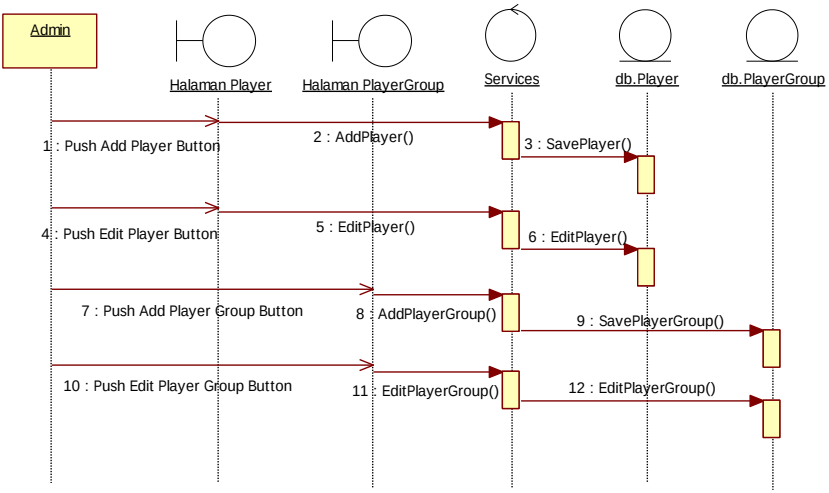
Pada kasus penggunaan ini, sistem akan menerima perintah dari pengguna admin untuk melakukan manajemen terhadap akun admin sendiri maupun manajemen akun dalam sisi *client* (*player*) diantaranya buat akun, update akun, dan hapus akun serta mengelompokkan *user* (untuk *client*).

Terdapat 2 aktor dalam kasus penggunaan ini yaitu admin dan super admin perbedaan hanya pada hak akses, pada super admin memiliki hak akses dalam pengelolaan admin pada sisi *web*. berada Spesifikasi kasus penggunaan ini dapat dilihat pada Tabel 3.3. Diagram aktivitas dan diagram urutan dari kasus penggunaan ini bisa dilihat pada Gambar 3.2 dan Gambar 3.3.

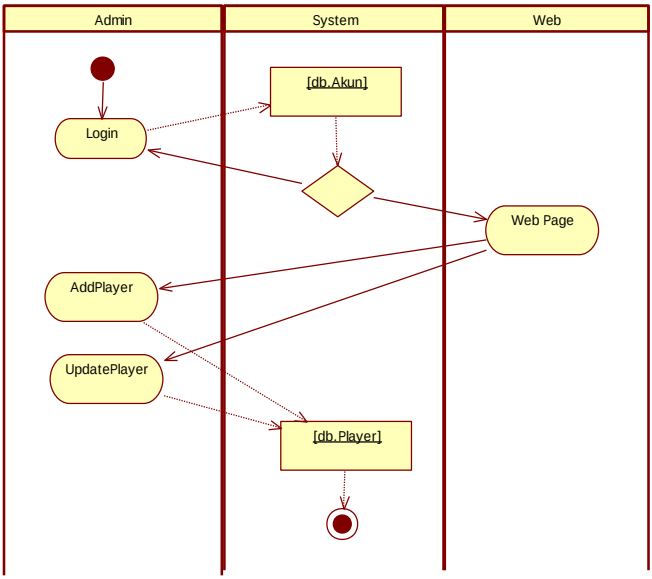
Tabel 3.3 Spesifikasi Kasus Penggunaan Manajemen akun

Nama	Manajemen akun
Kode	TA-UC0001
Deskripsi	melakukan manajemen *user* diantaranya buat akun, update akun, dan hapus akun serta mengelompokkan *user*.
Tipe	Fungsional

Pemicu	Pengguna masuk ke dalam menu player pada halaman *web*
Aktor	Admin, super admin.
Kondisi Awal	Belum atau sudah terdapat *User*
Aliran:	
-Kejadian Normal	1. Admin melakukan login 2. Admin masuk ke halaman Player 3. Admin klik pada hyperlink "Add Player" 4. Admin mengisi informasi Player 5. Admin menekan *save* jika proses telah selesai
-Kejadian Alternatif	2.A. Pengguna klik tombol *edit* 2.A.2 Admin dapat meng*edit* informasi player
Kondisi Akhir	Admin menyimpan informasi *user* terbaru ke dalam *database*.
Kebutuhan Khusus	Tidak ada



Gambar 3.2 Diagram Urutan Melakukan Manajemen Pengguna.



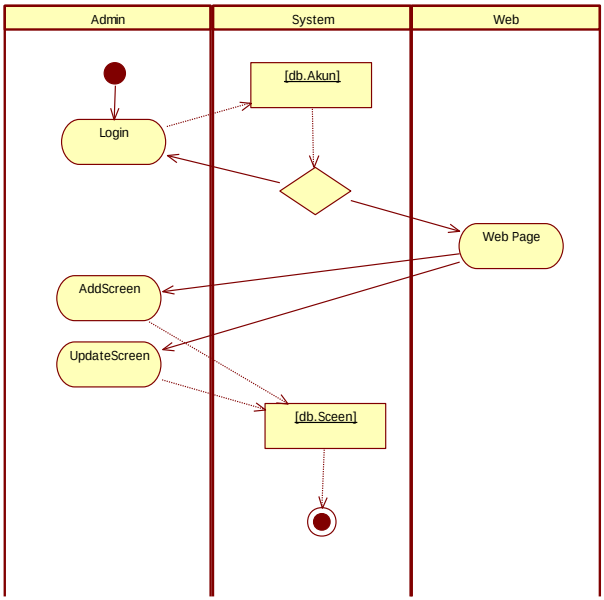
Gambar 3.3 Diagram Aktivitas Manajemen Akun

3.1.5.2. Manajemen *Screen*

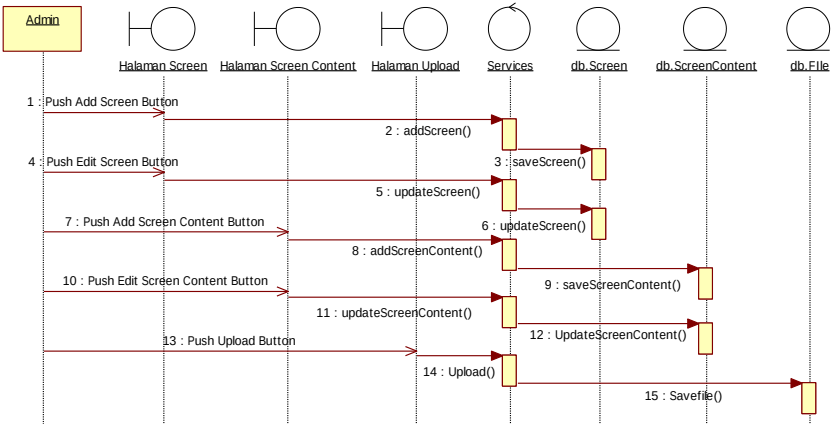
Pada kasus penggunaan ini, sistem akan menerima perintah dari pengguna admin untuk melakukan manajemen *screen.* Menentukan tampilan *screen* yang diinginkan. Terdapat 2 tampilan *screen* yang bisa ditampilkan yakni *playlist* video dan *slideshow image*. Sedangkan untuk membuat aplikasi lebih interaktif teradapat konten yang juga menempel pada tampilan utama. Untuk konten terdapat 3 macam tipe yaitu *image*, video, dan *web*. Spesifikasi kasus penggunaan ini dapat dilihat pada Tabel 3.4. Diagram aktivitas dan diagram urutan dari kasus penggunaan ini bisa dilihat pada Gambar 3.4 dan Gambar 3.5.

Tabel 3.4 Spesifikasi Kasus Penggunaan Manajamen *Screen*

Nama	Manajemen *Screen*
Kode	TA-UC0002
Deskripsi	Melakukan manajemen *screen*. Menentukan *screen* utama, konten *screen* beserta tipe konten yang diinginkan.
Tipe	Fungsional
Pemicu	Pengguna masuk ke halaman *screen* pada halaman *web*
Aktor	Pengguna
Kondisi Awal	Belum atau sudah terdapat daftar *screen* yang tersedia
Aliran:	
- **Kejadian Normal**	1. Admin melakukan login 2. Admin masuk ke halaman *Screen* 3. Admin klik pada hyperlink "Add Screen" 4. Admin mengisi tampilan *screen* yang dinginkan, dengan mengambil dari *playlist* atau *slideshow* yang telah tersedia 5. Admin menekan *save* jika proses telah selesai
- **Kejadian Alternatif**	2.A Admin mengunggah konten 2.B Admin membuat *playlist* atau *slideshow* dengan mengunggah video atau *image* terlebih dahulu. 2.C Admin dapat meng-*edit screen* maupun konten *screen* yang telah dibuat.
Kondisi Akhir	Admin menyimpan *screen* dan konten *screen* terbaru ke dalam *database*.
Kebutuhan Khusus	Tidak ada



Gambar 3.4 Diagram Aktivitas Melakukan Manajemen *Screen*



Gambar 3.5 Diagram Urutan Melakukan Manajemen *Screen*

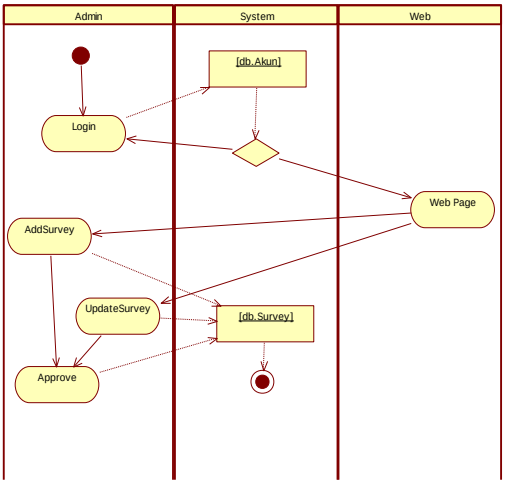
3.1.5.3. Membuat *Survey* dan Melihat *Report*

Admin dapat membuat dan mendesain *survey* sesuai dengan kebutuhan. Pertanyaan dapat dibuat oleh admin dan bentuk cara menjawab *survey* dengan *multiple choice* yang juga dapat disesuaikan dengan kebutuhan. Spesifikasi kasus penggunaan ini dapat dilihat pada Tabel 3.5. Diagram aktivitas dan diagram urutan dari kasus penggunaan ini bisa dilihat pada Gambar 3.6 dan Gambar 3.7.

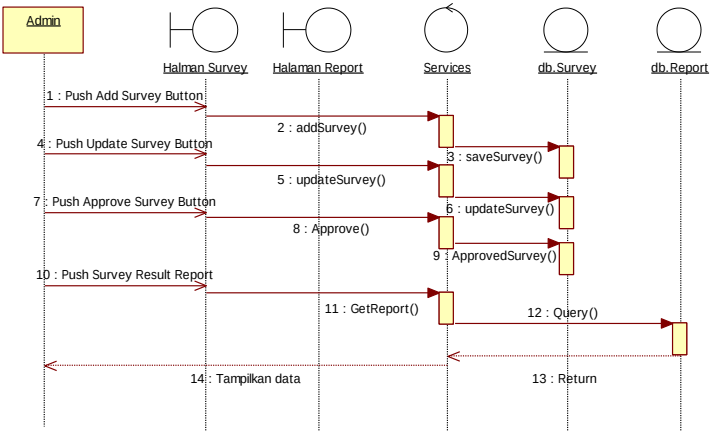
Tabel 3.5 Spesifikasi Kasus Penggunaan Membuat *Survey*

Nama	Membuat *Survey*
Kode	TA-UC0003
Deskripsi	Admin dapat membuat *survey* sesuai dengan kebutuhan.
Tipe	Fungsional
Pemicu	Pengguna masuk ke halaman *survey* pada halaman *web*
Aktor	Pengguna
Kondisi Awal	Belum atau sudah terdapat daftar *survey* yang tersedia
Aliran:	
- **Kejadian Normal**	1. Admin melakukan login 2. Admin masuk ke halaman *Survey* 3. Admin klik pada hyperlink "Add *Survey*" 4. Admin mengisi *survey* (pertanyaan, jenis pertanyaan, serta *approvement*) 5. Admin menekan *save* jika proses telah selesai
- **Kejadian Alternatif**	2.A.. Admin dapat melihat hasil *report* dari *survey* yang telah teerjawab
Kondisi Akhir	*Survey* akan tersimpan ke dalam *database*

Kebutuhan Khusus	Tidak ada



Gambar 3.6 Diagram Aktivitas Membuat *Survey* dan Melihat *Report*



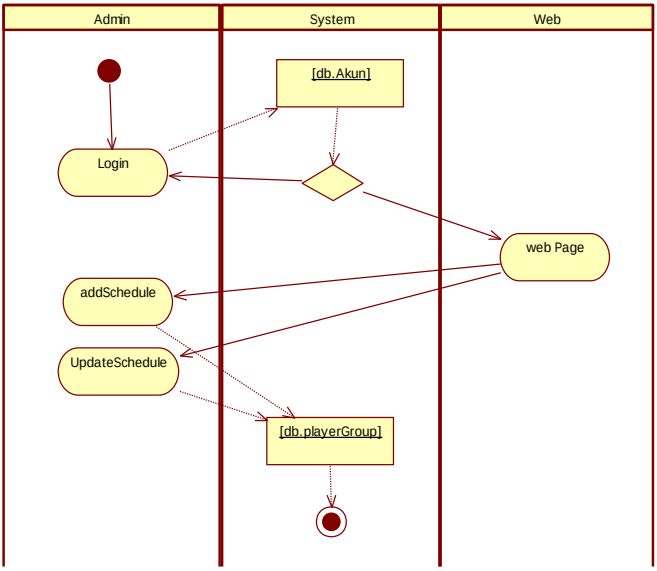
Gambar 3.7 Diagram Urutan Pengelolaan *Report*

3.1.5.4. Melakukan *Scheduling*

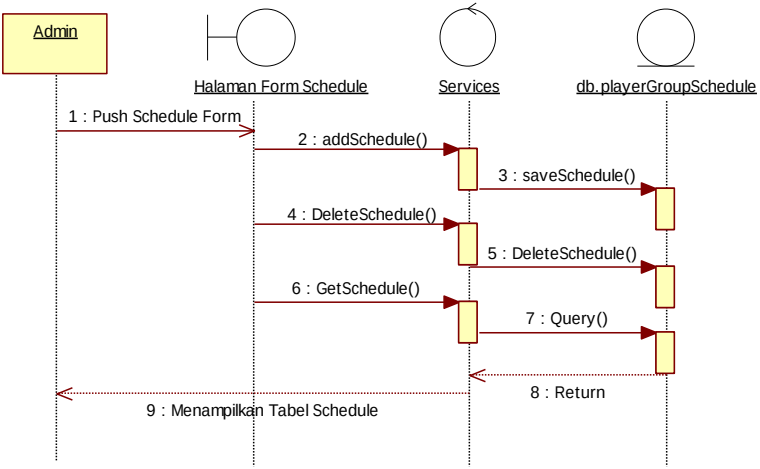
Admin dapat melakukan pengaturan *schedule* pada setiap player. *Scheduling* yang dilakukan berdasarkan hari dan jam, yang nantinya secara otomatis akan disimpan ke dalam *database* dan akan diproses pada *user* sesuai dengan jadwal yang telah tercantum. Spesifikasi kasus penggunaan ini dapat dilihat pada Tabel 3.6 Diagram aktivitas dan diagram urutan dari kasus penggunaan ini bisa dilihat pada Gambar 3.8 dan Gambar 3.9.

Tabel 3.6 Spesifikasi Kasus Penggunaan Melakukan *Scheduling*

Nama	Melakukan *Scheduling*
Kode	TA-UC0004
Deskripsi	Admin dapat melakukan pengaturan *schedule* pada setiap player. *Scheduling* yang dilakukan berdasarkan hari dan jam.
Tipe	Fungsional
Pemicu	Pengguna masuk ke halaman player group pada halaman *web* lalu pilih *edit schedule*
Aktor	Pengguna
Kondisi Awal	Belum atau sudah terisi daftar *schedule* yang tersedia
Aliran:	
- Kejadian Normal	1. Admin melakukan login 2. Admin masuk ke halaman Player Group 3. Admin klik pada hyperlink "*Schedule*" 4. Admin mengisi waktu *schedule* yang diinginkan (*form* hari dan jam) 5. Admin menekan *save* jika proses telah selesai
- Kejadian Alternatif	Tidak ada
Kondisi Akhir	*Schedule* terbaru akan tersimpan ke dalam *database*
Kebutuhan Khusus	Tidak ada



Gambar 3.8 Diagram Aktivitas untuk Melakukan *Scheduling*



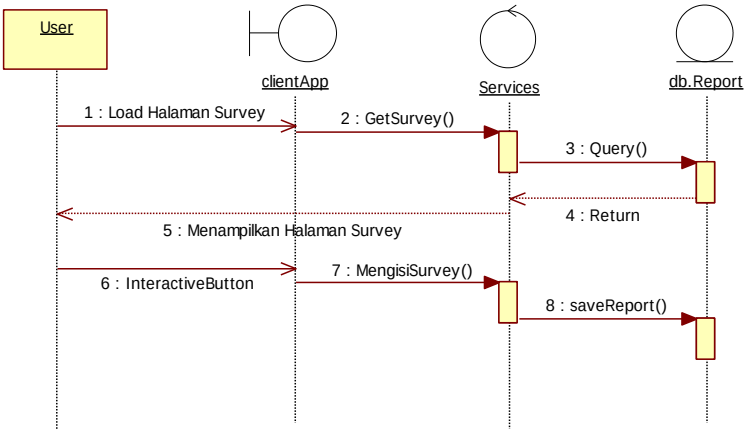
Gambar 3.9 Diagram Urutan untuk Melakukan *Scheduling*

3.1.5.5. Mengisi *Survey*

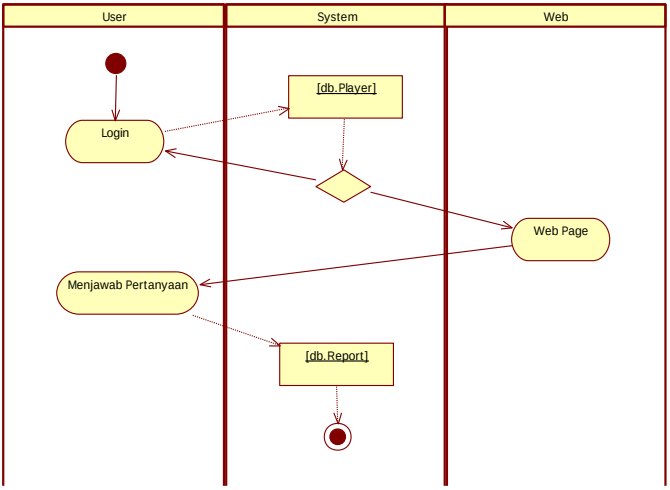
Mengisi *survey* tidak dilakukan pada halaman *web* dan tidak dilakukan oleh admin. Melainkan diisi dari aplikasi *client-desktop* yang akan diisi oleh setiap *user*. Setiap *user* dapat mengisi banyak *survey* yang artimya semua pengguna layanan Digital Signage dapat melakukan pengisian *survey* tersebut. Spesifikasi kasus penggunaan ini dapat dilihat pada. Tabel 3.7 Diagram urutan dan diagram aktivitas dari kasus penggunaan ini bisa dilihat pada Gambar 3.10 dan Gambar 3.10 Diagram Aktivitas Mengisi *Survey*.

Tabel 3.7 Spesifikasi Kasus Penggunaan Mengisi *Survey*

Nama	Mengisi *Survey*
Kode	TA-UC0005
Deskripsi	*User* dapat mengisi *survey* melalui aplikasi *client-desktop*
Tipe	Fungsional
Pemicu	Pengguna menekan tombol *survey* pada *screen content* yang tersedia
Aktor	Pengguna
Kondisi Awal	*Survey* telah disiapkan pada *client-desktop*
Aliran:	
-Kejadian Normal	1. *User* melakukan login 2. *User* memilih memilih *survey* pada *screen content* yang tersedia. 3. *User* mengisi *survey*.
-Kejadian Alternatif	Tidak ada
Kondisi Akhir	Sistem akan menyimpan hasil *survey* ke dalam *report database*
Kebutuhan Khusus	Tidak ada



Gambar 3.10 Diagram Aktivitas Mengisi *Survey*



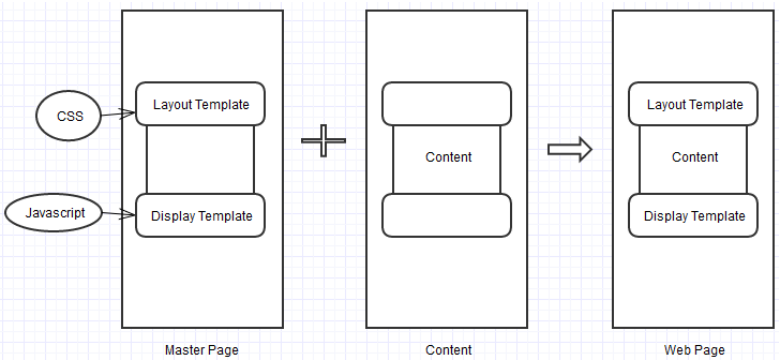
Gambar 3.11 Diagram Urutan Mengisi *Survey*

3.2. Perancangan Sistem Aplikasi *Web*

Penjelasan tahap perancangan perangkat lunak dibagi menjadi beberapa bagian yaitu perancangan diagram kelas, perancangan proses analisis, dan perancangan antarmuka. Untuk bab ini akan dijelaskan perancangan sistem pada aplikasi *Web*. *Model* yang digunakan dalam arsitektur ini ialah *model* tiga-lapis. Tiga lapisan pada arsitektur ini terdiri dari lapisan *Model, View,* dan *Controller* Lapisan *Controller* merupakan penghubung antara lapisan *Model* dengan *View.*

3.2.1. Perancangan Lapisan Antarmuka

dapat dilihat pada 3.1.2. Adalah konsep perancangan pada lapisan antarmuka di aplikasi *web* ini menggunakan Master Page, yang mana terdapat *page template* yang memiliki beberapa bentuk *form*at sama dan tidak berubah yang digunakan untuk semua page yang ada. Terdapat tiga kelas penyusun lapisan ini.



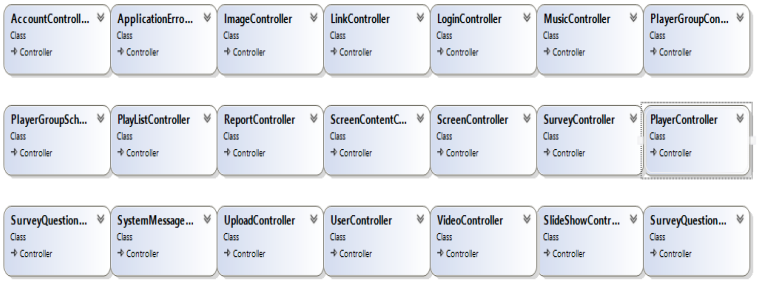
Gambar 3.12 Master Page yang Digunakan pada Aplikasi *Web*

Untuk Kelas-Kelas yang berfungsi sebagai *master page* terdapat pada folder ~/view/shared. Yang didalamnya terdapat kelas `_Layout.cshtml`. Sedangkan untuk kelas *view* lainya terdapat sub-sub halaman pada folder ~/view/ untuk setiap halaman misal: Login, Screen, Survey, Player Group, Player, Playlist,

Slideshow, Report, dsb. Untuk setiap *view* halaman terdapat 1 halaman index.cshtml yang mana digunakan halaman utama dari *sub-sub* halaman yang tersedia.

3.2.2. Perancangan Lapisan Kontrol

Kelas lapisan kontrol pada aplikasi *web* ini menunjukkan kelas-kelas penyusun lapisan kontrol. Kelas-kelas penyusun lapisan ini, yaitu berada pada folder '*controller*' yang telah disiapkan oleh *template* MVC 5. Lapisan kontroler bertugas untuk menghubungkan lapisan antarmuka dan data serta fungsionalitas-fungsionalitas khusus pada halaman sesuai dengan kebutuhan.



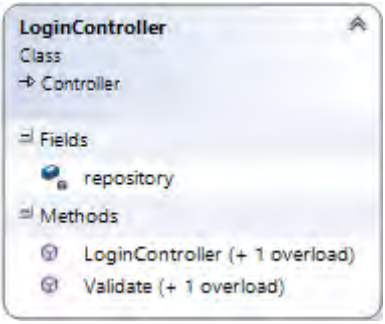
Gambar 3.11 Kelas Diagram Lapisan Kontrol

Sesuai dengan konsep MVC 5, folder-folder tersebut berisi semua *controller* yang dibutuhkan oleh beberapa *view*. *Controller* dan *view* tersebut pun saling berhubungan karena telah terkoneksi satu sama lain oleh *template* MVC 5. Pada subbab ini akan dijelaskan lebih detail untuk beberapa lapisan kontrol yang terpenting pada aplikasi *web* ini.

3.2.2.1. Perancangan Kelas `LoginController`

Seperti pada umumnya login, halaman ini khusus digunakan untuk menangani login dari admin. Pada Gambar 3.12

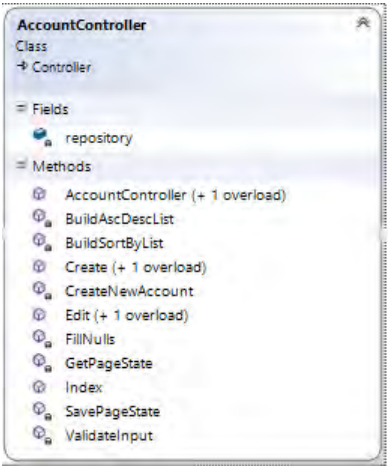
Terdapat *method* validate untuk melakukan proses validate *user* dan password dari hasil *form* login.



Gambar 3.12 Rancangan Kelas Kontroler Login

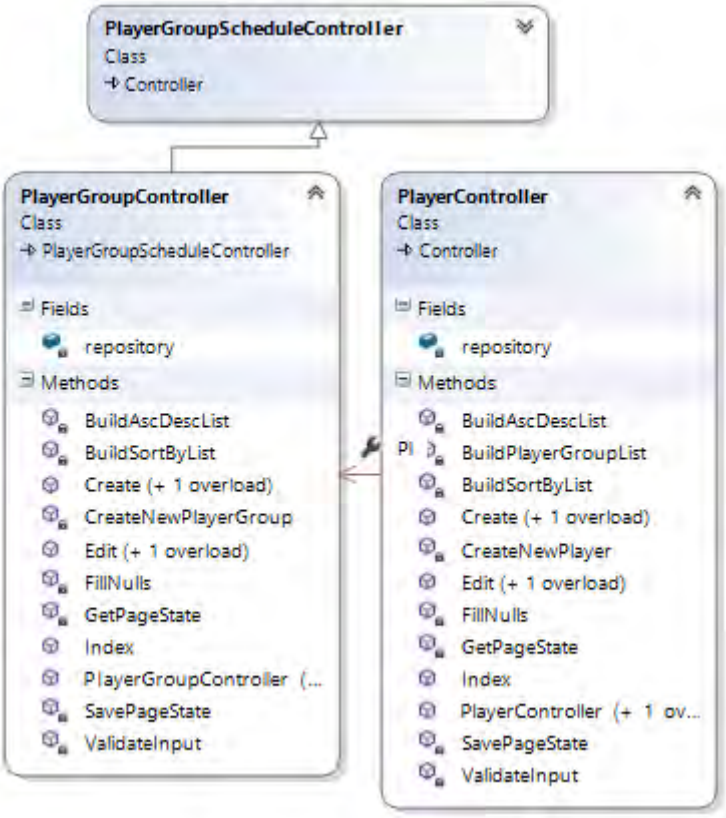
3.2.2.2. Perancangan Kelas `AccountController`

Kelas ini menangani proses yang terjadi pada fungsionalitas manajamen akun. *Controller* ini berisi tentang proses pengembalian data (*save*, *create*, *edit*, *delete*) pada serta beberapa fungsionalitas tambahan (*search, sort*) dan juga validasi pada setiap *form* yang tersedia (*add*)



Gambar 3.13 Rancangan Kelas Manajamen Akun

Untuk manajemen dan pengelompokkan *user* (*client*) pun tidak jauh beda karena secara konsep memiliki fungsionalitas yang sama.

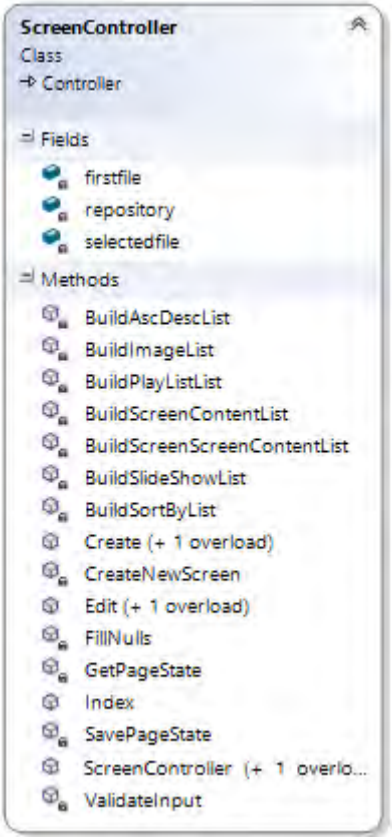


Gambar 3.14 Rancangan Kelas Manajemen Player dan Pengelompokkanya.

3.2.2.3. Perancangan Kelas `ScreenController`

Kelas ini menangani proses yang terjadi pada fungsionalitas manajamen akun. Kontroler ini berisi tentang proses

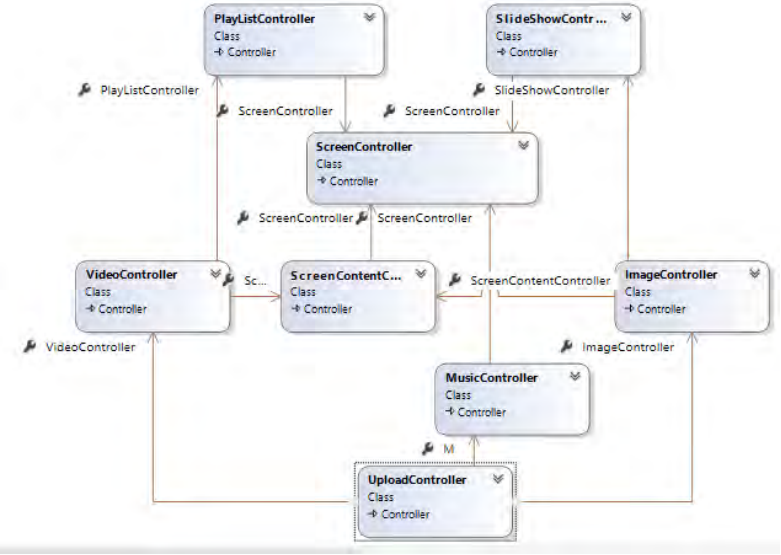
pengembalian data *(save, create, edit, delete*) pada serta beberapa fungsionalitas tambahan (*search, sort*), juga validasi pada setiap *form* yang tersedia (*add*), dan membuat *screen* berdasarkan kebutuhan yang diinginkan (*Playlist, Slideshow*) lalu *screen content* yang dibangun, kelas diagram dapat dilihat pada Gambar 3.15.



Gambar 3.15 Rancangan Kelas Manajemen *Screen*

Seperti Gambar 3.16 pada untuk beberapa sub-fungsionalitas yang tidak berhubungan secara langsung

berpengaruh terhadap *ScreenController* terdapat *PlaylistController*, *SlideshowController*, *VideoController*, *ScreenContentController*, *ImageController*, *MusicController*, dan *UploadController*.

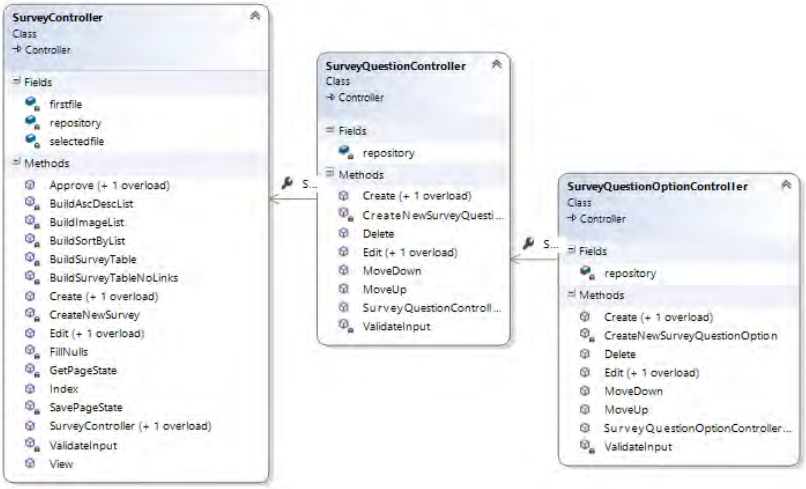


Gambar 3.16 Rancangan Kelas Manajemen *Screen* Secara Keseluruhan.

3.2.2.4. Perancagan Kelas *SurveyController*

Kelas ini menangani proses yang terjadi pada fungsionalitas manajemen akun. Kontroler ini berisi tentang proses pengembalian data (*save, create, edit, delete*) pada serta beberapa fungsionalitas tambahan (*search, sort*) juga validasi pada setiap *form* yang tersedia (*add*) dan melakukan persetujuan *survey* (*approve*).

Seperti pada Gambar 3.17 untuk melakukan pembuatan *survey* perlu melakukan tiga tahap yang mana setiap tahap diproses oleh kontroler masing-masing. Tiga tahap tersebut adalah *Create* pada `SurveyController` untuk mendefinisikan pertanyaan yang dibuat lalu *Create* pada `SurveyQuestionController` untuk membuat pertanyaan dan yang terakhir *Create* `SurveyQuestionOptionController` untuk mendefinisikan pilihan jawaban yang tersedia.



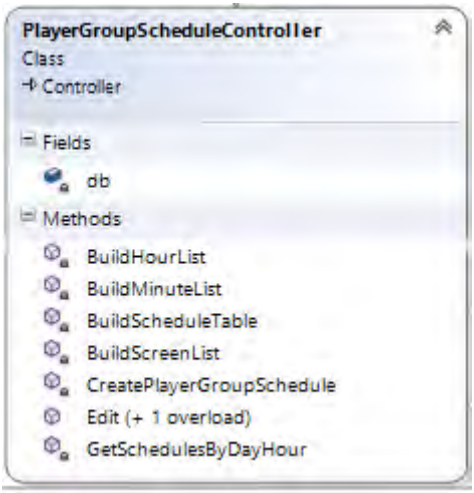
Gambar 3.17 Rancanga Kelas Pembuatan *Survey*

3.2.2.5. Perancangan Kelas `PlayerGroup-ScheduleController`

Seperti pada Gambar 3.18 Kelas ini menangani proses yang terjadi pada fungsionalitas manajamen akun. Kontroler ini berisi tentang proses pengembalian data (*get, save, create, edit, delete*) pada serta beberapa fungsionalitas tambahan (*search, sort*) dan juga validasi pada setiap *form* yang tersedia (*Create*)

Untuk melakukan penambahan *schedule*, masukkan berupa hari yang diinginkan untuk senin hingga minggu dan jam yang diinginkan untuk 00.00 hingga 23.00.

Untuk setiap group *user* memiliki *schedule* tersendiri dan bersifat independen dengan group *user* lain.



Gambar 3.18 Rancangan Kelas *Schedule*

3.2.3. Perancangan Lapisan Data

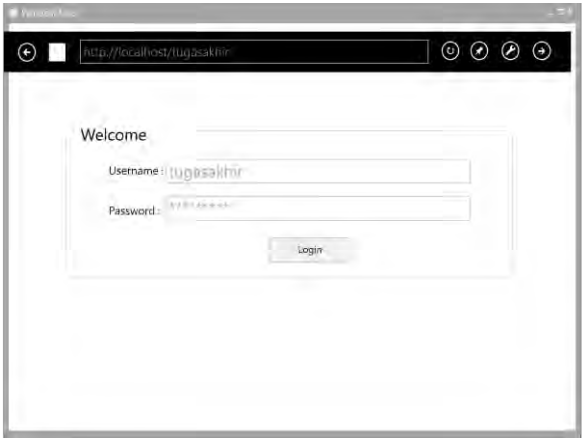
Diagram kelas pada lapisan data ditunjukkan oleh sebuah rancangan *database* berupa CDM atau PDM dapat dilihat pada Lampiran A.1 dan Lampiran A.2. Lapisan data ini hanya berhubungan dengan modul kontrol saja. Lapisan ini hanya bisa diakses oleh lapisan kontrol. Lapisan antarmuka tidak dapat berkomunikasi secara langsung dengan lapisan ini. Kelas-kelas penyusun lapisan ini adalah sebuah objek yang mewakili setiap entitas pada *database*.

3.2.4. Perancangan Antarmuka Pengguna

Bagian ini membahas mengenai perancangan antarmuka pada sistem. Terdapat 12 halaman *main-view* yang dibuat

berdasarkan fungsionalitas, beberapa halaman *link*, dan halaman login. Di setiap halaman *main-view* terdapat beberapa halaman *sub-view* yang setiap halamanya diguanakan sebagai penunjang *main-view* untuk mencapai fungsionalitas yang telah ditentukan. Pada subbab ini akan dijelaskan lebih detail untuk setiap lapisan antarmuka yang terdapat pada aplikasi *web* ini.

3.2.4.1. Rancangan Halaman Tampilan *Login*

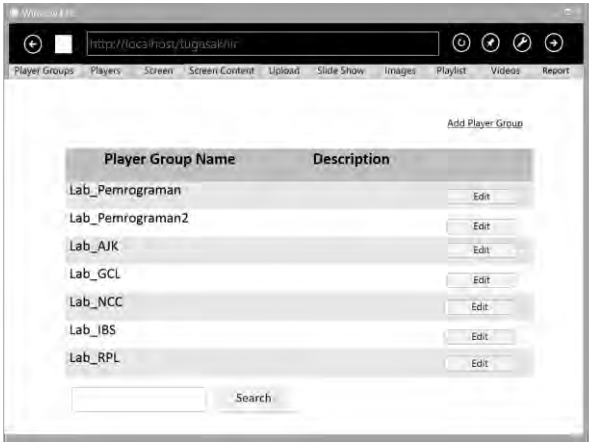


Gambar 3.19 Rancangan Halaman Tampilan *Login*

Gambar 3.19 ini merupakan tampilan login yang akan terliahat ketika halaman *web* pertama dibuka. Seperti pada umumnya *web* pengguna diharuskan login agar dapat memasuki halaman yang lain.

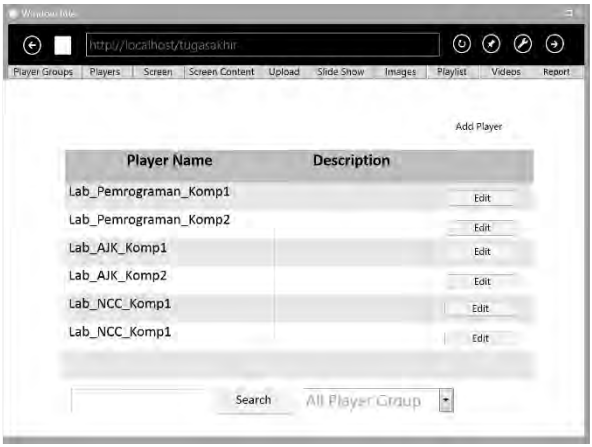
3.2.4.2. Rancangan Halaman Manajemen Akun

Halaman ini digunakan untuk melakukan manajemen akun. Diwakili oleh 2 halaman *main-view* yakni player group dan player untuk *client* dan halaman link account untuk admin.. Terdapat sub-view *add* dan *edit* untuk pada kedua halaman dan fungsi search untuk memfilter data yang diinginkan.



Gambar 3.20 Rancangan Antarmuka *Player Group*

Gambar 3.20 digunakan untuk mengelompokkan player berdasarkan group yang diinginkan. Dengan adanya pengelompokkan maka *server* dapat menentukan *schedule* setiap kelompok sesuai kebutuhan.

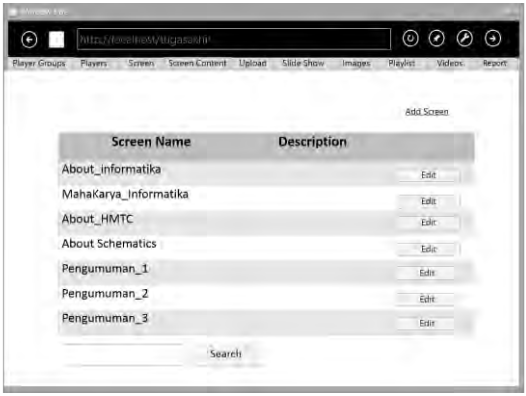


Gambar 3.21 Rancangan Antarmuka *Player*

Gambar 3.21 digunakan untuk menampilkan player yang tersedia. Admin dapat melakukan *edit*, *inactive*, serta membuat player baru

3.2.4.3. Rancangan Halaman Manajemen *Screen*

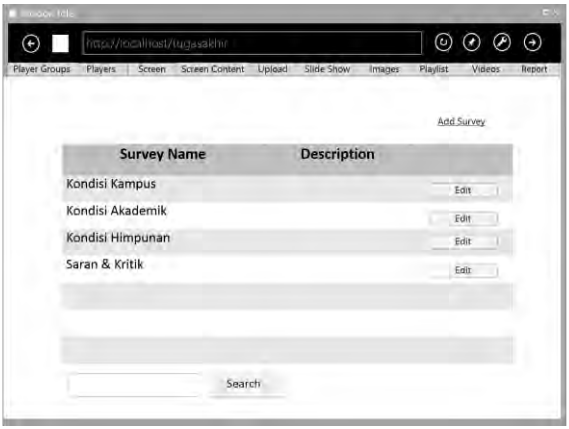
Pada Gambar 3.22 digunakan untuk melakukan manajemen *screen*. Secara fungsionalitas manajemen *screen* adalah pengelolaan *screen* secara keseluruhan yang akan ditampilkan pada player. Terdapat 8 main-menu yang membangun fungsionalitas manajemen *screen* yaitu *Screen, Screen Content, Upload, Slideshow, Image, Playlist, Video,* dan *Music*.



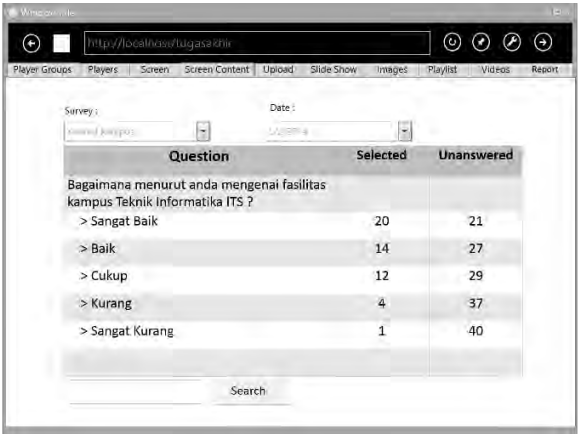
Gambar 3.22 Rancangan Antarmuka Pengelolaan *Screen*

3.2.4.4. Rancangan Halaman *Survey* dan *Report*

Halaman ini digunakan untuk mengelola *survey* beserta hasil *survey* tersebut (*report*). Keduanya dibangun berdasarkan kasus penggunaan 'Mengisi *Survey*. Terdapat 2 main-*view* yang disediakan pada aplikasi *web* ini yaitu *Survey* dan *Report*, seperti pada Gambar 3.23 dan Gambar 3.24.



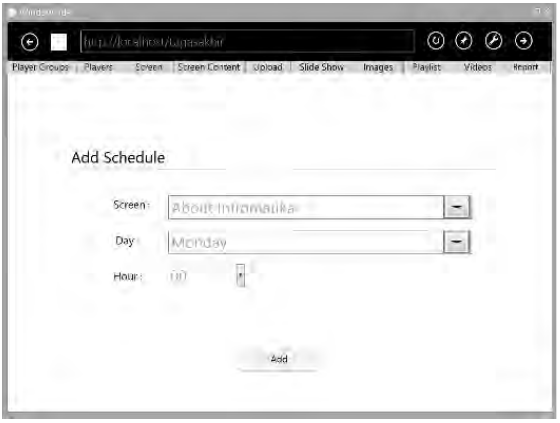
Gambar 3.23 RancanganAntarmuka *Survey*



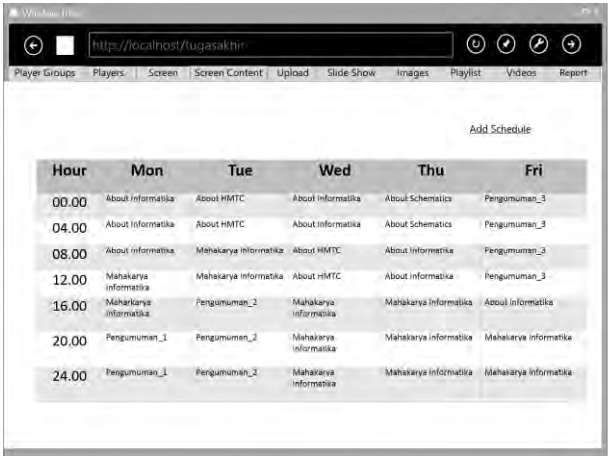
Gambar 3.24 Rancangan Antarmuka *Report*

3.2.4.5. Rancangan Halaman *Scheduling*

Pada Gambar 3.25 dan Gambar 3.26 digunakan untuk melakukan *scheduling* untuk *Client-Desktop* pada aplikasi *web*. Mendata semua *schedule* dan melakukan pengelolaan.



Gambar 3.25 Rancangan Antarmuka Pengelolan *Schedule*



Gambar 3.26 Rancangan Antarmuaka Tabel *Schedule*

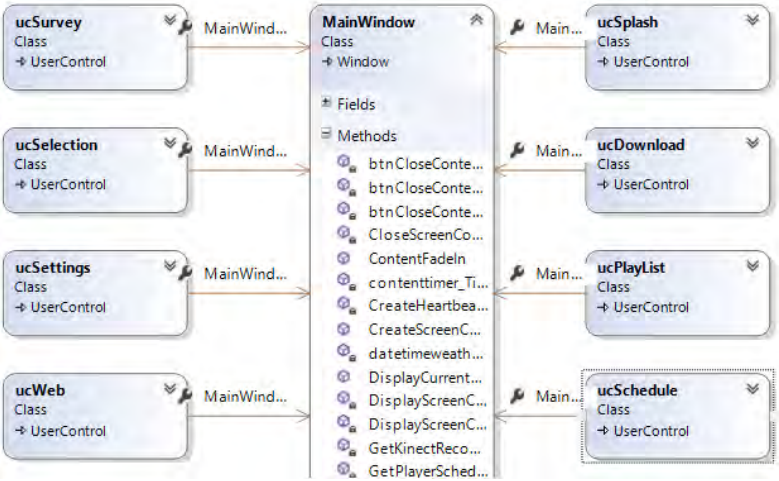
3.3. Perancangan Sistem Aplikasi *Client-Desktop*

Penjelasan tahap perancangan perangkat lunak dibagi menjadi beberapa bagian yaitu perancangan diagram kelas, perancangan proses analisis, dan perancangan antarmuka. Untuk

bab ini akan dijelaskan perancangan sistem pada aplikasi *client-desktop. Model* yang digunakan dalam arsitektur ini ialah *model* tiga-lapis. Tiga lapisan pada arsitektur ini terdiri dari lapisan *Model, View,* dan *ViewModel* Lapisan *ViewModel* merupakan penghubung antara lapisan *Model* dengan *View.*

3.3.1. Perancangan Diagram Kelas

Perancangan diagram kelas berisi rancangan dari kelas-kelas yang digunakan untuk membangun sistem. Pada bab ini, hubungan dan perilaku antar kelas dan antar digambarkan dengan lebih jelas.



Gambar 3.27 Diagram Kelas Lapisan Antarmuka pada *Client-Desktop*

3.3.2. Perancangan Lapisan *View*

Secara konsep lapisan antarmuka (*view*) pada aplikasi *web* dengan aplikasi *client-desktop* tidak jauh beda, yakni sama-sama memiliki sub-sub *view* yang digunakan untuk mewakili

*user*Control pada setiap halaman. Pada Gambar 3.14 Halaman utama terdapat pada `MainWindow.xaml`. Semua fungsi-fungsi penting dalam proses di lapisan antarmuka ini ditangani dalam halaman utama.

Selain Halaman utama lapisan antarmuka pada *client-desktop* juga memiliki sub-sub *view*. Setiap sub *view* memiliki peran pada fungsionalitasnya masing-masing, seperti `ur`*`Survey`* digunakan untuk menangani tampilan *survey*, `uc`*`Playlist`* digunakan untuk menangani tampilan *playlist* dan sama seperti sub-sub *view* lainya.

Pada subbab ini akan dijelaskan lebih detail untuk setiap lapisan *view* yang terdapat pada aplikasi *Client-Desktop*.

3.3.2.1. Rancangan Lapisan *View* pada Halaman Utama (MainWindow.xaml)

Lapisan *view* pada halaman utama seperti dijelaskan diatas yaitu berfungsi untuk menangani pemanggilan atas sub-sub view yang lain. Berperan sebagai protokoler untuk setiap halaman pada aplikasi *Client-Desktop*.

3.3.2.2. Rancangan Lapisan *View* pada Halaman Pembuka

Lapisan *view* pada halaman pembuka bertugas untuk menampilkan proses-proses awal pada aplikasi *Client-Desktop* seperti halaman *credit, download, login player*, dsb. *User*Control lapisan *view* pada halaman pembuka meliputi ucDownlaod, ucSplash, uc*Schedule*, dan ucRegister.

3.3.2.3. Rancangan Lapisan *View* pada Halaman Konten

Lapisan *view* pada halaman konten bertugas untuk menangani konten-konten yang tersedia seperti menampilkan media pemutar video, menampilkan slide-show, menampilkan *web* pada *web browser*, menampilkan *image*, maupun menampilkan

survey. *User*Control lapisan *view* pada halaman pembuka meliputi uc*Playlist*, uc*Slideshow*, uc*Web*, dan ucR*Survey*.

3.3.3. Perancangan *ViewModel*

Karena menggunakan konsep MVVM maka lapisan kontrol pada aplikasi *client-desktop* ini berupa *ViewModel*. Pada dasarnya komponen *ViewModel* berada menempel pada lapisan antarmuka yaitu berupa file.XAML.cs yang biasanya disebut dengan "*Code-Behind-File*".

Karena XAML.cs selalu ada dalam pembuatan lapisan antarmuka XAML maka lapisan kontrol juga selalu ada disetiap lapisan antarmuka yang telah dibuat.

Oleh karena itu pula pada aplikasi *client-desktop* untuk setiap sub-*view* diberi nama userControl yang artinya secara konsep selalu ada lapisan kontrol di setiap lapisan antarmuka yang ada pada *project*.

3.3.4. Perancangan *Model*

Model yang digunakan pada aplikasi *Client-Desktop* ini disesuaikan dengan setiap entitas yang ada pada *database* yang nantinya akan diproses oleh *web service* dari *framework* Vodigi open source.

Peran *Model* pada rancangan aplikasi *Client-Desktop* ini sebagai penampung value dari *database* yang diproses oleh *web* service ataupun dari aplikasi ini sendiri.

3.3.5. Perancangan Antarmuka Pengguna

Bagian ini membahas mengenai perancangan antarmuka pada sistem aplikasi *Client-Desktop*. Untuk antarmuka pengguna pada *Client-Desktop* bersifat dinamis artinya tampilan akan menampilkan halaman yang disesuaikan seperti: tampilan media player akan muncul jika load konten video, *web browser* akan muncul jika load konten *web*, dsb

BAB IV
IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan sistem. Bab ini berisi proses implementasi dari setiap kelas pada semua modul. Bahasa pemrograman yang digunakan adalah bahasa pemrograman C#. Bab ini juga akan pembahasanya dibagi menjadi 2 berdasarkan aplikasi *Web* dan aplikasi *Client-Desktop.*

4.1. Implementasi Lapisan Antarmuka Aplikasi *Web*

Lapisan antarmuka merupakan lapisan yang bertanggung jawab dengan tampilan sistem *web* kepada pengguna. Pada bagian ini akan dijelaskan secara terperinci mengenai implementasi kelas-kelas yang berada dalam lapisan ini. Urutan penjelasan kelas dikelompokkan berdasarkan modul-modul. Untuk halaman antarmka seperti dijelaskan pada 3.2 bahwa *code* lapisan antarmuka pada *web* dibangun dengan menggunakan masterpage yang dihasilkan dari *generating template* MVC 5 ASP.NET.

4.1.1. Implementasi Halaman Login

Untuk implementasi halaman login terdapat 2 aktor yang pada kasus penggunaanya yaitu admin dan anggota biasa, untuk perbedaan hak aksesnya hanya berada pada manajamen akun. Admin dapat melakukan manajamen akun sedangkan member biasa tidak dapat melakukanya.

4.1.2. Implementasi Halaman Utama

Implementasi pada halaman Utama berisi mengenai tampilan-tampilan yang membawahi setiap kasus penggunaan yang disesuaikan agar lebih mudah dalam hal pengoperasianya.

Seperti dijelaskan pada 3.2.4 bahwa pada halaman Utama terdapat 12 *Main-View* yang diesesuaikan dengan fungsionalitasnya. Secara implementasi 12 Main-*View* tersebut memiliki modul sendiri Antara satu dengan yang lainya, artinya

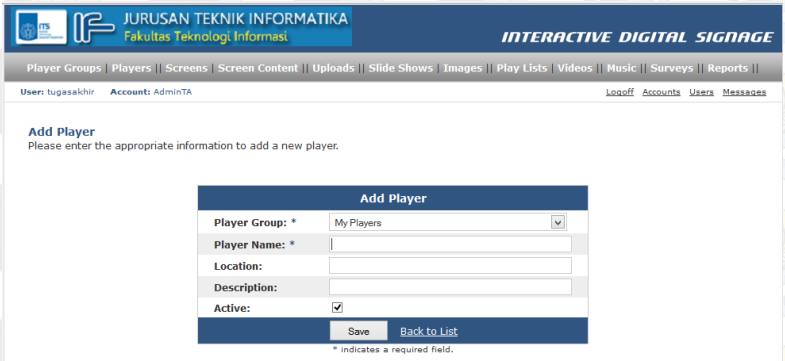
setiap Main-*View* memiliki *View* (Halaman), *Controller* yang sama, sedangkan untuk *model* diesusaikan dengan kebutuhan dan fungsionalitas pada setiap modul.

Untuk sub-bab ini akan dijelaskan lebih detil untuk implementasi serta proses kerja dari setiap halaman yang disesuaikan dengan kasus penggunaan.

4.1.2.1. Implementasi Halaman Manajemen Akun

Secara keseluruhan terdapat 3 halaman yang ditangani pada kasus penggunaan Manajamen Akun ini yaitu halaman Akun, halaman player, dan halaman group player.

Untuk ketiga halaman tersebut masih terdapat sub halaman penunjang yang secara fungsionalitas tidak jauh berbeda yaitu halaman Add Player, Add Account, dan Add Playergroup namun pada *form*-nya disesuaikan dengan kebutuhan masing-masing. Contoh halaman *form* Add Player dapat dilihat pada Gambar 4.1.



Gambar 4.1 Halaman *Form* Add Player

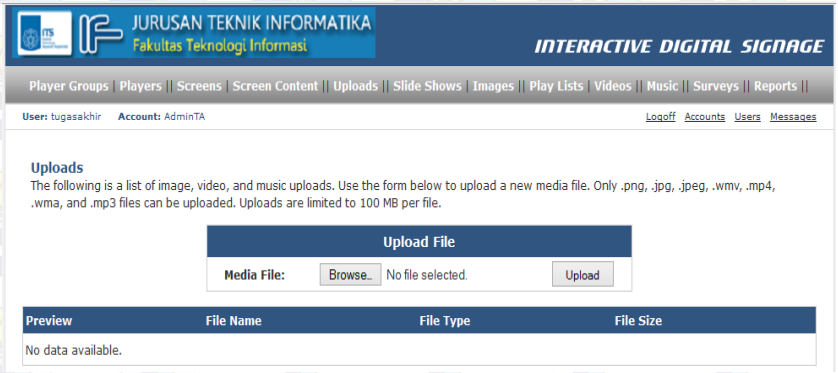
Selain itu juga terdapat fungsionalitas-fungsionalitas penunjang yang menempel pada halaman tersebut guna mempermudah melakaukan proses pada kasus pengguanaan

Manajemen Akun diantaranya yaitu *search, filtering, pengurutan ascending* dan *descending*.

4.1.2.2. Implementasi Halaman Manajemen *Screen*

Seperti dijelaskan pada bab 3.2.4.3 halaman ini bertugas menangani semua aktivitas yang berhubungan dengan manajemen *screen*. Juga dapat dilihat di kelas diagram pada gambar 3.23 bahwa secara fungsionalitas halaman ini terkait dengan banyak halaman lain agar memenuhi kebutuhan dari kasus penggunaan manajemen *screen*. Dibawah ini beberapa fungsionalitas dari kasus penggunaan manajemen *screen* yang meliputi halaman Manajemen *Screen*:

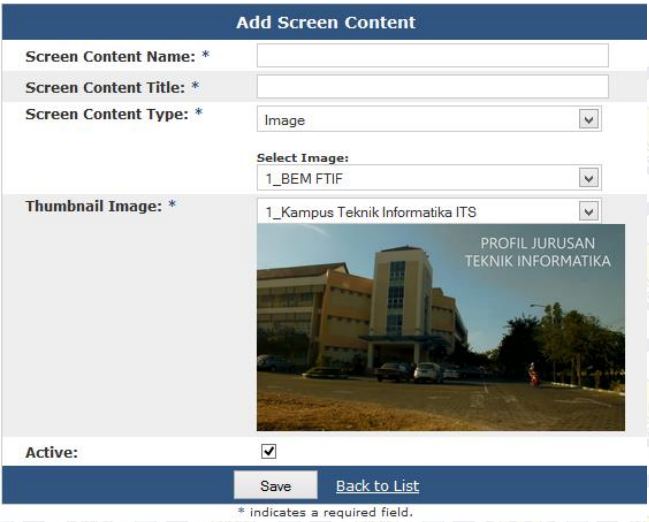
1) Menambahkan *screen*
   Untuk melakukan penambahan *screen user* admin terlebih dahulu harus membuat konten *screen* yang berisi *playlist* atau *slideshow* atau *image* seperti pada Gambar 4.3. Untuk membuat ketiganya *user* admin dapat melakukan upload terlebih dahulu seperti pada Gambar 4.2.



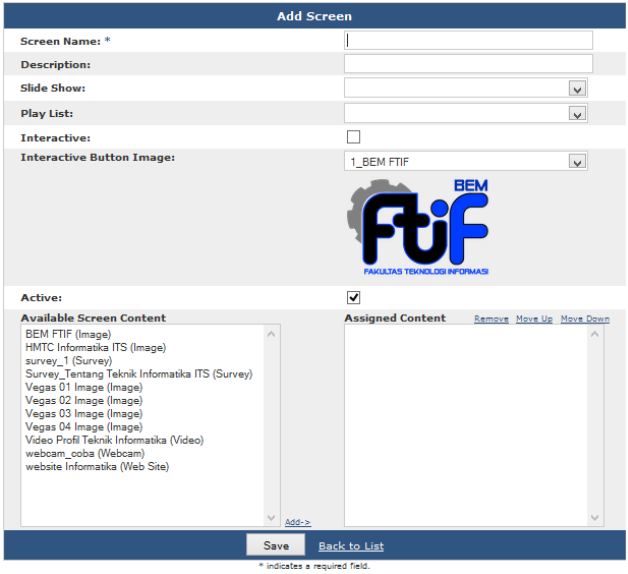
Gambar 4.2 Halaman Upload

Gambar 4.3 Halaman Add Slide Show

Setelah *playlist* atau *slideshow* atau *image* telah dibuat maka dapat dimasukkan ke dalam *screen content* atau membuat *screen* baru *content* seperti pada Gambar 4.4. Dan setelah *screen content* tersedia maka *screen* dapat dibuat dan diisi oleh *screen content* tersebut seperti pada gambar Gambar 4.5.

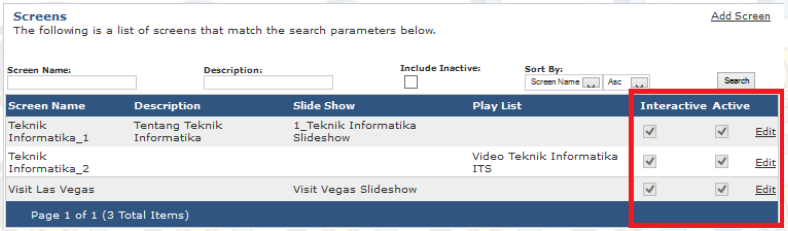


Gambar 4.4 Halaman Add *Screen Content*



Gambar 4.5 Halaman Add *Screen*

2) Manajemen *Screen*
Fungsionalitas yang disediakan untuk melakukan manajemen *screen* pada halaman ini adalah berupa *edit*, active atau deactive, dan *interactive* atau *non-interactive*. Seperti digambarkan pada Gambar 4.6.

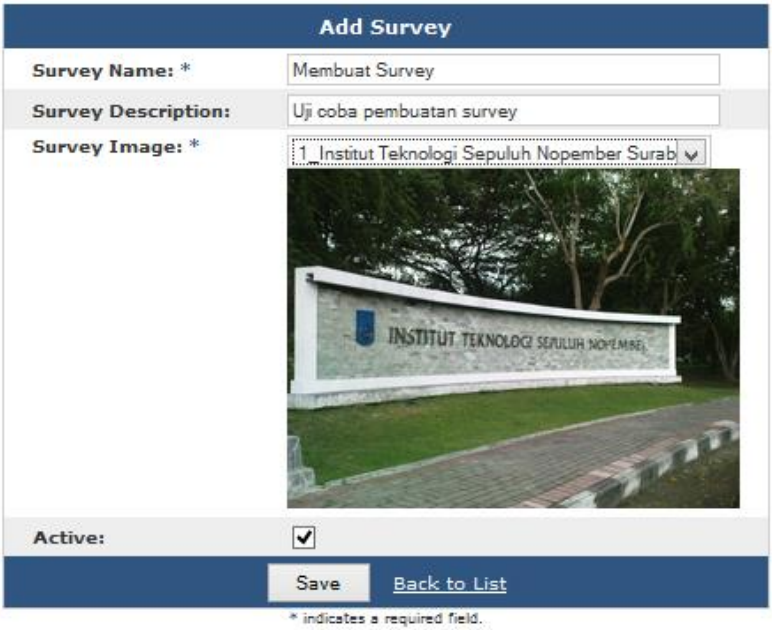


Gambar 4.6 Main-*view* Manajemen *Screen*

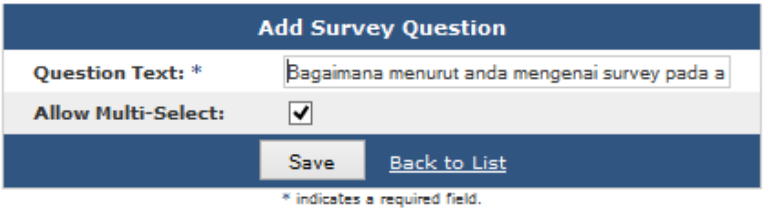
4.1.2.3. Implementasi Halaman Pengelolaan *Survey*

Seperti dijelaskan pada 3.2.2.4 halaman ini bertugas menangani semua aktivitas yang berhubungan *Survey dan Report*. Terdapat 2 halaman main-*view* yang yaitu halaman *Survey* dan halaman Log-*Report*. Dibawah ini beberapa fungsionalitas dari kasus penggunaan *Survey* dan *Report* yang meliputi halaman *Survey* dan *Report*:

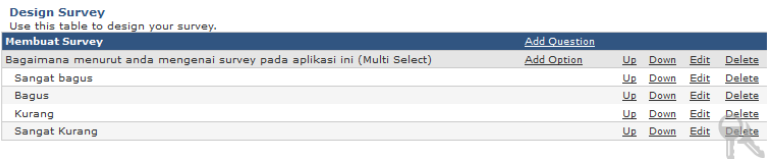
1) Membuat *Survey*
Untuk melakukan pembuatan *survey user* admin dapat masuk pada halaman Add *Survey* seperti pada Gambar 4.7. Setelah itu tambahkan pertanyaan *survey* dengan mengklik ‘add question’ lalu *user* admin akan masuk ke halaman Add Question seperti pada Gambar 4.8. Setelah itu pilih ‘add option’ untuk menambahkan pilihan jawaban yang tersedia jika sudah semua proses telah dilakukan maka akan ditambahkan kolom baru berupa design *survey* dan *user* admin dapat melakukan perubahan lagi sesuai dengan kebutuhan seperti pada Gambar 4.9.



Gambar 4.7 Halaman *Form* Add *Survey*



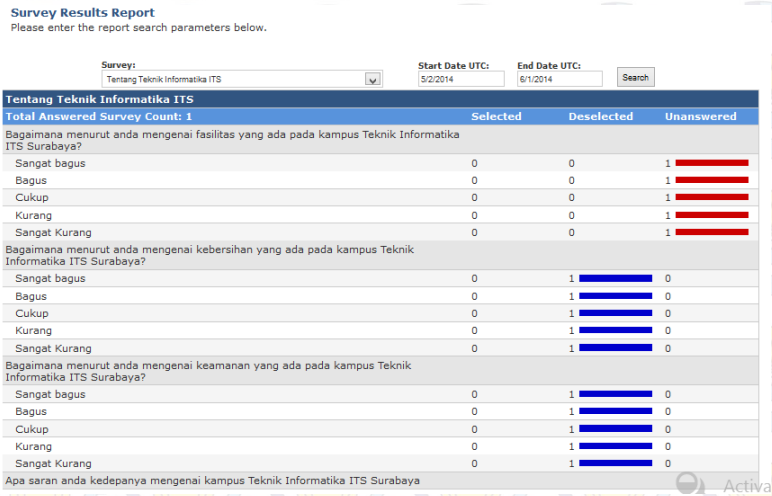
Gambar 4.8 Halaman *Form* Add *Survey* Question



Gambar 4.9 Kolom Design *Survey*

2) Melihat *Report*

Untuk melihat *report user* admin dapat masuk menuju halaman log-*report* pada *Survey Result Report* seperti pada Gambar 4.10 sehingga hasil jawaban dari *user Client-Desktop* dapat dianalisa lebih mudah.



Survey Results Report

Please enter the report search parameters below.

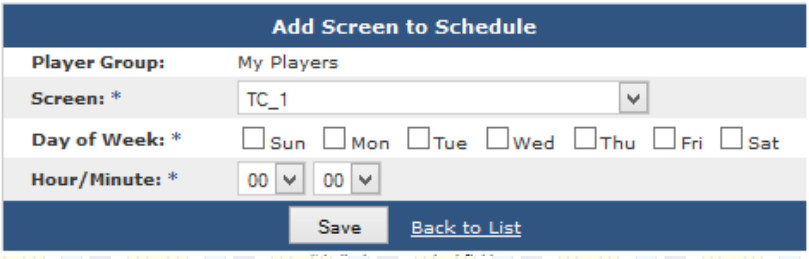
Survey: Tentang Teknik Informatika ITS | Start Date UTC: 5/2/2014 | End Date UTC: 6/1/2014 | Search

Tentang Teknik Informatika ITS Total Answered Survey Count: 1	Selected	Deselected	Unanswered
Bagaimana menurut anda mengenai fasilitas yang ada pada kampus Teknik Informatika ITS Surabaya?			
Sangat bagus	0	0	1
Bagus	0	0	1
Cukup	0	0	1
Kurang	0	0	1
Sangat Kurang	0	0	1
Bagaimana menurut anda mengenai kebersihan yang ada pada kampus Teknik Informatika ITS Surabaya?			
Sangat bagus	0	1	0
Bagus	0	1	0
Cukup	0	1	0
Kurang	0	1	0
Sangat Kurang	0	1	0
Bagaimana menurut anda mengenai keamanan yang ada pada kampus Teknik Informatika ITS Surabaya?			
Sangat bagus	0	1	0
Bagus	0	1	0
Cukup	0	1	0
Kurang	0	1	0
Sangat Kurang	0	1	0
Apa saran anda kedepanya mengenai kampus Teknik Informatika ITS Surabaya			

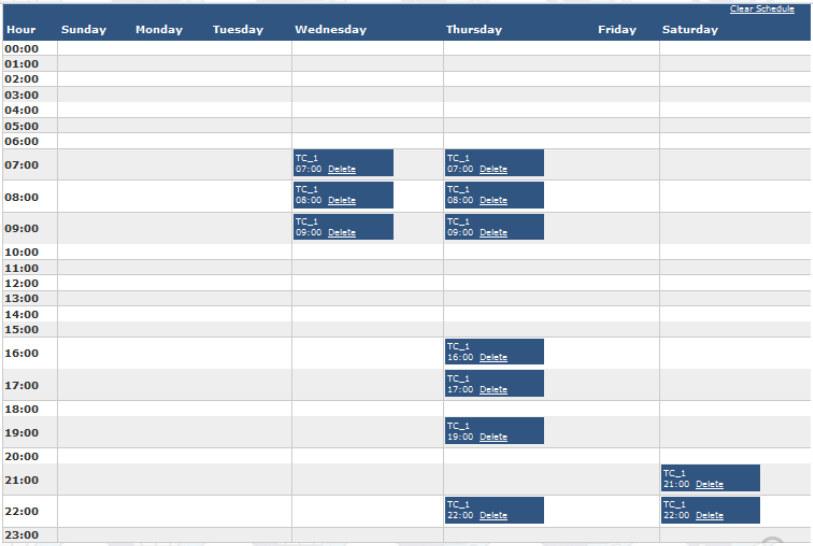
Gambar 4.10 Halaman *Survey Result Report*

4.1.2.4. Implementasi Halaman *Scheduling*

Seperti dijelaskan pada bab 3.2.4.5 halaman ini bertugas menangani semua aktivitas yang berhubungan dengan manajemen *Scheduling*. Untuk memudahkan dalam hal design maka halaman pada *scheduling* menempel pada halaman Player Group, yang artinya setiap Player Group memiliki *schedule* tersendiri. Secara implementasi *user* admin mengisi *form* pada halaman Add *Schedule* dan tersimpan dalam *database*, untuk semua data *schedule* pada setiap Player Group pun dapat langsung dilihat. Gambaran dapat dilihat pada Gambar 4.11 dan Gambar 4.12.



Gambar 4.11 *Form add schedule*



Gambar 4.12 Kolom Schedule

4.1.3. Implementasi Halaman Tambahan

Implementasi pada halaman tambahan hanya berisi tentang proses-proses tambahan yang bersifat *non*-fungsionalitas namun perlu seperti logoff, message, dan sebagainya.

4.2. Implementasi Lapisan Kontrol Aplikasi *Web*

Lapisan kontrol merupakan lapisan yang bertanggung jawab dengan tingkah laku sistem. Lapisan ini secara umum bertugas menghubungkan lapisan data dengan lapisan antarmuka. Pada bagian ini akan dijelaskan secara terperinci mengenai implementasi kelas-kelas yang berada dalam lapisan ini. Urutan penjelasan kelas dikelompokkan berdasarkan modul-modul.

4.2.1. Kelas `Login*Controller*`

Kelas `Login*Controller*` berisi mengenai proses login dari akun admin, proses yang dilakukan adalah validasi dari *form* yang tersedia yaitu *user*name dan password. Implementasi source code digambarkan pada Lampiran B.1.

4.2.2. Kelas `Account*Controller*`

Kelas `Account*Controller*` berisi mengenai proses manajemen akun yang telah dijelaskan pada bab 3.2.2.2. Secara implementasi kelas ini mengani pengelolaan data untuk entitas Account, *User*, Player, dan PlayerGroup. Implementasi source code digambarkan pada Lampiran B.2.

Pada modul ini kelas yang berhubungan secara asosiasi adalah `Player*Controller*`, dan kelas `PlayerGroup*Controller*`.

4.2.3. Kelas *`ScreenController`*

Kelas *`ScreenController`* berisi mengenai proses manajemen *screen* yang telah dijelaskan pada bab 3.2.2.3. Secara implementasi kelas ini menangani pengelolaan data untuk entitas *Screen*, *ScreenContent*, *Image*, Video, *Playlist*, *Slideshow*. Digambarkan pada Lampiran B.3.

Karena secara kasus penggunaan kelas ini memiliki fungsionalitas yang luas, maka modul ini memiliki berasosiasi dengan banyak kelas kontroler lainya. Kelas yang berhubungan secara asosiasi adalah *`PlaylistController`*,

SlideshowController, Video*Controller*, *ScreenContentController*, *ImageController*, Music*Controller*, dan Upload*Controller*.

4.2.4. Kelas *SurveyController*

Kelas *SurveyController* berisi mengenai proses pembuatan *survey* yang telah dijelaskan pada bab 3.2.2.4. Secara implementasi kelas ini menangani pengelolaan data untuk entitas *Survey*, *Survey*Question, *Survey*QuestionOption, Answered*Survey*, Answered*Survey*QuestionOption.

4.3. Implementasi Lapisan Data Aplikasi *Web*

Lapisan ini hanya berisi tentang kelas objek serta interface dan hanya berhubungan dengan kelas-kelas pada lapisan kontrol. Lapisan ini berfungsi sebagai penyimpan data sementara dari data yang diproses oleh kontroler.

Pada aplikasi *web* lapisan data disesuaikan dengan setiap entitas yang ada pada *database* yang nantinya akan diproses oleh kontroler.

4.4. Implementasi Lapisan Antarmuka (*View*) dan lapisan kontroler (*ViewModel*) *Client-Desktop*

Seperti dijelaskan pada 3.3 bahwa antarmuka pada *Client-Desktop* dibangun dengan MVVM *Design Pattern* yang mana lapisan antarmuka dan lapisan kontroler berkaitan.

Pada sub-bab ini juga akan dijelaskan lebih detil implementasi untuk beberapa halaman pokok pada aplikasi *Client-Desktop* ini.

4.4.1. Implementasi pada Halaman Utama (MainWindow.xaml)

Sesuai dengan rancangan aplikasi yang dijelaskan pada banyak bab sebelumnya (bab 3.3.2.1 dan bab 4.4) bahwa halaman utama pada aplikasi *Client-Desktop* ini merupakan protokol pada

semua halaman. Semua tampilan menempel pada tampilan utama (`MainWindow.xaml`). Sehingga secara implementasi peran dan fungsi pada sub-view atau *user*Control adalah sebagai fungsi yang akan tampil jika dipanggil atau dibutuhkan.Halaman ini akan bersifat dinamis sesuai dengan perintah sistem atau dari *user* jika terdapat mode interaktif.

Di dalam MainWindow.xaml tidak ada tampilan sebenarnya melainkan hanyalah sebuah halaman kosong yang menampung *user*Control-*user*Control lainya sehingga membuat halaman ini sebagian besar peranya pada kontroler.

4.4.2. Implementasi pada Halaman Pembuka

Seperti dijelaskan pada bab 3.3.2.2 *user*Control ini secara tampilan hanya bertugas untuk menampilkan proses-proses awal aplikasi *Client-Desktop* dijalankan. Dibawah ini *user*Control yang termasuk dalam kelompok halaman pembuka:

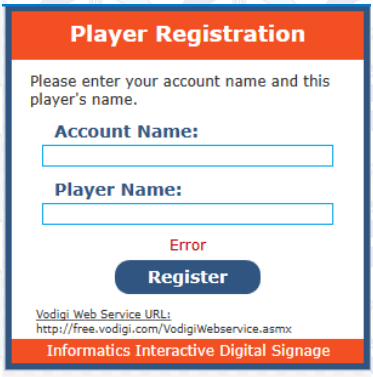
1) **ucDownload.xaml** : progress download dari *Client-Desktop* ke *Server*. *Progress bar* juga menunujukkan bahwa proses download sedang berlangsung dan proses download selesai jika *progress bar* tersebut penuh. Dan akan menampilkan peringatan jika gagal Tampilan dapat dilihat pada Gambar 4.13.

Gambar 4.13 Halaman *User*Control ucDownload

2) **ucSplash.xaml** : Salah satu *user*Control tambahan yang digunakan sebagai halaman pembuka. Tidak ada proses berlangsung pada saat halaman ini ditampilkan. Tampilan dapat dilihat padaGambar 4.14.

Gambar 4.14 Halaman *User*Control ucSplash

3) **ucRegister.xaml** : Secara konsep hampir sama dengan halaman login, namun halaman ini hanya muncul satu kali ketika telah didaftarkan kecuali konfigurasi pada playerconfiguration.xml diatur ulang atau dihapus. Tampilan dapat dilihat pada Gambar 4.15.



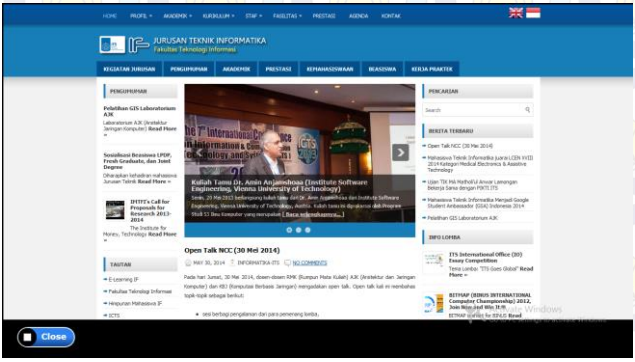
Gambar 4.15 Halaman *User*Control ucRegister.xaml

4) **uc*Schedule*.xaml** : *user*Control yang memiliki kesamaan dengan ucDownload.xaml yakni hanya menampilkan proses penentuan *schedule*, dan terdapat peringatan jika gagal.

4.4.3. Implementasi pada Halaman Konten

Halaman *user*Control paling penting dari aplikasi *Client-Desktop* ini. Halaman-halaman ini menangani semua bahan-bahan konten yang akan ditampilkan. Dibawah ini *user*Control yang termasuk dalam kelompok halaman Konten:

1) **uc*Playlist*.xaml** : Halaman yang hanya berisi media player yang disiapkan untuk menampung *playlist* yang akan ditampilkan. Halaman ini dipanggil oleh MainWindow ketika konten video akan ditampilkan.
2) **uc*Slideshow*.xaml** : Halaman yang dibangun untuk menyesuaikan banyak gambar yang akan ditampilkan secara bergantian. Halaman ini dipanggil oleh MainWindow ketika konten *slideshow* akan ditampilkan.
3) **uc*Web*.xaml** : Halaman yang hanya berisi *web browser* yang disiapkan untuk menampung alamat *web* yang dituju. Halaman ini dipanggil oleh MainWindow ketika konten *web* akan ditampilkan. Tampilan dapat dilihat pada Gambar 4.16.



Gambar 4.16 Tampilan Uc*Web*

4) **ucSelection.xaml dan ucSelectionBar.xaml** :

Halaman yang beruba fitur yang disiapkan untuk *screenContent* yang dapat dipilih oleh *user* player saat tampilan *screen* berjalan. Dipanggil ketika *user* player melakukan interaksi dengan *menu selection*. Tampilan dapat dilihat pada Gambar 4.17. untuk kotak merah mewakili tampilan ucSelection dan kotak hijau mewakili ucSelectionBar.

Gambar 4.17 Tampilan ucSelection dan ucSelectionBar

4.5. Implementasi Lapisan Data (*Model*) *Client-Desktop*

Model yang digunakan pada aplikasi *Client-Desktop* ini disesuaikan dengan setiap entitas yang ada pada *database* yang nantinya akan diproses oleh *web* service dari *framework* Vodigi open source.

Peran *Model* pada rancangan aplikasi *Client-Desktop* ini sebagai penampung value dari *database* yang diproses oleh *web* service ataupun dari aplikasi ini sendiri. Sebagai pendukung dalam pengiriman data maka pada perangkat lunak ini menggunakan *services framework* yakni vodigi *services* yang akan dijelaskan pada subbab dibawah ini.

4.5.1. Vodigi *Services*

Services yang diadopsi untuk digunakan dalam pengembangan aplikasi ini, sebagai pendukung dalam pengiriman data juga mempermudah dalam perancangan data yang dibuat.

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada *plugin* yang dikembangkan. Pengujian yang dilakukan adalah pengujian terhadap kebutuhan fungsionalitas sistem dan kegunaan sistem. Pengujian fungsionalitas mengacu pada kasus penggunaan pada bab tiga. Pengujian kegunaan program dilakukan dengan mengetahui tanggapan dari pengguna terhadap sistem. Hasil evaluasi menjabarkan tentang rangkuman hasil pengujian pada bagian akhir bab ini.

5.1. Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan Tugas Akhir ini dilakukan pada lingkungan dan alat kakas sebagai berikut:

Prosesor : Intel Core i7-3610QM
CPU @ 2.30GHz
Memori : 8.00 GB
Jenis Device : Laptop
Sistem Operasi : Microsoft Windows 8 Enterprise 64 bit
IDE Visual Studio : Visual Studio 2013

5.2. Dasar Pengujian

Pengujian pada Perangkat Lunak Digital Signage ini dilakukan dengan menggunakan laptop. Pengujian perangkat lunak ini menggunakan metode *black box* yang berfokus pada kebutuhan fungsional dan *non*-fungsional pada peranagkat lunak tersebut. Pengujian ini dilakukan untuk menguji apakah fungsionalitas yang diidentifikasi pada tahap kebutuhan benar-benar diimplementasi dan bekerja seperti yang semestinya serta sesuai dengan kebutuhan pada .

5.3. Skenario Pengujian

Pada subab ini akan dijelaskan mengenai skenario pengujian dari perangkat lunak Digital Signage yang terbagi menjadi 2 aplikasi yaitu aplikasi *web* dan aplikasi *Client-Desktop*.

5.3.1. Pengujian Fungsionalitas

Berikut ini akan dijelaskan mengenai pengujian mengenai pengujian yang dilakukan pada perangkat lunak Digital Signage berdasarkan kasus penggunaan pada bab 3.

5.3.1.1. Pengujian Manajamen Akun

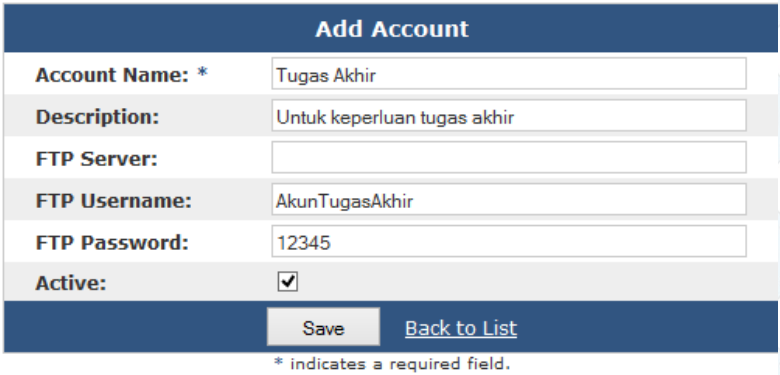
Pada bagian ini dijelaskan pengujian mengenai proses-proses yang ada pada kasus penggunaan Manajemen Akun. Proses-proses tersebut meliputi pembuatan akun serta pengelolaanya.

Tabel 5.1 Pengujian Proses Manajamen Akun

Test ID	TA-UJI-0101			
Tujuan Test	Mengecek apakah fungsi proses pembuatan akun berjalan normal			
Kondisi Awal	Pengguna masuk ke halaman akun			
Data Input	Presedur Pengujian	Hasil Diharapkan	Hasil Diperoleh	Kesimpulan
Input yang dimasukkan “Account Name”, “Desciption”, “*User*name”, “Password”, “Kondisi, “FTP *Server* (tidak	Mengisi *Form*	Data yang dimasukkan masuk ke dalam *database* dan dapat ditampilkan pada table halaman tabel akun.	Data berhasil masuk ke *database*, dan dapat ditampilkan pada halaman tabel akun.	Proses penambahan akun berhasil

harus)" aktif".				

Berikut ini adalah potongan gambar pengujian yang dijalankan. Gambar 5.1 menggambarkan halaman untuk memasukkan data Akun. Proses pengujian ini dikatakan berhasil jika data akun tersimpan di basis data dan tampil di halaman Akun.



Gambar 5.1 Halaman *Form* Pembuatan Akun

5.3.1.2. Pengujian Manajemen *Screen*

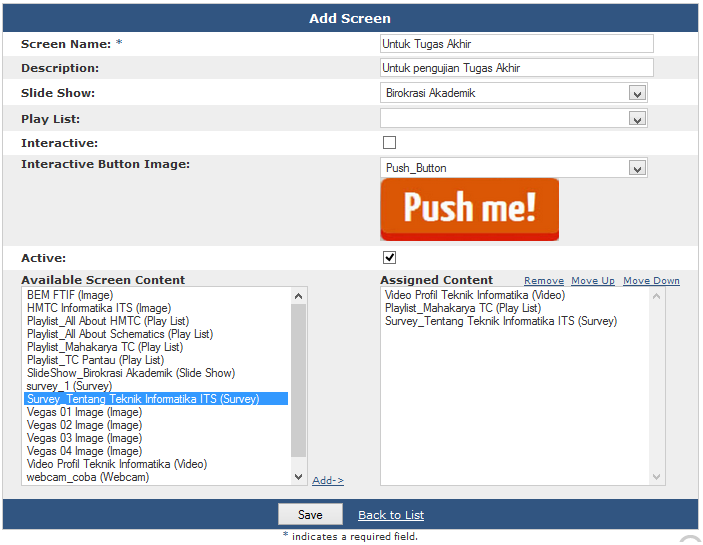
Pada bagian ini dijelaskan pengujian mengenai proses-proses yang ada pada kasus penggunaan Manajemen *Screen*. Proses-proses tersebut meliputi langkah membuat *screen* serta pengelolaanya.

Tabel 5.2 Pengujian Menambahkan *Screen*

Test ID	TA-UJI-0201			
Tujuan Test	Mengecek apakah fungsi proses pembuatan *Screen* berjalan normal			
Kondisi Awal	Pengguna masuk ke halaman *screen*			
Data Input	Presedur Pengujian	Hasil Diharapkan	Hasil Diperoleh	Kesimpulan

Input yang dimasukkan "*Screen* Name", "Desciption", "*Slideshow / playlist*", "mode *interactive*",, "thumbnail tombol", "aktif". "*Screen content*".	Mengisi *Form*	Data yang dimasukkan masuk ke dalam *database* dan dapat ditampilkan pada halaman tabel *screen*.	Data berhasil masuk ke *database*, dan dapat ditampilkan pada halaman tabel *screen*.	Proses penambahan *screen* berhasil

Berikut ini adalah potongan gambar pengujian yang dijalankan. Gambar 5.2 menggambarkan halaman untuk memasukkan data *screen*. Proses pengujian ini dikatakan berhasil jika data *screen* tersimpan di basis data dan tampil di halaman *screen*.



Gambar 5.2 Halaman *Form* Pembuatan *screen*

5.3.1.3. Pengujian Pengelolaan *Survey*

Pada bagian ini dijelaskan pengujian mengenai proses-proses yang ada pada kasus penggunaan Membuat *Survey* dan Melihat *report*. Proses-proses tersebut meliputi langkah membuat *survey* serta pengelolaanya, dan melihat hasil *report*.

Tabel 5.3 Pengujian Pengelolaan *Survey*

Test ID	TA-UJI-0301			
Tujuan Test	Mengecek apakah fungsi proses pembuatan *survey* berjalan normal			
Kondisi Awal	Pengguna masuk ke halaman *survey*			
Data Input	Presedur Pengujian	Hasil Diharapkan	Hasil Diperoleh	Kesimpulan
Input yang dimasukkan "*Survey* Name", "Desciption", "*Image*", "aktif", " pertanyaan"," opsi jawaban yang tersedia"	Mengisi *Form*	Data yang dimasukkan masuk ke dalam *database* dan dapat ditampilkan pada table halaman tabel *survey*.	Data berhasil masuk ke *database*, dan dapat ditampilkan pada halaman tabel *survey.*	Proses penambahan *Survey* berhasil

Berikut ini adalah potongan gambar pengujian yang dijalankan. Gambar 5.3 menggambarkan halaman untuk memasukkan data *survey*Proses pengujian ini dikatakan berhasil jika data *survey* tersimpan di basis data dan tampil di halaman *survey*.

Gambar 5.3 Halaman *Form* Pembuatan *survey*

5.3.1.4. Pengujian Melakukan *Scheduling*

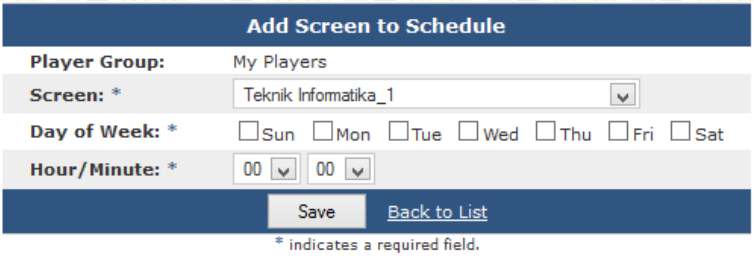
Pada bagian ini dijelaskan pengujian mengenai proses-proses yang ada pada kasus penggunaan Melakukan *Scheduling*. Proses-proses tersebut meliputi langkah mengisi *schedule*.

Tabel 5.4 Pengujian Menambahkan *Schedule*

Test ID	TA-UJI-0301
Tujuan Test	Mengecek apakah fungsi proses pembuatan *Schedule* berjalan normal
Kondisi Awal	Pengguna masuk ke halaman Player Group - *Schedule*

Data Input	Presedur Pengujian	Hasil Diharapkan	Hasil Diperoleh	Kesimpulan
Input yang dimasukkan "*Screen* yg ditampilkan", "hari", "jam",	Mengisi *Form*	Data yang dimasukkan masuk ke dalam *database* dan dapat ditampilkan pada table halaman tabel *schedule*	Data berhasil masuk ke *database*, dan dapat ditampilkan pada halaman tabel *schedule.*	Proses penambahan *schedule* berhasil

Berikut ini adalah potongan gambar pengujian yang dijalankan Gambar 5.4 menggambarkan halaman untuk memasukkan data *survey*. Proses pengujian ini dikatakan berhasil jika data *survey* tersimpan di basis data dan tampil di halaman *survey*.



Gambar 5.4 Halaman *Form* Pembuatan *Schedule*

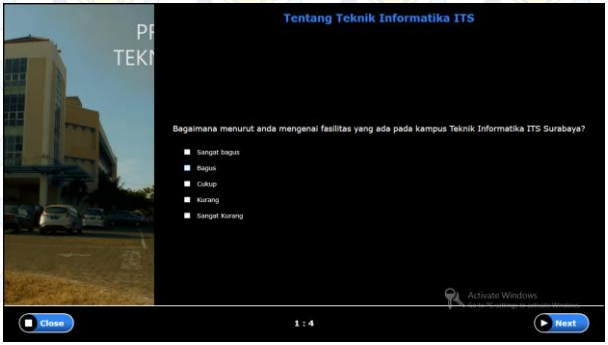
5.3.1.5. Pengujian Mengisi *Survey*

Pada bagian ini dijelaskan proses-proses yang ada pada kasus penggunaan Mengisi *Survey*. Proses-proses tersebut meliputi langkah mengisi *Survey* pada *Client-Desktop*.

Tabel 5.5 Pengujian Pengisian *Survey*

Test ID	TA-UJI-0301			
Tujuan Test	Mengecek apakah fungsi proses pengisian *survey* berjalan normal			
Kondisi Awal	Pengguna berada dalam halaman *survey*			
Data Input	Presedur Pengujian	Hasil Diharapkan	Hasil Diperoleh	Kesimpulan
Input yang dimasukkan jawaban optional yang telah disediakan	Mengisi *Form*	Data yang dimasukkan masuk ke dalam *database* dan dapat ditampilkan pada table halaman tabel *report.*	Data berhasil masuk ke *database*, dan dapat ditampilkan pada halaman tabel *report*	Proses penambahan *survey* berhasil

Berikut ini adalah gambar pengujian yang dijalankan. Gambar 5.5 Halaman *Form* Pengisian *Survey* menggambarkan halaman untuk memasukkan hasil *survey*. Proses pengujian ini dikatakan berhasil jika hasil *survey* tersimpan di basis data dan tampil di halaman *report.*



Gambar 5.5 Halaman *Form* Pengisian *Survey*

5.3.2. Pengujian *Non*-Fungsional

Pada bagian ini dijelaskan pengujian mengenai *non*-fungsional dari aplikasi *Web* maupun *Client-Desktop*. Kebutuhan *non* fungsionalitas ini ditinjau dari waktu yang dibutuhkan aplikasi untuk memproses data, dan keamanan aplikasi.

Berikut ini adalah prosedur pengujian berdasarkan waktu unggah file dan unduh dalam bentuk *image* dan video mapun per*form*a dalam pemasukan dan pengiriman data. Pengujian menggunakan jaringan intranet dan dilakukan secara bersamaan oleh 5 pengguna. Proses pengujian dibantu dengan menggunakan alat bantu hitung *Stopwatch*.

Tabel 5.6 Pengujian *Non*-Fungsionalitas

Test ID	TA-UJI-NF1		
Tujuan Test	Melakukan analisis waktu selama proses berlangsung		
Kondisi Awal	Aplikasi sudah dijalankan		
Data Input	Presedur Pengujian	Hasil Diharapkan	Hasil Diperoleh
Data input merupakan semua *form* yang menerima string dengan data yang berada di basis data kurang lebih 5 hingga 15 dalam setiap entitas. Waktu yang dihitunga hanya pada saat proses sistem tidak termasuk diluar itu.			

Masukkan dimasukkan pada *form* pembuatan akun, pembuatan player, dan pembuatan *survey* dan dilakukan oleh 5 orang secara bersamaan	Aplikasi dijalankan dari tahap awal sampai tahap memasukkan masukkan oleh sistem, lalu waktunya dihitung dengan menggunakan *Stopwatch*	Aplikasi berjalan dengan kecepatan kurang dari 1 menit	Waktu yang dibutuhkan selama proses berlangsung hingga semua masukkan telah masuk adalah 2.09 detik.
Data input merupakan file upload berupa video sebesar 72,98 MB dengan file awal yang telah ada di basis data sebanyak 3 hingga 5 file			
File yang dimasukkan berupa video dan dimasukkan ke dalam halaman *form* upload yang tersedia pada aplikasi *web*	Aplikasi *web* dijalankan dari tahap awal sampai tahap upload video oleh sistem, lalu waktunya dihitung dengan menggunakan *Stopwatch*	Aplikasi berjalan dengan kecepatan kurang dari 1 menit	Waktu yang dibutuhkan selama proses berlangsung hingga semua masukkan telah masuk adalah 6.22 detik.
Data input merupakan file upload berupa *Image* sebesar 2,8 MB dengan file awal yang telah ada di basis data sebanyak 3 hingga 5 file			

File yang dimasukkan berupa *image* dan dimasukkan ke dalam halaman *form* upload yang tersedia pada aplikasi *web*	Aplikasi *web* dijalankan dari tahap awal sampai tahap upload *image* oleh sistem, lalu waktunya dihitung dengan menggunakan *Stopwatch*	Aplikasi berjalan dengan kecepatan kurang dari 1 menit	Waktu yang dibutuhkan selama proses berlangsung hingga semua masukkan telah masuk adalah 0.33 detik.
Pengujian kecepatan unduh dari sisi *Client-Desktop* dengan file konten sebesar 510 MB			
File yang unduh berupa file konten yang akan ditampilkan pada sisi *Client-Desktop*	Aplikasi *Client-Desktop* dijalankan dari tahap awal sampai tahap unduh file konten, lalu waktunya dihitung dengan menggunakan *Stopwatch*	Aplikasi berjalan dengan kecepatan kurang dari 1 menit	Waktu yang dibutuhkan selama proses berlangsung hingga semua masukkan telah masuk adalah 9.88 detik.

5.3.3. Pengujian Kegunaan

Tugas Akhir ini juga membutuhkan pengujian aplikasi berdasarkan pengamatan pengguna lain yang belum pernah menggunakan Perangkat lunak Digital Singage ini. Tujuan dari pengujian kegunaan adalah memberikan penilaian tentang kelayakan Perangkat Lunak Digital Singage ini, kemudahan dalam penggunaan, maupun keseusaian kebutuhan pada lingkungan Teknik Informatika ITS Surabaya serta komentar-komentar terkait

dari fitur-fitur yang telah diimplementasikan.Pengujian ini dilakukan pengguna dengan metode *blackbox testing*.

Pengujian dilakukan oleh beberapa orang yang dipilih oleh penulis. Penulis memilih orang-orang yang berlatarbelakang pendidikan teknik komputer. Penulis juga memilih orang-orang yang pernah menggunakan aplikasi ASP.net dan WPF untuk melakukan pengoperasian pada aplikasi.

Pengujian dilakukan dengan memberikan kesempatan pada pengguna untuk mencoba sendiri Perangkat *lunak Digital Singage* yang telah dikembangkan. Uji coba yang dilakukan pengguna meliputi manajemen dan pengelolaan data pada sisi admin serta pengoperasianya, penggunaan aplikasi dari sisi *client*, dan melihat kesesuaian konten pada aplikasi yang disesuaikan dengan tujuan pembuatan perangkat lunak pada tugas akhir ini. Rincian isi kuesioner yang diberikan pada pengguna dapat dilihat padaTabel 5.4.

Tabel 5.7 Daftar Penguji Perangkat Lunak

No.	Nama	Profesi	Jurusan
1	Fernandes Sinaga	Mahasiswa	Teknik Informatika
2	Endang Wahyu P.	Mahasiswa	Teknik Informatika
3	Galang Amanda D.P.	Mahasiswa	Teknik Informatika
4	Chairaja Almas Djeni	Mahasiswa	Teknik Informatika
5	Bagus Ardiansyah	Mahasiswa	Teknik Informatika
6	Azi Prastyo	Mahasiswa	Teknik Informatika
7	Muh. Aunorafiq M.	Mahasiswa	Teknik Informatika
8	Muhammad Agil M.	Mahasiswa	Teknik Informatika
9	Suliadi Marsetya	Mahasiswa	Teknik Informatika
10	M Hanif Budiarto	Mahasiswa	Teknik Informatika

Tabel 5.8 Daftar Pertanyaan Kuesioner

No.	Pernyataan	Pilihan jawaban			
		Sangat setuju	Setuju	Tidak setuju	Sangat tidak setuju
1	Menurut saya aplikasi *web* ini mudah digunakan dan dipahami.	◯	◯	◯	◯
2	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen akun pada aplikasi *web* ini.	◯	◯	◯	◯
3	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen *screen* pada aplikasi *web* ini	◯	◯	◯	◯
4	Menurut saya, cukup mudah untuk pengoperasian membuat *survey* pada aplikasi *web* ini	◯	◯	◯	◯
5	Menurut saya, cukup mudah untuk pengoperasian melakukan *scheduling* pada aplikasi *web* ini	◯	◯	◯	◯
6	Menurut saya, tampilan pada aplikasi *web* sesuai dengan konten.	◯	◯	◯	◯
7	Menurut saya aplikasi *Client-Desktop* ini mudah digunakan dan dipahami.	◯	◯	◯	◯
8	Menurut saya, tampilan pada aplikasi *Client-Desktop* mudah dimengerti sehingga mudah untuk mengoperasikanya.	◯	◯	◯	◯
9	Menurut saya, konten yang ada pada tugas akhir ini sangat sesuai dengan	◯	◯	◯	◯

No.	Pernyataan	Pilihan jawaban			
		Sangat setuju	Setuju	Tidak setuju	Sangat tidak setuju
	kebutuhan Teknik Informatika ITS.				

Tabel 5.9 Hasil Kuesioner

No.	Pernyataan	Pilihan jawaban			
		Sangat setuju	Setuju	Tidak setuju	Sangat tidak setuju
1	Menurut saya aplikasi *web* ini mudah digunakan.	90 %	10%		
2	Menurut saya, cukup mudah untuk melakukan manajamen akun pada aplikasi *web* ini..	80%	20%		
3	Menurut saya, cukup mudah untuk melakukan manajamen *screen* pada aplikasi *web* ini	60%	40%		
4	Menurut saya, cukup mudah untuk membuat *survey* pada aplikasi *web* ini	60%	40%		
5	Menurut saya, cukup mudah untuk melakukan *scheduling* pada aplikasi *web* ini	30%	70%		
6	Menurut saya, tampilan pada aplikasi *web* mudah dimengerti sehingga mudah untuk mengoperasikanya.	90%	10%%		
7	Menurut saya aplikasi *Client-Desktop* ini mudah digunakan.	60%	40%		
8	Menurut saya, tampilan pada aplikasi *Client-Desktop* mudah dimengerti sehingga	90%	10%		

No.	Pernyataan	Pilihan jawaban			
		Sangat setuju	Setuju	Tidak setuju	Sangat tidak setuju
	mudah untuk mengoperasikanya.				
10	Menurut saya, konten yang ada pada tugas akhir ini sangat sesuai dengan kebutuhan Teknik Informatika ITS.	60%	40%		

Penguji Perangkat lunak Digital Signage ini adalah orang-orang yang pernah belum pernah menggunakan aplikasi Digital Singage Interaktif dan terintegrasi untuk melakukan *process* manajemen data maupun penggunaan dari sisi *client*. Daftar penguji Perangkat Lunak Digital Singage ini dapat dilihat pada Tabel 5.5. Hasil rekap kuesioner dapat dilihat pada Tabel 5.6. Hasil rekap kuesioner menunjukkan rangkuman jawaban dari seluruh penguji pada masing-masing pertanyaan.

5.4. Evaluasi Pengujian

Pada subab ini akan diberikan hasil evaluasi dari pengujian-pengujian yang telah dilakukan. Evaluasi yang diberikan meliputi evaluasi pengujian kebutuhan fungsional dan *non*-fungsional yang pada dilihat pada bab 5.3.1 dan 5.3.2 serta evaluasi pengujian kegunaan yang dapat dilihat pada bab 5.3.3.

5.4.1. Evaluasi Pengujian Fungsionalitas

Rangkuman mengenai hasil pengujian fungsionalitas dapat dilihat padaTabell 5.27. Berdasarkan data pada tabel tersebut, semua skenario pengujian berhasil dan program berjalan dengan baik. Sehingga bisa ditarik disimpulkan bahwa fungsionalitas dari program telah bisa bekerja sesuai dengan yang diharapkan. dapat diberikan paparan evaluasi sebagai berikut:

1) Proses menambahkan akun ke dalam basis data sistem telah berhasil dan berjalan seperti yang diharapkan. Hal ini terlihat pada pengujian [TA-UJI-0101], dimana detil pengujian dipaparkan secara rinci.
2) Proses menambahkan *screen* ke dalam basis data sistem telah berhasil dan berjalan seperti yang diharapkan. Hal ini terlihat pada pengujian [TA-UJI-0201], dimana detil pengujian dipaparkan secara rinci.
3) Proses menambahkan *survey* ke dalam basis data sistem telah berhasil dan berjalan seperti yang diharapkan. Hal ini terlihat pada pengujian [TA-UJI-0301], dimana detil pengujian dipaparkan secara rinci.
4) Proses melakukan *scheduling* ke dalam basis data sistem telah berhasil dan berjalan seperti yang diharapkan. Hal ini terlihat pada pengujian [TA-UJI-0401], dimana detil pengujian dipaparkan secara rinci.

5.4.2. Evaluasi Pengujian Kegunaan

Berdasarkan hasil kuesioner pada, dapat ditarik kesimpulan bahwa aplikasi web pada perangkat lunak *Digital Singnage* cukup mudah untuk digunakan dioperasikan hal ini dapat dilihat pada kuisioner dengan komposisi 90% sangat setuju dan 10% setuju sedangkan untuk memnuhi fungsionalitas dengan komposisi 64% sangat setuju dan 36% setuju diambil berdasarkan rata-rata pertanyaan 2 sampai 6, sedangkan untuk aplikasi *client-desktop* pada perangkat lunak Digital Signage membuktikan bahwa juga cukup mudah digunakan serta sesuai fungsionalitas dengan komposisi 75% sangat setuju dan 25% setuju., dan yang terakhir kesesuaian dengan kebutuhan kampus dengan komposisi 60% sangat setuju dan 40% setuju.

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan Tugas Akhir, saran-saran tentang pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan dating, dan kegunaan lanjutan dari hasil yang didapat dari Tugas Akhir ini.

6.1. Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian perangkat lunak yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Aplikasi *web* pada perangkat lunak Digital Signage ini telah berhasil melakukan pembuatan, pengelolaan dan manajemen sebuah data semua *user* yang dibutuhkan guna keperluan sistem pada perangkat lunak Digital Signage.
2. Aplikasi *web* pada perangkat lunak Digital Signage ini telah berhasil melakukan proses pembuatan, pengelolaan, dan manajemen semua konten yang dibutuhkan dalam sistem pada perangkat lunak *Digital Singage.*
3. Aplikasi *web* pada perangkat lunak Digital Signage ini telah berhasil melakukan proses pembuatan, pengelolaan, dan manajemen kebutuhan-kebutuhan fungsionalitas yang bersifat tambahan seperti *survey* dan analisa *report* sebagai penunjang nilai guna perangkat lunak *Digital Singage*.
4. Aplikasi *Client-Desktop* yang dibangun pada perangkat lunak Digital Signage ini telah terintegrasi dengan fasilitas Kinect SDK sebagai media kontroler, sehingga aplikasi lebih interaktif, inovatif, dan menarik.
5. Perangkat lunak yang dibangun telah terintegrasi dan dapat diakses secara *online* karena telah dirancang berbasis *Server-Client* dan *web services.*

6. Perangkat lunak yang dibangun telah memenuhi kebutuhan dalam hal informasi maupun proses pengembangan teknologi dalam lingkungan kampus Teknik Informatika ITS Surabaya.

6.2. Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi dan pengujian yang telah dilakukan.

1. Dengan pengembangan lebih lanjut pada sistem sehingga dapat digunakan oleh seluruh jurusan di ITS dengan menyesuaikan kebutuhan informasi dan keadaan lingkungan.
2. Dengan pengembangan lebih lanjut dalam hal analisa dari kebutuhan-kebutuhan *user* yang bisa didapat melalui kuisioner secara terintegrasi.
3. Penambahan fungsionalitas dalam hal masukkan yang lebih beragam dan menarik namun tetap *compatible* dengan fungsionalitas utama dari perangkat lunak Digital Signage ini.
4. Karena sistem ini bersifat *online* dan erat kaitanya dengan data dan perpindahanya sehingga dapat dikembangkan pula sistem penunjang lainya seperti dalam sisi per*forma* (*Hardware*, *server*, optimasi) maupun dalam sisi kemanan jaringan

DAFTAR PUSTAKA

[1] Ralph F. Grove, Eray Ozkan, "THE MVC-WEB DESIGN PATTERN," 2011.

[2] wikipedia, "Model View ViewModel - wikipedia, the free encyclopedia," May 2014. [*Online*].

[3] Shamlia, "Understanding the basics of MVVM design pattern," March 2013. [*Online*].

[4] stackoverflow, "Design Patterns used in WPF - Stackoverflow," December 2013. [*Online*].

[5] Wikipedia, "ASP.NET - wikipedia, the free encyclopedia," May 2014. [*Online*].

[6] M. Support, Microsoft, 2014. [*Online*]. Available: http://msdn.microsoft.com/en-us/library/ms731082%28v=vs.110%29.aspx.

[7] T. o. M. ASP.Net, "ASP.NET MVC 5," 2014. [*Online*].

[8] Wikipedia, "Internet Information Services - wikipedia, the free encyclopedia," May 2014. [*Online*].

[9] M. Support, "White Paper: Technical Overview of Internet Information Services (IIS) 6.0," 2014. [*Online*].

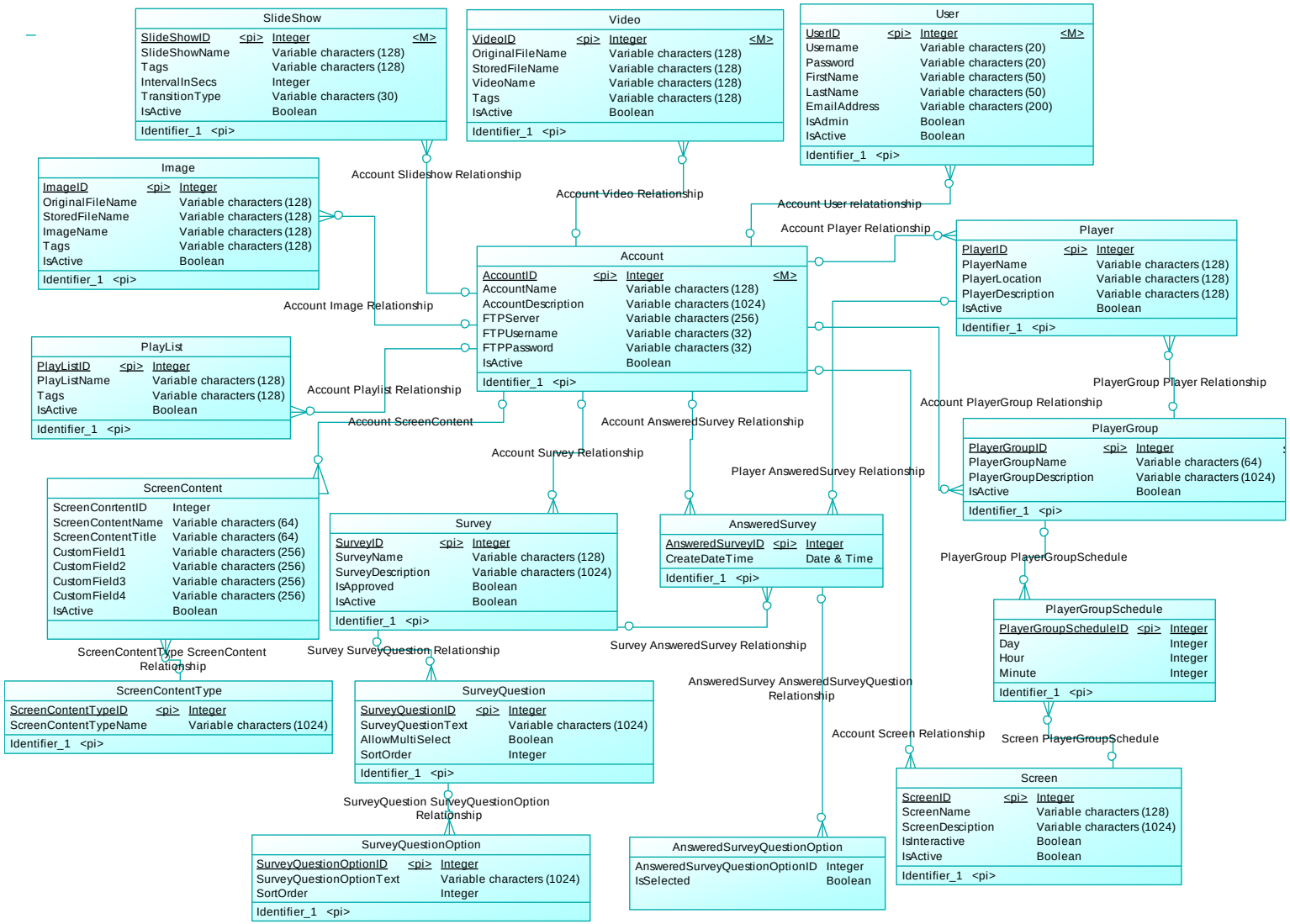
[10] M. N. Center, "Microsoft Releases SQL Server 2012 to Help Customers Manage “Any Data, Any Size, Anywhere”," March 2012.

[11] M. Ambitabha, G. Prithwijit, "Skeletal Tracking using Microsoft Kinect," 2012.

BIODATA PENULIS

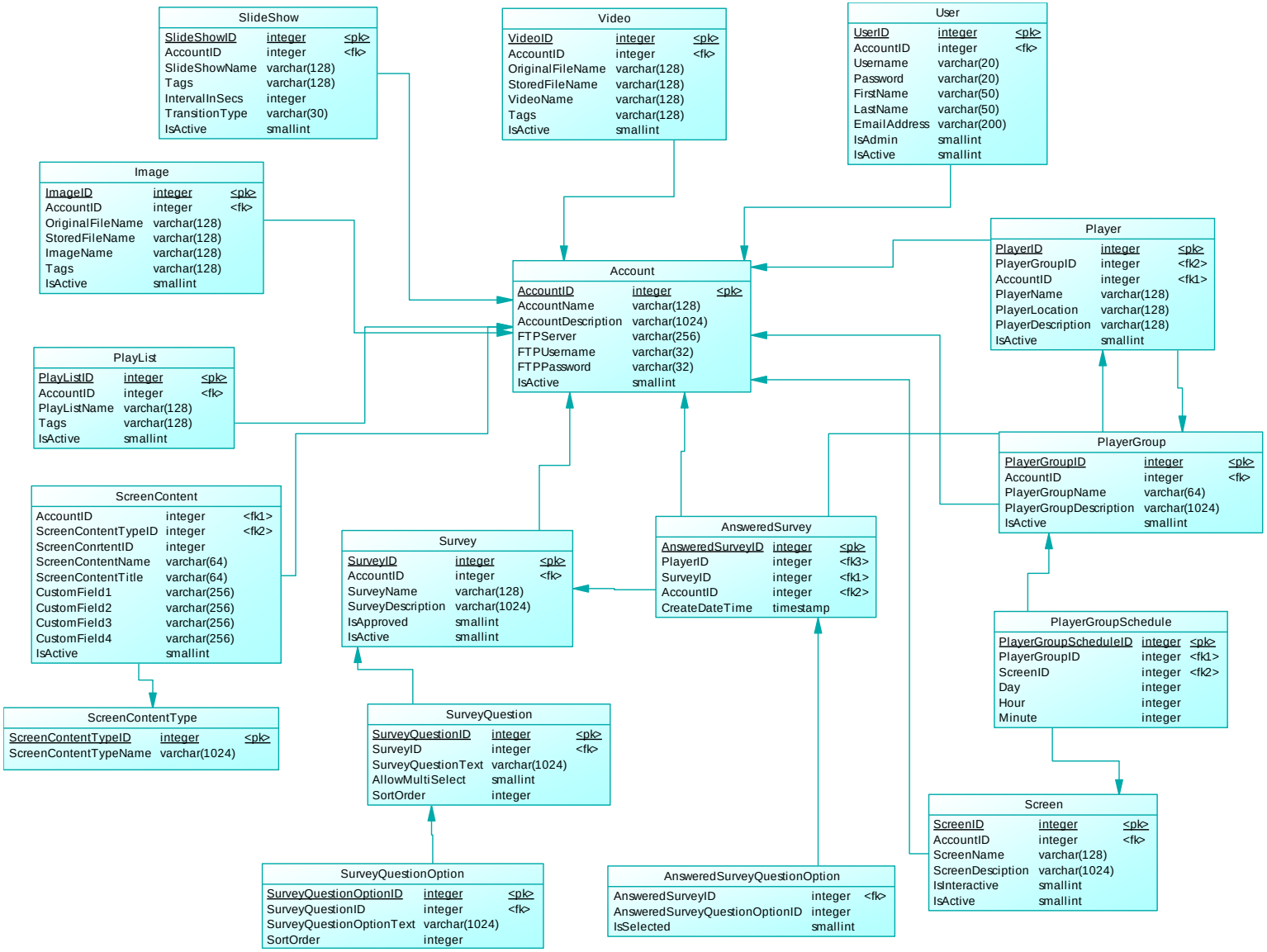
Penulis, **Ramadhani Tegar Perkasa**, lahir di Surabaya, 17 Maret 1992. Penulis menempuh pendidikan dasar mulai kelas 1 sampai 6 di SD Muhammadiyah IV Surabaya. Untuk pendidikan menengah, penulis tempuh di SMP Negeri 29 Surabaya dan selanjutnya di SMA Negeri 1 Surabaya. Selanjutnya penulis melanjutkan pendidikan sarjana di Jurusan Teknik Informatika, Institut Teknologi Sepuluh Nopember Surabaya. Penulis dalam menyelesaikan pendidikan S1 mengambil bidang minat Rekayasa Perangkat Lunak (*Software Engineering*) dan memiliki ketertarikan di bidang *Mobile Programming*, *Game Development*, *Web Development* dan *Desktop Software Development.* Penghargaan yang pernah diraih penulis selama duduk di bangku perkuliahan adalah 5 Besar pengembangan aplikasi SNITCH, Finalis GEMASTIK VI kategori Inovasi Perangkat Lunak, dan juara 5 Pengembangan Aplikasi Mobile UMDC MUGI (Microsoft *User* Group Indonesia). Selama perkuliahan penulis juga aktif dalam organisasi kemahasiswaan seperti Himpunan Mahasiswa Teknik Computer (HMTC), dan Badan Eksekutif Mahasiswa (BEM) FTIF. Penulis dapat dihubungi melalui email: rtegarp@gmail.com dan tegar10@mhs.if.its.ac.id

LAMPIRAN A. PERANCANGAN



Lampiran A.1 Perancagan *Conceptual Data Model*

[Halaman ini sengaja dikosongkan]



Lampiran A.2 Perancangan Physical Data *Model*

[Halaman ini sengaja dikosongkan]

LAMPIRAN B. IMPLEMENTASI

```
public class LoginController : Controller
    {
        ILoginRepository repository;

        public LoginController() : this(new EntityLoginRepository())
        { }

        public LoginController(ILoginRepository paramrepository)..

        // GET: /Login/
        public ActionResult Validate()
        {
            try
            {
                Session.Abandon();

                // Redirect to https is required
                if (ConfigurationManager.AppSettings["RequireHTTPS"] ==
"true")
                {
                    if (Request.Url.AbsoluteUri.StartsWith("http://"))

Response.Redirect(Request.Url.AbsoluteUri.Replace("http://", "https://"));
                }

                // Display the free links if appropriate
                ViewData["FreeLinks"] = "";
                if (ConfigurationManager.AppSettings["ShowFreeLinks"] ==
"true")
                    ViewData["FreeLinks"] = BuildFreeLinks();

                // Display the system messages, if any
                ViewData["SystemMessages"] = BuildSystemMessages();

                ViewData["Username"] = String.Empty;
                ViewData["Password"] = String.Empty;
                ViewData["ValidationMessage"] = String.Empty;
                ViewData["LoginInfo"] = "Please log in.";

                return View();
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Login", "Validate",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
```

```
            }
        }

        // POST: /Login/
        [HttpPost]
        public ActionResult Validate(FormCollection collection)
        {
            try
            {
                // Validate the login
                User user =
repository.ValidateLogin(Request.Form["txtUsername"].ToString(),
Request.Form["txtPassword"].ToString());

                ViewData["FreeLinks"] = "";
                if (ConfigurationManager.AppSettings["ShowFreeLinks"] ==
"true")
                    ViewData["FreeLinks"] = BuildFreeLinks();

                // Display the system messages, if any
                ViewData["SystemMessages"] = BuildSystemMessages();

                if (user == null)
                {
                    ViewData["Username"] =
Request.Form["txtUsername"].ToString();
                    ViewData["Password"] = String.Empty;
                    ViewData["ValidationMessage"] = "Invalid Login. Please
try again.";
                    ViewData["LoginInfo"] = "Please log in.";

                    return View();
                }
                else
                {
                    Session["User"] = user;
                    Session["UserAccountID"] = user.AccountID;

                    IAccountRepository acctrep = new
EntityAccountRepository();
                    Account account = acctrep.GetAccount(user.AccountID);
                    Session["UserAccountName"] = account.AccountName;

                    // Make sure the Account Folders exist
                    string serverpath = Server.MapPath("~/UploadedFiles");
                    if (!serverpath.EndsWith(@"\"))
                        serverpath += @"\";
                    System.IO.Directory.CreateDirectory(serverpath +
user.AccountID.ToString() + @"\Images");
```

```
                    System.IO.Directory.CreateDirectory(serverpath +
user.AccountID.ToString() + @"\Videos");
                    System.IO.Directory.CreateDirectory(serverpath +
user.AccountID.ToString() + @"\Music");

                    serverpath = Server.MapPath("~/Media");
                    if (!serverpath.EndsWith(@"\"))
                        serverpath += @"\";
                    System.IO.Directory.CreateDirectory(serverpath +
user.AccountID.ToString() + @"\Images");
                    System.IO.Directory.CreateDirectory(serverpath +
user.AccountID.ToString() + @"\Videos");
                    System.IO.Directory.CreateDirectory(serverpath +
user.AccountID.ToString() + @"\Music");

                    // If no player groups have been defined
IPlayerGroupRepository pgrep = new EntityPlayerGroupRepository();
                    IEnumerable<PlayerGroup> pgs =
pgrep.GetAllPlayerGroups(account.AccountID);
                    if (pgs == null || pgs.Count() == 0)
                    {
                        acctrep.CreateSampleData(account.AccountID);
                    }

                    // Log the login
                    ILoginLogRepository llrep = new
EntityLoginLogRepository();
                    LoginLog loginlog = new LoginLog();
                    loginlog.AccountID = user.AccountID;
                    loginlog.UserID = user.UserID;
                    loginlog.Username = user.Username;
                    loginlog.LoginDateTime =
DateTime.Now.ToUniversalTime();
                    llrep.CreateLoginLog(loginlog);
                    return RedirectToAction("Index", "PlayerGroup");
                }
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Login", "Validate POST",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
            }
        }
        private string BuildFreeLinks()
        {
            try
            {
```

```
                StringBuilder links = new StringBuilder();

                links.Append("<span id=\"freelink\"><a
href=\"http://www.vodigi.com/VodigiAccountSignUp.aspx\"
target=\"_blank\">New to Vodigi? Click to Request a New
Account.</a></span>");
                links.Append("<br />");
                links.Append("<span id=\"freelink\"><a
href=\"http://free.vodigi.com/osVodigiPlayerSetup52.msi\"
target=\"_blank\">Download the Vodigi Player Installer</a></span>");
                links.Append("<br />");
                links.Append("<span id=\"freelink\"><a
href=\"http://www.vodigi.com/VodigiPlayerDeviceConfiguration.pdf\"
target=\"_blank\">View the Vodigi Player Configuration Guide</a></span>");
                links.Append("<br />");
                links.Append("<span id=\"freelink\"><a
href=\"http://www.youtube.com/user/VodigiDigitalSignage?ob=0&feature=result
s_main\" target=\"_blank\">View Videos to Help You Get
Started</a></span>");

                return links.ToString();
            }
            catch { return String.Empty; }

        }
        private string BuildSystemMessages()..
}
```

Lampiran B.1 Implementasi Source Code Login*Controller*

```
public class AccountController : Controller
    {
        IAccountRepository repository;

        public AccountController() : this(new EntityAccountRepository())
        { }

        public AccountController(IAccountRepository paramrepository)..

        // GET: /Account/
        public ActionResult Index()
        {
            try
            {
                if (Session["UserAccountID"] == null)
                    return RedirectToAction("Validate", "Login");
                User user = (User)Session["User"];
```

```
                ViewData["LoginInfo"] =
Utility.BuildUserAccountString(user.Username,
Convert.ToString(Session["UserAccountName"]));
                if (user.IsAdmin)
                    ViewData["txtIsAdmin"] = "true";
                else
                    throw new Exception("You are not authorized to access
this page.");

                // Initialize or get the page state using session
                AccountPageState pagestate = GetPageState();

                // Set and save the page state to the submitted form values
if any values are passed
                if (Request.Form["lstAscDesc"] != null)
                {
                    pagestate.AccountName =
Request.Form["txtAccountName"].ToString().Trim();
                    pagestate.Description =
Request.Form["txtDescription"].ToString().Trim();
                    if
(Request.Form["chkIncludeInactive"].ToLower().StartsWith("true"))
                        pagestate.IncludeInactive = true;
                    else
                        pagestate.IncludeInactive = false;
                    pagestate.SortBy =
Request.Form["lstSortBy"].ToString().Trim();
                    pagestate.AscDesc =
Request.Form["lstAscDesc"].ToString().Trim();
                    pagestate.PageNumber =
Convert.ToInt32(Request.Form["txtPageNumber"].ToString().Trim());
                    SavePageState(pagestate);
                }

                // Add the session values to the view data so they can be
populated in the form
                ViewData["AccountName"] = pagestate.AccountName;
                ViewData["Description"] = pagestate.Description;
                ViewData["IncludeInactive"] = pagestate.IncludeInactive;
                ViewData["SortBy"] = pagestate.SortBy;
                ViewData["SortByList"] = new SelectList(BuildSortByList(),
"Value", "Text", pagestate.SortBy);
                ViewData["AscDescList"] = new
SelectList(BuildAscDescList(), "Value", "Text", pagestate.AscDesc);

                // Determine asc/desc
                bool isdescending = false;
                if (pagestate.AscDesc.ToLower().StartsWith("d"))
```

```
                    isdescending = true;

                // Get a Count of all filtered records
                int recordcount =
repository.GetAccountRecordCount(pagestate.AccountName,
pagestate.Description, pagestate.Include Inactive);

                // Determine the page count
                int pagecount = 1;
                if (recordcount > 0)
                {
                    pagecount = recordcount / Constants.PageSize;
                    if (recordcount % Constants.PageSize != 0) // Add a
page if there are more records
                    {
                        pagecount = pagecount + 1;
                    }
                }

                // Make sure the current page is not greater than the page
count
                if (pagestate.PageNumber > pagecount)
                {
                    pagestate.PageNumber = pagecount;
                    SavePageState(pagestate);
                }

                // Set the page number and account in viewdata
                ViewData["PageNumber"] =
Convert.ToString(pagestate.PageNumber);
                ViewData["PageCount"] = Convert.ToString(pagecount);
                ViewData["RecordCount"] = Convert.ToString(recordcount);
                ViewResult result =
View(repository.GetAccountPage(pagestate.AccountName,
pagestate.Description, pagestate.Include Inactive, pagestate.SortBy,
isdescending, pagestate.PageNumber, pagecount));
                result.ViewName = "Index";
                return result;
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Account", "Index",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
            }
        }

        // GET: /Account/Create
        public ActionResult Create()
```

```
        {
            try
            {
                if (Session["UserAccountID"] == null)
                    return RedirectToAction("Validate", "Login");
                User user = (User)Session["User"];
                ViewData["LoginInfo"] =
Utility.BuildUserAccountString(user.Username,
Convert.ToString(Session["UserAccountName"]));
                if (user.IsAdmin)
                    ViewData["txtIsAdmin"] = "true";
                else
                    throw new Exception("You are not authorized to access
this page.");

                ViewData["ValidationMessage"] = String.Empty;
                return View(CreateNewAccount());
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Account", "Create",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
            }
        }

        // POST: /Account/Create
        [HttpPost]
        public ActionResult Create(Account account)
        {
            try
            {
                if (Session["UserAccountID"] == null)
                    return RedirectToAction("Validate", "Login");
                User user = (User)Session["User"];
                ViewData["LoginInfo"] =
Utility.BuildUserAccountString(user.Username,
Convert.ToString(Session["UserAccountName"]));
                if (user.IsAdmin)
                    ViewData["txtIsAdmin"] = "true";
                Else throw new Exception("You are not authorized to access
this page.");
                if (ModelState.IsValid)
                {
                    // Set NULLs to Empty Strings
                    account = FillNulls(account);
                    string validation = ValidateInput(account);
                    if (!String.IsNullOrEmpty(validation))
```

```
                    {
                        ViewData["ValidationMessage"] = validation;
                        return View(account);
                    }
                    repository.CreateAccount(account);
                    repository.CreateSampleData(account.AccountID);
                    CommonMethods.CreateActivityLog((User) Session["User"],
"Account", "Add", "Added account '" + account.AccountName + "' - ID: " +
account.AccountID.ToString());

                    return RedirectToAction("Index");
                }
                return View(account);
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Account", "Create POST",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
            }
        }

        // GET: /Account/Edit/5
        public ActionResult Edit(int id)
        {
            try
            {
                if (Session["UserAccountID"] == null)
                    return RedirectToAction("Validate", "Login");
                User user = (User)Session["User"];
                ViewData["LoginInfo"] =
Utility.BuildUserAccountString(user.Username,
Convert.ToString(Session["UserAccountName"]));
                if (user.IsAdmin)
                    ViewData["txtIsAdmin"] = "true";
                else
                    throw new Exception("You are not authorized to access
this page.");

                Account account = repository.GetAccount(id);
                ViewData["ValidationMessage"] = String.Empty;

                return View(account);
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Account", "Edit",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
```

```
            }
        }

        // POST: /Account/Edit/5
        [HttpPost]
        public ActionResult Edit(Account account)
        {
            try
            {
                if (Session["UserAccountID"] == null)
                    return RedirectToAction("Validate", "Login");
                User user = (User)Session["User"];
                ViewData["LoginInfo"] =
Utility.BuildUserAccountString(user.Username,
Convert.ToString(Session["UserAccountName"]));
                if (user.IsAdmin)
                    ViewData["txtIsAdmin"] = "true";
                else
                    throw new Exception("You are not authorized to access
this page.");
                if (ModelState.IsValid)
                {
                    // Set NULLs to Empty Strings
                    account = FillNulls(account);
                    string validation = ValidateInput(account);
                    if (!String.IsNullOrEmpty(validation))
                    {
                        ViewData["ValidationMessage"] = validation;
                        return View(account);
                    }
                    repository.UpdateAccount(account);
                    CommonMethods.CreateActivityLog((User)Session["User"],
"Account", "Edit", "Edited account '" + account.AccountName + "' - ID: " +
account.AccountID.ToString());

                    return RedirectToAction("Index");
                }

                return View(account);
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Account", "Edit POST",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
            }
        }
```

```
        // Support Methods
        private List<SelectListItem> BuildSortByList()..
        private List<SelectListItem> BuildAscDescList()..
        private AccountPageState GetPageState()..
        private void SavePageState(AccountPageState pagestate)..
        private Account FillNulls(Account account)..
        private string ValidateInput(Account account)..
        private Account CreateNewAccount()
        {
            Account account = new Account();
            account.AccountID = 0;
            account.AccountName = String.Empty;
            account.AccountDescription = String.Empty;
            account.FTPServer = String.Empty;
            account.FTPUsername = String.Empty;
            account.FTPPassword = String.Empty;
            account.IsActive = true;
            return account;
        }
    }
```

Lampiran B.2 Implementasi *Source Code* `AccountController`

```
public class ScreenController : Controller
    {
        IScreenRepository repository;
        string firstfile = String.Empty;
        string selectedfile = String.Empty;

        public ScreenController() : this(new EntityScreenRepository())
        { }

        public ScreenController(IScreenRepository paramrepository)..

        // GET: /Screen/
        public ActionResult Index()
        {
            try
            {
                if (Session["UserAccountID"] == null)
                    return RedirectToAction("Validate", "Login");
                User user = (User)Session["User"];
                ViewData["LoginInfo"] =
Utility.BuildUserAccountString(user.Username,
Convert.ToString(Session["UserAccountName"]));
                if (user.IsAdmin)
                    ViewData["txtIsAdmin"] = "true";
                else
```

```
                    ViewData["txtIsAdmin"] = "false";

                // Initialize or get the page state using session
                ScreenPageState pagestate = GetPageState();

                // Get the account id
                int accountid = 0;
                if (Session["UserAccountID"] != null)
                    accountid = Convert.ToInt32(Session["UserAccountID"]);

                // Set and save the page state to the submitted form values
if any values are passed
                if (Request.Form["lstAscDesc"] != null)
                {
                    pagestate.AccountID = accountid;
                    pagestate.ScreenName =
Request.Form["txtScreenName"].ToString().Trim();
                    pagestate.Description =
Request.Form["txtDescription"].ToString().Trim();
                    if
(Request.Form["chkIncludeInactive"].ToLower().StartsWith("true"))
                        pagestate.IncludeInactive = true;
                    else
                        pagestate.IncludeInactive = false;
                    pagestate.SortBy =
Request.Form["lstSortBy"].ToString().Trim();
                    pagestate.AscDesc =
Request.Form["lstAscDesc"].ToString().Trim();
                    pagestate.PageNumber =
Convert.ToInt32(Request.Form["txtPageNumber"].ToString().Trim());
                    SavePageState(pagestate);
                }

                // Add the session values to the view data so they can be
populated in the form
                ViewData["AccountID"] = pagestate.AccountID;
                ViewData["ScreenName"] = pagestate.ScreenName;
                ViewData["Description"] = pagestate.Description;
                ViewData["IncludeInactive"] = pagestate.IncludeInactive;
                ViewData["SortBy"] = pagestate.SortBy;
                ViewData["SortByList"] = new SelectList(BuildSortByList(),
"Value", "Text", pagestate.SortBy);
                ViewData["AscDescList"] = new
SelectList(BuildAscDescList(), "Value", "Text", pagestate.AscDesc);

                // Determine asc/desc
                bool isdescending = false;
                if (pagestate.AscDesc.ToLower().StartsWith("d"))
```

```
                    isdescending = true;

                // Get a Count of all filtered records
                int recordcount =
repository.Get[Screen]RecordCount(pagestate.AccountID, pagestate.[Screen]Name,
pagestate.Description, pagestate.Include[Inactive]);

                // Determine the page count
                int pagecount = 1;
                if (recordcount > 0)
                {
                    pagecount = recordcount / Constants.PageSize;
                    if (recordcount % Constants.PageSize != 0) // Add a
page if there are more records
                    {
                        pagecount = pagecount + 1;
                    }
                }

                // Make sure the current page is not greater than the page
count
                if (pagestate.PageNumber > pagecount)
                {
                    pagestate.PageNumber = pagecount;
                    [Save]PageState(pagestate);
                }

                // Set the page number and account in viewdata
                ViewData["PageNumber"] =
Convert.ToString(pagestate.PageNumber);
                ViewData["PageCount"] = Convert.ToString(pagecount);
                ViewData["RecordCount"] = Convert.ToString(recordcount);

                // We need to add the Slide Show and/or Play List name
                IEnumerable<[Screen]> screens =
repository.Get[Screen]Page(pagestate.AccountID, pagestate.[Screen]Name,
pagestate.Description, pagestate.Include[Inactive], pagestate.SortBy,
isdescending, pagestate.PageNumber, pagecount);
                List<[Screen]View> screenviews = new List<[Screen]View>();
                I[Slideshow]Repository ssrep = new
Entity[Slideshow]Repository();
                I[Playlist]Repository plrep = new Entity[Playlist]Repository();
                foreach ([Screen] screen in screens)
                {
                    [Screen]View sv = new [Screen]View();
                    sv.[Screen]ID = screen.[Screen]ID;
                    sv.AccountID = screen.AccountID;
                    sv.[Screen]Name = screen.[Screen]Name;
                    sv.[Screen]Description = screen.[Screen]Description;
```

```
                    sv.SlideshowID = screen.SlideshowID;
                    if (sv.SlideshowID > 0)
                    {
                        Slideshow ss = ssrep.GetSlideshow(sv.SlideshowID);
                        sv.SlideshowName = ss.SlideshowName;
                    }
                    else
                    {
                        sv.SlideshowName = String.Empty;
                    }
                    sv.PlaylistID = screen.PlaylistID;
                    if (sv.PlaylistID > 0)
                    {
                        Playlist pl = plrep.GetPlaylist(sv.PlaylistID);
                        sv.PlaylistName = pl.PlaylistName;
                    }
                    sv.IsInteractive = screen.IsInteractive;
                    sv.ButtonImageID = screen.ButtonImageID;
                    sv.IsActive = screen.IsActive;

                    screenviews.Add(sv);
                }

                ViewResult result = View(screenviews);
                result.ViewName = "Index";
                return result;
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Screen", "Index",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
            }
        }

        // GET: /Screen/Create
        public ActionResult Create()
        {
            try
            {
                if (Session["UserAccountID"] == null)
                    return RedirectToAction("Validate", "Login");
                User user = (User)Session["User"];
                ViewData["LoginInfo"] =
Utility.BuildUserAccountString(user.Username,
Convert.ToString(Session["UserAccountName"]));
                if (user.IsAdmin)
                    ViewData["txtIsAdmin"] = "true";
```

```
                else
                    ViewData["txtIsAdmin"] = "false";

                ViewData["ValidationMessage"] = String.Empty;
                ViewData["ImageList"] = new SelectList(BuildImageList(""),
"Value", "Text", "");
                ViewData["ImageUrl"] = firstfile;
                ViewData["SlideshowList"] = new
SelectList(BuildSlideshowList(), "Value", "Text", "");
                ViewData["PlaylistList"] = new
SelectList(BuildPlaylistList(), "Value", "Text", "");
                ViewData["ScreenContentList"] = new
SelectList(BuildScreenContentList(), "Value", "Text", "");
                ViewData["ScreenScreenContentList"] = new
SelectList(BuildScreenScreenContentList(0), "Value", "Text", "");
                ViewData["ScreenScreenContent"] = String.Empty;

                int accountid = 0;
                if (Session["UserAccountID"] != null)
                    accountid = Convert.ToInt32(Session["UserAccountID"]);
                ViewData["ImageFolder"] =
ConfigurationManager.AppSettings["MediaRootFolder"] +
Convert.ToString(Session["UserAccountID"]) + @"/Images/";

                return View(CreateNewScreen());
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Screen", "Create",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
            }
        }

        // POST: /Screen/Create
        [HttpPost]
        public ActionResult Create(Screen screen)
        {
            try
            {
                if (Session["UserAccountID"] == null)
                    return RedirectToAction("Validate", "Login");
                User user = (User)Session["User"];
                ViewData["LoginInfo"] =
Utility.BuildUserAccountString(user.Username,
Convert.ToString(Session["UserAccountName"]));
                if (user.IsAdmin)
                    ViewData["txtIsAdmin"] = "true";
                else
```

```
                    ViewData["txtIsAdmin"] = "false";

                if (ModelState.IsValid)
                {
                    // Set NULLs to Empty Strings
                    screen = FillNulls(screen);
                    screen.AccountID =
Convert.ToInt32(Session["UserAccountID"]);
                    screen.SlideshowID =
Convert.ToInt32(Request.Form["lstSlideshow"]);
                    screen.PlaylistID =
Convert.ToInt32(Request.Form["lstPlaylist"]);
                    string buttonimageguid =
Request.Form["lstButtonImage"];

                    IImageRepository imgrep = new EntityImageRepository();
                    Image img = imgrep.GetImageByGuid(buttonimageguid);
                    if (img != null)
                        screen.ButtonImageID = img.ImageID;
                    else
                        screen.ButtonImageID = 0;

                    string validation = ValidateInput(screen);
                    if (!String.IsNullOrEmpty(validation))
                    {
                        ViewData["ValidationMessage"] = validation;
                        ViewData["ImageList"] = new
SelectList(BuildImageList(Request.Form["lstButtonImage"]), "Value", "Text",
Request.Form["lstButtonImage"]);
                        ViewData["ImageUrl"] = selectedfile;
                        ViewData["SlideshowList"] = new
SelectList(BuildSlideshowList(), "Value", "Text",
Request.Form["lstSlideshow"]);
                        ViewData["PlaylistList"] = new
SelectList(BuildPlaylistList(), "Value", "Text",
Request.Form["lstPlaylist"]);
                        ViewData["ScreenContentList"] = new
SelectList(BuildScreenContentList(), "Value", "Text", "");
                        ViewData["ScreenScreenContentList"] = new
SelectList(BuildScreenScreenContentList(screen.ScreenID), "Value", "Text",
"");
                        ViewData["ScreenScreenContent"] =
Request.Form["txtScreenScreenContent"];

                        int accountid = 0;
                        if (Session["UserAccountID"] != null)
                            accountid =
Convert.ToInt32(Session["UserAccountID"]);
```

```
                        ViewData["ImageFolder"] =
ConfigurationManager.AppSettings["MediaRootFolder"] +
Convert.ToString(Session["UserAccountID"]) + @"/Images/";

                        return View(screen);
                    }
                    else
                    {
                        repository.CreateScreen(screen);


CommonMethods.CreateActivityLog((User)Session["User"], "Screen", "Add",
                            "Added screen '" + screen.ScreenName + "' - ID:
" + screen.ScreenID.ToString());

                        // Create a xref for each screen content in the
screen
                        IScreenScreenContentXrefRepository sscrep = new
EntityScreenScreenContentXrefRepository();
                        string[] ids =
Request.Form["txtScreenScreenContent"].ToString().Split('|');
                        int i = 1;
                        foreach (string id in ids)
                        {
                            if (!String.IsNullOrEmpty(id.Trim()))
                            {
                                ScreenScreenContentXref ssc = new
ScreenScreenContentXref();
                                ssc.DisplayOrder = i;
                                ssc.ScreenID = screen.ScreenID;
                                ssc.ScreenContentID = Convert.ToInt32(id);
                                sscrep.CreateScreenScreenContentXref(ssc);
                                i += 1;
                            }
                        }

                        return RedirectToAction("Index");
                    }
                }

                return View(screen);
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Screen", "Create POST",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
            }
        }
```

```
        //
        // GET: /Screen/Edit/5
        public ActionResult Edit(int id)
        {
            try
            {
                if (Session["UserAccountID"] == null)
                    return RedirectToAction("Validate", "Login");
                User user = (User)Session["User"];
                ViewData["LoginInfo"] =
Utility.BuildUserAccountString(user.Username,
Convert.ToString(Session["UserAccountName"]));
                if (user.IsAdmin)
                    ViewData["txtIsAdmin"] = "true";
                else
                    ViewData["txtIsAdmin"] = "false";
                Screen screen = repository.GetScreen(id);
                ViewData["ValidationMessage"] = String.Empty;
                IImageRepository imgrep = new EntityImageRepository();
                Image img = imgrep.GetImage(screen.ButtonImageID);
                ViewData["ImageList"] = new
SelectList(BuildImageList(img.StoredFilename), "Value", "Text",
img.StoredFilename);
                ViewData["ImageUrl"] = selectedfile;
                ViewData["SlideshowList"] = new
SelectList(BuildSlideshowList(), "Value", "Text", screen.SlideshowID);
                ViewData["PlaylistList"] = new
SelectList(BuildPlaylistList(), "Value", "Text", screen.PlaylistID);
                ViewData["ScreenContentList"] = new
SelectList(BuildScreenContentList(), "Value", "Text", "");
                ViewData["ScreenScreenContentList"] = new
SelectList(BuildScreenScreenContentList(screen.ScreenID), "Value", "Text",
"");

                // Get the content ids for the screen
                string ids = String.Empty;
                IScreenScreenContentXrefRepository sscrep = new
EntityScreenScreenContentXrefRepository();
                IEnumerable<ScreenScreenContentXref> sscs =
sscrep.GetScreenScreenContentXrefs(screen.ScreenID);
                foreach (ScreenScreenContentXref ssc in sscs)
                {
                    ids += "|" + ssc.ScreenContentID.ToString();
                }
                ViewData["ScreenScreenContent"] = ids;

                int accountid = 0;
```

```
                if (Session["UserAccountID"] != null)
                    accountid = Convert.ToInt32(Session["UserAccountID"]);
                ViewData["ImageFolder"] =
ConfigurationManager.AppSettings["MediaRootFolder"] +
Convert.ToString(Session["UserAccountID"]) + @"/Images/";

                return View(screen);
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Screen", "Edit",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
            }
        }
        // POST: /Screen/Edit/5
        [HttpPost]
        public ActionResult Edit(Screen screen)
        {
            try
            {
                if (Session["UserAccountID"] == null)
                    return RedirectToAction("Validate", "Login");
                User user = (User)Session["User"];
                ViewData["LoginInfo"] =
Utility.BuildUserAccountString(user.Username,
Convert.ToString(Session["UserAccountName"]));
                if (user.IsAdmin)
                    ViewData["txtIsAdmin"] = "true";
                else
                    ViewData["txtIsAdmin"] = "false";

                if (ModelState.IsValid)
                {
                    // Set NULLs to Empty Strings
                    screen = FillNulls(screen);
                    screen.SlideshowID =
Convert.ToInt32(Request.Form["lstSlideshow"]);
                    screen.PlaylistID =
Convert.ToInt32(Request.Form["lstPlaylist"]);
                    string buttonimageguid =
Request.Form["lstButtonImage"];
                    IImageRepository imgrep = new EntityImageRepository();
                    Image img = imgrep.GetImageByGuid(buttonimageguid);
                    if (img != null)
                        screen.ButtonImageID = img.ImageID;
                    else
                        screen.ButtonImageID = 0;
```

```
                    string validation = ValidateInput(screen);
                    if (!String.IsNullOrEmpty(validation))
                    {
                        ViewData["ValidationMessage"] = validation;
                        ViewData["ImageList"] = new
SelectList(BuildImageList(Request.Form["lstButtonImage"]), "Value", "Text",
Request.Form["lstButtonImage"]);
                        ViewData["ImageUrl"] = selectedfile;
                        ViewData["SlideshowList"] = new
SelectList(BuildSlideshowList(), "Value", "Text",
Request.Form["lstSlideshow"]);
                        ViewData["PlaylistList"] = new
SelectList(BuildPlaylistList(), "Value", "Text",
Request.Form["lstPlaylist"]);

                        int accountid = 0;
                        if (Session["UserAccountID"] != null)
                            accountid =
Convert.ToInt32(Session["UserAccountID"]);
                        ViewData["ImageFolder"] =
ConfigurationManager.AppSettings["MediaRootFolder"] +
Convert.ToString(Session["UserAccountID"]) + @"/Images/";
                        return View(screen);
                    }
                    else
                    {
                        // Update the screen
                        repository.UpdateScreen(screen);
CommonMethods.CreateActivityLog((User)Session["User"], "Screen", "Edit",
                            "Edited screen '" + screen.ScreenName + "' -
ID: " + screen.ScreenID.ToString());
                        IScreenScreenContentXrefRepository xrefrep = new
EntityScreenScreenContentXrefRepository();

                        // Delete existing xrefs for the screen

xrefrep.DeleteScreenScreenContentXrefs(screen.ScreenID);

                        // Create a xref for each screen content in the
screen
                        IScreenScreenContentXrefRepository sscrep = new
EntityScreenScreenContentXrefRepository();
                        string[] ids =
Request.Form["txtScreenScreenContent"].ToString().Split('|');
                        int i = 1;
                        foreach (string id in ids)
                        {
                            if (!String.IsNullOrEmpty(id.Trim()))
```

```
                            {
                                ScreenScreenContentXref ssc = new
ScreenScreenContentXref();
                                ssc.DisplayOrder = i;
                                ssc.ScreenID = screen.ScreenID;
                                ssc.ScreenContentID = Convert.ToInt32(id);
                                sscrep.CreateScreenScreenContentXref(ssc);
                                i += 1;
                            }
                        }

                        return RedirectToAction("Index");
                    }
                }

                return View(screen);
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Screen", "Edit POST",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
            }
        }

        // Support Methods
        private List<SelectListItem> BuildSortByList()..
        private List<SelectListItem> BuildAscDescList()..
        private List<SelectListItem> BuildSlideshowList()
        {
            // Get the account id
            int accountid = 0;
            if (Session["UserAccountID"] != null)
                accountid = Convert.ToInt32(Session["UserAccountID"]);

            // Build the slide show list
            List<SelectListItem> items = new List<SelectListItem>();

            // Create a blank item
            SelectListItem blank = new SelectListItem();
            blank.Text = String.Empty;
            blank.Value = "0";
            items.Add(blank);
            ISlideshowRepository ssrep = new EntitySlideshowRepository();
            IEnumerable<Slideshow> sss = ssrep.GetAllSlideshows(accountid);
            foreach (Slideshow ss in sss)
            {
                SelectListItem item = new SelectListItem();
                item.Text = ss.SlideshowName;
```

```
                item.Value = ss.SlideshowID.ToString();
                items.Add(item);
            }

            return items;
        }

        private List<SelectListItem> BuildPlaylistList()
        {
            // Get the account id
            int accountid = 0;
            if (Session["UserAccountID"] != null)
                accountid = Convert.ToInt32(Session["UserAccountID"]);

            // Build the play list list
            List<SelectListItem> items = new List<SelectListItem>();

            // Create a blank item
            SelectListItem blank = new SelectListItem();
            blank.Text = String.Empty;
            blank.Value = "0";
            items.Add(blank);

            IPlaylistRepository plrep = new EntityPlaylistRepository();
            IEnumerable<Playlist> pls = plrep.GetAllPlaylists(accountid);
            foreach (Playlist pl in pls)
            {
                SelectListItem item = new SelectListItem();
                item.Text = pl.PlaylistName;
                item.Value = pl.PlaylistID.ToString();
                items.Add(item);
            }
            return items;
        }

        private List<SelectListItem> BuildScreenContentList()..
        private ScreenPageState GetPageState()..
        private void SavePageState(ScreenPageState pagestate)..
        private Screen FillNulls(Screen screen)..
        private Screen CreateNewScreen()..
        private string ValidateInput(Screen screen)..
    }
```

Lampiran B.3 Implementasi *Source Code* `ScreenController`

```
public class SurveyController : Controller
    {
        ISurveyRepository repository;
        string firstfile = String.Empty;
        string selectedfile = String.Empty;

        public SurveyController()
            : this(new EntitySurveyRepository())
        { }

        public SurveyController(ISurveyRepository paramrepository)..

        // GET: /Survey/
        public ActionResult Index()
        {
            try
            {
                if (Session["UserAccountID"] == null)
                    return RedirectToAction("Validate", "Login");
                User user = (User)Session["User"];
                ViewData["LoginInfo"] =
Utility.BuildUserAccountString(user.Username,
Convert.ToString(Session["UserAccountName"]));
                if (user.IsAdmin)
                    ViewData["txtIsAdmin"] = "true";
                else
                    ViewData["txtIsAdmin"] = "false";

                // Initialize or get the page state using session
                SurveyPageState pagestate = GetPageState();

                // Get the account id
                int accountid = 0;
                if (Session["UserAccountID"] != null)
                    accountid = Convert.ToInt32(Session["UserAccountID"]);

                // Set and save the page state to the submitted form values
if any values are passed
                if (Request.Form["lstAscDesc"] != null)
                {
                    pagestate.AccountID = accountid;
                    pagestate.SurveyName =
Request.Form["txtSurveyName"].ToString().Trim();
                    if
(Request.Form["chkOnlyApproved"].ToLower().StartsWith("true"))
                        pagestate.OnlyApproved = true;
                    else
                        pagestate.OnlyApproved = false;
```

```
                    if
(Request.Form["chkIncludeInactive"].ToLower().StartsWith("true"))
                        pagestate.IncludeInactive = true;
                    else
                        pagestate.IncludeInactive = false;
                    pagestate.SortBy =
Request.Form["lstSortBy"].ToString().Trim();
                    pagestate.AscDesc =
Request.Form["lstAscDesc"].ToString().Trim();
                    pagestate.PageNumber =
Convert.ToInt32(Request.Form["txtPageNumber"].ToString().Trim());
                    SavePageState(pagestate);
                }

                // Add the session values to the view data so they can be
populated in the form
                ViewData["AccountID"] = pagestate.AccountID;
                ViewData["SurveyName"] = pagestate.SurveyName;
                ViewData["OnlyApproved"] = pagestate.OnlyApproved;
                ViewData["IncludeInactive"] = pagestate.IncludeInactive;
                ViewData["SortBy"] = pagestate.SortBy;
                ViewData["SortByList"] = new SelectList(BuildSortByList(),
"Value", "Text", pagestate.SortBy);
                ViewData["AscDescList"] = new
SelectList(BuildAscDescList(), "Value", "Text", pagestate.AscDesc);

                // Determine asc/desc
                bool isdescending = false;
                if (pagestate.AscDesc.ToLower().StartsWith("d"))
                    isdescending = true;

                // Get a Count of all filtered records
                int recordcount =
repository.GetSurveyRecordCount(pagestate.AccountID, pagestate.SurveyName,
pagestate.OnlyApproved, pagestate.IncludeInactive);

                // Determine the page count
                int pagecount = 1;
                if (recordcount > 0)
                {
                    pagecount = recordcount / Constants.PageSize;
                    if (recordcount % Constants.PageSize != 0) // Add a
page if there are more records
                    {
                        pagecount = pagecount + 1;
                    }
                }
```

```
                // Make sure the current page is not greater than the page
count
                if (pagestate.PageNumber > pagecount)
                {
                    pagestate.PageNumber = pagecount;
                    SavePageState(pagestate);
                }

                // Set the page number and account in viewdata
                ViewData["PageNumber"] =
Convert.ToString(pagestate.PageNumber);
                ViewData["PageCount"] = Convert.ToString(pagecount);
                ViewData["RecordCount"] = Convert.ToString(recordcount);

                // Set the image folder
                ViewData["ImageFolder"] =
ConfigurationManager.AppSettings["MediaRootFolder"] +
Convert.ToString(Session["UserAccountID"]) + @"/Images/";

                ViewResult result =
View(repository.GetSurveyPage(pagestate.AccountID, pagestate.SurveyName,
pagestate.OnlyApproved, pagestate.IncludeInactive, pagestate.SortBy,
isdescending, pagestate.PageNumber, pagecount));
                result.ViewName = "Index";
                return result;
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Survey", "Index",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
            }
        }

        // GET: /Survey/Create
        public ActionResult Create()
        {
            try
            {
                if (Session["UserAccountID"] == null)
                    return RedirectToAction("Validate", "Login");
                User user = (User)Session["User"];
                ViewData["LoginInfo"] =
Utility.BuildUserAccountString(user.Username,
Convert.ToString(Session["UserAccountName"]));
                if (user.IsAdmin)
                    ViewData["txtIsAdmin"] = "true";
                else
                    ViewData["txtIsAdmin"] = "false";
```

```
                ViewData["ValidationMessage"] = String.Empty;
                ViewData["ImageList"] = new SelectList(BuildImageList(""),
"Value", "Text", "");
                ViewData["ImageUrl"] = firstfile;
                ViewData["ImageFolder"] =
ConfigurationManager.AppSettings["MediaRootFolder"] +
Convert.ToString(Session["UserAccountID"]) + @"/Images/";

                return View(CreateNewSurvey());
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Survey", "Create",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
            }
        }

        // POST: /Survey/Create
        [HttpPost]
        public ActionResult Create(Survey survey)
        {
            try
            {
                if (Session["UserAccountID"] == null)
                    return RedirectToAction("Validate", "Login");
                User user = (User)Session["User"];
                ViewData["LoginInfo"] =
Utility.BuildUserAccountString(user.Username,
Convert.ToString(Session["UserAccountName"]));
                if (user.IsAdmin)
                    ViewData["txtIsAdmin"] = "true";
                else
                    ViewData["txtIsAdmin"] = "false";

                if (ModelState.IsValid)
                {
                    // Set NULLs to Empty Strings
                    survey = FillNulls(survey);
                    survey.AccountID =
Convert.ToInt32(Session["UserAccountID"]);

                    IImageRepository imgrep = new EntityImageRepository();
                    Image img =
imgrep.GetImageByGuid(Request.Form["lstImage"]);
                    if (img != null)
                        survey.SurveyImageID = img.ImageID;
```

```
                    else
                        survey.SurveyImageID = 0;

                    string validation = ValidateInput(survey);
                    if (!String.IsNullOrEmpty(validation))
                    {
                        ViewData["ValidationMessage"] = validation;
                        ViewData["ImageList"] = new
SelectList(BuildImageList(Request.Form["lstImage"]), "Value", "Text",
Request.Form["lstImage"]);
                        ViewData["ImageUrl"] = selectedfile;
                        ViewData["ImageFolder"] =
ConfigurationManager.AppSettings["MediaRootFolder"] +
Convert.ToString(Session["UserAccountID"]) + @"/Images/";
                        return View(survey);
                    }
                    else
                    {
                        repository.CreateSurvey(survey);


CommonMethods.CreateActivityLog((User)Session["User"], "Survey", "Add",
                            "Added survey '" + survey.SurveyName + "' - ID:
" + survey.SurveyID.ToString());

                        return RedirectToAction("Edit", "Survey", new { id
= survey.SurveyID });
                    }
                }
                return View(survey);
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Survey", "Create POST",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
            }
        }

        // GET: /Survey/Edit/5
        public ActionResult Edit(int id)
        {
            try
            {
                if (Session["UserAccountID"] == null)
                    return RedirectToAction("Validate", "Login");
                User user = (User)Session["User"];
```

```
                ViewData["LoginInfo"] =
Utility.BuildUserAccountString(user.Username,
Convert.ToString(Session["UserAccountName"]));
                if (user.IsAdmin)
                    ViewData["txtIsAdmin"] = "true";
                else
                    ViewData["txtIsAdmin"] = "false";

                Survey survey = repository.GetSurvey(id);
                ViewData["ValidationMessage"] = String.Empty;

                IImageRepository imgrep = new EntityImageRepository();
                Image img = imgrep.GetImage(survey.SurveyImageID);
                ViewData["ImageList"] = new
SelectList(BuildImageList(img.StoredFilename), "Value", "Text",
img.StoredFilename);
                ViewData["ImageUrl"] = selectedfile;
                ViewData["ImageFolder"] =
ConfigurationManager.AppSettings["MediaRootFolder"] +
Convert.ToString(Session["UserAccountID"]) + @"/Images/";

                ViewData["SurveyTable"] = BuildSurveyTable(survey);

                return View(survey);
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Survey", "Edit",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
            }
        }

        // POST: /Survey/Edit/5
        [HttpPost]
        public ActionResult Edit(Survey survey)
        {
            try
            {
                if (Session["UserAccountID"] == null)
                    return RedirectToAction("Validate", "Login");
                User user = (User)Session["User"];
                ViewData["LoginInfo"] =
Utility.BuildUserAccountString(user.Username,
Convert.ToString(Session["UserAccountName"]));
                if (user.IsAdmin)
                    ViewData["txtIsAdmin"] = "true";
                else
```

```
                    ViewData["txtIsAdmin"] = "false";

                if (ModelState.IsValid)
                {
                    // Set NULLs to Empty Strings
                    survey = FillNulls(survey);

                    IImageRepository imgrep = new EntityImageRepository();
                    Image img =
imgrep.GetImageByGuid(Request.Form["lstImage"]);
                    if (img != null)
                        survey.SurveyImageID = img.ImageID;
                    else
                        survey.SurveyImageID = 0;

                    string validation = ValidateInput(survey);
                    if (!String.IsNullOrEmpty(validation))
                    {
                        ViewData["ValidationMessage"] = validation;
                        ViewData["ImageList"] = new
SelectList(BuildImageList(Request.Form["lstImage"]), "Value", "Text",
Request.Form["lstImage"]);
                        ViewData["ImageUrl"] = selectedfile;
                        ViewData["ImageFolder"] =
ConfigurationManager.AppSettings["MediaRootFolder"] +
Convert.ToString(Session["UserAccountID"]) + @"/Images/";

                        ViewData["SurveyTable"] = BuildSurveyTable(survey);
                        return View(survey);
                    }

                    repository.UpdateSurvey(survey);

                    CommonMethods.CreateActivityLog((User)Session["User"],
"Survey", "Edit",
                                                    "Edited survey '" +
survey.SurveyName + "' - ID: " + survey.SurveyID.ToString());

                    return RedirectToAction("Index");
                }

                return View(survey);
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Survey", "Edit POST",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
            }
```

```
        }

        // GET: /Survey/Approve/5
        public ActionResult Approve(int id)
        {
            try
            {
                if (Session["UserAccountID"] == null)
                    return RedirectToAction("Validate", "Login");
                User user = (User)Session["User"];
                ViewData["LoginInfo"] =
Utility.BuildUserAccountString(user.Username,
Convert.ToString(Session["UserAccountName"]));
                if (user.IsAdmin)
                    ViewData["txtIsAdmin"] = "true";
                else
                    ViewData["txtIsAdmin"] = "false";

                Survey survey = repository.GetSurvey(id);
                ViewData["ValidationMessage"] = String.Empty;

                IImageRepository imgrep = new EntityImageRepository();
                Image img = imgrep.GetImage(survey.SurveyImageID);
                ViewData["ImageList"] = new
SelectList(BuildImageList(img.StoredFilename), "Value", "Text",
img.StoredFilename);
                ViewData["ImageUrl"] = selectedfile;
                ViewData["ImageFolder"] =
ConfigurationManager.AppSettings["MediaRootFolder"] +
Convert.ToString(Session["UserAccountID"]) + @"/Images/";

                ViewData["SurveyTable"] = BuildSurveyTableNoLinks(survey);

                return View(survey);
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Survey", "Approve",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
            }
        }

        //
        // POST: /Survey/Approve/5

        [HttpPost]
        public ActionResult Approve(Survey survey)
```

```
        {
            try
            {
                if (Session["UserAccountID"] == null)
                    return RedirectToAction("Validate", "Login");
                User user = (User)Session["User"];
                ViewData["LoginInfo"] =
Utility.BuildUserAccountString(user.Username,
Convert.ToString(Session["UserAccountName"]));
                if (user.IsAdmin)
                    ViewData["txtIsAdmin"] = "true";
                else
                    ViewData["txtIsAdmin"] = "false";

                if (ModelState.IsValid)
                {
                    // Set NULLs to Empty Strings
                    survey = FillNulls(survey);

                    survey.IsApproved = true;
                    repository.UpdateSurvey(survey);

                    CommonMethods.CreateActivityLog((User)Session["User"],
"Survey", "Approve",
                                                    "Approved survey '" +
survey.SurveyName + "' - ID: " + survey.SurveyID.ToString());

                    return RedirectToAction("Index");
                }

                return View(survey);
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Survey", "Approve POST",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
            }
        }

        // GET: /Survey/View/5
        public ActionResult View(int id)
        {
            try
            {
                if (Session["UserAccountID"] == null)
                    return RedirectToAction("Validate", "Login");
                User user = (User)Session["User"];
```

```
                ViewData["LoginInfo"] =
Utility.BuildUserAccountString(user.Username,
Convert.ToString(Session["UserAccountName"]));
                if (user.IsAdmin)
                    ViewData["txtIsAdmin"] = "true";
                else
                    ViewData["txtIsAdmin"] = "false";

                Survey survey = repository.GetSurvey(id);
                ViewData["ValidationMessage"] = String.Empty;

                IImageRepository imgrep = new EntityImageRepository();
                Image img = imgrep.GetImage(survey.SurveyImageID);
                ViewData["ImageList"] = new
SelectList(BuildImageList(img.StoredFilename), "Value", "Text",
img.StoredFilename);
                ViewData["ImageUrl"] = selectedfile;
                ViewData["ImageFolder"] =
ConfigurationManager.AppSettings["MediaRootFolder"] +
Convert.ToString(Session["UserAccountID"]) + @"/Images/";

                ViewData["SurveyTable"] = BuildSurveyTableNoLinks(survey);

                return View(survey);
            }
            catch (Exception ex)
            {
                Helpers.SetupApplicationError("Survey", "View",
ex.Message);
                return RedirectToAction("Index", "ApplicationError");
            }
        }
        // Support Methods
        private List<SelectListItem> BuildSortByList()..
        private List<SelectListItem> BuildAscDescList()..
        private SurveyPageState GetPageState()..
        private void SavePageState(SurveyPageState pagestate)..
        private Survey CreateNewSurvey()..
        private Survey FillNulls(Survey survey)..

        private string BuildSurveyTable(Survey survey)
        {
            try
            {
                StringBuilder sb = new StringBuilder();

                string root = Request.Url.OriginalString.Substring(0,
Request.Url.OriginalString.ToLower().LastIndexOf("/survey/"));
```

```
                if (!root.EndsWith("/")) root += "/";

                sb.AppendLine("<table style=\"border-spacing:0;border-
collapse:collapse;\" class=\"surveytable\">");
                sb.AppendLine("<tr class=\"surveyrow\">");
                sb.AppendLine("<td class=\"gridtext\">" + survey.SurveyName
+ "</td>");
                sb.AppendLine("<td class=\"gridtext\"
style=\"width:110px;\"><span class=\"surveyquestionlink\"
onclick=\"window.location='" + root + "SurveyQuestion/Create/" +
survey.SurveyID.ToString() + "'\">Add Question</span></td>");
                sb.AppendLine("<td class=\"gridtext\"
style=\"width:25px;\"></td>");
                sb.AppendLine("<td class=\"gridtext\"
style=\"width:45px;\"></td>");
                sb.AppendLine("<td class=\"gridtext\"
style=\"width:35px;\"></td>");
                sb.AppendLine("<td class=\"gridtext\"
style=\"width:55px;\"></td>");
                sb.AppendLine("</tr>");

                // Loop through each question and question option
                ISurveyQuestionRepository qrep = new
EntitySurveyQuestionRepository();
                ISurveyQuestionOptionRepository orep = new
EntitySurveyQuestionOptionRepository();
                List<SurveyQuestion> questions =
qrep.GetSurveyQuestions(survey.SurveyID).ToList();

                foreach (SurveyQuestion question in questions)
                {
                    sb.AppendLine("<tr class=\"questionrow\">");
                    string selectionmode = " (Single Select)";
                    if (question.AllowMultiSelect) selectionmode = " (Multi
Select)";
                    sb.AppendLine("<td class=\"gridtext\">" +
question.SurveyQuestionText + selectionmode + "</td>");
                    sb.AppendLine("<td class=\"gridtext\"><span
class=\"surveyquestionoptionlink\" onclick=\"window.location='" + root +
"SurveyQuestionOption/Create/" + question.SurveyQuestionID.ToString() +
"'\">Add Option</span></td>");
                    sb.AppendLine("<td class=\"gridtext\" style=\"text-
align:center;\"><span class=\"surveyquestionoptionlink\"
onclick=\"window.location='" + root + "SurveyQuestion/MoveUp/" +
question.SurveyQuestionID.ToString() + "'\">Up</span></td>");
                    sb.AppendLine("<td class=\"gridtext\" style=\"text-
align:center;\"><span class=\"surveyquestionoptionlink\"
onclick=\"window.location='" + root + "SurveyQuestion/MoveDown/" +
question.SurveyQuestionID.ToString() + "'\">Down</span></td>");
```

```
                    sb.AppendLine("<td class=\"gridtext\" style=\"text-
align:center;\"><span class=\"surveyquestionoptionlink\"
onclick=\"window.location='" + root + "SurveyQuestion/Edit/" +
question.SurveyQuestionID.ToString() + "'\">Edit</span></td>");
                    sb.AppendLine("<td class=\"gridtext\" style=\"text-
align:center;\"><span class=\"surveyquestionoptionlink\"
onclick=\"window.location='" + root + "SurveyQuestion/Delete/" +
question.SurveyQuestionID.ToString() + "'\">Delete</span></td>");
                    sb.AppendLine("</tr>");

                    // Loop through each question option
                    List<SurveyQuestionOption> options =
orep.GetSurveyQuestionOptions(question.SurveyQuestionID).ToList();
                    foreach (SurveyQuestionOption option in options)
                    {
                        sb.AppendLine("<tr class=\"optionrow\">");
                        sb.AppendLine("<td class=\"gridtext\"
style=\"padding-left:15px;\">" + option.SurveyQuestionOptionText +
"</td>");
                        sb.AppendLine("<td class=\"gridtext\"></td>");
                        sb.AppendLine("<td class=\"gridtext\" style=\"text-
align:center;\"><span class=\"surveyquestionoptionlink\"
onclick=\"window.location='" + root + "SurveyQuestionOption/MoveUp/" +
option.SurveyQuestionOptionID.ToString() + "'\">Up</span></td>");
                        sb.AppendLine("<td class=\"gridtext\" style=\"text-
align:center;\"><span class=\"surveyquestionoptionlink\"
onclick=\"window.location='" + root + "SurveyQuestionOption/MoveDown/" +
option.SurveyQuestionOptionID.ToString() + "'\">Down</span></td>");
                        sb.AppendLine("<td class=\"gridtext\" style=\"text-
align:center;\"><span class=\"surveyquestionoptionlink\"
onclick=\"window.location='" + root + "SurveyQuestionOption/Edit/" +
option.SurveyQuestionOptionID.ToString() + "'\">Edit</span></td>");
                        sb.AppendLine("<td class=\"gridtext\" style=\"text-
align:center;\"><span class=\"surveyquestionoptionlink\"
onclick=\"window.location='" + root + "SurveyQuestionOption/Delete/" +
option.SurveyQuestionOptionID.ToString() + "'\">Delete</span></td>");
                        sb.AppendLine("</tr>");
                    }
                }

                sb.Append("</table>");

                return sb.ToString();
            }
            catch { return String.Empty; }
        }

        private string BuildSurveyTableNoLinks(Survey survey)
```

```
        {
            try
            {
                StringBuilder sb = new StringBuilder();

                string root = Request.Url.OriginalString.Substring(0,
Request.Url.OriginalString.ToLower().LastIndexOf("/survey/"));
                if (!root.EndsWith("/")) root += "/";

                sb.AppendLine("<table style=\"border-spacing:0;border-
collapse:collapse;\" class=\"surveytable\">");
                sb.AppendLine("<tr class=\"surveyrow\">");
                sb.AppendLine("<td class=\"gridtext\">" + survey.SurveyName
+ "</td>");
                sb.AppendLine("<td class=\"gridtext\"
style=\"width:110px;\"></td>");
                sb.AppendLine("<td class=\"gridtext\"
style=\"width:25px;\"></td>");
                sb.AppendLine("<td class=\"gridtext\"
style=\"width:45px;\"></td>");
                sb.AppendLine("<td class=\"gridtext\"
style=\"width:35px;\"></td>");
                sb.AppendLine("<td class=\"gridtext\"
style=\"width:55px;\"></td>");
                sb.AppendLine("</tr>");

                // Loop through each question and question option
                ISurveyQuestionRepository qrep = new
EntitySurveyQuestionRepository();
                ISurveyQuestionOptionRepository orep = new
EntitySurveyQuestionOptionRepository();
                List<SurveyQuestion> questions =
qrep.GetSurveyQuestions(survey.SurveyID).ToList();

                foreach (SurveyQuestion question in questions)
                {
                    sb.AppendLine("<tr class=\"questionrow\">");
                    string selectionmode = " (Single Select)";
                    if (question.AllowMultiSelect) selectionmode = " (Multi
Select)";
                    sb.AppendLine("<td class=\"gridtext\">" +
question.SurveyQuestionText + selectionmode + "</td>");
                    sb.AppendLine("<td class=\"gridtext\"></td>");
                    sb.AppendLine("<td class=\"gridtext\" style=\"text-
align:center;\"></td>");
                    sb.AppendLine("<td class=\"gridtext\" style=\"text-
align:center;\"></td>");
                    sb.AppendLine("<td class=\"gridtext\" style=\"text-
align:center;\"></td>");
```

```
                    sb.AppendLine("<td class=\"gridtext\" style=\"text-
align:center;\"></td>");
                    sb.AppendLine("</tr>");

                    // Loop through each question option
                    List<SurveyQuestionOption> options =
orep.GetSurveyQuestionOptions(question.SurveyQuestionID).ToList();
                    foreach (SurveyQuestionOption option in options)
                    {
                        sb.AppendLine("<tr class=\"optionrow\">");
                        sb.AppendLine("<td class=\"gridtext\"
style=\"padding-left:15px;\">" + option.SurveyQuestionOptionText +
"</td>");
                        sb.AppendLine("<td class=\"gridtext\"></td>");
                        sb.AppendLine("<td class=\"gridtext\" style=\"text-
align:center;\"></td>");
                        sb.AppendLine("<td class=\"gridtext\" style=\"text-
align:center;\"></td>");
                        sb.AppendLine("<td class=\"gridtext\" style=\"text-
align:center;\"></td>");
                        sb.AppendLine("<td class=\"gridtext\" style=\"text-
align:center;\"></td>");
                        sb.AppendLine("</tr>");
                    }
                }
                sb.Append("</table>");
                return sb.ToString();
            }
            catch { return String.Empty; }
        }
        private List<SelectListItem> BuildImageList(string currentfile)..
    }
```

Lampiran B.4 Implementasi *Source Code* `SurveyController`

[Halaman ini sengaja dikosongkan]

LAMPIRAN C. KUISIONER

No.	Pernyataan	Pilihan jawaban			
		Sangat setuju	Setuju	Tidak setuju	Sangat tidak setuju
1	Menurut saya aplikasi web ini mudah digunakan dan dipahami.	✓	○	○	○
2	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen akun pada aplikasi web ini.	✓	○	○	○
3	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen *screen* pada aplikasi web ini	✓	○	○	○
4	Menurut saya, cukup mudah untuk pengoperasian membuat *survey* pada aplikasi web ini	○	✓	○	○
5	Menurut saya, cukup mudah untuk pengoperasian melakukan *scheduling* pada aplikasi web ini	○	✓	○	○
6	Menurut saya, tampilan pada aplikasi web sesuai dengan konten	✓	○	○	○
7	Menurut saya aplikasi *Client-Desktop* ini mudah digunakan dan dipahami.	✓	○	○	○
8	Menurut saya, tampilan pada aplikasi *Client-Desktop* mudah dimengerti sehingga mudah untuk mengoperasikanya.	✓	○	○	○
9	Menurut saya, konten yang ada pada tugas akhir ini sangat sesuai dengan kebutuhan Teknik Informatika ITS.	✓	○	○	○

Nama : Muh. Aunorafiq Musa

Kuisioner B.1. Muh. Aunorafiq Musa

No.	Pernyataan	Pilihan jawaban			
		Sangat setuju	Setuju	Tidak setuju	Sangat tidak setuju
1	Menurut saya aplikasi web ini mudah digunakan dan dipahami.	✓	○	○	○
2	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen akun pada aplikasi web ini.	✓	○	○	○
3	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen *screen* pada aplikasi web ini	○	✓	○	○
4	Menurut saya, cukup mudah untuk pengoperasian membuat *survey* pada aplikasi web ini	○	✓	○	○
5	Menurut saya, cukup mudah untuk pengoperasian melakukan *scheduling* pada aplikasi web ini	○	✓	○	○
6	Menurut saya, tampilan pada aplikasi web sesuai dengan konten	✓	○	○	○
7	Menurut saya aplikasi *Client-Desktop* ini mudah digunakan dan dipahami.	○	✓	○	○
8	Menurut saya, tampilan pada aplikasi *Client-Desktop* mudah dimengerti sehingga mudah untuk mengoperasikanya.	✓	○	○	○
9	Menurut saya, konten yang ada pada tugas akhir ini sangat sesuai dengan kebutuhan Teknik Informatika ITS.	○	✓	○	○

Nama : Muhammad Agil Mahbuby

Kuisioner B.2. Muhammad Agil Mahbuby

No.	Pernyataan	Pilihan jawaban			
		Sangat setuju	Setuju	Tidak setuju	Sangat tidak setuju
1	Menurut saya aplikasi web ini mudah digunakan dan dipahami.	✓	○	○	○
2	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen akun pada aplikasi web ini.	✓	○	○	○
3	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen *screen* pada aplikasi web ini	✓	○	○	○
4	Menurut saya, cukup mudah untuk pengoperasian membuat *survey* pada aplikasi web ini	○	✓	○	○
5	Menurut saya, cukup mudah untuk pengoperasian melakukan *scheduling* pada aplikasi web ini	✓	○	○	○
6	Menurut saya, tampilan pada aplikasi web sesuai dengan konten	✓	○	○	○
7	Menurut saya aplikasi *Client-Desktop* ini mudah digunakan dan dipahami.	✓	○	○	○
8	Menurut saya, tampilan pada aplikasi *Client-Desktop* mudah dimengerti sehingga mudah untuk mengoperasikanya.	✓	○	○	○
10	Menurut saya, konten yang ada pada tugas akhir ini sangat sesuai dengan kebutuhan Teknik Informatika ITS.	○	✓	○	○

Nama : M. Hanif B.

Kuisioner B.3. M. Hanif B.

No.	Pernyataan	Pilihan jawaban			
		Sangat setuju	Setuju	Tidak setuju	Sangat tidak setuju
1	Menurut saya aplikasi web ini mudah digunakan dan dipahami.	✓	○	○	○
2	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen akun pada aplikasi web ini.	○	✓	○	○
3	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen *screen* pada aplikasi web ini	✓	○	○	○
4	Menurut saya, cukup mudah untuk pengoperasian membuat *survey* pada aplikasi web ini	✓	○	○	○
5	Menurut saya, cukup mudah untuk pengoperasian melakukan *scheduling* pada aplikasi web ini	✓	○	○	○
6	Menurut saya, tampilan pada aplikasi web sesuai dengan konten	○	✓	○	○
7	Menurut saya aplikasi *Client-Desktop* ini mudah digunakan dan dipahami.	✓	○	○	○
8	Menurut saya, tampilan pada aplikasi *Client-Desktop* mudah dimengerti sehingga mudah untuk mengoperasikanya.	✓	○	○	○
9	Menurut saya, konten yang ada pada tugas akhir ini sangat sesuai dengan kebutuhan Teknik Informatika ITS.	✓	○	○	○

Nama : Endang Wahyu P.

Kuisioner B.4. Endang Wahyu P.

No.	Pernyataan	Pilihan jawaban			
		Sangat setuju	Setuju	Tidak setuju	Sangat tidak setuju
1	Menurut saya aplikasi web ini mudah digunakan dan dipahami.	✓	○	○	○
2	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen akun pada aplikasi web ini.	○	✓	○	○
3	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen *screen* pada aplikasi web ini	○	✓	○	○
4	Menurut saya, cukup mudah untuk pengoperasian membuat *survey* pada aplikasi web ini	○	✓	○	○
5	Menurut saya, cukup mudah untuk pengoperasian melakukan *scheduling* pada aplikasi web ini	✓	○	○	○
6	Menurut saya, tampilan pada aplikasi web sesuai dengan konten	✓	○	○	○
7	Menurut saya aplikasi *Client-Desktop* ini mudah digunakan dan dipahami.	○	✓	○	○
8	Menurut saya, tampilan pada aplikasi *Client-Desktop* mudah dimengerti sehingga mudah untuk mengoperasikanya.	✓	○	○	○
9	Menurut saya, konten yang ada pada tugas akhir ini sangat sesuai dengan kebutuhan Teknik Informatika ITS.	✓	○	○	○

Nama : Azi Prastyo

Kuisioner B.5. Azi Prastyo

No.	Pernyataan	Pilihan jawaban			
		Sangat setuju	Setuju	Tidak setuju	Sangat tidak setuju
1	Menurut saya aplikasi web ini mudah digunakan dan dipahami.	✓			
2	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen akun pada aplikasi web ini.	✓			
3	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen *screen* pada aplikasi web ini		✓		
4	Menurut saya, cukup mudah untuk pengoperasian membuat *survey* pada aplikasi web ini	✓			
5	Menurut saya, cukup mudah untuk pengoperasian melakukan *scheduling* pada aplikasi web ini	✓			
6	Menurut saya, tampilan pada aplikasi web sesuai dengan konten	✓			
7	Menurut saya aplikasi *Client-Desktop* ini mudah digunakan dan dipahami.		✓		
8	Menurut saya, tampilan pada aplikasi *Client-Desktop* mudah dimengerti sehingga mudah untuk mengoperasikanya.	✓			
9	Menurut saya, konten yang ada pada tugas akhir ini sangat sesuai dengan kebutuhan Teknik Informatika ITS.		✓		

Nama : Chairaja Almas Djeni

Kuisioner B.6. Chairaja Almas Djeni

No.	Pernyataan	Pilihan jawaban			
		Sangat setuju	Setuju	Tidak setuju	Sangat tidak setuju
1	Menurut saya aplikasi web ini mudah digunakan dan dipahami.	✓	○	○	○
2	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen akun pada aplikasi web ini.	✓	○	○	○
3	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen *screen* pada aplikasi web ini	✓	○	○	○
4	Menurut saya, cukup mudah untuk pengoperasian membuat *survey* pada aplikasi web ini	○	✓	○	○
5	Menurut saya, cukup mudah untuk pengoperasian melakukan *scheduling* pada aplikasi web ini	✓	○	○	○
6	Menurut saya, tampilan pada aplikasi web sesuai dengan konten	✓	○	○	○
7	Menurut saya aplikasi *Client-Desktop* ini mudah digunakan dan dipahami.	✓	○	○	○
8	Menurut saya, tampilan pada aplikasi *Client-Desktop* mudah dimengerti sehingga mudah untuk mengoperasikanya.	✓	○	○	○
9	Menurut saya, konten yang ada pada tugas akhir ini sangat sesuai dengan kebutuhan Teknik Informatika ITS.	✓	○	○	○

Nama : Bagus Ardiansyah

Kuisioner B.7. Bagus Ardiansyah

No.	Pernyataan	Pilihan jawaban			
		Sangat setuju	Setuju	Tidak setuju	Sangat tidak setuju
1	Menurut saya aplikasi web ini mudah digunakan dan dipahami.	○	✓	○	○
2	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen akun pada aplikasi web ini.	✓	○	○	○
3	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen *screen* pada aplikasi web ini	○	✓	○	○
4	Menurut saya, cukup mudah untuk pengoperasian membuat *survey* pada aplikasi web ini	○	✓	○	○
5	Menurut saya, cukup mudah untuk pengoperasian melakukan *scheduling* pada aplikasi web ini	✓	○	○	○
6	Menurut saya, tampilan pada aplikasi web sesuai dengan konten	✓	○	○	○
7	Menurut saya aplikasi *Client-Desktop* ini mudah digunakan dan dipahami.	✓	○	○	○
8	Menurut saya, tampilan pada aplikasi *Client-Desktop* mudah dimengerti sehingga mudah untuk mengoperasikanya.	○	✓	○	○
9	Menurut saya, konten yang ada pada tugas akhir ini sangat sesuai dengan kebutuhan Teknik Informatika ITS.	✓	○	○	○

Nama : FERNANDES P. SINAGA.

Kuisioner B.8. Fernandes P. Sinaga

No.	Pernyataan	Pilihan jawaban			
		Sangat setuju	Setuju	Tidak setuju	Sangat tidak setuju
1	Menurut saya aplikasi web ini mudah digunakan dan dipahami.	✓	○	○	○
2	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen akun pada aplikasi web ini.	✓	○	○	○
3	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen *screen* pada aplikasi web ini	✓	○	○	○
4	Menurut saya, cukup mudah untuk pengoperasian membuat *survey* pada aplikasi web ini	✓	○	○	○
5	Menurut saya, cukup mudah untuk pengoperasian melakukan *scheduling* pada aplikasi web ini	○	✓	○	○
6	Menurut saya, tampilan pada aplikasi web sesuai dengan konten	✓	○	○	○
7	Menurut saya aplikasi *Client-Desktop* ini mudah digunakan dan dipahami.	✓	○	○	○
8	Menurut saya, tampilan pada aplikasi *Client-Desktop* mudah dimengerti sehingga mudah untuk mengoperasikanya.	✓	○	○	○
9	Menurut saya, konten yang ada pada tugas akhir ini sangat sesuai dengan kebutuhan Teknik Informatika ITS.	○	✓	○	○

Nama : Suliadi Marsetya

Kuisioner B.9. Suliadi Marsetya

No.	Pernyataan	Pilihan jawaban			
		Sangat setuju	Setuju	Tidak setuju	Sangat tidak setuju
1	Menurut saya aplikasi web ini mudah digunakan dan dipahami.	✓	○	○	○
2	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen akun pada aplikasi web ini.	✓	○	○	○
3	Menurut saya, cukup mudah untuk melakukan pengoperasian manajamen *screen* pada aplikasi web ini	✓	○	○	○
4	Menurut saya, cukup mudah untuk pengoperasian membuat *survey* pada aplikasi web ini	○	✓	○	○
5	Menurut saya, cukup mudah untuk pengoperasian melakukan *scheduling* pada aplikasi web ini	○	✓	○	○
6	Menurut saya, tampilan pada aplikasi web sesuai dengan konten	✓	○	○	○
7	Menurut saya aplikasi *Client-Desktop* ini mudah digunakan dan dipahami.	○	✓	○	○
8	Menurut saya, tampilan pada aplikasi *Client-Desktop* mudah dimengerti sehingga mudah untuk mengoperasikanya.	✓	○	○	○
9	Menurut saya, konten yang ada pada tugas akhir ini sangat sesuai dengan kebutuhan Teknik Informatika ITS.	✓	○	○	○

Nama : Galang A. D. P.

Kuisioner B.10. Galang Amanda D. P.