

KISIM		YAPRAK NO : 01
YAPILAN İŞ	Tanışma ve Bilgi Alma	TARİH : 20/07/2015



Stajın ilk günü olması sebebiyle **Tegsoft** firması ve yaptıkları işler ile ilgili bilgi alındı. Tegsoft'un genel olarak telekomünikasyon alanlarında hizmet verdiği; tamamen açık kaynak kodlu olarak **Tobe** yazılım alt yapısı ile geliştirilmiş, **Müşteri İlişkileri Yönetimi (TegsoftCRM)**, **İnsan Kaynakları Yazılımı (TegsoftHR)** ile **IP PBX** ve **Contact Center yazılımları (TegsoftPBX ve TegsoftCC)** ürünleri ile şirketler için daha iyi, daha hızlı ve maliyeti daha düşük çözümler sunduğu; araştırma ve geliştirmeye odaklanan bir firma olduğu bilgisi verildi. Ayrıca Tegsoft'un, geliştirdiği ürünlerin yanısıra, **Avaya/Nortel CTI** ve **IVR** yazılımlarının geliştirilmesi ile yazılım ve eğitim projelerinin yönetimi alanlarında da hizmet vermekte olduğu, deneyimli uzmanları aracılığı ile eğitim ve danışmanlık hizmetleri sunduğu belirtildi.

Şirketin geliştirdiği yazılımlar gösterildi, Bu yazılımların distribütörler aracılığıyla Türkiye'deki birçok firmaya hizmet verdiği anlatıldı.

Staj süresince yapılacak çalışmalar ile ilgili bilgi verildi. Çağrı merkezleri için, müşteri temsilcilerinin verdiği hizmet kalitelerinin değerlendirilmesi amacıyla; müşteri temsilcilerinin bilgisayar ekranlarının kaydedilmesi(**ScreenCasting**) ve kaydedilen ekranın, denetleyen kişilerce izlenebilmesini(**ScreenPlaying**) sağlayan Java Applet'ler geliştirileceği belirtildi ve buna yönelik araştırma yapmamız istendi.

KISIM		YAPRAK NO : 02
YAPILAN İŞ	Gerekli Ortam ve Yazılımların Kurulması	TARİH : 21/07/2015
<div data-bbox="564 291 1015 504" data-label="Image"></div> <p>Staj süresince kullanılacak işletim sistemi olan Ubuntu kuruldu. Gerekli güncellemeler yapıldı. Sürücüler yüklendi.</p> <p>Aşağıdaki komutlarla Oracle Java 8 yüklendi. Ve JAVA_HOME ortam değişkeninin setlenmesi gibi bazı ayarları yapıldı.</p> <pre>sudo apt-add-repository ppa:webupd8team/java sudo apt-get update sudo apt-get install oracle-java8-installer</pre> <div data-bbox="564 927 1021 1046" data-label="Image"></div> <p>Eclipse geliştirme ortamı yüklendi ve gerekli ayarları yapıldı.</p> <div data-bbox="518 1184 1066 1305" data-label="Image"></div> <p>Proje yönetimi için Git kuruldu. GitHub'ın kullanımıyla ilgili araştırmalar yapıldı.</p> <p>ScreenCasting ve ScreenPlaying uygulamaları için araştırma yapmamız istendi. Buna yönelik gün boyunca bilgisayarın ekran görüntüsünün alınması, alınan görüntünün formatının ne olması gerektiği, görüntünün boyutunun nasıl küçültülebileceği ile ilgili araştırmalar yapıldı. Araştırmalar sonucunda png formatının çizim ve grafik için uygun olduğunu, jpeg formatının ise fotoğraf ve realistik resimler için uygun olduğu gözlemlendi. Bu yüzden jpeg formatı üzerine yoğunlaşıldı.</p> <p>ScreenPlaying tarafında ise alınan görüntülerin video olarak encode'lanması ile resim resim sunucuya gönderilmesinden sonra videoya dönüştürülmesi senaryoları incelendi. Burada karşılaşılan en önemli zorluk, müşteri temsilcisinin kullandığı bilgisayarın harddiskinin kullanılamamasıdır. Bir diğer problem ise müşteri temsilcisinden gelen ekran görüntüsü ile sunucudan gelen ses kaydının senkron şekilde oynatılması zorunluluğudur. Oynatma işlemi için ise HTML5 ve Flash Player seçenekleri araştırıldı. Trendin HTML5 yönünde olması ve harici eklenti gerektirmemesinden dolayı araştırmalar bu doğrultuda yoğunlaştırıldı.</p>		
KONTROL SONUCU		

KISIM		YAPRAK NO : 03
YAPILAN İŞ	Ekran Görüntüsü Alma	TARİH : 22/07/2015
<p>Java'da ekran görüntüsü alma ve jpeg formatı ile kaydetme ile ilgili araştırmalar yapıldı. ScreenCapture adlı tam ekran görüntü alan sınıf aşağıdaki gibi kodlandı.</p> <pre> public class ScreenCapture { public void fullScreenCapture(String fileName){ try { Robot robot = new Robot(); //Görüntüsü alınacak koordinatlar Rectangle screenRect = new Rectangle(Toolkit.getDefaultToolkit().getScreenSize()); //Verilen koordinatlardaki görüntünün alınması BufferedImage screenFullImage = robot.createScreenCapture(screenRect); File file = new File("screencapture.jpg"); //Alınan görüntünün JPEG formatında kaydedilmesi ImageIO.write(screenFullImage, "jpg", file); } catch (AWTException IOException ex) { System.err.println(ex); } } } </pre> <p>Dosya büyüklüklerinin, ağ trafiğinde sebebiyet verecek olumsuz etkilerini azaltmak amacıyla kaydedilen resmin sıkıştırılması ile ilgili araştırmalar yapıldı.</p> <p>Alınan görüntülerin mp4 şeklinde videoya dönüştürülmesi araştırıldı. Video Encoding olaylarının, C ve C++ gibi donanıma yakın dillerde daha hızlı olması nedeniyle çok az kaynak bulunabildi. Video encoding'in, oluşturulacak video'nun boyutunu ciddi oranda düşürecek olması nedeniyle araştırmalar bu alanda yoğunlaştırıldı.</p>		
KONTROL SONUCU		

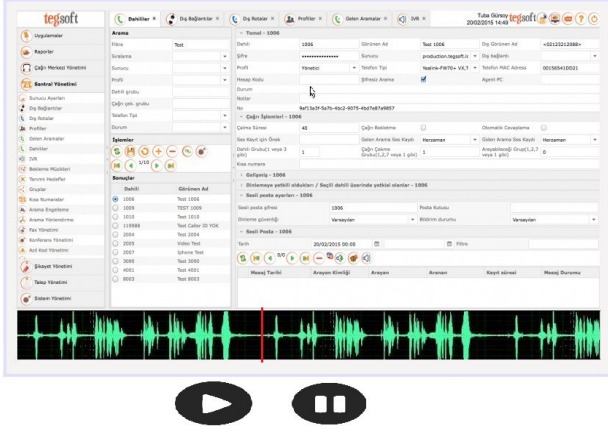
KISIM		YAPRAK NO : 04
YAPILAN İŞ	Resim Sıkıştırma	TARİH : 23/07/2015
<p>Java'da resim sıkıştırma yöntemleri araştırmaya devam edildi. Ekran görüntüsü alınan resmi sıkıştıran <i>CompressJPEGFile</i> sınıfı aşağıdaki gibi kodlandı.</p> <pre> public class CompressJPEGFile { public Float quality = 0.3f; public void toCompress(BufferedImage image, String compressImagePath) throws IOException{ File compressedImageFile = new File(compressImagePath); OutputStream os =new FileOutputStream(compressedImageFile); Iterator<ImageWriter>writers = ImageIO.getImageWritersByFormatName("jpg"); ImageWriter writer = (ImageWriter) writers.next(); ImageOutputStream ios = ImageIO.createImageOutputStream(os); writer.setOutput(ios); ImageWriteParam param = writer.getDefaultWriteParam(); param.setCompressionMode(ImageWriteParam.MODE_EXPLICIT); param.setCompressionQuality(quality); writer.write(null, new IIOMemoryImage(image, null, null), param); os.close(); ios.close(); writer.dispose(); } } </pre> <p><i>quality</i> değişkeninin 0,3fe kadar çekilmesi, resmin kalitesini ciddi şekilde düşürse de resim boyutunu yüzde 45'lere varan oranda azaltmıştır.</p> <p>Burdan sonraki aşamada resimler video olarak encode edilerek boyutun azaltılması planlanmıştır.</p>		
KONTROL SONUCU		

KISIM		YAPRAK NO : 05
YAPILAN İŞ	Java içerisinde Harici Program Çalıştırma	TARİH : 24/07/2015
<p>Java ile video encode edilememesi senaryosunda; müşteri temsilcisinin bilgisayarından gelen görüntülerin, Java'dan harici program çağırılması ile sunucuda belirli zamanlarda encode edilmesi işleminin nasıl yapılabileceği araştırıldı.</p> <p>Harici program olarak FFmpeg adındaki açık kaynak kodlu, cross-platform çalışan multimedia aracı incelendi. Bu aracın, harici olarak Java Applet'ten çağırılması ile ilgili olarak da aşağıdaki ExecuteCommand sınıfı oluşturuldu.</p> <pre>public class ExecuteCommand { public void execute(String cmd) throws IOException, InterruptedException{ Runtime runtime = Runtime.getRuntime(); Process process = runtime.exec(cmd); process.waitFor(); BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(process.getInputStream())); String line = ""; while ((line=bufferedReader.readLine())!=null) { System.out.println(line); } process.destroy(); } }</pre> <p>Günün geri kalanında ScreenPlaying uygulaması için HTML5, CSS ve JavaScript ile ilgili araştırma çalışması yapılmıştır.</p>		
KONTROL SONUCU		

KISIM		YAPRAK NO : 06								
YAPILAN İŞ	Proje Analiz Dökümanının Oluşturulması	TARİH : 27/07/2015								
<p>Yapılacak ScreenCast ve ScreenPlay projelerinin tasarımı, ne yapılacağı, hangi teknolojilerinin kullanıldığı ile ilgili bir toplantı yapıldı. İlgili mühendis bunlarla ilgili Proje Analiz Dökümanı çıkarmamızı istedi. Aşağıdaki Proje Analiz Dökümanı oluşturuldu.</p> <p>Proje Adı: Kalite Denetim Ekran Kayıt Modülü</p> <p>Analiz Tarihi : 27/07/2015</p> <p>Döküman Sürümü : 1.0</p> <p>Sorumlular : Fethi Erdiñ UZUN, Murat HAS, Emre YALÇIN, Orhan YILMAZ</p> <p>Onay : Mehmet Eray Gürsoy</p> <p>Aktörler ve Roller</p> <table><tr><th>Aktör</th><th>Rol</th></tr><tr><td>Müşteri Temsilcisi</td><td><ul style="list-style-type: none">Müşteri temsilcisi gelen çağrıya cevap vererek ekran kaydını başlatır, çağrı sonlandırıldığında ekran videosu kaydedilir.</td></tr><tr><td>Uygulama Katmanı</td><td><ul style="list-style-type: none">Müşteri temsilcisi çağrıyı sonlandırdığında kaydedilen video sunucuya gönderilir.Sunucudan alınan ses ile video eşleştirilerek izlenmeye hazır duruma getirilir.</td></tr><tr><td>Denetleyen Yetkili</td><td><ul style="list-style-type: none">Denetleyici modül tarafından kaydedilen görüntü ve sesi oynatır.</td></tr></table> <p>Proje Adımları</p> <ul style="list-style-type: none">AnalizYazılım GeliştirmeKurulumTestKabulEğitim <p>Kullanılacak Teknolojiler</p> <ul style="list-style-type: none">Java ve Java AppletHTML5JavaScriptHTTP ServerMonte Media Library <p>Yapılacak Çalışmalar</p> <ul style="list-style-type: none">Monte Media Library kullanılarak ekranın video kaydını alacak bir Java Applet oluşturulmasıVideo kaydının uygun şekilde encode edilmesiAlınan video kaydının sunucuya gönderilmesiSunucudan ses ve video kaydının alınmasıVideo ve ses kaydını eş zamanlı oynatacak bir HTML5 Player hazırlanması			Aktör	Rol	Müşteri Temsilcisi	<ul style="list-style-type: none">Müşteri temsilcisi gelen çağrıya cevap vererek ekran kaydını başlatır, çağrı sonlandırıldığında ekran videosu kaydedilir.	Uygulama Katmanı	<ul style="list-style-type: none">Müşteri temsilcisi çağrıyı sonlandırdığında kaydedilen video sunucuya gönderilir.Sunucudan alınan ses ile video eşleştirilerek izlenmeye hazır duruma getirilir.	Denetleyen Yetkili	<ul style="list-style-type: none">Denetleyici modül tarafından kaydedilen görüntü ve sesi oynatır.
Aktör	Rol									
Müşteri Temsilcisi	<ul style="list-style-type: none">Müşteri temsilcisi gelen çağrıya cevap vererek ekran kaydını başlatır, çağrı sonlandırıldığında ekran videosu kaydedilir.									
Uygulama Katmanı	<ul style="list-style-type: none">Müşteri temsilcisi çağrıyı sonlandırdığında kaydedilen video sunucuya gönderilir.Sunucudan alınan ses ile video eşleştirilerek izlenmeye hazır duruma getirilir.									
Denetleyen Yetkili	<ul style="list-style-type: none">Denetleyici modül tarafından kaydedilen görüntü ve sesi oynatır.									

Projenin Amacı: Çağrı merkezlerinde müşteri temsilcilerinin görüşme kayıtlarının yanı sıra bilgisayar ekranlarının da kayıt altına alınarak denetleyiciye eş zamanlı görüntü-ses oynatımı sunulması

Muhtemel Ekran Görüntüsü



Proje planının oluşturulmasının ardından iş bölümü yapılarak **HTML Player** oluşturulması kısmı bana verildi. Gün boyunca **HTML5** ve **JavaScript** ile ilgili konu çalışması ve alıştırımlar yapıldı. Örnek kodlar incelendi.

KISIM		YAPRAK NO : 07
YAPILAN İŞ	HTML5 Player I	TARİH : 28/07/2015

HTML5 ile birlikte tarayıcıların, eklentisiz video ve audio oynatabilmeleri; mobil cihazların da **HTML5**'i desteklemeleri, ilgiyi tamamen bu yeni web teknolojisine kaydırıldı. *Apple, Google, Microsoft* gibi şirketlerin de tamamen desteğini alan **HTML5**, günümüzde bir web standardı haline geldi. Bu yüzden *Kalite Denetim Ekran Kayıt Modülü'nün* Player kısmının bu teknoloji kullanılarak tasarlanması kararı alındı.

Öncelikle *SPlay.html, SPlay.css, SPlay.js* proje dosyaları oluşturuldu. *HTML, CSS* ve *JavaScript* kodlarının birbirinden ayrılması karmaşıklığı ciddi oranda azalttığı için bu şekilde bir tercih yapıldı. Ardından aşağıdaki gibi bir görsel arayüz oluşturuldu.



Bu arayüzü oluşturan *SPlay.html* dosyası ise aşağıdaki şekildedir.

```

<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="SPlay.css">
  <script src="SPlay.js"></script>
</head>
<body>
<center>
<div id="video_player_box">
  <video id="videofile" poster="poster.jpg" >
    <source src="movie.mp4">
  </video>
  <div id="video_controls_bar">
    <button id="playpausebtn" >Play</button>
    <input id="seekslider" type="range" min="0" max="100" value="0" step="1">
    <span id="curtimetext">00:00</span> / <span id="durtimetext">00:00</span>
    <button id="mutebtn">Mute</button>
    <input id="volumeslider" type="range" min="0" max="100" value="100" step="1">
    <button id="fullscreenbtn">[ ]</button>
  </div>
</center>
</div>
</body>
</html>

```


Oluşturulan arayüzde **HTML5**'in varsayılan oynatıcı arayüzü, video ve sesin ayrı ayrı alınıp, birlikte oynatılabilmesi için devredışı bırakılması gerekiyordu. Bu işlemin gerçekleşmesi için `<video controls>.....</video>` etiketinden **controls** anahtar kelimesinin çıkarılması yeterli oldu. Ancak tam ekrana geçince varsayılan **HTML5 oynatıcı arayüzü** tekrar açığa çıkmaktaydı. Bunu engellemek için **CSS** dosyasına aşağıdaki kod eklendi.

```
video::-webkit-media-controls
{
    display:none !important;
}
```

Bu sayede oynatıcı arayüzünün bizim özelleştirmiş olduğumuz tasarımda kalması sağlandı.

Sonraki aşama olan **Play/Pause**, **İleri/Geri sarma** gibi olayların gerçekleştirilmesi için **JavaScript** kodlarının yazılması gerekiyordu. Gerekli araştırma yapılarak **SPlay.js** dosyası aşağıdaki şekilde oluşturuldu.

```
var video, pause, playbtn;

function init()
{
    video = document.getElementById("videofile");
    playbtn=document.getElementById("playpausebtn");
    playbtn.addEventListener("click", playpause,false );
}

window.onload = init;

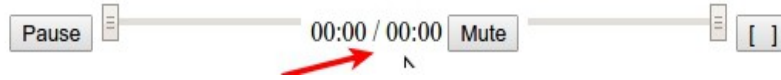
function playpause()
{
    if(video.paused)
    {
        video.play();
        playbtn.innerHTML="pause";
    }
    else
    {
        video.pause();
        playbtn.innerHTML = "play";
    }
}
```

Böylece **Play** tuşuna basıldığında video'nun oynatılması ve isminin **pause** olarak değişimi ile tersi durumunda ise video'nun duraklaması sağlanmış oldu.

KISIM		YAPRAK NO : 08
YAPILAN İŞ	HTML5 Player II	TARİH : 29/07/2015
<p>Dün başlanan HTML5 Player çalışmasına devam edildi. Projenin JavaScript tarafına <i>İleri-Geri sar- ma</i>, <i>video zamanını gösterme</i> ve <i>tam ekran yapma</i> fonksiyonları eklendi. <i>Splay.js</i> aşağıdaki şekilde güncellendi.</p> <pre> var video, pause, playbtn, fullScreenbtn, seekbar; function init() { video = document.getElementById("videofile"); playbtn=document.getElementById("playpausebtn"); fullScreenbtn=document.getElementById("fullscreenbtn"); seekbar=document.getElementById("seekslider"); playbtn.addEventListener("click", playpause,false); fullScreenbtn.addEventListener("click", toggleFullScreen, false); seekbar.addEventListener("change", seek, false); video.addEventListener("timeupdate", seekprogress, false) ; } window.onload = init; function playpause() { if(video.paused) { video.play(); playbtn.innerHTML="pause"; } else { video.pause(); playbtn.innerHTML = "play"; } } function seek() { var time= video.duration * (seekbar.value / 100); video.currentTime=time; } function seekprogress() { var seekvalue=(100 / video.duration) * video.currentTime; seekbar.value=seekvalue; } function toggleFullScreen() { if (video.requestFullScreen) { video.requestFullScreen(); } else if (video.webkitRequestFullScreen) { video.webkitRequestFullScreen(); } else if (video.mozRequestFullScreen) { video.mozRequestFullScreen(); } } </pre>		
KONTROL SONUCU		

KISIM		YAPRAK NO : 09
YAPILAN İŞ	Video ve Ses Kontrollerinin Eklenmesi	TARİH : 30/07/2015

HTML5 Player'ın yapımına devam edildi. Videonun anlık zamanının gösterilebilmesi için **Splay.js** içerisindeki **seekProgress()** fonksiyonu aşağıdaki şekilde güncellendi.



```
function seekProgress()
{
    var value = video.currentTime * (100 / video.duration);
    seekslider.value = value;
    var curmins = Math.floor(video.currentTime / 60);
    var cursecs = Math.floor(video.currentTime - curmins * 60);
    var durmins = Math.floor(video.duration / 60);
    var dursecs = Math.floor(video.duration - durmins * 60);
    if(cursecs < 10){ cursecs = "0"+cursecs; }
    if(dursecs < 10){ dursecs = "0"+dursecs; }
    if(curmins < 10){ curmins = "0"+curmins; }
    if(durmins < 10){ durmins = "0"+durmins; }
    curtimetext.innerHTML = curmins+":"+cursecs;
    durtimetext.innerHTML = durmins+":"+dursecs;
}
```

Müşteri temsilcisinin bilgisayar ekranından gelen görüntü ile müşteri ile konuşmasının ses kaydı ayrı ayrı ancak senkron bir şekilde oynatılacağından, bu senaryoya uygun olarak **Splay.html** dosyasına aşağıdaki **audio tag**'ı eklendi.

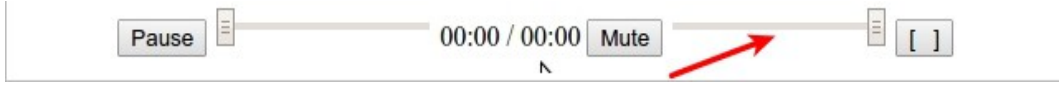
```
<audio id="audio" >
    <source src="audio.mp3" type="audio/mpeg">
</audio>
```



Mute/Unmute butonunun aktifleştirilmesi için **Splay.js** dosyasına aşağıdaki fonksiyon eklendi.

```
function audio.mute()
{
    if(audio.muted)
    {
        audio.muted = false;
        mutebtn.innerHTML = "Mute";
    }
    else
    {
        audio.muted = true;
        mutebtn.innerHTML = "Unmute";
    }
}
```

Ardından sesi artırıp azaltma işmini sağlamak için volume *seekslider*'ı için aşağıdaki fonksiyon yazıldı.



```
function setvolume()
{
    audio.volume = volumeslider.value / 100;
}
```

Son olarak video ve ses dosyasının aynı anda *play/pause* işlemi için *Splay.js* dosyasındaki *playPause()* fonksiyonu aşağıdaki şekilde güncellendi.

```
function playPause()
{
    //oynat/durdur
    if (video.paused)
    {
        video.play();
        audio.play();
        playbtn.innerHTML = "Pause";
    }
    else
    {
        video.pause();
        audio.pause();
        playbtn.innerHTML = "Play";
    }
}
```

Böylece play/pause butonuna basıldığında ses ve videonun aynı anda başlatılıp/durdurulması sağlanmış oldu.

KISIM		YAPRAK NO : 10
YAPILAN İŞ	İnternette Yer Alan Ses ve Video Dosyalarının Senkron Oynatılması	TARİH : 31/07/2015

Video ileri alındığında *audio'nun* da *video'nun* bulunduğu zamana geçmesi için **Splay.js** dosyasındaki *seek()* fonksiyonu aşağıdaki gibi güncellenmiştir. Böylece *video* ileri alındığında, sesin de aynı zamana alınması sağlanmış oldu.

```
function seek()
{
    time = video.duration * (seekbar.value / 100);
    video.currentTime = time;
    audio.currentTime = time;
}
```

HTML5 Player'da oynatılan ses ve video dosyaları yerel diskte bulunan dosyalardı. Ancak *video* ve *audio'nun* farklı sunuculardan gelecek olması nedeniyle bu senaryoya uygun olarak ses ve video dosyaları

<http://fethierdincuzun.com/wp-content/uploads/2015/08/output.mp4>
<http://fethierdincuzun.com/wp-content/uploads/2015/08/audio.mp3>

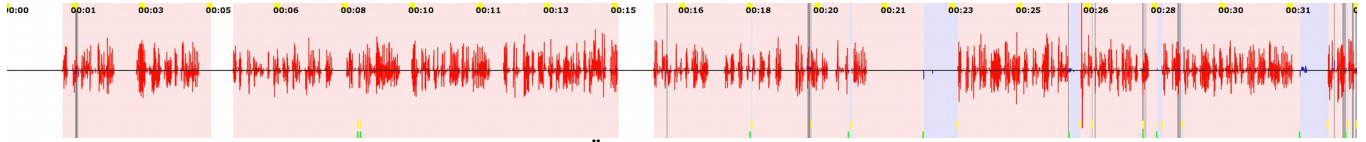
adreslerine yüklendi ve **Splay.html** dosyasındaki *audio* ve *video* etiketlerine bu adresler eklendi.

Video ve *audio* İnternet ortamında yer alması nedeniyle donma ve senkronizasyon problemlerini aşmak için bir **buffer**'lama işlemi gerekiyordu. Bu yüzden bu işlemi sağlayacak bir özelliğin olup olmadığı araştırıldı. *Video* ve *audio tag*'larına *preload="auto"* ifadesinin eklenmesiyle tarayıcının internet hızına göre otomatik **buffer**'lama işlemi yaptığı öğrenildi. Yapılan testlerde *video* ve *audio* dosyasının senkron şekilde sunucudan oynatılabildiği gözlemlendi. Bu özellikle birlikte kaba hatlarıyla **HTML5 Player** iskeleti oluşturulmuş oldu.

Bundan sonraki aşama için ilgili mühendisten bilgi alındı ve yapılacak yeni değişiklikler için görünüm için CSS çalışıldı.

KISIM		YAPRAK NO : 11
YAPILAN İŞ	Arayüz Geliştirmeleri	TARİH : 03/08/2015

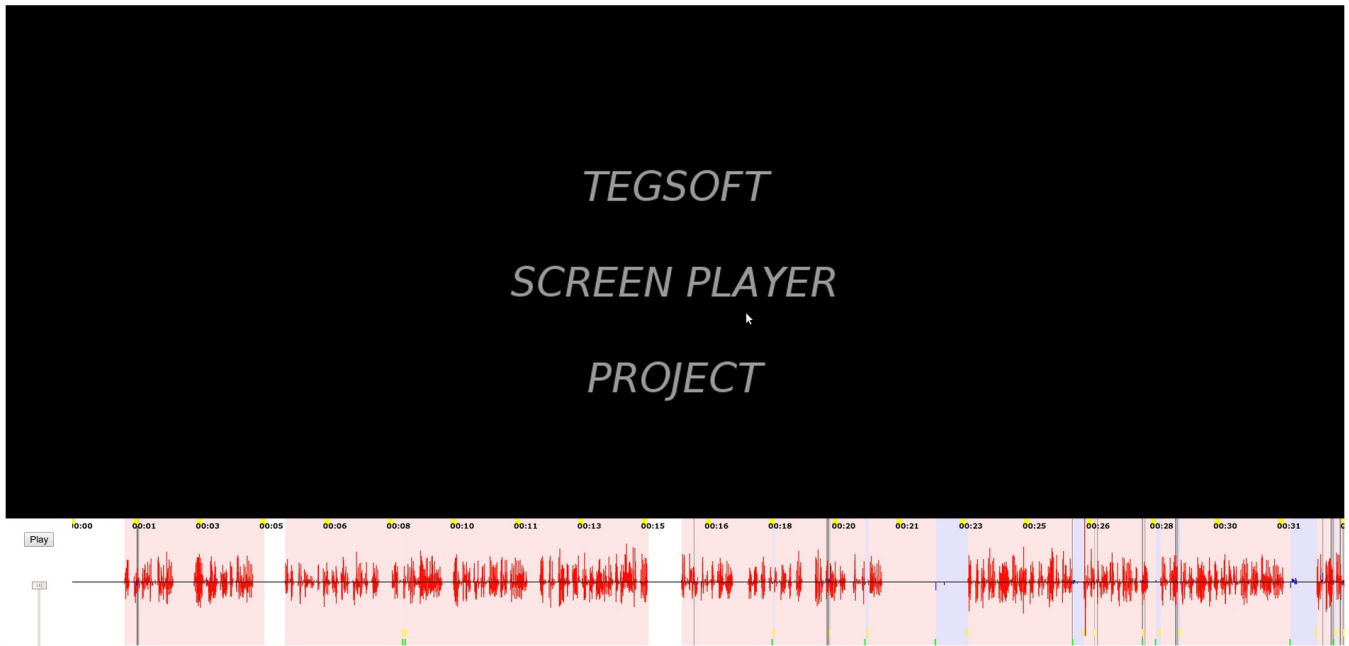
HTML5 Player'ın kabahatları oluşturulduktan sonra ilgili mühendisten görünüm ile ilgili yapılması gerekenler hakkında bilgi alındı. Her ses kaydının **infografik resimlerinin** olduğu ve **seekslider** yerine bu resimlerin arayüzde yer alması istendi. Ayrıca **Mute/Unmuted**, **FullScreen** gibi butonlara ihtiyaç duyulmayacağımız için arayüzden kaldırabileceğimiz söylendi. Ek olarak **seekslider** yerine her ses kaydına özel olarak gelecek **infografik** resmin ilgili zaman aralığına tıklandığında video ve audio'unun da aynı zaman aralığına **seek** etmesi gerektiği bilgisi verildi.



Örnek infografik resim

Bunun üzerine öncelikle **CSS** ile arayüz değişiklikleri yapılması; ardından da **HTML5 Canvas** ya da **SVG** ile birlikte **JavaScript** kullanılarak infografik resmin interaktif özelliğe kavuşturulması gerekmektedir.

Öncelikle **CSS** ile oluşturulması gereken arayüz özellikleri üzerinde duruldu. Ve gün boyunca süren çalışmalar sonucunda ilgili mühendisin istediği aşağıdaki arayüz oluşturuldu.



Bu arayüz için yazılan CSS kodları ise aşağıdaki şekildedir.

```
video::-webkit-media-controls
{
    display:none !important;
}

div#video_player_box
{
    background:#000;
    max-width: 100%;
    height: auto;
}
```

```
div#videoarea
{
  max-width: 100%;
  height: auto;;
}
```

```
div#videofile
{
  max-width: 100%;
  height: auto;
}
```

```
div#audio
{
  visibility: hidden;
  display: none;
}
```

```
div#video_controls_bar
{
  max-width: 100%;
  height: auto;
  background: #333;
  color:#CCC;
}
```

```
div#seekarea
{
  max-width: 95%;
  float: right;
  height: auto;
}
```

```
img
{
  max-width: 100%;
  height: auto;
}
```

```
div#controlarea
{
  float: left;
  max-width: 5%;
  height: auto;
}
```

```
button#playpausebtn
{
  margin-top: 20%;
  max-width: 100%;
  display: block;
  float: top;
  height:auto;
  clear: right;
}
```

```
input#volumeslider
{
  margin-left: 0%;
  margin-right: 50%;
  margin-top: 90%;
  display: block;
  float: bottom;
  height: auto;
  width: auto;
  max-height: 100%;
  max-width: 100%;

  -ms-transform: rotate(-90deg); /* IE 9 */
  -webkit-transform: rotate(-90deg); /* Chrome, Safari, Opera */
  transform: rotate(-90deg);
}

button#fullscreenbtn
{
  display: none;
}

canvas#seekdot
{
  width: 100%;
  height: auto;
}

#seekslider
{
  display: none;
}
```


KISIM		YAPRAK NO : 12
YAPILAN İŞ	İnfografik Resme SeekSlider Özelliği Kazandırılması	TARİH : 04/08/2015
<p>İstenilen arayüz tasarlandıktan infografik resme <i>seekslider</i> özelliği kazandırmak için çalışmalara başlandı. İnfografik resmin ilgili zaman aralığına tıklandığında audio ve video'nun da aynı aralığa gelmesini sağlamak için video ve audio süreleri ile resmin genişliği arasında bir bağıntı kurmak gerekiyordu. Resme tıklanılan noktanın <i>x koordinat değeri / resmin genişliğe oranı</i> video ve audio'da hangi noktaya gidilmesi gerektiğini gösteren bir bağıntı olabilirdi. Geniş çaplı bir araştırmadan sonra aşağıdaki kodlarla öncelikle resmin genişlik değeri alındı.</p> <pre> image = document.getElementById("image"); image_width = image.width; </pre> <p>Ardından resme tıklanılan noktanın <i>x koordinat değerini</i> bulan aşağıdaki <i>getPosition()</i> JavaScript fonksiyonu yazıldı. Bu fonksiyonla <i>x/resim genişliği</i> oranı, <i>time/video veya audio'nun toplam süresi</i> oranı kullanılarak istenilen bağıntı kurulmuş oldu.</p> <pre> function getPosition(event) { x = event.x; y = event.y; if (event.x != undefined && event.y != undefined) { x = event.x; y = event.y; } else // Firefox method to get the position { x = event.clientX + document.body.scrollLeft + document.documentElement.scrollLeft; y = event.clientY + document.body.scrollTop + document.documentElement.scrollTop; } x -= image.offsetLeft; y -= image.offsetTop; ratio = Math.abs((x / image_width) * 100); time = (ratio * video.duration) / 100; video.currentTime = time; audio.currentTime = time; } </pre> <p>Sonrasında <i>SPlay.js</i> dosyasında yer alan <i>init()</i> fonksiyonu içerisine aşağıdaki kod eklenerek resme her tıklandığında <i>getPosition()</i> fonksiyonunun çağırılması sağlandı.</p> <pre> image.addEventListener("click", getPosition, false); </pre> <p>Ancak pencere küçültüldüğünde koordinat değerleri yenilenmediği için resme tıklandığında video ve audio'yu yanlış süreye aldığı gözlemlendi. Bu nedenle bir süre bu problemin çözümü üzerinde duruldu. Pencere resizing olaylarında <i>init()</i> fonksiyonu yeniden çağırılarak bu problem de çözüldü.</p> <pre> window.onresize = init; </pre>		
KONTROL SONUCU		

KISIM		YAPRAK NO : 13
YAPILAN İŞ	Video/Audio'nun Anlık Oynatıldığı Zamanı Gösteren SVG Elementinin Eklenmesi	TARİH : 05/08/2015

Video ve audio'nun anlık oynatıldığı zamanı gösteren bir işaret gerekti. Bunun için **svg etiketi** kullanılarak **Splay.html** dosyasında aşağıda yer alan doğru şekilde bir çubuk oluşturuldu.

```
<svg id="svg_seek">
<line id="line" x1="0" y1="0" x2="0" y2="30" style="stroke:rgb(0,0,0);stroke-width:10" />
</svg>
```

Infografik resmin genişliği, video'nun toplam süresine bölünüp; video'nun o an oynatılan zamanı ile çarpılarak **svg** elementinin o noktada çizimi sağlandı. Bunu sağlayan **drawSeekStick()** **Javascript** fonksiyonu **Splay.js** dosyasına eklendi.

```
function drawSeekStick()
{
//oynatılan videonun neresinde olduğumuzu gösteren çubuğun yerini güncelliyor
rate = (image.width / video.duration) * video.currentTime;
seekstick.setAttribute("x1", rate);
seekstick.setAttribute("x2", rate);
}
```

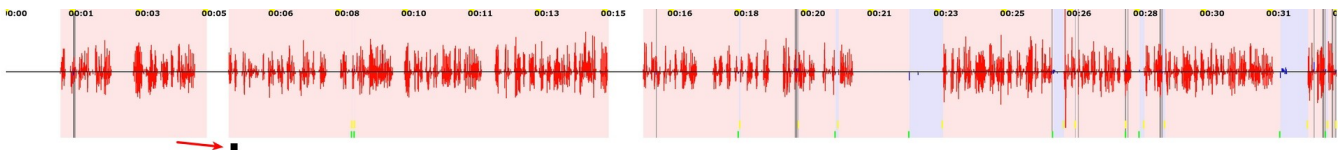
Ardından bu fonksiyon **seekProgress()** fonksiyonunda çağrıldı,

```
function seekProgress()
{
drawSeekStick();
}
```

```
seekStick = document.getElementById("line");
```

Aşağıdaki kodla da video oynatımı boyunca bu fonksiyonun çağrımı sağlandı.

```
video.addEventListener("timeupdate", seekProgress, false);
```



Oynatılan videonun neresinde olduğunu gösteren çubuk

Böylece yukarıda resimde okla gösterilen oynatılan medyanın anlık olarak neresinde olduğunu gösteren **seekStick** elementi eklenmiş oldu.

KISIM		YAPRAK NO : 14
YAPILAN İŞ	Klavye Kısayolları Eklenmesi	TARİH : 06/08/2015

Video ve audio oynatılırken kullanıcı kontrollerini kolaylaştırmak için klavye kısayol tuşları eklenmesi gerekliydi. Gerekli araştırma yapıldıktan sonra **Splay.js** dosyasına aşağıdaki kod eklenerek öncelikle klavye tuşlarına olay dinleyici eklendi.

```
window.addEventListener("keydown", checkkey, false);
```

Basılan tuşların **unicode karşılıklarına** göre işlem yapan aşağıdaki **checkKey()** fonksiyonu yazılarak yukarıdaki dinleyicide çağrıldı. Bu fonksiyonda kullanıcının **32 unicode değerine** sahip **Tab** tuşuna basması halinde **video ve audio'nun Play/Pause** etmesi; **38 unicode değerli yukarı yön tuşu** ile **ses artırımı** ve **40 unicode değerli aşağı yön tuşu** ile **ses azaltımı** yapılabilmesi sağlandı.

```
function checkKey(e)
{
  var code = e.keyCode;
  if (code == 32)
  {//32 matches Tab Button for PLayer/Pause
    playpause();
  }
  else if (code == 38)
  {
    audio.volume += 0.1;
  }
  else if (code == 40)
  {
    audio.volume -= 0.1;
  }
}
```

Arayüz üzerinde fare ile sağ tıklandığında bir menü çıkmaması için aşağıdaki kod **init()** fonksiyonu içerisinde eklendi.

```
window.oncontextmenu = function ()
{
  return false;
}
```

Video üzerine çift tıklandığında arayüzün tam ekran moduna geçmesi ya da tam ekran modundayken normal durumuna dönmesi için **“dblclick”** olayında **toggleFullScreen()** fonksiyonunun çağırılması aşağıdaki şekilde gerçekleştirildi.

```
video.addEventListener("dblclick", toggleFullScreen, false);
```

Tam ekran moduna geçiş farklı tarayıcılarda farklı şekilde gerçekleştiği için tarayıcıya göre kontrol, bir sonraki sayfadaki şekilde sağlandı.

```

function toggleFullScreen()
{
    if (!document.fullscreenElement && // alternative standard method
        !document.mozFullScreenElement && !document.webkitFullscreenElement && !
document.msFullscreenElement) { // current working methods
        if (document.documentElement.requestFullscreen) {
            document.documentElement.requestFullscreen();
        } else if (document.documentElement.msRequestFullscreen) {
            document.documentElement.msRequestFullscreen();
        } else if (document.documentElement.mozRequestFullScreen) {
            document.documentElement.mozRequestFullScreen();
        } else if (document.documentElement.webkitRequestFullscreen) {
            document.documentElement.webkitRequestFullscreen(Element.ALLOW_KEYBOARD_INPUT);
        }
    } else {
        if (document.exitFullscreen) {
            document.exitFullscreen();
        } else if (document.msExitFullscreen) {
            document.msExitFullscreen();
        } else if (document.mozCancelFullScreen) {
            document.mozCancelFullScreen();
        } else if (document.webkitExitFullscreen) {
            document.webkitExitFullscreen();
        }
    }
}
}

```

KISIM		YAPRAK NO : 15
YAPILAN İŞ	Responsive Layout Oluşturulması	TARİH : 07/08/2015

Eklenen kısayollardan sonra butonlara ihtiyaç kalmadığı için ilgili mühendis butonların arayüzden kaldırılmasını istedi. Bunun üzerine butonlar kaldırıldı ve arayüz aşağıdaki sade şekle dönüştü.



Arayüzün pencerenin küçültülmesi veya büyütülmesine bağlı olarak görünümünü koruması istendi. *Splay.css* dosyasındaki elementlerin **max-width değerleri 100%** ve **height değerleri auto** olarak setlenerek **responsive'lik** sağlanmaya çalışıldı. Ancak video elementinin **aspect ratio**'su nedeniyle **CSS** kodlarıyla **responsive'lik** sağlanamadı. Bunun için aşağıdaki **videoScale()** **JavaScript** fonksiyonu ile pencerenin büyüklüğüne göre video'nun genişliğinin ayarlanması sağlandı.

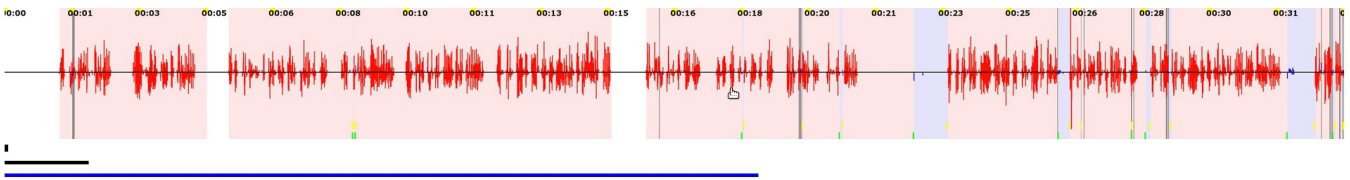
```
function videoScale()
{
    videobox.height = window.outerHeight;

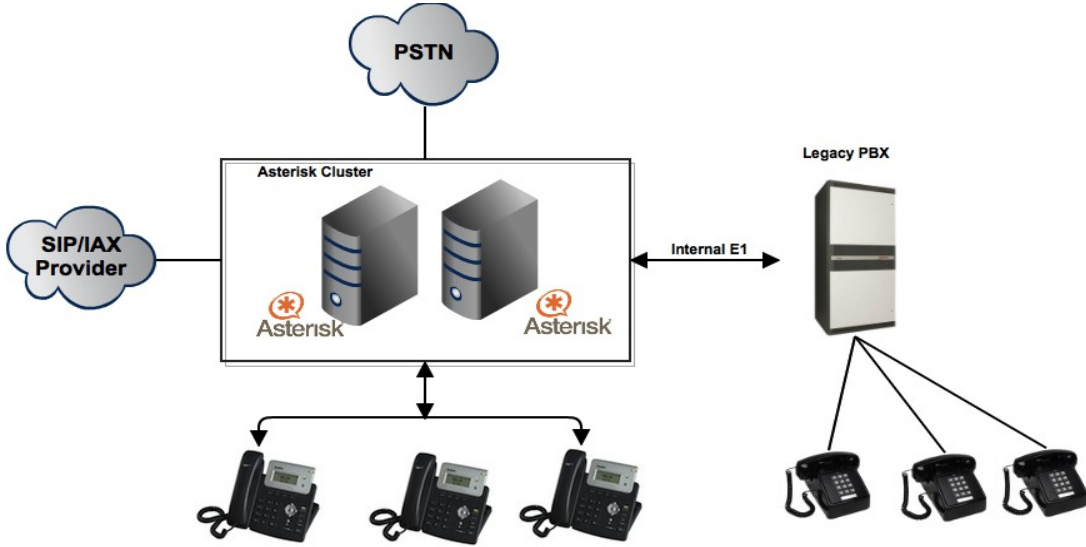
    if (window.innerWidth < 160)
    {
        video.width = 80;
    }
    else if (window.innerWidth < 320)
    {
        video.width = 160;
    }
    else if (window.innerWidth < 480)
    {
        video.width = 320;
    }
    else if (window.innerWidth < 640)
    {
        video.width = 480;
    }
    else if (window.innerWidth < 800)
    {
        video.width = 640;
    }
}
```

```
else if (window.innerWidth < 1000)  
{  
    video.width = 800;  
}  
else  
    video.width = 960;  
}
```

Ardında ***init()*** fonksiyonu içerisinde bu fonksiyon eklenerek ***resize*** işlemlerinde sürekli çağırımı sağlandı. Bu sayede pencere küçültülse de büyültülse de görünümünü koruması sağlandı.

```
Init()  
{  
    videoScale();  
}
```

KISIM		YAPRAK NO : 16
YAPILAN İŞ	Buffered Çizgilerinin Oluşturulması	TARİH : 10/08/2015
<p>Video ve audio farklı sunucularda bulunduğu için hangisinin ne kadar bufferlandığının gösterilmesi internet bağlantısının yavaş olduğu durumlarda kullanıcıya bilgi vermesi çok yararlı olabilirdi. Çünkü video veya audio'nun birinin gecikmesi senkronizasyon problemine neden olmaktaydı. Bu nedenle video ve audio'nun her birinin ne kadarının bufferlandığını gösteren bir seek şekli oluşturuldu.</p> <pre> <svg id="bufferedVideoProgress" > <line id="bufferedVideoLine" x1="0" y1="1" x2="1" y2="1" style="stroke:black;stroke-width:10"/> </svg> <svg id="bufferedAudioProgress" > <line id="bufferedAudioLine" x1="0" y1="1" x2="1" y2="1" style="stroke:blue;stroke-width:10"/> </svg> </pre> <p>Ardından aşağıdaki showBufferedVideoProgress() ve showBufferedAudioProgress() fonksiyonları oluşturuldu. Bu fonksiyonlar “progress” olayında çağrılarak, infografik ses resmi altında; video ve audio'nun bufferlanma miktarlarına göre siyah ve mavi çizgiler oluşturulmasını sağladılar.</p> <pre> video.addEventListener("progress", showBufferedVideoProgress, false); audio.addEventListener("progress", showBufferedAudioProgress, false); function showBufferedVideoProgress() { bufferedVideoEnd = video.buffered.end(video.buffered.length - 1); bufferedVideoRatio = (window.innerWidth / video.duration); bufferedVideoLine.setAttribute("x1", 0); bufferedVideoLine.setAttribute("x2", (bufferedVideoEnd * bufferedVideoRatio)); } function showBufferedAudioProgress() { bufferedAudioEnd = audio.buffered.end(audio.buffered.length - 1); bufferedAudioRatio = (window.innerWidth / audio.duration); bufferedAudioLine.setAttribute("x1", 0); bufferedAudioLine.setAttribute("x2", (bufferedAudioEnd * bufferedAudioRatio)); } </pre>  <p>Böylece video ve audio'nun ne kadarının yüklendiği arayüzde görüntülenmesi sağlanmış oldu.</p>		
KONTROL SONUCU		

KISIM		YAPRAK NO : 17
YAPILAN İŞ	TFTP Server Entegrasyonu	TARİH : 11/08/2015
<p>Başka bir stajyer arkadaşın yürüttüğü; FTP' nin temel fonksiyonel şekli olarak ifade edilen basit bir dosya transfer protokolü olan TFTP Server bitince, Tegsoft yazılımına entegrasyonu için bir toplantı yapıldı. 6 saat süren toplantı boyunca TegSoft yazılımı ve kaynak kodları hakkında geniş çaplı bilgi verildi. Adım adım TFTP Server'in sisteme entegrasyonu gösterildi.</p> <p>Toplantı süresince şirketin kullandığı,</p> <ul style="list-style-type: none"> Ajax bazlı; veritabanı araçları ile kullanıcı arayüzü bileşenlerini de içeren açık kaynak kodlu ticari Java yazılımı tobe, Ticari mobil ve web uygulamaları için kolay grafik arayüz oluşturmada kullanılan; Java temelli ve açık kaynak kodlu ZK framework'ü, Zorlu iş yükleri için ölçeklenebilir ve açıldığı gibi kullanıma hazır hale gelen; operasyonel olarak da çok hızlı çalışan IBM DB2 veritabanının yapısı ve diğer veritabanlarına göre avantajları, IP bazlı telefonlarda Internet Protokol Ağları üzerinden video, ses ve anlık mesaj iletişimi için kullanılan SIP ağ protokol, IP santral özelliklerini barındıran, açık kaynak kodlu bir PBX(telephone Private Branch Exchange) yazılımı olan asterisk yazılımı ile ilgili bilgiler verildi. 		
 <p>The diagram illustrates a telephony system architecture. At the top, a cloud labeled 'PSTN' is connected to a central box labeled 'Asterisk Cluster'. This cluster contains two server icons, each with an 'Asterisk' logo. To the left of the cluster is a cloud labeled 'SIP/IAX Provider'. To the right, a box labeled 'Legacy PBX' is connected to the cluster via a double-headed arrow labeled 'Internal E1'. Below the Asterisk Cluster, three IP phone icons are connected to the cluster. Below the Legacy PBX, three traditional desk phone icons are connected to the PBX.</p>		
KONTROL SONUCU		

KISIM		YAPRAK NO : 18
YAPILAN İŞ	Buffered Çizgilerinin Oluşturulması II	TARİH : 12/08/2015

svg ile oluşturulan **buffered göstergesi** **Google Chrome** tarayıcısında sorun çıkarmazken, **Firefox** tarayıcısında düzgün çalışmadı. Uzun süre problem aşılmaya çalışılsa da bir sonuç elde edilemedi. Bunun üzerine **HTML5**'in **<canvas>** etiketi ile **bufferlanan video** ve **audio**'nun gösterilmesi denendi.

HTML5 ile gelen **canvas** etiketinin kullanımı araştırıldıktan sonra **Splay.html** dosyasında aşağıdaki **canvas elementleri** oluşturuldu.

```
<div>
<canvas id="bufferedVideoLine" style="border:1px solid #c3c3c3;">
</canvas>
</div>
```

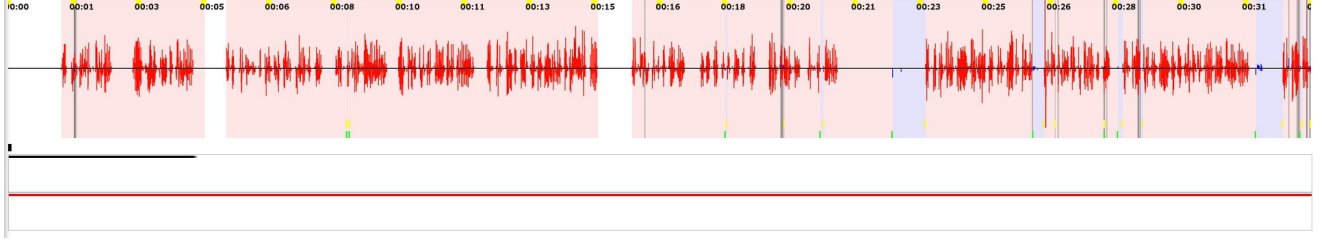
```
<div>
<canvas id="bufferedAudioLine" style="border:1px solid #c3c3c3;">
</canvas>
</div>
```

Sonrasında **showBufferedVideoProgress()** ve **showBufferedAudioProgress()** **JavaScript** fonksiyonları aşağıdaki şekilde oluşturuldu. Bu fonksiyonlarda **video'nun buffer'lanmış** aralıklarını öğrenmek için **DOM**'un **video.buffered** özelliği kullanıldı. **video.buffered.length** toplam **buffer** aralığını verirken, **video.buffered.start(index)** ve **video.buffered.end(index)** fonksiyonları ise bu aralıkların bitiş ve başlangıç zamanlarını vermektedir. Elde edilen bu değerler **canvas**'ın **context.fillRect()** fonksiyonu ile çizdirilmiştir.

```
function showBufferedVideoProgress()
{
    bufferedVideoRatio = (window.innerWidth / video.duration);
    var buffered=video.buffered;
    videoContext=bufferedVideoLine.getContext("2d");

    if (buffered)
    {
        for (var i=0; i<buffered.length; i++)
        {
            var start = buffered.start(i) * bufferedVideoRatio;
            var end = buffered.end(i) * bufferedVideoRatio;
            videoContext.fillRect(start, 3, Math.max(2, end-start), 10);
        }
    }
}
```

```
function showBufferedAudioProgress()
{
    bufferedAudioRatio = (window.innerWidth / audio.duration);
    var buffered=audio.buffered;
    audioContext=bufferedAudioLine.getContext("2d");
    audioContext.fillStyle = "#FF0000";
    if (buffered)
    {
        for (var i=0; i<buffered.length; i++)
        {
            var start = buffered.start(i) * bufferedAudioRatio;
            var end = buffered.end(i) * bufferedAudioRatio;
            audioContext.fillRect(start, 3, Math.max(2, end-start), 10);
        }
    }
}
```



Buffered Göstergeleri

Ancak yine **Google Chrome**'da düzgün **render** edilen gösterge **Firefox** tarayıcısında sorun çıkardı. Denenen bir çok yönteme rağmen **Firefox**'ta düzgün bir sonuç elde edilemedi.

KISIM		YAPRAK NO : 19
YAPILAN İŞ	Arayüz İyileştirmeleri	TARİH : 13/08/2015
<p>HTML5 Player ekranına çift tıklandığında infografik audio resmi seçili kalmaktaydı. Bu hatayı düzeltmek için CSS ile seçim özelliği kapatıldı.</p> <pre> img { -khtml-user-select: none; -o-user-select: none; -moz-user-select: none; -webkit-user-select: none; user-select: none; } </pre> <p>Fare infografik resim üzerine gelince işaretçinin ok yerine seçim moduna dönüşmesi için cursor:pointer kodu eklendi.</p> <pre> img { cursor: pointer; } </pre> <p>Ses artırımı azaltımı için kullanılan yukarı-aşağı yön tuşlarına basıldığında ekran aşağı yukarı scroll etmekteydi. Bunu önlemek için de Splay.js dosyasındaki checkKey() fonksiyonuna aşağıdaki tüm kısayol işlevlerini devredışı bırakan kod eklendi.</p> <pre> checkKey(e) { ... e.preventDefault(); ... } </pre>		
KONTROL SONUCU		