

LabSense

Tunahan KANBAK, Murat UYANIK

Gerçekleştirim Belgesi **Yazılım Geliştirme Projesi**

Bilgi Teknolojileri Sertifika Programı **İDEA: ODTÜ Sanal Kampüsü**

21 Temmuz 2022

1 Öz

Bu rapor, daha önceki raporlarda analiz edilen müşteri ihtiyaçlarının tasarım raporunda tanımlanmış mimari yapıya olabildiğince sadık kalınarak oluşturulan ürünün detaylarını içermektedir. Ürünün gerçekleştirimi sırasında kullanılan veritabanı tabloları, ara yüz görüntüleri, kod parçaları bu rapor içerisinde aktarılmıştır. Tasarım sırasında karşılaşılan problemler ve bunları çözümleri hakkında bilgiler raporlandırılmıştır.

2 Karşılaşılan Sorunlar ve Uygulanan Çözümler

LabSense yazılımının gerçekleştirimi sırasında tasarım aşamasında belirlenen mimariye genel olarak uyulmuş olsa da bazı işlemlerin gerçekleştirilebilmesi için yardımcı fonksiyonların tanımlanması gerekmiştir. Bu eklemeler küçük çaplı olup programın genel yapısına ve akışına etki etmemiştir. Ancak, tasarımda ön görülemeyen ve yapıyı/çalışma mantığını etkileyen sapmalar aşağıda aktarılmıştır.

LabSense yazılımının gerçekleştirimi sırasında karşılaşılan en temel problem, veritabanı servisinin paralel çalışan “dash.callback”lerini yönetmesi sırasında oluşmuştur. LabSense yazılımı üç temel işlevin farklı sayfalarda yer aldığı bir yazılımdır. Her sayfanın veritabanıyla olan ilişkisi ayrı veritabanı fonksiyonları (deney_sonucu_goruntule() ve deney_talebi_goruntule() vb.) ile yönetilmektedir. Bu fonksiyonları kullanan “dash.callback” fonksiyonları aynı sinyale (“login”) bağlandığı zaman veritabanında tutarsız hatalar (hem hata alınan işlem hem de hata mesajları tutarlı değildir.) alındığı ve programın kendisini sonlandırdığı görülmüştür. İlk çözüm olarak try-except blokları ile hata kurtarılmaya çalışılmıştır ancak bu bloklar ile hata kurtarılamadan programın yine de kendini sonlandırdığı görülmüştür. Bir sonraki çözüm olarak her sayfa içerisindeki veritabanı fonksiyonlarını çağıran “dash.callback” sinyalleri ortak “login” sinyali yerine “active-tab” sinyaline çevrilmiştir. Böylece seçilen tab’a geçiş yapıldığı sırada sadece o tab için gerekli veritabanı işlemleri gerçekleştirilerek problem çözülebilmektedir.

Veritabanı iletişimi “active_tab” sinyaline bağlandığı için Sorgulama (Query) sayfası her aktifleştğinde veritabanı ile iletişim kurularak göreceli olarak büyük verilerin transferi yapılmaktaydı. Bu durum, sayfanın her açılışında yazılımın fazla veri işleyerek yavaş tepki vermesine sebep olmaktaydı. Bu sayfada, iletişim kurulan veri tablosunun tamamının çekilmesi kullanımı rahatlatıldığı için hala yazılım içerisinde mevcut bir özelliktir. Ancak büyük verilerin tekrar tekrar çekilmemesi için oturuma bağlı bir değişken ile verinin daha önce çekilip çekilmediği kontrol edilerek her kullanıcı oturumu için tek seferlik veri çekimi gerçekleştirilmiştir.

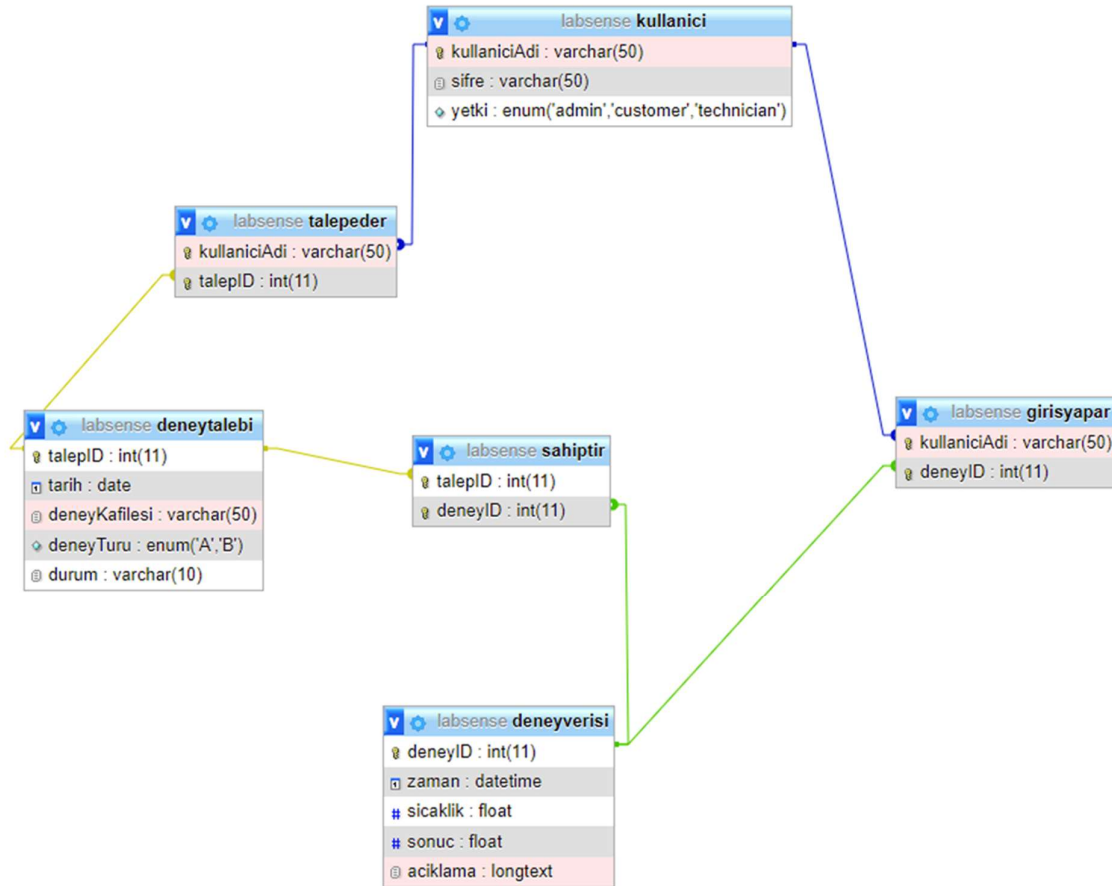
Karşılaşılan diğer problem ise mantık hatası temelli bir problemdir. Veritabanında saklanan deney verileri tarih etiketlerine sahip olup gösterilen grafiğin x-ekseni bu tarih verisini kullanmaktadır. Ancak farklı tarihlerde yapılan deneyler grafik üzerinde gösterildiğinde x-ekseninde çok dağınık veri kümeleri (örneğin karşılaştırılmak istenen numune hem ay başı hem de ay sonu test edildiğinde grafikte ayrık iki veri oluşmaktadır.) oluşmuştur. Ancak, analiz dokümanında da belirtildiği üzere bu grafiğin amacı kullanıcıların deney verilerini kıyaslayabileceği bir arayüz oluşturmaktadır. Bu probleme çözüm olarak, verilerin kendi

içerisindeki zaman verisinin “zaman-farkı” (timedelta) verilerine dönüştürülmesi belirlenmiştir. Müşteri tarafından belirtilen bir bilgi olarak her deney verisi grubu, içerisinde 1 adet “KS” açıklamasına sahip deney sonucu içermektedir. Bu açıklama özelliği t=0 kabul edilerek diğer deney verileri için -/+ “zaman-farkı” verileri türetilmiştir. “KS” açıklamasını içermeyen deney veri gruplarında ise ilk veri baz alınarak “zaman-farkı” verisi türetilerek grafikte kullanılmıştır.

3 Yazılım Bileşenleri ve Görevleri

3.1 Veritabanı

Veritabanı gerçekleştirimi tasarım raporunda belirtilen özelliklere sadık kalınarak gerçekleştirilmiştir. Veritabanı Yönetim Sistemi olarak MySQL kullanılmış olup lokal sistemde yapılan çalışmalar süresince veritabanı kontrolü WAMPServer yazılımında yer alan PHPMyAdmin aracılığı ile gerçekleştirilmiştir. Veritabanı motoru olarak “InnoDB” seçilmiştir. Bu sayede farklı tablolardaki birincil anahtarlar ilişki tablolarında yabancı anahtar olarak kullanılarak tablo ilişkileri kurulmuştur. Tablo ilişkilerinin yer aldığı MySQL şeması aşağıda paylaşılmıştır.



Şekil 1 LabSense Yazılımı MySQL Veritabanı Şeması

LabSense yazılımı için kullanılan veritabanı tablolarına ait tablo oluşturu kod parçaları aşağıda yer almaktadır.

- deneytalebi tablosu:

```
CREATE TABLE IF NOT EXISTS `deneytalebi` (  
  `talepID` int(11) NOT NULL AUTO_INCREMENT,  
  `tarih` date NOT NULL,  
  `deneyKafilesi` varchar(50) CHARACTER SET utf8 NOT NULL,  
  `deneyTuru` enum('A','B') CHARACTER SET utf8 NOT NULL,  
  `durum` varchar(10) CHARACTER SET utf8 NOT NULL,  
  PRIMARY KEY (`talepID`)  
) ENGINE=InnoDB AUTO_INCREMENT=172 DEFAULT CHARSET=latin1;
```

- deneyverisi tablosu:

```
CREATE TABLE IF NOT EXISTS `deneyverisi` (  
  `deneyID` int(11) NOT NULL AUTO_INCREMENT,  
  `zaman` datetime NOT NULL,  
  `sicaklik` float NOT NULL,  
  `sonuc` float NOT NULL,  
  `aciklama` longtext CHARACTER SET utf8 NOT NULL,  
  PRIMARY KEY (`deneyID`)  
) ENGINE=InnoDB AUTO_INCREMENT=991 DEFAULT CHARSET=latin1;
```

- girisyapar tablosu:

```
CREATE TABLE IF NOT EXISTS `girisyapar` (  
  `kullaniciAdi` varchar(50) NOT NULL,  
  `deneyID` int(11) NOT NULL,  
  PRIMARY KEY (`kullaniciAdi`,`deneyID`),  
  KEY `girisyapar_fkkey_deneyID` (`deneyID`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

- kullanıcı tablosu:

```
CREATE TABLE IF NOT EXISTS `kullanici` (  
  `kullaniciAdi` varchar(50) NOT NULL,  
  `sifre` varchar(50) NOT NULL,  
  `yetki` enum('admin','customer','technician') NOT NULL,  
  PRIMARY KEY (`kullaniciAdi`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

- sahiptir tablosu:

```
CREATE TABLE IF NOT EXISTS `sahiptir` (
  `talepID` int(11) NOT NULL,
  `deneyID` int(11) NOT NULL,
  PRIMARY KEY (`talepID`,`deneyID`),
  KEY `sahiptir_fkkey_deneyID` (`deneyID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

- talepeder tablosu:

```
CREATE TABLE IF NOT EXISTS `talepeder` (
  `kullaniciAdi` varchar(50) NOT NULL,
  `talepID` int(11) NOT NULL,
  PRIMARY KEY (`kullaniciAdi`,`talepID`),
  KEY `talepeder_fkkey_talepID` (`talepID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

LabSense arayüzünün veritabanına veri aktarırken veya veritabanından veri çekerken kullandığı SQL sorguları ve amaçları aşağıda verilmiştir.

- Kullanıcının kıyaslama ve sonuçları görmesi adına veritabanına aşağıdaki sorgu ile ulaşarak veri çekilir.

SELECT s.deneyID, s.talepID, DV.sicaklik, DV.sonuc, DV.aciklama, DV.zaman **FROM** deneyverisi DV, talepeder T, sahiptir S **WHERE** DV.deneyID=S.deneyID **AND** T.talepID=S.talepID

- Laboratuvar çalışanının açık talepler olup olmadığını kontrol edebilmesi için veritabanından aşağıdaki sorgu aracılığı ile kapatılmamış deneyler çekilir.

SELECT talepID **FROM** deneytalebi **WHERE** durum = "open"

- Giriş yapan kullanıcının yetki seviyesi aşağıdaki veritabanı sorgusu ile kontrol edilir.

SELECT yetki **FROM** kullanıcı **WHERE** kullanıcıAdi=%s **AND** sifre=%s

3.2 LabSense Arayüz Modülleri

LabSense arayüzü plotly tarafından oluşturulmuş dash kütüphanesinden yararlanılarak oluşturulmuştur. LabSense arayüz gerçekleştirimi için Python 3.9 kullanılmış olup kullanılan kütüphaneler aşağıda paylaşılmıştır.

- Dash
 - Html
 - Dcc
 - Input
 - Output
 - State
 - Ctx

- Exceptions
- Plotly
 - Express
 - Graph_objects
- Dash_bootstrap_components
- Dash_mantime_components
- Numpy
- Pandas
- Datetime

Modüller içerisinde yer alan fonksiyonların gerçekleştirimleri Ekler kısmında verilmiştir.

3.2.1 login_page.py Modülü

Bu modül kullanıcı adı ve şifre bilgisini kullanıcıdan alarak veritabanı ile ilişki kurar. Veritabanından gelen yanıtı göre hatalı giriş uyarısı verir veya yetkiye uygun tablaları aktifleştirir.

Login_page.py modülü yetki kontrolü ardından kendi bulunduğu tab'ı deaktive eder. Bu modülün işlevselliğini tekrar kullanmak için Logout butonuna tıklanarak login_page.py modülüne dönülebilir. Ancak, login_page.py modülüne döndüğünde diğer tablalar , içlerindeki veriyi koruyarak, tekrardan deaktive hale gelir.

login_page.py modülünde yer alan bir adet callback fonksiyonu bulunmaktadır.

- yetkiKontrol(n_clicks, tab, uname, pword): Arayüzden alınan kullanıcı adı ve şifresini veritabanına ileterek yetki kontrolünü gerçekleştirir. Veritabanından iletilen yetki seviyesine göre kullanıcıya uygun tablaları aktifleştirir.
 - n_clicks: login butonuna kaç kere basıldığıнын tutulduğu değişkendir.
 - tab: anlık aktif edilen tab bilgisini tutar.
 - uname: login_page.py sayfasında yer alan kullanıcı adı kısmındaki veridir.
 - pword: login_page.py sayfasında yer alan şifre kısmındaki veridir.

3.2.2 request_page.py Modülü

Bu modül kullanıcıların yeni deney talebinde bulunmalarını sağlamaktadır. Yeni talebin veritabanına başarılı bir şekilde işlenmesinin ardından kullanıcıya yaptığı talebin ID'sini ileterek kullanıcıyı bilgilendirir. Aksi halde veritabanında bir hata ile karşılaşarak deney talebini işleyemediğini belirtir.

request_page.py içerisinde 3 farklı callback fonksiyonu yer almaktadır. Bu fonksiyonlar ve görevleri aşağıda paylaşılmıştır.

- talebilsle(sub_click, modal_click, uname, sub_date, exp_name, exp_type): Bu fonksiyon arayüzde yer alan veri giriş bölgelerindeki verileri toplayarak bir sözlük haline getirir. Daha sonra bu sözlük veritabanına işlenmek üzere veritabanına iletilir.

Veritabanının dönüşüne göre kullanıcıya yeni talep numarasını iletir veya talebin alınamadığı bilgisini verir.

- sub_click: Arayüzde yer alan gönder butonuna kaç kere basıldığını tutan değişkendir. Kullanıcının gönder tuşuna basıp basmadığını anlamak için kullanılır.
- modal_click: Talebin işlenmesine üzerine kullanıcıya gösterilen uyarı mesajının kapatıldığını ileten değişkendir.
- uname: arayüzde yer alan kullanıcı kısmındaki veriyi tutan değişkendir.
- sub_date: arayüzde yer alan tarih alanındaki veriyi tutan değişkendir.
- exp_name: arayüzde yer alan deney kfilesi alanındaki veriyi tutan değişkendir.
- exp_type: arayüzde yer alan deney türü alanındaki veriyi tutan değişkendir.
- tabloyuSıfırla(response): Bu fonksiyon arayüz üzerinden talep gönderimi tamamlandıktan sonra verinin orijinal hale gelmesini sağlayarak kullanıcının yeni talep yapabilmesi için formu yeniler.
 - response: Talebin veri tabanına işlenip işlenemediğini belirten değişkendir.
- setKullanıcıAdı(log_click, uname): Deney talebinde bulunan kullanıcının adı login_page.py modülünden alınarak buradaki tabloya işlenir. Böylece kullanıcının kullanıcı adını tekrar tekrar girmesine gerek kalmaz.
 - log_click: login_page.py’da bulunan login tuşuna kaç kere tıklandığını tutan değişkendir.
 - uname: login_page.py arayüzünde yer alan kullanıcı adı alanındaki veriyi tutan değişkendir.

3.2.3 query_page.py Modülü

Bu modül kullanıcıların tamamlanmış deney verilerini inceleyebileceği bir arayüz sunmaktadır. Aynı arayüz içerisinde incelenen verinin indirilebilmesi de mümkündür.

query_page.py içerisinde 3 adet callback ve 1 adet yardımcı fonksiyonu bulunmaktadır.

- sonuclarıGoster(experiment_list): kullanıcının ekrandaki çok seçmeli liste aracılığı ile seçtiği verileri filtreleyerek tablo ve grafik oluşturmunu gerçekleştirir. Bu callback fonksiyonu normalizetoKS() isminde bir yardımcı fonksiyon kullanır. Bu yardımcı fonksiyonun amacı pandas.DataFrame objesi olarak tutulan ve tablo/grafik oluşturulurken kullanılacak verilerden “zaman-farkı” verisinin türetilmesidir.
- deneySonucuGuncelle(activated, loaded): Bu callback fonksiyonu “Query” tabının ilk aktifleştirildiği sırada veritabanından deney sonuçlarının çekilmesini sağlar. Veritabanından sürekli veri akışını engellemek için “loaded” adında bir dcc.Store() objesi tutarak verinin daha önce çekilip çekilmediğini kontrol eder.
- hamVeriindir(click, data): Kullanıcının grafik ve tablo üzerinden incelediği veriyi indirmek için kullandığı callback fonksiyonudur.
 - click: İndir butonuna kaç kere basıldığını tutan değişkendir.
 - data: kullanıcının anlık olarak grafik ve tablo üzerinde incelediği verinin tutulduğu sözlük veri yapısına sahip değişkendir.

- `normalizetoKS(group)`: kullanıcının incelediği deney verilerinin talep numarasına göre gruplandırılmasının ardından “zaman-farkı” verisinin türetilmesini sağlayan yardımcı fonksiyondur.
 - `group`: `talepID`’lere göre gruplandırılmış `pandas.DataFrame` objesidir.

3.2.4 `new_data_page.py` Modülü

Bu modül laboratuvar kullanıcısının tamamlanan deneylere ait verilerin girişini yapabildiği arayüzdür. Bu ekran aracılığı ile açık talepler görüntülenebilir ve seçilen talebe ait veri girişi tablo aracılığı ile veritabanına iletilebilir.

`new_data_page.py` içerisinde 5 adet callback fonksiyonu bulunmaktadır.

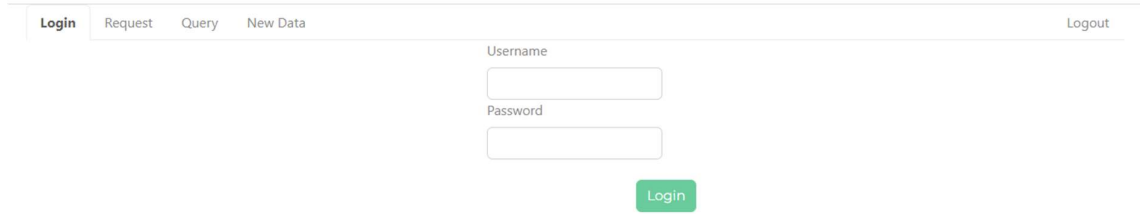
- `formuGüncelle(n_clicks_add, n_clicks_remove, submission, rows)`: Kullanıcının arayüzde yer alan “satır ekle” butonuna bastığında satır eklenmesini, “satır sil” butonuna bastığında son satırın silinmesini ve “gönder” butonuna basıldığında tablonun sıfırlanmasını sağlayan fonksiyondur.
 - `n_clicks_add`: “satır ekle” butonuna kaç kere tıklandığını tutan değişkendir.
 - `n_clicks_remove`: “satır sil” butonuna kaç kere tıklandığını tutan değişkendir.
 - `submission`: veri tabanına verinin iletilip iletilmediğini belirten değişkendir.
 - `rows`: tabloda anlık bulunan satırları tutan değişkendir.
- `veriyiisle(n_clicks, date, children, code)`: Kullanıcının “gönder” tuşuna basmasının ardından arayüzde yer alan verileri toplayarak bir `pandas.DataFrame` objesine dönüştüren fonksiyondur.
 - `n_clicks`: “gönder” butonuna kaç kere tıklandığını tutan değişkendir.
 - `date`: arayüzde yer alan tarih alanındaki veriyi tutan değişkendir.
 - `children`: arayüzde yer alan tablonun tüm iç yapısının tutulduğu değişkendir.
 - `code`: arayüzde yer alan deney adı alanındaki veriyi tutan değişkendir.
- `veriyiAktar(close_click, dataframe, code, operator)`: `veriyiisle()` fonksiyonunu veriyi işleme işlemini tamamlamasının ardından arayüzden alınan verilerin veritabanına iletilmesini sağlayan ve kullanıcıyı sonuç hakkında bilgilendiren fonksiyondur.
 - `close_click`: verinin veritabanına aktarılıp aktarılamadığı hakkında gösterilen alarmin kapatılıp kapatılmadığını belirten değişkendir.
 - `dataframe`: `veriyiisle()` tarafından oluşturulan `pandas.dataframe` objesinin sözlük veriyapısına sahip halini tutan değişkendir.
 - `code`: arayüzde yer alan deney adı alanındaki veriyi tutan değişkendir.
 - `operator`: arayüzde yer alan kullanıcı adı alanındaki veriyi tutan değişkendir.
- `setKullanıcıAdı(log_click, uname)`: Deney talebinde bulunan kullanıcının adı `login_page.py` modülünden alınarak buradaki tabloya işlenir. Böylece kullanıcının kullanıcı adını tekrar tekrar girmesine gerek kalmaz.
 - `log_click`: `login_page.py`’da bulunan login tuşuna kaç kere tıklandığını tutan değişkendir.
 - `uname`: `login_page.py` arayüzünde yer alan kullanıcı adı alanındaki veriyi tutan değişkendir.

- acikTalepGüncelle(res, activated): arayüzde yer alan deney adı alanında görülen açık talep numaralarının veritabanından çekilmesini sağlar.
 - res: yeni veri girişi yapıp yapılmadığını belirten değişkendir.
 - activated: anlık aktif tab verisinin tutulduğu değişkendir.

4 Yazılımdan Kesitler

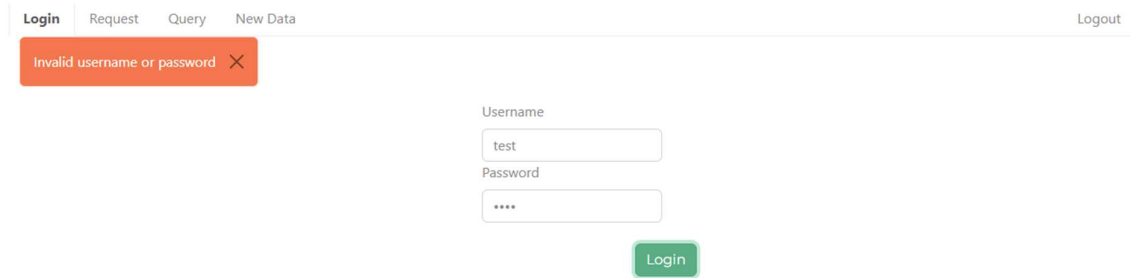
4.1 Giriş Sekmesi

Giriş sekmesi kullanıcı bilgilerinin yazılıma iletildiği arayüzdür. Aşağıdaki görselde de görüldüğü üzere 2 adet veri girişi alanı mevcut olup bir adet gönder butonu mevcuttur.



Şekil 2 Labsense Kullanıcı Giriş Arayüzü

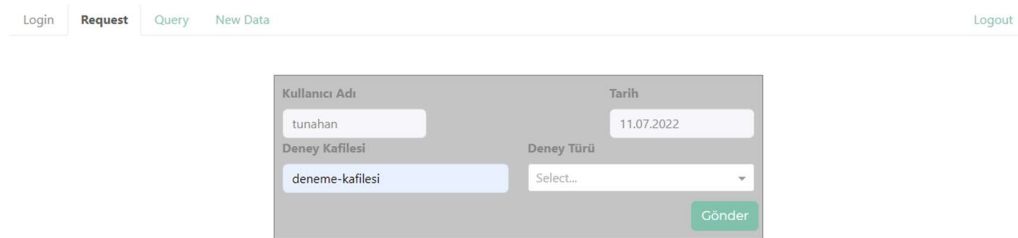
Hatalı giriş yapılması durumunda kullanıcıya bilgi verilmektedir. Örnek bir kullanım aşağıdaki görselde gösterilmektedir.



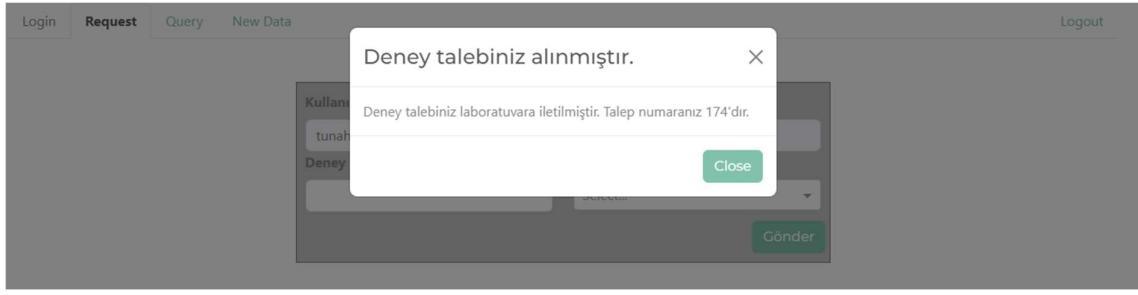
Şekil 3 Hatalı Kullanıcı Girişi Hatası

4.2 Talep Sekmesi

Bu sekmede kullanıcılar yeni deney talebi yapabilmekte olup başarılı talep yapılması durumunda kullanıcıya daha sonra sorgulama ekranında kullanacağı bir barkod numarası iletilir. Aşağıdaki görselde kullanıcının doldurması gereken iki alan belirtilmiştir. Diğer iki alan giriş yapan kullanıcının adı ve günün tarihine göre otomatik doldurulmaktadır.



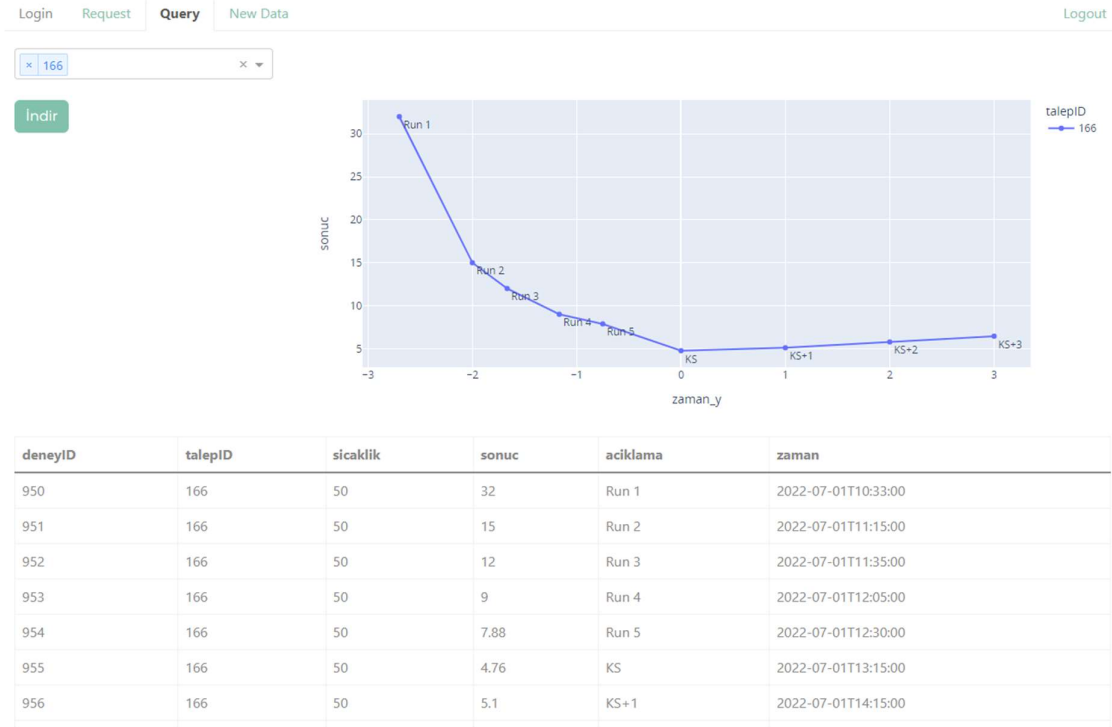
Şekil 4 Deney Talep Giriş Ekranı



Şekil 5 Başarılı Deney Talebi Sonrası Barkod Numarası Mesajı

4.3 Sorgu Sekmesi

Bu sekmede, kullanıcılar tamamlanmış deneylerin sonuçlarını inceleyebilir. Sekme içerisinde bir adet çok seçmeli liste aracılığı ile erişilmek istenilen deney sonuçları seçilir. Seçim yapılan deney taleplerine ait deney verileri tablo ve grafikler aracılığı ile kullanıcıya sunulur. Talep eden kullanıcılar ham verisi indirebilir. Aşağıdaki görselde örnek bir sorgu gösterilmiştir.



Şekil 6 Örnek Deney Sorgulaması

4.4 Yeni Veri Sekmesi

Bu sekmede, laboratuvar kullanıcısı tamamlanan deneylerin verilerini veritabanına yükler. Ekranda yer alan bütün alanların doldurulmasının ardından kullanıcı verileri veritabanına yönlendirebilir.

[Login](#) [Request](#) [Query](#) [New Data](#) [Logout](#)

Deney Adı

Açık Deneyler

Teknisyen

tunahan

Tarih

01.07.2022

Tanım	Zaman	Değer	Sıcaklık
Run 1	13 : 13	4	55
Run 2	14 : 14	5	55
Run 3	15 : 15	6	55
Run 4	16 : 16	7	55

Add Row

Remove Row

Gönder

Şekil 7 Örnek Yeni Veri Giriş Ekranı

5 Kullanıcı Kılavuzu

5.1 Giriş Sekmesi

Giriş sekmesi kullanıcıların sisteme erişimini gerçekleştirmek üzere kullanılabileceği arayüzü oluşturmaktadır.

Bu sekmede yer alan 2 adet veri alanı bulunmaktadır. Bu veri alanları;

- Kullanıcı Adı: Kullanıcıların kullanıcı adını girmesi gereken alandır. utf8_general_ci uyumlu her karakterin kullanılabileceği ve uzunluk sınırı bulunmayan verileri kabul eder. Belirtilen formata sahip olmayan karakterler veritabanı ile kıyaslanamaz.
- Şifre: Kullanıcıların kullanıcı şifresini girmesi gereken alandır. utf8_general_ci uyumlu her karakterin kullanılabileceği ve uzunluk sınırı bulunmayan verileri kabul eder. Belirtilen formata sahip olmayan karakterler veritabanı ile kıyaslanamaz.

“Giriş Yap” butonu aracılığı ile kullanıcı adı ve şifresi veritabanı ile kıyaslanarak yetki kontrolü gerçekleştirilir. Kullanılan kullanıcı adı ve şifresi ikilisi veritabanında bulunmadığı takdirde kullanıcıya hatalı kullanıcı adı veya şifresi hakkında uyarı mesajı iletilir. Veri alanları boş bırakılarak “Giriş Yap” butonu kullanıldığında da hatalı kullanıcı adı ve şifresi şeklinde uyarı verilir.

Kullanıcı adı ve şifresinin veritabanı ile eşlemesi durumunda yetki seviyesi veritabanından alınır ve yetki seviyesine göre sekmeler aktif hale gelir. Yetki kontrol kapsamında üç farklı yetki bulunmaktadır.

- Customer: Laboratuvar müşterilerini temsil eden yetki seviyesidir. Request ve Query sekmelerini kullanabilir.
- Technician: Laboratuvarı temsil eden yetki seviyesidir. New Data sekmesine erişim sağlar.
- Admin: Sistem yöneticisi için tanımlanan yetki seviyesidir. Bütün sekmelere erişim sağlayabilir.

5.2 Talep Sekmesi

Talep sekmesi laboratuvar müşterilerinin yeni deney taleplerini laboratuvara iletebileceği bir arayüzdür.

Talep sekmesinde dört adet veri alanı bulunmaktadır.

- Kullanıcı Adı: Kullanıcının giriş sekmesinde kullandığı kullanıcı adı verisi bu veri alanına işlenir. Böylece kullanıcı talep yaparken hangi hesapta olduğunu kontrol edebilir. Bu veri alanı değiştirilemez.
- Tarih: Kullanıcının talep yaptığı gün otomatik olarak bu alana işlenir. Bu veri alanı değiştirilemez.
- Deney Kfilesi: Veritabanına iletmek üzere kullanıcı tarafından belirtilen numune adıdır. Herhangi bir karakter veya uzunluk limiti bulunmamaktadır. Eşsiz bir isim olması gerekmektedir.
- Deney Türü: Yapılacak deneyin türünü belirten tek seçmeli bir liste elementidir.

Veri alanlarının tamamı doldurulduktan sonra kullanıcı “Gönder” butonu aracılığı ile talebini iletebilir. Talebin başarılı iletilmesi durumunda kullanıcıya bir mesaj kutusu aracılığı ile sorgulama esnasında kullanması gereken ID numarası iletilir.


5.3 Sorgu Sekmesi

Sorgu sekmesi laboratuvar müşterilerinin tamamlanan deneylerine ait sonuçları inceleyebildiği, grafik aracılığı ile kıyaslayabildiği ve incelenen veriyi indirebildiği bir arayüzdür.

Sorgu sekmesinde bir adet veri alanı bulunmaktadır.

- Deney listesi: Kullanıcının incelemek istediği deneyleri talep ID ile seçtiği çoktan seçmeli bir liste elementidir.

Deney listesinin belirlenmesinin ardından kullanıcıya seçilen talep ID’lere ait deney sonuçları grafik ve tablo üzerinden gösterilir. Tablo içerisinde [“deneyID”, “talepID”, “sıcaklık”, “sonuç”, “açıklama”, “zaman”] verileri gösterilmektedir. Grafik üzerinde inceleme yapıldığında ise kullanıcı aynı veri noktasında birden fazla veriyi kıyaslayabilir veya seçilen verileri grafik üzerinde filtreleyebilir.

“İndir” butonuna tıklandığında tabloda anlık olarak görülen verinin tamamı “.csv” formatında kullanıcı bilgisayarına indirilir. Aynı zamanda, grafikte yer alan  sembolüne tıklanarak görüntülenen grafik de kullanıcı bilgisayarına indirilebilir.

5.4 Yeni Veri Sekmesi

Yeni veri sekmesi, laboratuvarın tamamlanan deneyleri işleyebileceği bir arayüzdür.

Yeni veri sekmesinde toplamda 7 farklı veri giriş alanı bulunmaktadır.

- Deney Adı: Daha önce müşteriler tarafından talep edilen deneylerin ID’lerinin yer aldığı tek seçmeli bir liste elementidir.

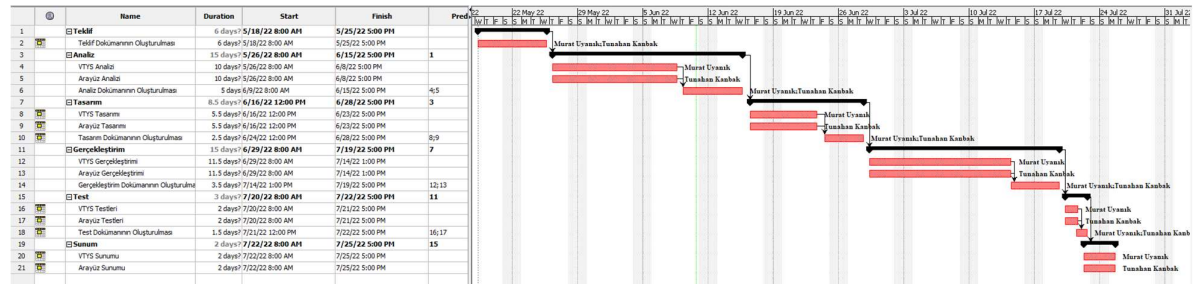
- Teknisyen: Giriş yapan kullanıcının adının belirtildiği veri giriş alanıdır. Kullanıcının giriş yap ekranında kullandığı bilgiye göre otomatik olarak doldurulur. Bu veri alanı değiştirilemez.
- Tarih: Deneyin yapıldığı tarihi gün/ay/yıl olarak belirtmek için kullanılan takvim veri giriş elemanıdır.
- Tanım: Sorgu sayfasında “açıklama” kolonuna denk gelen veridir. Yazı temelli olup ilgili deney sonucunu tanımlayıcı bir açıklama belirtilmesi için kullanılır.
- Zaman: Sadece rakam girişi yapılabilen ve deney tarihindeki hangi saatte/dakikada deney yapıldığını belirtmek için kullanılır.
- Değer: Deney sonucunun belirtildiği numerik veri girişi alanıdır. 0-100 arası herhangi bir sayıyı kabul etmektedir.
- Sıcaklık: Deney sıcaklığının belirtildiği numerik veri girişi alanıdır. 0-60 arası herhangi bir sayıyı kabul etmektedir.

Bütün veri alanları doldurulduktan sonra kullanıcı “Gönder” butonunu kullanarak verileri veritabanına işleyerek talebi kapatabilir ancak herhangi bir veri alanının boş kalması durumunda kullanıcıya veri alanlarının boş bırakılmayacağı uyarısı yapılır ve veriler işlenmez. Verinin başarılı bir şekilde veritabanına işlenmesinin ardından giriş tablosu formatı sıfırlanarak orijinal haline döner.

“Satır ekle” ve “Satır sil” butonları aracılığı ile veri girişinin yapıldığı tablonun satır sayısı değiştirilebilir.

6 Proje Planı ve Uygulama Takvimi

Proje Çağlayan modeli ile geliştirilmektedir. Projenin ilerleyen fazlarında yer alan Tasarım ve Gerçekleştirim aşamaları için yapılan iş bölümü aşağıda paylaşıldığı gibidir:



7 Ekler

LabSense arayüz gerçekleştirimi için kullanılan callback fonksiyonları bu başlık altında aktarılmıştır. LabSense gerçekleştirimi için kullanılan kodların tamamına <https://github.com/TunahanKanbak/LabSense> adresinden ulaşılabilir.

7.1 login_page.py

7.1.1 yetkiKontrol()

```
@app.callback(
    Output("request-tab", "disabled"),
    Output("query-tab", "disabled"),
    Output("logout-tab", "disabled"),
    Output("new-data-tab", "disabled"),
    Output("login-tab", "disabled"),
    Output("tabs", "active_tab"),
    Output("login-error-alert", "is_open"),
    Input("login-control-button", "n_clicks"),
    Input("tabs", "active_tab"),
    State("username-box", "value"),
    State("password-box", "value"),
    prevent_initial_call=True
)
def yetkiKontrol(n_clicks, tab, uname, pword):
    if ctx.triggered_id == "login-control-button":
        if n_clicks is None:
            raise PreventUpdate

        permission_level = labDBmanager.objel.kullanici_sec(uname, pword)
        print(permission_level)

        if permission_level == "admin":
            return False, False, False, False, True, "request-tab", False
        elif permission_level == "technician":
            return dash.no_update, dash.no_update, False, False, True, "new-data-tab",
dash.no_update
        elif permission_level == "customer":
            return False, False, False, dash.no_update, True, "request-tab",
dash.no_update
        else:
            return dash.no_update, dash.no_update, dash.no_update, dash.no_update,
dash.no_update, \
            dash.no_update, True

    if ctx.triggered_id == "tabs" and tab == "logout-tab":
        return True, True, True, True, False, "login-tab", False

    raise PreventUpdate
```

7.2 request_page.py

7.2.1 talebilsle()

```
@app.callback(
    Output('request_modal', 'is_open'),
    Output('title_request_modal', 'children'),
    Output('body_request_modal', 'children'),
    Output('submission_result', 'data'),
    Input('submit_request', 'n_clicks'),
    Input('close_request_modal', 'n_clicks'),
    State('request_user', 'value'),
    State("request_submission_date", 'value'),
    State("request_exp_name", 'value'),
    State("request_exp_type", 'value'),
    prevent_initial_call=True
)
def talebilsle(sub_click, modal_click, uname, sub_date, exp_name, exp_type):
    if ctx.triggered_id == 'close_request_modal':
        return False, dash.no_update, dash.no_update, dash.no_update

    data_dict = {'kullaniciAdi': uname,
                 'talepTarihi': sub_date,
                 'deneyKafilesi': exp_name,
                 'deneyTipi': exp_type}

    if None in data_dict.values() or '' in data_dict.values():
        raise PreventUpdate

    sql_response, request_ID =
labDBmanager.objel.deney_talebi_isle(data_dict['kullaniciAdi'],
data_dict['deneyTipi'],
data_dict['deneyKafilesi'],
data_dict['talepTarihi'])

    if sql_response:
        modal_shown = True
        modal_title = 'Deney talebiniz alınmıştır.'
        modal_text = 'Deney talebiniz laboratuvara iletilmiştir.\nTalep numaranız
{ }\n'dır.'.format(request_ID)
        sub_result = True
        return modal_shown, modal_title, modal_text, sub_result
    else:
        modal_shown = True
        modal_title = 'Deney talebiniz alınamamıştır!'
        modal_text = 'Deney talebiniz alınamamıştır. Lütfen bütün alanları
doldurduğunuzu kontrol ediniz.'
        sub_result = False
        return modal_shown, modal_title, modal_text, sub_result
```

7.2.2 tabloyuSifirla()

```
@app.callback(
    Output('submission_form', 'children'),
    Input('submission_result', 'data')
)
def tabloyuSifirla(response):
    if response:
        return [
            dbc.Row([
                dbc.Col([
                    dbc.Label("Kullanıcı Adı", html_for="request_user",
style={"font-weight": "bold"}),
                    dbc.Input(type="text", disabled=True, id="request_user")
                ]),
                dbc.Col([
                    dbc.Label("Tarih", html_for="request_submission_date",
style={"font-weight": "bold"}),
                    dbc.Input(type="date", disabled=True, value=date.today(),
id="request_submission_date")
                ], width={"offset": 4})
            ]),
            dbc.Row([
                dbc.Col([
                    dbc.Label("Deney Kfilesi", html_for="request_exp_name",
style={"font-weight": "bold"}),
                    dbc.Input(type="text", id="request_exp_name"),
                ]),
                dbc.Col([
                    dbc.Label("Deney Türü", html_for="request_exp_type",
style={"font-weight": "bold"}),
                    dcc.Dropdown(
                        options={
                            "A": "A Tipi Kültür",
                            "B": "B Tipi Kültür"
                        }, id="request_exp_type"
                    )
                ])
            ]),
            dbc.Row([
                dbc.Col([
                    dbc.Button('Gönder', id='submit_request')
                ], width={'size': 'auto', 'offset': 10})
            ], style={'padding-top': '10px'})
        ]
    else:
        return dash.no_update
```


7.2.3 setKullaniciAdi()

```
@app.callback(  
    Output("request_user", "value"),  
    Input("login-control-button", "n_clicks"),  
    State("username-box", "value")  
)  
def setKullaniciAdi(log_click, uname):  
    if log_click is None:  
        raise PreventUpdate  
    else:  
        return uname
```

7.3 query_page.py

7.3.1 sonuclariGoster()

```
@app.callback(
    Output("result-table", "children"),
    Output('result-graph', 'figure'),
    Output('current-data', 'data'),
    Input("result-list-picker", "value"),
    prevent_initial_call=True
)
def sonuclariGoster(experiment_list):
    if len(experiment_list) == 0:
        empty_graph = go.Figure().add_annotation(x=2, y=2, text="No Data to Display",
                                                    font=dict(family="sans serif", size=25,
                                                                color="crimson"),
                                                    showarrow=False, yshift=10)

        return None, empty_graph, {}

    global df_of_results
    df_of_results_filtered =
df_of_results[df_of_results['talepID'].isin(experiment_list)].sort_values(by=['talepID',
'zaman'])
    #Table formati
    table_header = [html.Thead(html.Tr([html.Th(column) for column in
df_of_results_filtered.columns]))]
    table_body = [html.Tbody([html.Tr([html.Td(data) for data in row])
                                for index, row in df_of_results_filtered.iterrows()])]

    #Graph formati
    try:
        dateCol =
df_of_results_filtered.groupby('talepID').apply(normalizeToKS)['zaman']
        dateCol = dateCol/np.timedelta64(1, 'h')
        mergedDF = df_of_results_filtered.merge(dateCol, how='inner', left_index=True,
right_index=True)
    except:
        print('Dataframe time corruption occured.')
        raise PreventUpdate
    graph = px.line(mergedDF, x='zaman_y', y='sonuc', text='aciklama', color='talepID',
hover_name="talepID",
                    hover_data={"zaman_y": False, "sicaklik": True, "sonuc": True,
"talepID": False, "aciklama": False})
    graph.update_traces(textposition="bottom right")
    graph.update_layout(hovermode="x unified")
    return table_header + table_body, graph, df_of_results_filtered.to_dict()
```

7.3.2 deneySonucuGuncelle()

```
@app.callback(  
    Output('result-list-picker', 'options'),  
    Output('data-loaded', 'data'),  
    Input("tabs", "active tab"),  
    State('data-loaded', 'data'),  
    prevent_initial_call=True  
)  
def deneySonucuGuncelle(activated, loaded):  
    if activated == "query-tab" and not(loaded):  
        try:  
            global df_of_results  
            df_of_results = labDBmanager.objel.deney_sonucu_goruntule()  
            list_of_results = df_of_results.loc[:, 'talepID'].unique()  
  
            if list_of_results is None:  
                raise PreventUpdate  
            else:  
                return list_of_results, True  
        except:  
            raise PreventUpdate  
    else:  
        raise PreventUpdate
```

7.3.3 hamVeriİndir()

```
@app.callback(  
    Output('download-dataframe-csv', 'data'),  
    Input('download-raw-data-button', 'n_clicks'),  
    State('current-data', 'data'),  
    prevent_initial_call=True  
)  
def hamVeriİndir(click, data):  
    df = pd.DataFrame(data)  
    return dcc.send_data_frame(df.to_csv,  
    "Result_{}.csv".format(int(datetime.now().timestamp())))
```

7.3.4 normalizeToKS()

```
def normalizeToKS(group):  
    #KS bilgisinin bulunduğu deney gruplarında olcumlerin zaman degerleri KS bolgeleri 0  
    kabul edilerek bagil degerler  
    #atanacaktır.  
    #KS bilgisinin bulunmadigi deneylerde ise ilk olcum zamani KS sayilacaktır.  
    try:  
        group['zaman'] = group['zaman'] - group[group['aciklama'].str.upper() ==  
        'KS']['zaman'].iloc[0]  
    except:  
        group['zaman'] = group['zaman'] - group['zaman'].min()  
    return group
```

7.4 new_data_page.py

7.4.1 formu_güncelle()

```
@app.callback(
    Output('viscosity_table', 'children'),
    Input('add_row', 'n_clicks'),
    Input('remove_row', 'n_clicks'),
    Input('new_data_submission_result', 'data'),
    State('viscosity_table', 'children'),
    prevent_initial_call=True
)
def formuGüncelle(n_clicks_add, n_clicks_remove, submission, rows):
    if ctx.triggered_id == 'add_row' and n_clicks_add is not None:
        rows.append(
            dbc.Row([
                dbc.Col(
                    dbc.Input(placeholder='Run X', type='text'), width={'size': 3}
                ), dbc.Col(
                    dmc.TimeInput(), width={'size': 3}
                ), dbc.Col(
                    dbc.Input(placeholder='4.2', type='number', min=0, max=100),
                    width={'size': 3}
                ),
                dbc.Col(
                    dbc.Input(placeholder='50', type='number', min=20, max=60),
                    width={'size': 3}
                )
            ])
        )
        return rows
    elif ctx.triggered_id == 'remove_row' and n_clicks_remove is not None:
        if len(rows) > 2:
            rows.pop()
            return rows
        else:
            raise PreventUpdate
    elif ctx.triggered_id == 'new_data_submission_result':
        if submission:
            return [
                dbc.Row([
                    dbc.Col(dbc.Label('Tanım'), width={'size': 3}, style={'font-weight':
'bold'}),
                    dbc.Col(dbc.Label('Zaman'), width={'size': 3}, style={'font-weight':
'bold'}),
                    dbc.Col(dbc.Label('Değer'), width={'size': 3}, style={'font-weight':
'bold'}),
                    dbc.Col(dbc.Label('Sıcaklık'), width={'size': 3}, style={'font-
weight': 'bold'})
                ]),
                dbc.Row([
                    dbc.Col(
                        dbc.Input(placeholder='Run X', type='text'), width={'size': 3}
                    ), dbc.Col(
                        dmc.TimeInput(), width={'size': 3}
                    ), dbc.Col(
                        dbc.Input(placeholder='4.2', type='number', min=0, max=100),
                        width={'size': 3}
                    ),
                    dbc.Col(
                        dbc.Input(placeholder='50', type='number', min=20, max=60),
                        width={'size': 3}
                    )
                ])
            ]
        else:
            raise PreventUpdate
    else:
        raise PreventUpdate
```

7.4.2 veriyiisle()

```
@app.callback(
    Output('new_data_submission', 'data'),
    Output("empty-data-field-error", 'is_open'),
    Input('submit data', 'n_clicks'),
    State('date', 'value'),
    State('viscosity_table', 'children'),
    State('code', 'value'),
    prevent_initial_call=True
)
def veriyiisle(n_clicks, date, children, code):
    if any([n_clicks is None, date is None, date == '', code is None, code == '']):
        return dash.no_update, True

    df = pd.DataFrame({'Tag': [],
                       'Time': [],
                       'Result': [],
                       'Temperature': []})

    for r, row in enumerate(children):
        if r == 0:
            continue
        else:
            new_row = []

            for c, col in enumerate(row['props']['children']):
                if 'value' in col['props']['children']['props'].keys():
                    new_row.append(col['props']['children']['props']['value'])
                else:
                    new_row.append(np.nan)

            df.loc[len(df)] = new_row

    if df.isna().sum().sum() != 0:
        return dash.no_update, True

    #Sonuc formundaki zaman kolonu veri girisi yapilan gunu baz almaktadır. Gun degeri
operator
    #tarafindan belirtilen manuel tarihe gore duzeltilir.
    date = datetime.strptime(date, '%Y-%m-%d')

    df['Time'] = df['Time'].astype('datetime64[ns]')
    df['Time'] = df['Time'].apply(lambda dt: dt.replace(day=date.day, month=date.month))

    return df.to_dict(), False
```

7.4.3 veriyiAktar()

```
@app.callback(
    Output('new_data_modal', 'is_open'),
    Output('title_new_data_modal', 'children'),
    Output('body_new_data_modal', 'children'),
    Output('new_data_submission_result', 'data'),
    Input("close_new_data_modal", 'n_clicks'),
    Input('new_data_submission', 'data'),
    State('code', 'value'),
    State('operator', 'value'),
    prevent_initial_call=True
)
def veriyiAktar(close_click, dataframe, code, operator):
    if ctx.triggered_id == 'close_new_data_modal':
        return False, dash.no_update, dash.no_update, dash.no_update

    sql_response = labDBmanager.objel.deney_verisi_isle(code, operator,
pd.DataFrame(dataframe))

    if sql_response:
        modal_shown = True
        modal_title = 'Deney sonucu alınmıştır.'
        modal_text = 'Deney sonucu sisteme işlenmiştir.'
        sub_result = True
        return modal_shown, modal_title, modal_text, sub_result
    else:
        modal_shown = True
        modal_title = 'Deney sonucu alınamamıştır.'
        modal_text = 'Deney sonucu sisteme işlenememiştir.\nLütfen bütün veri alanlarını
doldurduğunuzdan emin olunuz.'
        sub_result = False
        return modal_shown, modal_title, modal_text, sub_result
```


7.4.4 setKullaniciAdi()

```
@app.callback(  
    Output("operator", "value"),  
    Input("login-control-button", "n_clicks"),  
    State("username-box", "value")  
)  
def setKullaniciAdi(log_click, uname):  
    if log_click is None:  
        raise PreventUpdate  
    else:  
        return uname
```

7.4.5 acikTalepGuncelle()

```
@app.callback(
    Output('code', 'options'),
    Output('code', 'value'),
    Input('new_data_submission_result', 'data'),
    Input("tabs", "active tab"),
    prevent_initial_call=True
)
def acikTalepGuncelle(res, activated):
    if (ctx.triggered_id == 'new_data_submission_result' and res) or \
        (ctx.triggered_id == 'tabs' and activated == 'new-data-tab'):
        list_of_open_request = labDBmanager.objel.deney_talebi_goruntule()
        if list_of_open_request is None:
            return ['Sayfayı yenileyin'], None
        else:
            return [{ 'label': exp, 'value': exp} for exp in list_of_open_request], None
    else:
        raise PreventUpdate
```