



**ESKİŞEHİR OSMANGAZİ ÜNİVERSİTESİ
MÜHENDİSLİK - MİMARLIK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
2023-2024 GÜZ YARIYILI**

**VERİ TABANI YÖNETİM SİSTEMLERİ
DERSİ**

SUNUCU SAĞLIĞI İZLEME UYGULAMASI

FİNAL RAPORU

Şube-A Grup-2

152120201039 Hakan Yavaş (Koordinatör)

152120201098 Emre Kart

152120201054 Alperen Güneş

Firma Adı: MERGEN YAZILIM A.Ş

Proje Yetkilisi: Efe Eroğlu, efe.eroglu@mergentech.com.tr, 05076665543

Prof. Dr. AHMET YAZICI

Dr. Öğr. Üyesi SİNEM BOZKURT KESER

OCAK 2024

1. Giriş.....	4
1.1. Proje Amacı ve Hedefi.....	4
1.2. Kullanılan Teknolojiler.....	4
1.3. Yapılan çalışmalar.....	5
1.4. Gelinan nokta.....	5
2. Gereksinimler.....	6
1. Sunumdaki/1.Rapordaki gereksinimlerden sonra eklenenler.....	6
2.1. Fonksiyonel Olmayan Gereksinimler.....	6
2.1.1 Kullanılabilirlik.....	6
2.1.2 Performans.....	7
2.1.3 Güvenlik.....	7
2.1.4 Uyumluluk.....	7
2.1.5 Esneklik.....	8
2.1.6 Ek Özellikler.....	8
2.2. Fonksiyonel Gereksinimler.....	8
2.2.2 Uygulama Arayüz Gereksinimleri.....	8
2.2.3 Yazılım Arayüz Gereksinimleri.....	10
2.2.3.1 SQL Veritabanı Yönetimi.....	10
2.2.3.2 Rest API.....	10
2.2.3.3 Kotlin.....	11
2.3. Veritabanı Gereksinimleri.....	11
2.3.1 Sistem Gereksinimleri:.....	11
2.3.2 Güvenlik Gereksinimleri:.....	11
2.3.3 Veri Bütünlüğü Gereksinimleri:.....	12
2.3.4 Performanslı Veri Erişimi:.....	12
2.3.5 Uyumluluk ve Entegrasyon:.....	12
2.3.6 Veri Modelleri:.....	12
2.3.6.1 Kullanıcılar:.....	12
2.3.6.2 Sunucular.....	13
2.3.6.3 Hastaneler.....	13
2.3.6.4 Lokasyonlar.....	13
2.3.6.5 Sağlık Verileri.....	13
2.3.6.6 Kesinti Durumları.....	14
2.3.6.7 Bildirimler.....	14
2.3.7. Veritabanı ve İşleyışı.....	14
3. Veri Yapısı.....	15
3.1. ER Diyagram tasarımlı.....	15
3..1.1 Veritabanı Tabloları ve Açıklamaları:.....	15
4. Sorgular.....	21
5. VTYS Araç Kullanımı.....	22
5.1. Stored Procedures.....	22

5.2. Triggers.....	29
5.3 Hata Ayıklama (Error handling).....	30
6. Kullanıcı Arayüzü (User Interface).....	32
7. Diğer Entegrasyonlar.....	37
8. Çalışma Adam-saat değerleri.....	38
Görev Dağılımı:.....	38
Adam Saat Verileri.....	39
9. Sonuç ve Değerlendirme.....	41

1. Giriş

1.1. Proje Amacı ve Hedefi

Proje amacı ve hedefi, Mergen Yazılım A.Ş'nin farklı lokasyonlarda bulunan sunucularının çalışma ve sağlık durumlarını detaylı bir şekilde izlemek ve kontrol etmek için geliştirilen bir mobil tabanlı uygulama geliştirmektir. Bu proje, firmanın Türkiye genelindeki farklı lokasyonlardaki sunucularının anlık çalışma ve detaylı sağlık durumlarını tek bir uygulama üzerinden izleyebilmesine olanak sağlayarak, manuel yöntemlerle sunucu kontrolü yapmaktan kaynaklanan zorlukları ortadan kaldırmayı hedeflemektedir.

Firmanın şu anda kullandığı yöntem, anonim bir uygulama ile sunucuların tek tek eklenip ardından takip edilmesi üzerine kuruludur. Ancak, bu mevcut yöntem yalnızca sunucuların anlık çalışma durumunu göstermeye sınırlıdır ve sunucuların detaylı özellikleri, kaynak kullanım verileri veya bulundukları ortam koşulları gibi detayları sunmamaktadır. Bu noktada Mergen Yazılım A.Ş'nin ihtiyacı, sunucu bilgilerini kendi veritabanlarında detaylı olarak depolayabileceği, veritabanından gelen verileri tüm cihazlara eş zamanlı olarak gönderebilecek bir uygulamadır. Bu uygulama, belirli aralıklarla sunucuların çalışma durumunu izler, olası sorunları tespit eder ve kullanıcılarla anlık bildirimler göndererek kritik durumları önceden belirleyebilme imkanı sağlar. Ayrıca, geçmiş durum verilerini depolar ve kullanıcılarla, bu verileri analiz etme olanağı sunar. Uygulama aynı zamanda veri güvenliğini sağlamak ve kullanıcı erişimini yönetmek için uygun güvenlik önlemlerini alırken, kullanıcıların herhangi bir yerden sistem durumunu izleyip yönetebileceğii bir uzaktan erişim imkanı sunar.

Bu projenin hedefleri şunlardır:

- Sunucuların anlık çalışma ve detaylı sağlık durumlarının izlenmesi.
- Kullanıcıların sunucuların detaylı özellikleri, kaynak kullanım verileri, ve bulundukları ortam koşulları gibi kritik faktörleri görebilmesi.
- Anlık bildirimlerle kritik durumların önceden belirlenmesi ve hızlı müdahale imkanı sağlanması.
- Veri güvenliğinin ve kullanıcı erişiminin sağlanması.
- Uzaktan erişim imkanı ve kullanıcı dostu bir arayüzün sunulması.
- Pazarda bulunan benzer uygulamaların aksine kısıtlı erişim ve performans sorunları gibi sorunların çözülerek firmanın rekabet üstünlüğünün artırılması.

1.2. Kullanılan Teknolojiler:

Programlama Dilleri: Java, Kotlin

İş Takip: Click Up

Frameworkler: Spring Boot, Hibernate, JPA

Tasarım: Figma, Adobe XD

Kullanılan Mimariler: Repository Pattern

API Test: Postman

Veritabanı Arayüzü: SQL Server Management Studio

Veritabanı: MSSQL*

Veritabanı ER Diyagram Tasarımı: Visual Paradigm

Sanal Sunucu: Docker

IDE: IntelliJ IDEA, Visual Studio Code, Android Studio

*Kullanılması planlanan PostgreSQL veritabanı öncesinde, uygulamanın T-SQL veritabanı üzerinde Hibernate kullanılarak gerçekleştirilenmiştir. Projede oluşan zaman problemleri ve teknik sorunlar nedeniyle T-SQL'de bulunan stored procedure, trigger ve bunların API entegrasyonu PostgreSQL üzerinde sağlanamamıştır.

1.3.Yapılan çalışmalar

Proje sürecinde, sunucuların detaylı izlenmesi ve sağlık durumlarının takibi için geliştirilen mobil tabanlı uygulamanın gereksinimleri başlangıçta belirlendi daha sonra ilk adım olarak, backend kısmı için gerekli API işlemleri gerçekleştirildi. Bu aşama, uygulamanın veri işlemlerini yönetebilmesi için temel bir yapı oluşturdu. Ardından, mobil uygulama için gereksinimlere uygun bir tasarım Figma aracılığıyla hazırlandı. Tasarım, kullanıcı arayüzüünü ve işlevselligi ön planda tutarak detaylı bir şekilde oluşturuldu. Android Studio üzerinden tasarımın uygulama aşamasına geçildi. Figma'da hazırlanan tasarım, burada hayatı geçirilerek işlevsellik kazandı. Tasarımın kodlaması sırasında, kullanıcı deneyimini artırmak ve gereksinimlere uygunluğu sağlamak için işlevler oluşturuldu. Bu adımların tamamlanmasıyla birlikte tasarım süreci sonlandı ve uygulama, kullanıcı dostu arayüzüyle birlikte fonksiyonel hale getirildi. Son aşamada, backend ile frontend arasında iletişimini sağlamak amacıyla Retrofit kütüphanesi kullanıldı. Bu sayede, oluşturulan API'ler aracılığıyla veritabanına erişim sağlandı. Bu süreçler, uygulamanın veri alışverişi ve işlemlerini güvenli ve verimli bir şekilde gerçekleştirebilmek adına önemli adımları oluşturdu.

1.4.Gelinен нокта

Geliştirilen mobil tabanlı uygulama ile Mergen Yazılım A.Ş'nin farklı lokasyonlardaki sunucularının çalışma ve sağlık durumları detaylı bir şekilde izlenip kontrol edilebilecek bir seviyeye ulaşıldı. Ek olarak uygulama üzerinden yeni hastaneler ve sunucular eklenebilir. Uygulama, sunucuların detaylı özellikleri, kaynak kullanım verileri ve bulundukları ortam koşulları gibi kritik bilgileri gösterirken kullanıcılarla hastane, sunucu ekleme işlemlerinde esneklik sunmaktadır.. Veri güvenliği ve kullanıcı erişimi için uygun önlemler alınmış, uzaktan erişim imkanı ve kullanıcı dostu bir arayüz sunulmuştur. Bu sayede firma, benzer uygulamalara kıyasla kısıtlı erişim ve performans sorunları gibi zorlukları aşmıştır.

2. Gereksinimler

1. Sunumdaki/1.Rapordaki gereksinimlerden sonra eklenenler

Yeniden yapılandırma ve geliştirme süreci boyunca, sağlık veritabanının gereksinimleri detaylandırılmış ve genişletilmiştir. Veritabanı şeması, veri girişi, rol yönetimi ve veri alışveriş gibi bir dizi önemli özellik eklenmiş ve iyileştirilmiştir. Veritabanı şeması, gereksinimlerin daha detaylı şekilde belirtilmesiyle güncellenmiştir. Sağlık verileri daha ayrıntılı olarak tanımlanmıştır. Kullanıcı rolleri oluşturulmuş ve yönetimi sağlanmıştır, bu sayede kullanıcıların veri eklemesine uygulama içinden olanak tanınmıştır. Veritabanına sunucu ve hastane ekleme işlemleri uygulanmış aynı zamanda csv dosyaları import edilerek daha hızlı eklemeler yapılabiliyor bu da kullanıcıların veri yönetimini daha etkin bir şekilde yapabilmesine olanak tanımıştir. Google hesabı ile giriş yapma gereksinimi eklenmiş ve sosyal giriş entegrasyonu sağlanmıştır. Sağlık veritabanının kapsamı genişletilmiştir. Sağlık verileri ve kesintiler, dışa aktarılabilir duruma getirilmiş, bu sayede veri paylaşımı ve raporlama kolaylığı sağlanmıştır. Store Procedures ve Triggerlar ile tetikleyiciler oluşturulmuş ve entegre edilmiştir, veritabanı işlemleri ve veri bütünlüğü güvence altına alınmıştır. Hata işleme mekanizmaları tasarlanmış ve uygulanmış, sisteme olası hata durumlarına karşı önlemler alınmıştır. Bu geliştirmelerle, gereksinimleri daha detaylı şekilde karşılanmış, veri yönetimi ve kullanıcı işlevselligi önemli ölçüde artırılmıştır. Yapılan bu değişikliklerle veritabanı hem daha geniş hem de daha kullanışlı hale getirilmiştir. Kullanılması planlanan PostgreSQL veritabanı öncesinde, uygulamanın T-SQL veritabanı üzerinde Hibernate kullanılarak gerçekleştirilmesi sağlanmıştır. Projede oluşan zaman problemleri ve teknik sorunlar nedeniyle T-SQL'de bulunan stored procedure, trigger ve bunların API entegrasyonu PostgreSQL üzerinde sağlanamamıştır.

2.1.Fonksiyonel Olmayan Gereksinimler

2.1.1 Kullanılabilirlik

Kullanılabilirlik, uygulamanın kolayca anlaşılır ve erişilebilir olmasıyla ilgilidir. Sağlık İzleme uygulamasının **kullanılabilirlik** açısından şu özellikleri sağlaması beklenmektedir:

1. Kolay Kullanıcı Arayüzü: Kullanıcı dostu bir arayüz tasarımı ile, kullanıcıların uygulamayı rahatlıkla kullanabilmesi hedeflenmektedir.
2. Hızlı Erişim: Ana ekran üzerinden temel işlevlere hızlı erişim, kullanıcı deneyimini artırmak için önemlidir.

3. Doğrulama ve Kayıt Süreçleri: Kullanıcıların kolayca kayıt olabilmesi ve doğrulama işlemlerini sorunsuz bir şekilde tamamlayabilmesi için adımların anlaşılır olması gerekmektedir. Bunun için kullanıcı giriş, kayıt ve doğrulama sayfaları istenilen şekilde tasarlancalı.

2.1.2 Performans

Projede geliştirilecek uygulamanın en önemli yapı taşlarından birisi de performanstır. Firmanın şu anda kullanmakta olduğu uygulama performans açısından yeterli değildir. Bu sebeple istenilen **performans gereksinimleri** şu şekildedir:

1. Uygulama açılış süresi asgari düzeyde tutulmalıdır.
2. Giriş ve kayıt olma ekranlarındaki bekleme süreleri uzun olmamalıdır.
3. Doğrulama ekranında girilmesi gereken kodun E-posta ile ulaşma süresi uzun olmamalıdır.
4. Sayfalar arası geçişlerde takılma, aşırı kaynak tüketimi olmamalıdır.
5. Uygulama ve veritabanındaki işlemler mümkün olan en kısa sürede gerçekleşmeli, bir istek için uzun sürelerde beklenmemelidir.

2.1.3 Güvenlik

Projede kullanıcı, sunucu ve lokasyon bilgileri edinilip veritabanında tutulacağından, uygulamanın kullanıldığı her senaryoda güvenlik öncelikli konular arasında olacaktır.

Güvenli bir sistem sağlamak adına beklenen **güvenlik gereksinimleri** şu şekildedir:

1. Kullanıcı adı ve şifre bilgileri uygulama ve veritabanı arasında taşınırken HTTPS güvenlik protokolü ile taşınmalıdır.
2. Hatalı giriş denemelerini engellemek adına güvenlik duvarları(firewall) entegre edilmelidir.
3. Sunucuların bilgi ve güvenliğini sağlamak adına kullanıcılar arasında yetkilendirme yapılmalıdır. Bunun nedeni sunucunun kritik bilgilerini sadece yetkisi olan kullanıcıların değiştirebilmesidir. Şöyled ki:
 - a) Yönetici: Sunucu ekleme, silme, detay bilgileri görüntüleme gibi işlemleri gerçekleştirebilir.
 - b) Görüntüleyici: Sunucu ekleme ve silme yetkisi bulunmaz, sunucuların sadece temel bilgilerini (sunucu adı, sunucu ip, aktiflik durumu) görüntüleyebilir.

2.1.4 Uyumluluk

Projede geliştirilecek olan uygulamanın mobil tabanlı olacağının Giriş kısmında söz edilmişti. Her kullanıcının kullanmakta olduğu telefonların teknik özellikleri, kullanılan Android versiyonu gibi faktörler farklı olabileceğinden uygulamanın sahip olması gereken **uyumluluk gereksinimleri** şu şekildedir:

1. Uygulama Android 7.0 ve üzeri platformları desteklemelidir.
2. Uygulama farklı çözünürlük ve boyutlara uyumlu olmalıdır.
3. Uygulama E-Posta servisleri ile tam uyum içinde çalışmalıdır.
4. Uygulama her ağ koşulunda minimum paket gereksinimi ile çalışmalıdır.
(3G, 4G, WI-FI vb.)
5. Yabancı ülke çalışanları için çoklu dil desteği eklenmelidir. Burada öncelik İngilizce'dir.

2.1.5 Esneklik

Geliştirilecek uygulamanın olası hatalardan arındırılması, yeni sayfaların ve özelliklerin eklenmesi adına esnek olması önem arz etmektedir. Uygulamanın bu konuda sahip olması gereken **esneklik gereksinimleri** şu şekildedir:

1. Sistemin veritabanı esnek olarak geliştirilmelidir, firmanın olası bir altyapı değişikliğine en kısa sürede cevap verilebilmelidir.
2. Uygulamanın tasarımı modüler olmalıdır, tasarıma eklenebilecek ve çıkarılabilen özellikler olması muhtemel olduğundan gelişen kullanıcı isteklerine en hızlı şekilde yanıt verecek tasarıma sahip olmalıdır.

2.1.6 Ek Özellikler

Yukarıda verilmiş olan ana başlık gereksinim maddelerinin dışında firma tarafından uygulamada bulunması istenilen **ek gereksinimler** şu şekildedir:

1. Uygulama çökme ve hata verilerini gerektiğiinde anlık bildirime ek olarak E-Posta ile de gönderebilmelidir.
2. Uygulamada bulunan sunucuların verileri, istenilen zaman aralığında raporlanabilmelidir. Bu raporlama verileri Excel dosyası şeklinde verilmelidir.
3. Kullanıcılar karşılaştıkları sorunları geribildirim menüsü ile geliştiricilere iletebilmelidir.

4. Sunucular için toplu şekilde işlem yapabilme seçeneği geliştirilmelidir, belirli sunucuların toplu olarak güncellenmesi gerekebileceğinden bu seçenek önem arz etmektedir.

2.2 Fonksiyonel Gereksinimler

2.2.2 Uygulama Arayüz Gereksinimleri

Bu bölüm, tasarılanacak olan uygulamanın, kullanıcı arayüzüne ilişkin gereksinimleri detaylandırılmaktadır. Aşağıda belirtilen başlıklar, kullanıcıların uygulamayı etkili bir şekilde kullanabilmeleri için gerekli olan sayfaların gereksinimlerini içermektedir.

1. Kullanıcıların sisteme giriş yapabilmeleri için istenilen sayfadır. Bu sayfa, kullanıcı adı ve şifre ile giriş yapma imkanı sunmalıdır. Ayrıca, hatalı giriş durumunda kullanıcıya bilgilendirme sağlanmalıdır. Aynı zamanda kullanıcı Google hesabı üzerinden giriş yapabilmelidir. Ek olarak, beni hatırla butonu bulunmalıdır. Butona tıklandığında kullanıcı sonraki girişler için hatırlanmalı ve giriş ekranına geçmeden ana sayfaya ulaşmalıdır. Kullanıcıların şifrelerini unuttuğu zaman yeniden belirlemek için şifremi unuttum butonu gerekmektedir. Kullanıcıdan E-Posta adresini istemelidir ve kullanıcıya şifre sıfırlama bağlantısı veya doğrulama kodu gönderilmelidir.
2. Yeni kullanıcıların sisteme kayıt olabilmeleri için bir sayfa gerekmektedir. İlgili sayfa, kullanıcı bilgilerini içeren form alanları ve kayıt işlemini tamamlamak için gerekli butonlar içermelidir. Bu form alanlarında kullanıcı adı, E-Posta ve şifre istenmelidir, kullanıcıdan alınan bilgilerden sonra doğrulama sayfasına yönlendirilebilir. Kullanıcı istege bağlı olarak Google hesabı üzerinden hızlı bir şekilde kayıt sağlayabilmelidir.

Yeni kullanıcı kaydı yapıldığında, kullanıcı doğrulama işlemini gerçekleştirmek için kullanıcı doğrulama sayfası beklenmektedir. Kullanıcının, kayıt olma sayfasında girilen E-Posta adresine doğrulama kodu gönderilmelidir, bu doğrulama kodu sistemde karşılaştırma yapmalı eğer doğruysa kayıt işlemi tamamlanmalıdır.

3. Uygulamaya giriş yapıldığında anasayfa ile karşılaşılması beklenmektedir. Bu sayfa, kullanıcıya genel bir bakış sunmalı ve gerekli işlevleri içermelidir. Bunlardan bahsetmek gerekirse;
 - a. **Liste Formunda Sunucular:** Sunucuların listelendiği bir sayfa olmalıdır. Her sunucu için kısa bilgiler yer almalıdır.
 - b. **Özel Sunucu Seçme:** Her sunucunun yanında özel sunucu olarak seçmek için bir buton bulunmalıdır.

- c. **Sunucu Durumu:** Sunucunun aktif veya pasif durumunu gösteren bir göstergesi (örneğin, yeşil daire) yer almmalıdır. Örneğin sunucuda bir problem olduğu takdirde göstergesi kırmızı olabilir.
 - d. **Sunucu Detaylarına Erişim:** Herhangi bir sunucunun detayları görüntülenmek istenildiğinde, ilgili sunucunun üzerine tıklanabilmelidir. Bu işlem sonucunda sunucu detay ekranını istenmektedir.
4. Kullanıcıların yeni sunucuları uygulamaya eklemelerini sağlayan bir sayfa gerekmektedir. İstenen sayfa, sunucu bilgilerini girmek için gerekli form alanlarını içermelidir. Bunlar veritabanı gereksinimlerinden elde edilen veriler doğrultusunda: Hastane adı, Sunucu IP, sunucu adı, RAM, depolama tipi, depolama kapasitesi ve sunucu işletim sistemi olmalıdır. Girilen sunucu bilgilerini veritabanına kaydetmek için bir buton bulunmalıdır.
 5. Kullanıcıların belirli bir sunucunun detaylı bilgilerini görüntüleyebilecekleri sayfalar istenmektedir. Listelenmiş sunuculardan herhangi birinin üzerine tıklandığında sunucu detay sayfasına yönlendirilmeli. Sunucu detay sayfasının içinde hastane adı, sunucu adı, sunucunun IP adresi, RAM, depolama tipi, depolama kapasitesi, CPU kullanımı, RAM kullanımı, disk kullanımı, sunucu sıcaklığı, ortam sıcaklığı, güç kullanımı gibi bilgiler görüntülenebilir.
 6. Kullanıcıların önceden seçilmiş yani özel olarak takip ettikleri sunucularını ana sayfada olduğu gibi listeleyebilecekleri seçilmiş özel sunucuların listelenmesi beklenmektedir.
 7. Sistemden gelen bildirimleri görüntülemek için bildirim görüntüleme sayfası beklenmektedir. Kullanıcıların alınan bildirimleri görüntüleyebilmesi için uygulamanın ‘Footer’ kısmında bildirimler sayfasına erişebileceğİ bir alan bulunmalıdır. Bildirimler gerekiğinde silinebilmelidir.
 8. Kullanıcıların profil bilgilerini görüntüleyebilecekleri ve düzenleyebilecekleri bir sayfa beklenmektedir. İlgili sayfadan, kullanıcı bilgilerini güncelleme hizmeti beklenmektedir aynı zamanda kullanıcıların kullanıcı adı ve E-Posta ve profil resmi görüntülenebilecektir. Bildirimleri kapat ve çıkış yap gibi eylemler istenmektedir.

2.2.3 Yazılım Arayüz Gereksinimleri

2.2.3.1 SQL Veritabanı Yönetimi

Projede kullanılacak veritabanı PostgreSQL olarak belirlenmiştir. PostgreSQL açık kaynak kodlu ve dokümantasyon desteği ile öne çıkmaktadır. SQL dili olarak PostgreSQL'in desteklediği PL/SQL kullanılacaktır. PostgreSQL sunucusu Docker üzerinden ayağa kaldırılacak ve veritabanı arayüzü olarak DBeaver üzerinden etkileşime geçilecektir fakat daha sonra uygulamanın T-SQL veritabanı üzerinde Hibernate kullanılarak gerçekleştirilmesi sağlanmıştır. Projede oluşan zaman problemleri ve teknik sorunlar nedeniyle T-SQL'de

bulunan stored procedure, trigger ve bunların API entegrasyonu PostgreSQL üzerinde sağlanamamıştır.

2.2.3.2 Rest API

Uygulama ve veritabanı arasındaki bağlantı Rest API aracılığıyla sağlanacaktır. Rest API geliştirilirken Java programlama dili ve Java'ya entegre edilebilen Spring Boot Framework'ü kullanılacaktır.

Spring Boot kullanılmasının başlıca nedenleri:

1. Hız ve performans avantajı
2. Hibernate/JPA ile ORM kullanmaya olanak sağlama. (Gereksinimlerde belirtildiği üzere gelecekteki altyapı değişikliklerine karşı uyumlu olmak adına ORM kullanılması gerekmektedir.)
3. Entegre gelen Tomcat sunucusu ile ayrıca sunucu kurmadan uygulama ayağa kaldırılabilir durumdadır.

2.2.3.3 Kotlin

Mobil uygulamada kotlin programlama dili kullanılacaktır. Android uygulamalar geliştirmek için tercih edilen Kotlin; temiz, okunabilir ve esnek bir yapıya sahiptir. Android tabanlı uygulamalarda esnek, güvenilir geliştirme süreçlerine imkan tanıyacaktır. Ayrıca, Kotlin'in Java ile uyumlu olması, mevcut Java kütüphanelerinin kullanımını da kolaylaştırır.

Geniş Topluluk ve Destek: Kotlin'in hızla büyüyen bir geliştirici topluluğu vardır. Ayrıca, JetBrains gibi güçlü bir şirketin desteklediği bir dil olması, sürekli güncellemeler ve iyileştirmelerle desteklenmesini sağlar.

Daha Güvenli ve Daha Az Hata: Kotlin, null referanslarına karşı daha güvenli bir yapı sunar. "Nullable" ve "Non-Nullable" tür sistemleri ile geliştiricilerin hata yapma olasılığını azaltır ve daha sağlam bir kod yazımı sağlar.

Bu dilin projede kullanılmasının temel nedenleri:

1. Kotlin, güçlü bir statik tip güvenliği
2. Java ile uyumluluk gibi özellikleriyle daha az kod yazarak daha verimli uygulama programlanabilmektedir.
3. Kotlin'de xml kullanılarak programlama yapılacaktır. Bu da tasarıma sonradan müdahale etme imkanı tanıyacaktır.
4. API aracılığıyla veriler JSON formatında gelip işlenecektir. Bundan yola çıkarak Recycleview kullanılarak bildirimler listeleneciktir.

2.3.Veritabanı Gereksinimleri

Projedeki en önemli fonksiyonel gereksinim veritabanı gereksinimleri olacaktır. Uygulamada gerçekleşen işlemlerin veritabanındaki karşılıkları olacağından, veritabanının geliştirilmesi önem arz etmektedir. Kullanıcı kaydı, girişi; sunucu ekleme, silme, listeleme, filtreleme gibi özelliklerin tümü veritabanı üzerinden yapılacak sorgular ile gerçekleşeceğini, uygulamadaki veritabanı gereksinimleri şu şekildedir:

2.3.1 Sistem Gereksinimleri:

- Veritabanı Windows, Linux ve MacOS'te kullanılabilir.
- Veritabanı altyapısı: T-SQL
- Dil Desteği: SQL, PL/SQL

2.3.2 Güvenlik Gereksinimleri:

- Veritabanına erişim ile ilgili kısıtlar bulunmalıdır.
- Veritabanının çalışmış olduğu sunucu güvenlik duvarları ile korunmalıdır.

2.3.3 Veri Bütünlüğü Gereksinimleri:

Veritabanındaki verilerin bütünlüğünün sağlanması adına veritabanı yedeklemeleri kritik veriler anlık olarak başka bir sunucu üzerinde, ehemmiyeti az olan veriler ise kullanıcının tercihine göre günlük, haftalık ve aylık şekilde yedeklenebilir.

2.3.4 Performanslı Veri Erişimi:

Veritabanındaki verilerin uygulama ve API arasındaki aktarım süresi anlık veriler için önem arz ettiğinden veri akışları önem arz etmektedir. Hızlı veri akışı için kullanılması gereken yöntemler aşağıdaki gibidir:

Veritabanına gelen sorgular mümkün olduğunca saklı yordamlar (Stored Procedures) ile gelmelidir, bu şekilde Ad-Hoc query'e göre daha fazla performans elde edilir.

Kullanılan “Stored Procedure”lerin performansının belirli aralıklarla takip edilmesi için gerekli testler geliştirilmelidir, performans kaybı durumunda kullanıcıya bilgi sağlanmalıdır.

Veritabanı ilişkileri olabildiğince sade ve normalizasyonu gerektiği derecede yapılmalıdır.

2.3.5 Uyumluluk ve Entegrasyon:

Veritabanına mobil uygulama ile API üzerinden erişileceğinden geliştirilen veritabanı API entegrasyonuna uygun olmalıdır, bununla beraber JSON formatında veri aktarımına da olanak sağlamalıdır.

2.3.6 Veri Modelleri:

Uygulama işleyişi süresince bir çok ilişki içerecektir. Bu ilişkiler ve içermesi gereken özellikler aşağıda belirtilmiştir.

2.3.6.1 Kullanıcılar:

Kullanıcıların sisteme kayıt olurken girmiş olduğu kullanıcı adı (username) ve şifre bilgileri veritabanında tutulurken şifrelerek tutulmalıdır.

Verilerin şifrelenmesinde yaygın ve sürdürülebilir kriptografi uygulamaları kullanılmalıdır.

Kullanıcıların E-Posta adresleri, kayıt tarihleri, kullanıcı tipleri (yetki durumları) ve uygulamaya son giriş tarihleri de bir kullanıcı için veritabanına eklenecek bilgiler arasındadır.

Kullanıcılar bu bilgilerini veritabanı erişimine ihtiyaç duymadan uygulama arayüzü üzerinden değiştirebilmelidir.

Kullanıcılar birden fazla yetkiye sahip olabilmelidir.

Üye olmuş kullanıcılar, yönetici gerekliliği olmadan sunucu düzenlemelerini gerçekleştirebilmelidirler. Bu tip kullanıcıları ‘editör’ şeklinde isimlendirebiliriz.

Kullanıcıların sunucular üzerinde yaptıkları ekleme, silme, güncelleme değişiklikleri tutulmalıdır.

2.3.6.2 Sunucular

Sunucular veritabanına detaylı olarak eklenmelidir ve silinebilmelidir. Eklenecek özellikler arasında sunucunun adı, sunucunun IP adresi, sunucunun işletim sistemi, sunucunun ram miktarı, sunucuda kullanılan depolama aygıtinin tipi ve kapasitesi bulunmaktadır.

Ayrıca her sunucu içerisinde hangi hastaneye ait olduğu bilgisini de içermelidir. Bu şekilde belirli bir hastaneye göre filtreleme yapıldığında tüm sunuculara erişilebilmelidir.

2.3.6.3 Hastaneler

Hastaneler veritabanına adları ve bulundukları şehirler ile kayıt edilmelidir. Bu şekilde şehre göre filtreleme yapılabilmelidir.

2.3.6.4 Lokasyonlar

Hastanelerin bulunduğu lokasyonlar için şu anda sadece şehir isminin bulunması yeterli olacağından lokasyonlar veritabanına şehirler ve şehirlerin plaka numaraları ile eklenmelidir, plaka numarası aynı zamanda şehrın ayırt edici özelliği (ID) olacaktır.

2.3.6.5 Sağlık Verileri

Sunucuların takip edilecek sağlık durumlarına ait verilerin tamamı veritabanında bulunmalıdır.

Tutulması gereken veriler şu şekildedir: Sunucunun CPU, RAM, disk kullanımları, sunucu sıcaklığı, sunucunun bulunduğu ortamın sıcaklığı, sunucunun bulunduğu ortamda güç tüketimi.

Tüm bu verilere ek olarak sunucunun çalışır durumda olduğunun tespiti için de “heartbeat” verisi tutulmalıdır. Böylece verinin alındığı saatte sunucunun çalışıp çalışmadığı da bildirilmiş olur. Ancak gerektiğinde veritabanına kaydetmeden sürekli “heartbeat” gönderilip çalışma durumunun kontrol edilmesi de gerekmektedir.

Sunuculara ait sağlık verileri belirli zaman aralıklarıyla veritabanında saklanmalıdır.

2.3.6.6 Kesinti Durumları

Gerektiğinde kesinti verilerini ayrı olarak listeleme ve raporlama ihtiyacı duyulabileceğinden kesinti bilgileri kesintinin ne zaman olduğu, neden dolayı gerçekleştiği ve kesintinin anlık durumu (çözülüp - çözülmemiş) ile birlikte tutulmalıdır.

2.3.6.7 Bildirimler

Bildirimler veritabanında yer tutmamak adına lokal bir veritabanında saklanmalıdır.

Okunan bildirimler manuel olarak da silinebileceği gibi, 30 gün içinde otomatik olarak silinmelidir.

Kullanıcılar bildirimleri diledikleri zaman kapatabilmelidir.

2.3.7.Veritabanı ve İşleyişi

Uygulamanın veritabanı 2 farklı uygulama üzerinde geliştirilecektir. Birincisi sunucu üzerinden çalışmakta olan T-SQL ve lokal veritabanı olarak SQLite kullanılacaktır. Bunun sebebi gereksinimlerde istenildiği üzere T-SQL veritabanında bildirimlerin yer kaplamaması ve sunuculara “ping” gönderme işlemi gerçekleştirilirken lokal veritabanındaki sunucu bilgilerinden faydalanaılması içindir.

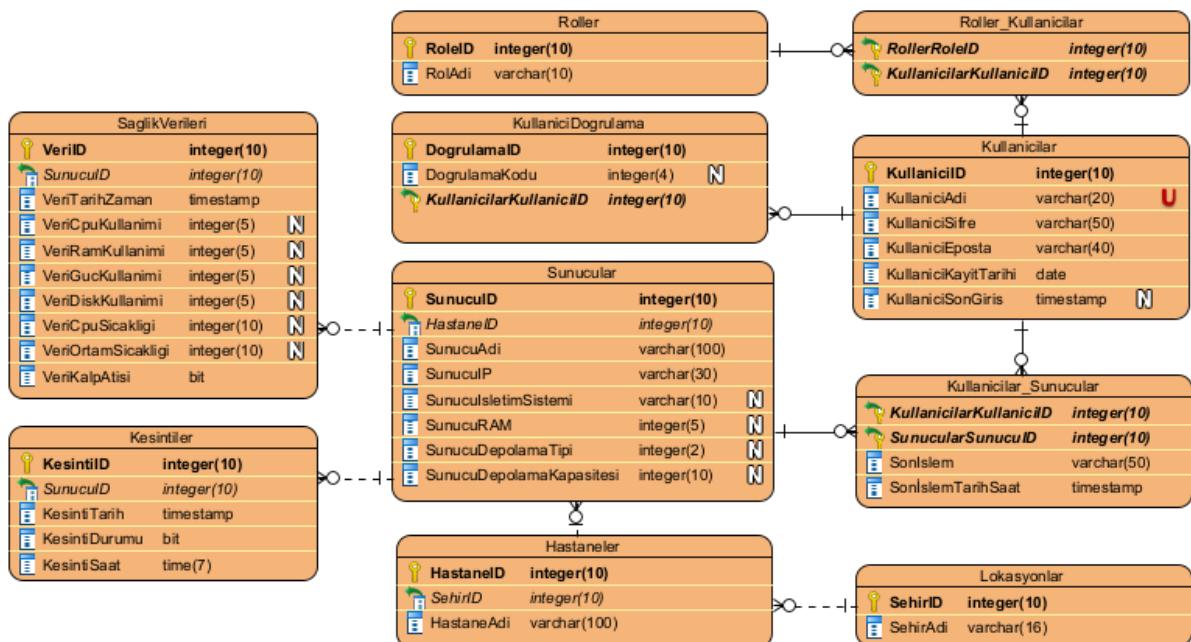
Uygulamanın geliştirilme sürecinde olmasa dahi, canlı uygulama kullanıma geçtiğinde verilerin güvenliği önem arz ettiğinden, veritabanına gelen istekler UFW (Uncomplicated

Firewall) kullanılarak denetlenecek ve işlenecektir. Bu sayede veritabanı bir güvenlik duvarı (firewall) ile korunmuş olacaktır.

Veritabanında bulunan veriler, verilerin bütünlüğünün korunması amacıyla yedekleme işlemine tabi tutulacaktır. Yedekleme işlemleri T-SQL içerisinde bulunan fonksiyonlarla ve Linux sunucusu üzerinde bulunan “cron” görev zamanlayıcısı ile gerçekleştirilecektir.

3. Veri Yapısı

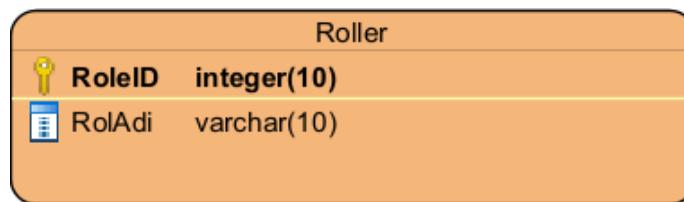
3.1. ER Diyagram tasarıımı



Şekil x: ER Diyagram

3..1.1 Veritabanı Tabloları ve Açıklamaları:

Roller:



Şekil 3: Roller ER Diyagram

(PK) RoleID (Integer): Kullanıcıların sahip olabileceği roller için benzersiz olarak belirlenen sayı.

RoleAdı (Varchar): Rolün adı, örneğin: Yönetici, Editör, Kullanıcı.

Roller-Kullanıcılar:



Şekil 4: Roller-Kullanıcılar ER Diyagram

Roller ve Kullanıcılar tablosu arasındaki many-to-many ilişkiyi yönetmeyi sağlayan tablodur.

İçerisinde RoleID ve KullanıcıID alanlarını barındırır. Bu iki alan beraber benzersiz bir anahtar oluştururlar, böylece her bir kullanıcı bir role birden çok kez sahip olarak tutulmaz. Ayrıca her bir kullanıcı da bu sayede birden fazla role sahip olabilir.

Kullanıcı Doğrulama:

Kullanıcı doğrulamaya ait bilgilerin tutulduğu tablodur.



Şekil 5: KullanıcıDoğrulama ER Diyagram

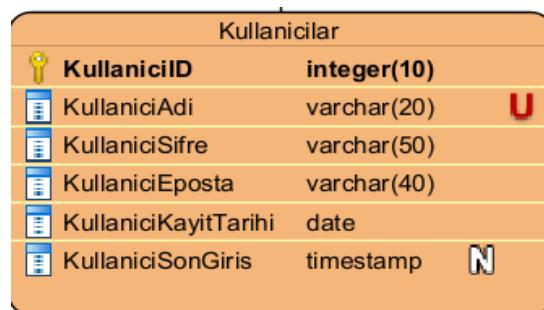
DoğrulamaID (PK Integer): kullanıcı doğrulaması için oluşturulan kodların id si

KullanıcıID (FK Integer): Her bir kullanıcı için benzersiz olarak belirlenen sayı.

Doğrulama Kodu(Integer): Kullanıcıların kayıt doğrulama işlemleri için oluşturulan kod.

Kullanıcılar:

Kullanıcılara ait bilgilerin tutulduğu tablodur.



Sekil 5: Kullanicilar ER Diyagram

KullanıcıID (PK Integer): Her bir kullanıcı için benzersiz olarak belirlenen sayı.

KullanıcıAdı (Varchar): Kullanıcının sisteme giriş yapabileceği benzersiz ad alanı.

KullanıcıSifre (Varchar): Kullanıcının belirleyeceği şifre.

KullanıcıEposta (Varchar): Kullanıcının E-Posta adresi.

KullanıcıKayitTarihi (Date): Kullanıcının kayıt olduğu tarih.

KullaniciSonGiris (timestamp): Kullanıcının sisteme son giriş yaptığı tarih ve saat.

Kullanıcılar-Sunucular:

Kullanıcılar_Sunucular		
Kullanıcılar	KullaniciID	integer(10)
Sunucular	SunucuID	integer(10)
	SonIslem	varchar(50)
	SonIslemTarihSaat	timestamp

Şekil 6: Kullanıcılar-Sunucular ER Diyagram

Bir kullanıcı birden fazla sunucu üzerinde işlem yapabileceği, bir sunucu da birden çok kişi tarafından yönetilebileceği için many-to-many ilişkisi sağlamak için kullanılan tablodur. İçerisinde KullaniciID ve SunucuID alanları dışında son yapılan işlemin açıklamasını ve yapıldığı tarih saati tutar.

Sunucular:

Sunucu bilgilerinin detaylı bir şekilde bulunduğu tablodur.

Sunucular		
SunucuID	integer(10)	
HastaneID	integer(10)	
SunucuAdı	varchar(100)	
SunucuIP	varchar(30)	
SunucusletimSistemi	varchar(10)	N
SunucuRAM	integer(5)	N
SunucuDepolamaTipi	integer(2)	N
SunucuDepolamaKapasitesi	integer(10)	N

Şekil 7: Sunucular ER Diyagram

SunucuID (PK Integer): Her bir sunucu için benzersiz olan sayı.

HastaneID (FK Integer): Bir sunucunun hangi hastaneye ait olduğunu takip edebilmek için tutulan yabancı anahtardır.

SunucuAdı (Varchar): Sunucunun adı, örneğin: “Tekirdağ Şehir Hastanesi - Randevu Kayıt Servisi Sunucu 1”

SunucuIP (Varchar): Sunucunun IP adresi.

SunucuIsletimSistemi (Varchar): Sunucunun sahip olduğu işletim sistemi bilgisi.

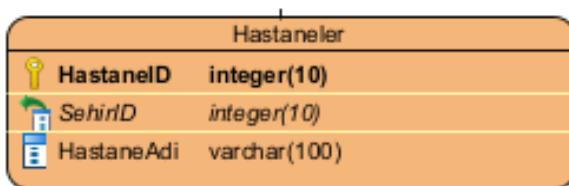
SunucuRAM (Integer): Sunucunun sahip olduğu bellek miktarı.

SunucuDepolamaTipi (Integer): Sunucuda hangi tip depolama türü (HDD veya SSD) olduğunun bilgisi.

SunucuDepolamaKapasitesi (Integer): Sunucudaki depolamanın kapasitesi.

Hastaneler:

Hastane bilgilerini içeren tablodur.



Şekil 8: Hastaneler ER Diyagram

HastaneID (PK Integer): Her bir hastane için benzersiz olan sayı.

SehirID (FK Integer): Bir hastanenin hangi şehirde(isterlerde sadece şehir tutulması istendiğinden ismi LokasyonID yerine SehirID şeklinde) bulunduğu bilgisi.

HastaneAdi (Varchar): Hastane adı.

Lokasyonlar:

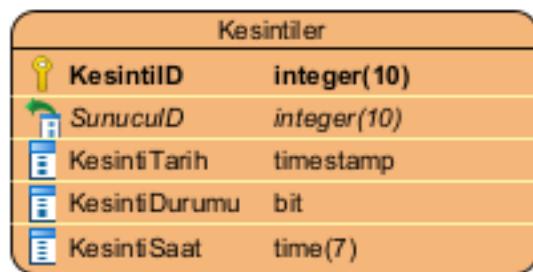
Isterlerde belirtildiği üzere şehir adlarının ve ID değeri olarak şehrin plakasının tutulduğu tablodur.



Şekil 9: Lokasyonlar ER Diyagram

Kesintiler:

Sunucuda kesinti olduğunda oluşan sorunun loglandığı tablodur.



Şekil 10: Kesintiler ER Diyagram

KesintiID (PK Integer): Her bir kesinti durumu için benzersiz olan sayı. Bu değerin tutulma sebebi daha sonrasında sorunun takibinin kolay olmasına.

SunucuID(FK Integer): Oluşan kesintinin hangi sunucuda meydana geldiğini ve sunucunun o anki verilerine ulaşmak için bağlantı kurulan yabancı anahtardır.

KesintiTarih (Timestamp): Kesintinin gerçekleştiği tarihi tutar.

KesintiSaat (Time): Kesintinin gerçekleştiği saat dakika saniyeyi tutar.

KesintiDurumu (Bit): Kesintinin anlık durumunu bildiren bit değeridir. Aktif (1), Pasif(0) şeklinde 2 değer alır.

Sağlık Verileri:

Sunucuya ait sağlık verilerinin tutulduğu tablodur.

SaglikVerileri		
VeriID	integer(10)	
SunucuID	integer(10)	
VeriTarihZaman	timestamp	
VeriCpuKullanimi	integer(5)	N
VeriRamKullanimi	integer(5)	N
VeriGucKullanimi	integer(5)	N
VeriDiskKullanimi	integer(5)	N
VeriCpuSicakligi	integer(10)	N
VeriOrtamSicakligi	integer(10)	N
VeriKalpAtisi	bit	

Şekil 11: Sağlık Verileri ER Diyagram

VeriID (PK Integer): Her bir veri için benzersiz olan sayı.

SunucuID (FK Integer): Verinin hangi sunucuya ait olduğunu bilgisi için bağlantı kurulan yabancı anahtardır.

VeriTarihZaman (Timestamp): Verinin alındığı tarih ve saat.

VeriCpuKullanimi (Integer): Verinin alındığı zamandaki yüzde olarak CPU kullanımı.

VeriRamKullanimi (Integer): Verinin alındığı zamandaki yüzde olarak RAM kullanımı.

VeriGucKullanimi (Integer): Verinin alındığı zamandaki güç tüketimi.

VeriDiskKullanimi (Integer): Verinin alındığı zamandaki yüzde olarak disk kullanımı.

VeriCpuSicakligi (Integer): Verinin alındığı zamandaki CPU sıcaklığı.

VeriOrtamSicakligi (Integer): Verinin alındığı zamandaki ortamın sıcaklığı.

VeriKalpAtisi (Bit): Verinin alındığı zamandaki anlık olarak çalışma durumu. (İsterlerde belirtildiği üzere belirli zaman aralıklarla alınan bu değer tabloda tutulurken uygulama süresince anlık olarak takip verileri lokal bir veritabanında saklanmaktadır.)

4. Sorgular

Uygulamanın geliştirilmesi sırasında Stored Procedure'ler ile 2 adet karmaşık sorgu oluşturulmuştur. Bunlar 5.1.Stored Procedures kısmında kod gösterimi yapılan **sp_ReportInterruptionsByMonthly** (1) ve **sp_ReportInterruptionsByDaily** (2) saklı yordamlarıdır. Bu saklı yordamlar içlerinde GROUP BY ve ORDER BY kullanımları ile kullanıcılara sunucunun bilgileri hakkında detaylı bir tablo sunmaktadır. 1 numaralı yordam sunucuların 1 yıl içerisinde hangi aylarda ne kadar kesinti yaşadıklarını, 2 numaralı yordam ise sunucuların bir hafta içinde gün bazında ne kadar kesinti yaşadıklarını kullanıcıya sunmaktadır. Bu sorgular sonucu dönen tabloların satır ve sütun bilgileri önce Restful API'a sonrasında ise Kotlin tarafına iletilerek işlenmiş ve Sunucu Detay Sayfasında kullanıcılara sunulmuştur. Ayrıca uygulamaya entegre olmasa dahi, tüm sunuculardan veri gelen bir sistem simülle edilmek için **sp_AddHealthdataRandomly** saklı yordamı oluşturulmuştur. Bu yordam da içerisinde döngü ve TSQL fonksiyonları barındırarak otomatik olarak kullanıcının istediği değerde veri (Healthdata) üretmektedir. Bu yordam sayesinde uygulamanın gerçek verileri gönderen bir sisteme entegre olduğunda çalışabileceği simülle edilmiştir.

Bu sorgular Rest API tarafından aşağıdaki şekilde çağrırlarak kullanılmıştır.

```
@Query(value = "EXEC sp_ReportInterruptionsByMonthly :serverid", nativeQuery = true)
List<Object[]> getInterruptsReportMonthly(@Param("serverid") Integer serverid);

@Query(value = "EXEC sp_ReportInterruptionsByDaily :serverid", nativeQuery = true)
List<Object[]> getInterruptsReportDaily(@Param("serverid") Integer serverid);
```

5. VTYS Araç Kullanımı

5.1. Stored Procedures

Veritabanının geliştirilme aşamasında veri güvenliği, bütünlüğü ve kullanılabilirlik olması açısından saklı yordamlar ve tetikleyicilerin kullanımı sağlanmıştır. Bu saklı yordamlar aşağıdaki şekilde tanımlanmıştır:

1. sp_GetHealthdataByServerid

```
CREATE PROC [dbo].[sp_GetHealthdataByServerid]
@Serverid INT
AS
BEGIN
    SELECT *
    FROM
        Healthdata
    WHERE
        serverid = @Serverid
    ORDER BY CONVERT(datetime,datadatetime) DESC
END;
```

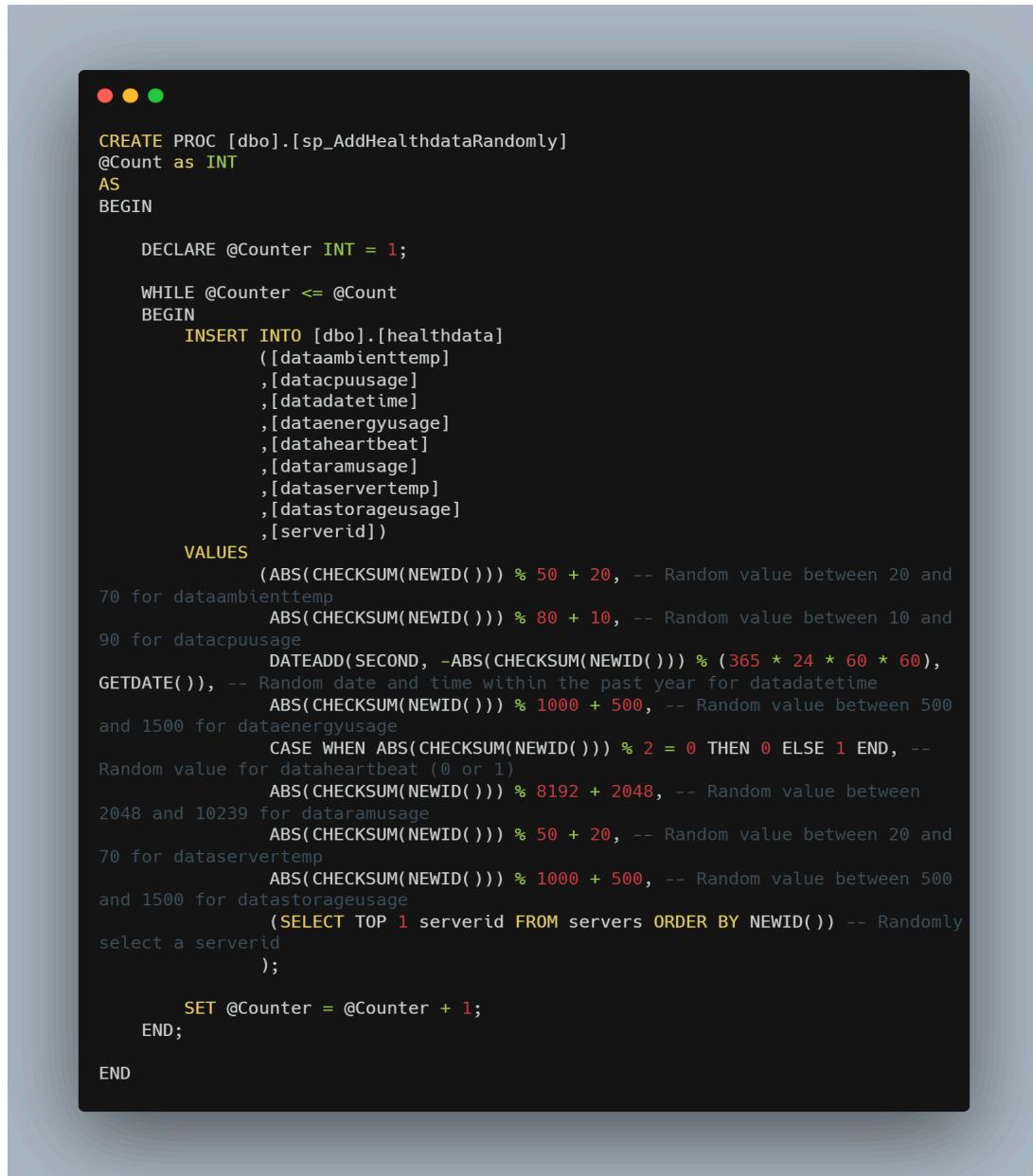
Bu saklı yordam, Healthdata tablosundan belirli bir sunucunun sağlık verilerini almak için kullanılır. @Serverid parametresi, istenen sunucunun kimliğini belirtmek için kullanılır ve sonuçlar tarih/saat değerine göre en yeni olandan en eskiye doğru sıralanır. Bu saklı prosedürü çağrırmak için @Serverid parametresiyle bir tamsayı değeri sağlanmalıdır.

2. sp_GetInterruptsByServerId

```
CREATE PROCEDURE [dbo].[sp_GetInterruptsByServerId]
    @serverid INT
AS
BEGIN
    SELECT
        interruptid,
        interruptdate,
        interruptstatus,
        interrupttime,
        serverid
    FROM
        interrupts
    WHERE
        serverid = @serverid
    ORDER BY CONVERT(date,interruptdate) DESC, CONVERT(time,interrupttime) DESC
END;
```

Bu saklı yordam, belirli bir sunucunun interrupts tablosundaki kesinti verilerini alır. @serverid parametresi, istenen sunucunun kimliğini belirtir. Sonuçlar, kesinti tarihine göre en yeni olanın en üstte olacak şekilde sıralanır. Eğer aynı tarih içinde birden fazla kayıt varsa, saat değerine göre de en yeni olan en üstte olacak şekilde sıralanır. Bu şekilde, en son kesintiler ilk olarak listelenir.

3. sp_AddHealthdataRandomly



```
CREATE PROC [dbo].[sp_AddHealthdataRandomly]
@Count as INT
AS
BEGIN

    DECLARE @Counter INT = 1;

    WHILE @Counter <= @Count
    BEGIN
        INSERT INTO [dbo].[healthdata]
        ([dataambienttemp]
        ,[datacpuusage]
        ,[datadatetime]
        ,[dataenergyusage]
        ,[dataheartbeat]
        ,[dataramusage]
        ,[dataservertemp]
        ,[datastorageusage]
        ,[serverid])
        VALUES
            (ABS(CHECKSUM(NEWID())) % 50 + 20, -- Random value between 20 and
            70 for dataambienttemp
            ABS(CHECKSUM(NEWID())) % 80 + 10, -- Random value between 10 and
            90 for datacpuusage
            DATEADD(SECOND, -ABS(CHECKSUM(NEWID())) % (365 * 24 * 60 * 60),
            GETDATE()), -- Random date and time within the past year for datadatetime
            ABS(CHECKSUM(NEWID())) % 1000 + 500, -- Random value between 500
            and 1500 for dataenergyusage
            CASE WHEN ABS(CHECKSUM(NEWID())) % 2 = 0 THEN 0 ELSE 1 END, -- Random
            value for dataheartbeat (0 or 1)
            ABS(CHECKSUM(NEWID())) % 8192 + 2048, -- Random value between
            2048 and 10239 for dataramusage
            ABS(CHECKSUM(NEWID())) % 50 + 20, -- Random value between 20 and
            70 for dataservertemp
            ABS(CHECKSUM(NEWID())) % 1000 + 500, -- Random value between 500
            and 1500 for datastorageusage
            (SELECT TOP 1 serverid FROM servers ORDER BY NEWID()) -- Randomly
            select a serverid
        );

        SET @Counter = @Counter + 1;
    END;
END;
```

Bu saklı yordam, bir tane @Count adında bir tamsayı parametresi alır. Bu parametre, eklenmesi istenen rastgele sağlık verisi sayısını belirtir. Veri oluşturma işlemi, rastgele değerler kullanılarak yapılır ve gerçek sağlık verileriyle doğrudan ilişkili değildir.

4. sp_GetInterruptsByServerIdAndDateRangeInOneWeek

```
CREATE PROCEDURE sp_GetInterruptsByServerIdAndDateRangeInOneWeek
    @serverId INT
AS
BEGIN
    SELECT *
    FROM interrupts i
    WHERE serverid = @serverId
        AND CONVERT(date, i.interruptdate) >= DATEADD(DAY, -7, GETDATE())
    ORDER BY CONVERT(date, i.interruptdate) DESC;
END;
```

Bu saklı yordan, belirtilen sunucunun kesinti verilerini son bir haftalık dönemde almak için kullanılabilir. @serverId parametresi belirli bir sunucunun kimliğini belirtir ve bu sunucunun kesinti verileri tarih sırasına göre listelenir.

5. sp_GetInterruptsByServerIdAndDateRangeInOneMonth

```
CREATE PROCEDURE sp_GetInterruptsByServerIdAndDateRangeInOneMonth
    @serverId INT
AS
BEGIN
    SELECT *
    FROM interrupts i
    WHERE serverid = @serverId
        AND CONVERT(date, i.interruptdate) >= DATEADD(MONTH, -1, GETDATE())
    ORDER BY CONVERT(date, i.interruptdate) DESC;
END;
```

Bu saklı yordam, belirtilen sunucunun kesinti verilerini son bir ay içinde almak için kullanılabilir. @serverId parametresi belirli bir sunucunun kimliğini belirtir ve bu sunucunun kesinti verileri tarih sırasına göre listelenir.

6. sp_GetInterruptsByServerIdAndDateRangeInOneYear

```
CREATE PROCEDURE sp_GetInterruptsByServerIdAndDateRangeInOneYear
    @serverId INT
AS
BEGIN
    SELECT *
    FROM interrupts i
    WHERE serverid = @serverId
        AND CONVERT(date, i.interruptdate) >= DATEADD(YEAR, -1, GETDATE())
    ORDER BY CONVERT(date, i.interruptdate) DESC;
END;
```

Bu saklı yordam, belirtilen sunucunun kesinti verilerini son bir yıl içinde almak için kullanılabilir. @serverId parametresi belirli bir sunucunun kimliğini belirtir ve bu sunucunun kesinti verileri tarih sırasına göre listelenir.

7. sp_GetHealthDataByServerIdAndDateRangeInOneWeek

```
CREATE PROCEDURE sp_GetHealthDataByServerIdAndDateRangeInOneWeek
    @serverId INT
AS
BEGIN
    SELECT *
    FROM healthdata h
    WHERE serverid = @serverId
        AND CONVERT(date, h.datadatetime) >= DATEADD(DAY, -7, GETDATE())
    ORDER BY CONVERT(date, h.datadatetime) DESC;
END;
```

Bu saklı yordam, belirtilen sunucunun sağlık verilerini son bir haftalık dönem içinde almak için kullanılabilir. @serverId parametresi belirli bir sunucunun kimliğini belirtir ve bu sunucunun sağlık verileri tarih sırasına göre listelenir.

8. sp_GetHealthDataByServerIdAndDateRangeInOneMonth

```
CREATE PROCEDURE sp_GetHealthDataByServerIdAndDateRangeInOneMonth
    @serverId INT
AS
BEGIN
    SELECT *
    FROM healthdata h
    WHERE serverid = @serverId
        AND CONVERT(date, h.datadatetime) >= DATEADD(MONTH, -1, GETDATE())
    ORDER BY CONVERT(date, h.datadatetime) DESC;
END;
```

Bu saklı yordam, belirtilen sunucunun sağlık verilerini son bir aylık dönemde almak için kullanılabilir. @serverId parametresi belirli bir sunucunun kimliğini belirtir ve bu sunucunun sağlık verileri tarih sırasına göre listelenir.

9. sp_GetHealthDataByServerIdAndDateRangeInOneYear

```
CREATE PROCEDURE sp_GetHealthDataByServerIdAndDateRangeInOneYear
    @serverId INT
AS
BEGIN
    SELECT *
    FROM healthdata h
    WHERE serverid = @serverId
        AND CONVERT(date, h.datadatetime) >= DATEADD(YEAR, -1, GETDATE())
    ORDER BY CONVERT(date, h.datadatetime) DESC;
END;
```

Bu saklı yordam, belirtilen sunucunun sağlık verilerini son bir yıllık dönemde almak için kullanılabilir. @serverId parametresi belirli bir sunucunun kimliğini belirtir ve bu sunucunun sağlık verileri tarih sırasına göre listelenir.

10. sp_ReportInterruptsByMonthly



```
CREATE PROCEDURE [dbo].[sp_ReportInterruptsByMonthly]
@ServerId AS INT
AS
BEGIN
    SELECT DATENAME(MONTH, CONVERT(date, i.interruptdate)) AS MONTH_,
           COUNT(*) AS INTERRUPTCOUNT
    FROM interrupts i
    WHERE serverid = @ServerId
    GROUP BY DATENAME(MONTH, CONVERT(date, i.interruptdate)), DATEPART(MONTH,
    CONVERT(date, i.interruptdate))
END
```

Bu saklı yordam, belirtilen sunucunun kesinti verilerini aylık bazda raporlamak için kullanılabilir. Her bir ay için kesinti sayılarını ve ay isimlerini @ServerId parametresine göre gruplar ve raporlar.

11. sp_ReportInterruptsByDaily

```
CREATE PROCEDURE sp_ReportInterruptsByDaily
@ServerId AS INT
AS
BEGIN
SELECT DATENAME(dw, CONVERT(date, i.interruptdate)) AS DW,
       COUNT(*) AS INTERRUPTCOUNT
FROM interrupts i
WHERE serverid = @ServerId
GROUP BY DATENAME(dw, CONVERT(date, i.interruptdate))
ORDER BY
    CASE DATENAME(dw, CONVERT(date, i.interruptdate))
        WHEN 'Monday' THEN 1
        WHEN 'Tuesday' THEN 2
        WHEN 'Wednesday' THEN 3
        WHEN 'Thursday' THEN 4
        WHEN 'Friday' THEN 5
        WHEN 'Saturday' THEN 6
        WHEN 'Sunday' THEN 7
    END;
END
```

Bu saklı yordam, belirtilen sunucunun kesinti verilerini günlük bazda raporlamak için kullanılır. Her bir gün için kesinti sayısını hesaplar ve haftanın günlerine göre bu sayıları sıralar.

5.2.Triggers

1. **trg_InsertInterruptHeartbeatFalse**

```
CREATE TRIGGER [dbo].[trg_InsertInterruptHeartbeatFalse]
ON [dbo].[healthdata]
AFTER INSERT
AS
BEGIN
    IF (SELECT COUNT(*) FROM inserted WHERE dataheartbeat = 0) > 0
    BEGIN
        INSERT INTO interrupts (
            interruptdate,
            interruptstatus,
            interrupttime,
            serverid
        )
        SELECT
            CAST(datadatetime AS date) AS interruptdate,
            1 AS interruptstatus,
            CAST(datadatetime AS time) AS interrupttime,
            serverid
        FROM inserted
        WHERE dataheartbeat = 0;
    END
END;
```

Bu tetikleyici, "healthdata" tablosuna bir veri eklendiğinde çalışır. Eğer eklenen verilerdeki dataheartbeat sütunu "False" ise, bu tetikleyici interrupts tablosuna yeni bir kesinti (interrupt) kaydı ekler. Bu işlem, belirli bir durum gerçekleştiğinde otomatik olarak bir eylem gerçekleştirmek üzere tasarlanmıştır. Burada, dataheartbeat sütununda 0 olan verilerin eklenmesi durumunda bir "interrupt" kaydı oluşturulur.

5.3 Hata Ayıklama (Error handling)

Uygulamanın hata yönetiminin entegre edilmesi kısmında kullanıcı doğrulama fonksiyonu sağlanmıştır. Hata ayıklama işlemlerinin SQL tarafından yapılması işlemleri, bu mekanizmaların Rest API ve Kotlin uygulaması üzerindeki entegrasyon çalışmaları, kullanılacak teknolojilerde belirtilen * durumundan kaynaklı olarak teslim süresine dek tam olarak sağlanamamıştır. Entegrasyonu sağlanan **sp_VerifyUser** saklı yordamı aşağıdaki gibidir. Bu saklı yordam Rest API sunucusuna kullanıcının kaydı esnasında oluşturduğu kullanıcı adı ve sunucu içerisinde oluşturulan doğrulama kodunu parametre olarak alır. İçerisinde bulunan kontrol mekanizmaları öncelikle kullanıcının varlığını kullanıcılar (users) tablosundan kontrol eder, daha sonrasında KullaniciDogrulama (user_verification_codes)

sayfasında kullanıcı için oluşturulmuş olan doğrulama kodu ile parametre olarak gelen doğrulama kodunun aynı olup olmadığını kontrol eder. Kontroller sonucunda integer olarak değer döndürür, bu değer Rest API kısmında işlenerek hata yönetimi tamamlanmış olur.

```
CREATE PROCEDURE [dbo].[sp_VerifyUser]
    @Username NVARCHAR(255),
    @VerificationCode NVARCHAR(255),
    @VerificationResult INT OUTPUT
AS
BEGIN
    DECLARE @UserCount INT;
    SET @VerificationResult = 0;

    SELECT @UserCount = COUNT(*)
    FROM Users
    WHERE Username = @Username;

    IF @UserCount = 1
    BEGIN
        IF EXISTS (
            SELECT 1
            FROM user_verification_codes
            WHERE username = @Username
                AND verification_code = @VerificationCode
        )
        BEGIN
            UPDATE Users
            SET UserAuthority = '1'
            WHERE Username = @Username;

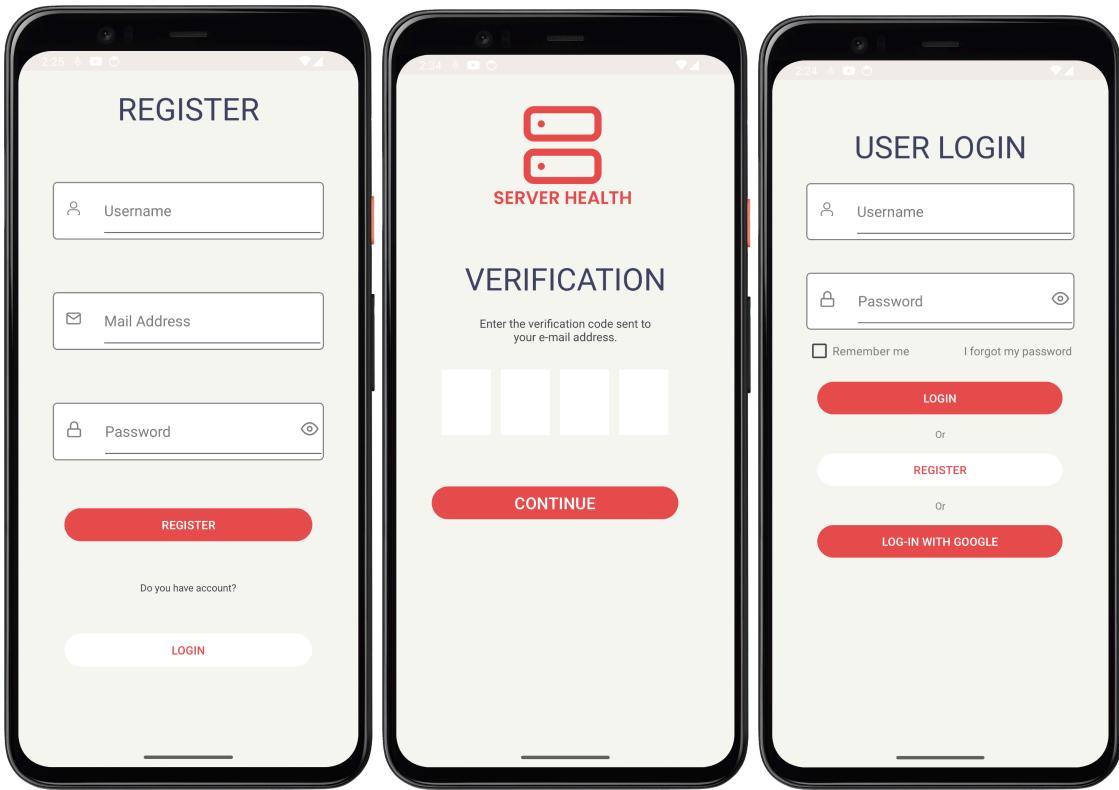
            SET @VerificationResult = 1;
            DELETE FROM user_verification_codes WHERE username = @Username
        END
        ELSE
        BEGIN
            SET @VerificationResult = 2;
        END
    END
    ELSE
    BEGIN
        SET @VerificationResult = 3;
    END
END;
```

```
private int verifyUserInDatabase(String username, String verificationCode) {
    try (Connection connection =
DriverManager.getConnection("jdbc:sqlserver://localhost:1433;encrypt=true;trustServerCertificate=true;databaseName=HealthMonitorApp;user
=sa;password=Sunucu123")) {
        try (CallableStatement statement = connection.prepareCall("{call sp_VerifyUser(?, ?, ?)}")) {
            statement.setString(1, username);
            statement.setString(2, verificationCode);
            statement.registerOutParameter(3, Types.INTEGER);

            statement.execute();
            return statement.getInt(3);
        }
    } catch (SQLException e) {
        e.printStackTrace();
        return VerificationStatus.UNKNOWN_ERROR;
    } catch (Exception e) {
        e.printStackTrace();
        return VerificationStatus.UNKNOWN_ERROR;
    }
}

public class VerificationStatus {
    public static final int SUCCESS = 1;
    public static final int VERIFICATION_FAILED = 2;
    public static final int USER_NOT_FOUND = 3;
    public static final int UNKNOWN_ERROR = 0;
}
```

6. Kullanıcı Arayüzü (User Interface)



Giriş Sayfası:

Uygulamamızın giriş sayfası, kullanıcıların hesaplarına erişebilecekleri bir başlangıç noktasıdır. Bu sayfada, kullanıcılar iki farklı yöntemle giriş yapabilirler:

Google Hesabı ile Giriş: Kullanıcılar, Google hesaplarıyla kolayca giriş yapabilirler.

Kullanıcı Adı ve Şifre ile Giriş: Kayıtlı kullanıcılar, önceden belirledikleri kullanıcı adı ve şifreleriyle giriş yapabilirler. Eğer henüz bir hesapları yoksa, kayıt ol butonuna tıklayarak kayıt sayfasına erişebilirler.

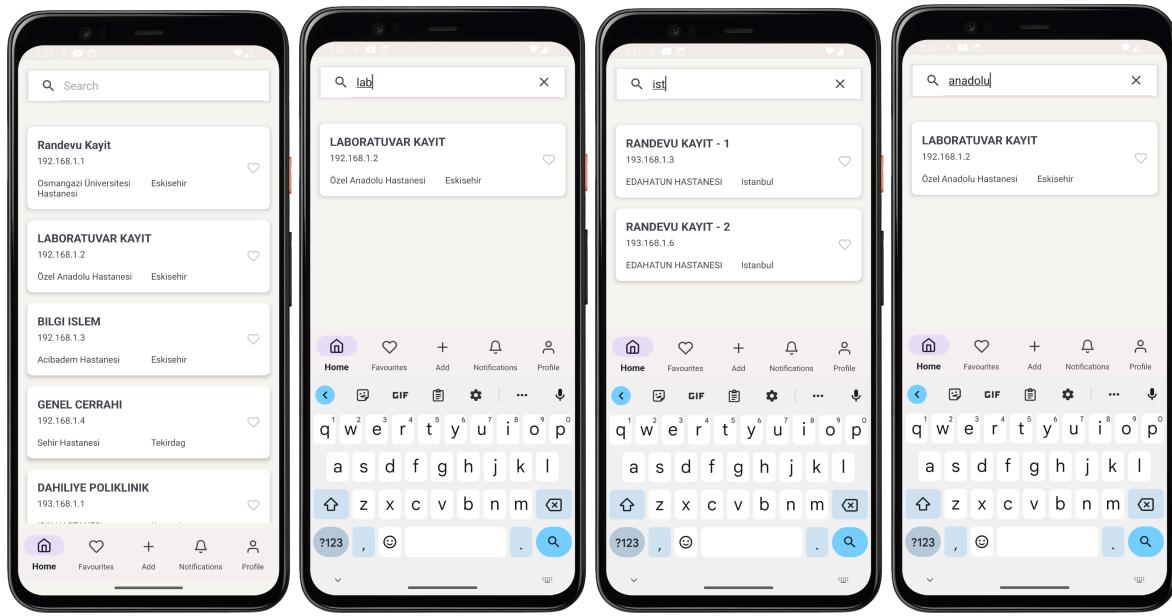
Kayıt Sayfası:

Kayıt ol sayfasında, kullanıcılar temel bilgilerini girebilirler. Burada kullanıcı adı, e-posta adresi ve şifrelerini belirleyerek, kullanıcı doğrulaması yaptıktan sonra kayıt işlemini tamamlayabilirler.

Doğrulama Aşaması:

Kayıt olurken belirtilen e-posta adresine doğrulama amaçlı bir mail gönderilir. Bu mailde kullanıcıya, kayıt işlemini tamamlamak için gereken dört haneli bir doğrulama kodu ilettilir. Bu kod, uygulamadaki doğrulama kodu alanına girilerek kayıt işlemi tamamlanır.

Doğrulama adımlını tamamlayan kullanıcılar, kullanıcı adı ve şifreleriyle giriş yaparak ana sayfaya erişebilirler. Artık hesaplarına erişmiş olup, uygulamanın tüm fonksiyonlarını kullanabilirler.



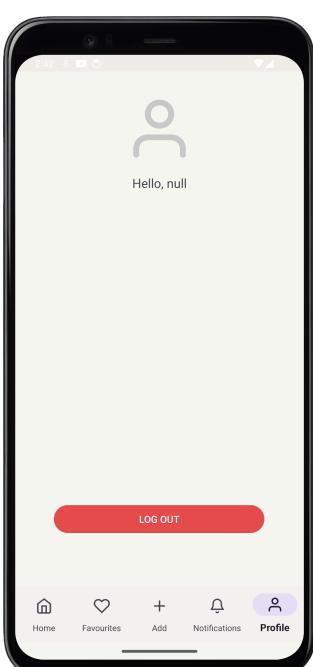
Anasayfa:

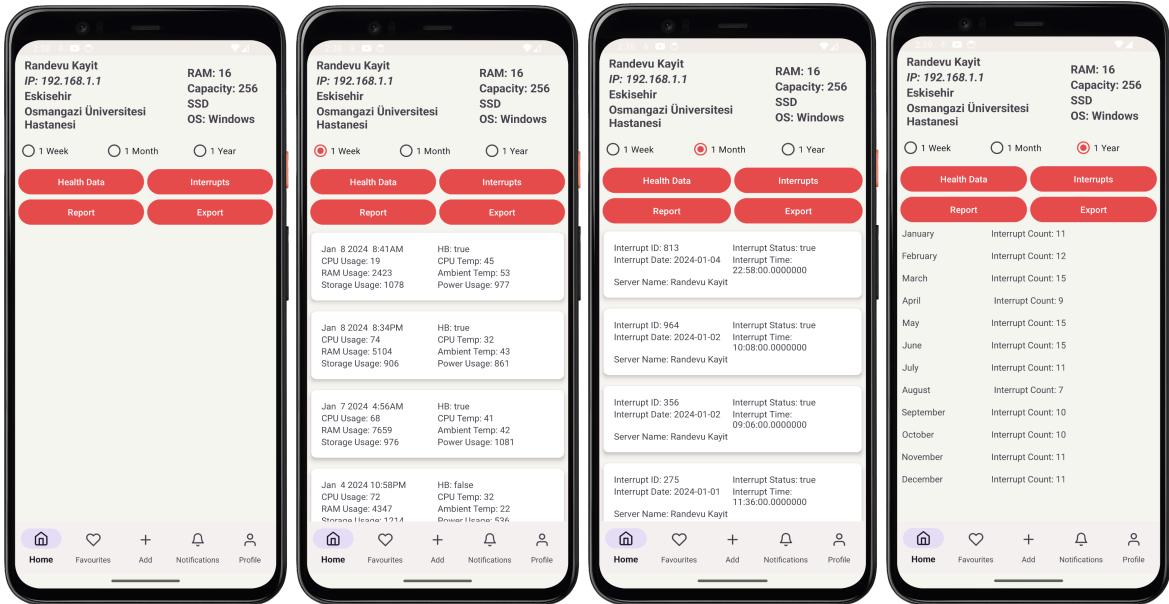
Uygulamamız, alta yer alan bir menü çubuğu ile sayfalar arası gezinmeyi sağlar. Bu menü çubuğunun sol alt köşesinde varsayılan olarak "Anasayfa" seçeneği bulunmaktadır.

Anasayfada, sunucular listelenir ve her sunucunun adı ve altında ilgili temel bilgiler, yani hastane adı ve şehir adı görüntülenir. Uygulamanın anasayfasının üst kısmında bulunan arama bölümü, kullanıcıların hızlı bir şekilde ilgili sunuculara erişebilmelerini sağlar. Bu arama bölümü sayesinde, kullanıcılar arama kutusuna ilgili şehir adını, hastane adını veya sunucu adını yazarak istedikleri sunucuya hızlıca erişebilirler. Örneğin, bir şehir adı veya hastane adı yazarak o konumda bulunan veya o hastaneye ait sunucuları kolayca bulabilirler. Aynı şekilde, sunucu adı yazarak da doğrudan ilgili sunucuya erişebilirler.

Profil:

Uygulamanın alt barında bulunan sağ köşede yer alan "Profil" menüsüne tıklandığında, kullanıcı uygulama üzerindeki hesabından çıkış yapabilir. Bu işlem, kullanıcıyı oturumu kapatma işlemi gerçekleştirerek giriş sayfasına yönlendirir. Kullanıcılar, profil menüsünden çıkış yaparak daha sonra tekrar giriş yapabilirler.

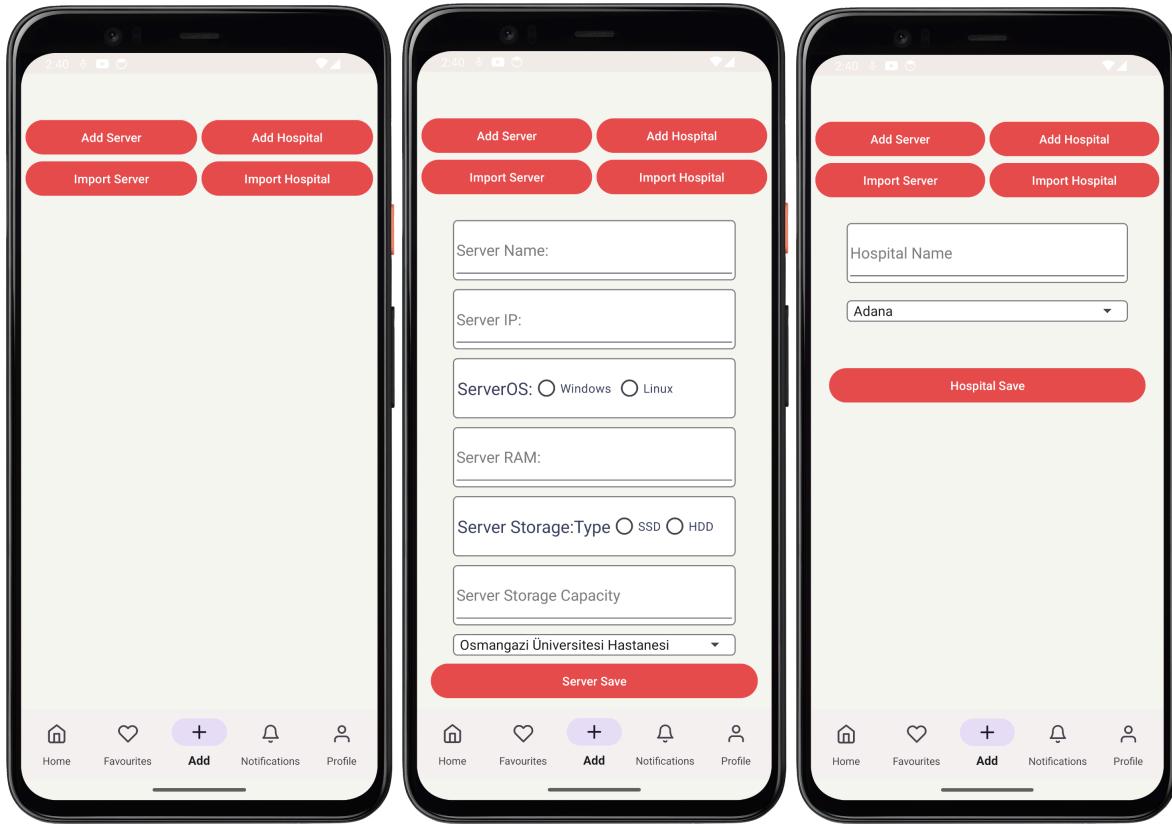




Sunucu Detayları:

Listelenen sunuculara tıklanabilir ve tıklandığında sunucuya özgü detay sayfası açılır. Bu detay sayfasında sunucunun adı, IP adresi, işletim sistemi, RAM, depolama tipi ve kapasitesi gibi detaylar sunulur. Ayrıca, haftalık, aylık ve yıllık seçenekler sunan butonlar bulunur.

Sağlık Verileri ve Interrupt Bilgileri: Haftalık, aylık veya yıllık seçeneklerden biri seçildiğinde, "Sağlık Verileri" veya "Interrupt" butonlarına tıklanabilir. Bu butonlar, seçilen zaman aralığına göre sağlık verilerini veya kesinti bilgilerini görüntüleme imkanı sunar. Kullanıcı, tercih ettiği zaman dilimine göre verileri görüntüleyebilir. Bu yapı, kullanıcıların sunucuların temel bilgilerini görmesini ve daha detaylı bilgilere erişmelerini sağlar. Ayrıca, zaman aralıklarına göre sağlık verilerini veya kesinti bilgilerini görüntüleyerek verileri analiz etme imkanı sunar. Bu analiz işlemini kolaylaştırmak amacıyla uygulamada seçimize göre ulaştığımız bilgileri report butonu yardımı ile haftalık listelenen verilerin günlere göre analizini yıllık listelenen verilerin aylara göre analizini kolayca görebilirsiniz. aynı zamanda export butonuna tıklandığında ise cav formunda görüntülediğiniz interrupt verilerini yada sağlık verileri dışa aktarabilirsınız.



Sunucu ve Hastane Ekleme İşlevi:

Uygulamanın alt barında yer alan "+" simbolüne tıklandığında, sunucu veya hastane eklemek için kullanılan bir sayfa açılır. Bu sayfada kullanıcılar, dört farklı butonla karşılaşır: "Hastane Ekleme", "Sunucu Ekleme", "Hastane ve Sunucu İçe Aktarma" butonları bulunmaktadır.

Hastane Ekleme: Bu seçenek, kullanıcıların bir şehir seçerek hastane eklemelerini sağlar. Önce bir şehir seçilir ve daha sonra hastane adı yazılır. Arka planda, şehirlere plaka numaraları ile ID'ler atanır.

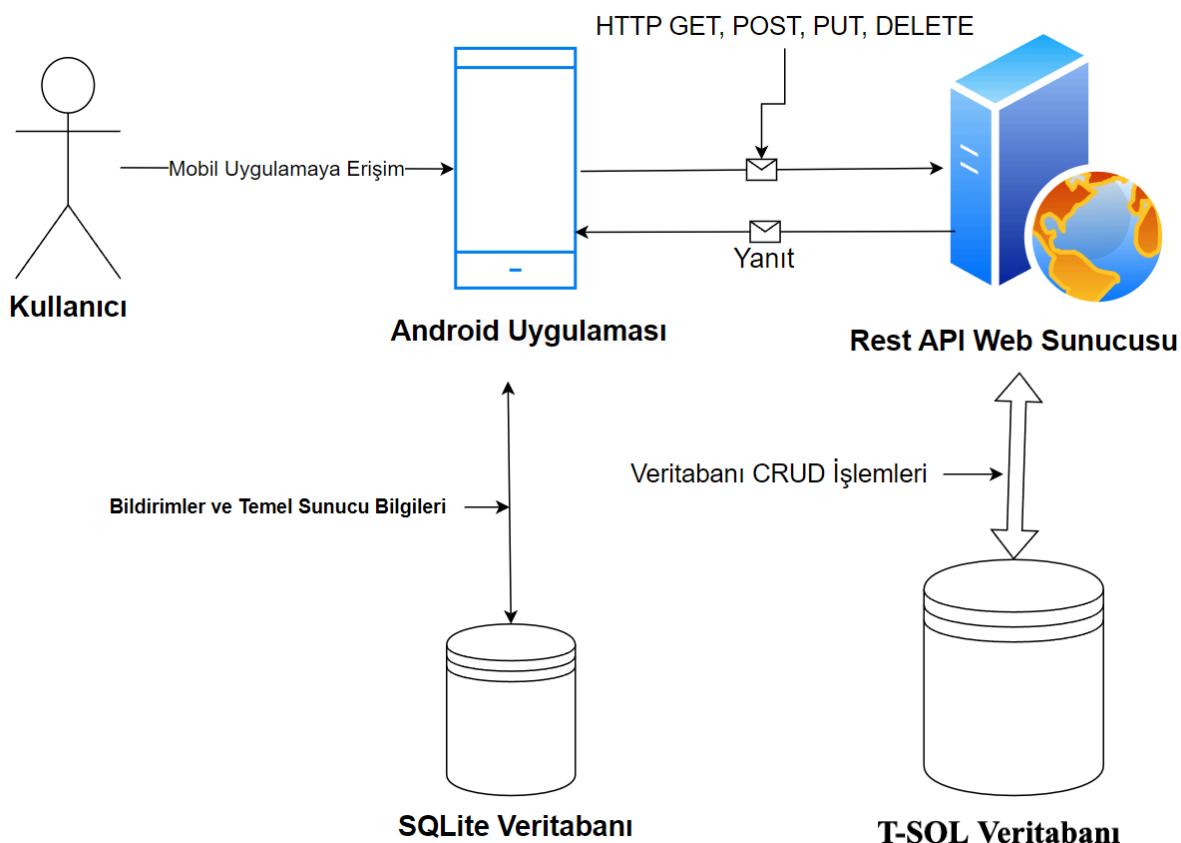
Sunucu Ekleme: Bu seçenek, kullanıcıların sunucu bilgilerini girmelerine olanak tanır. Sunucunun adı, IP adresi, sunucu işletim sistemi, RAM miktarı, depolama birimi ve kapasitesi gibi bilgiler girilir. Ardından eklenmiş hastanelerden biri seçilerek sunucu ekleme işlemi tamamlanır.

Hastane ve Sunucu İçe Aktarma:

CSV Dosyası İçe Aktarma: Kullanıcılar, CSV formatında bulunan verileri doğrudan uygulamaya aktarabilirler. Bu yöntemle toplu halde hastane veya sunucu verilerini içe aktarabilirler.

Bu sayfada kullanıcılar, istedikleri yöntemi seçerek veri girişini hızlı ve kolay bir şekilde tamamlayabilirler.

Uygulamanın Genel İşleyişi



Şekil xx: Uygulama Genel İşleyiş Diyagram

Şekil xx'de görüldüğü üzere tasarlanan uygulama işleyişi 4 farklı katmandan oluşmaktadır. Bunlar Andorid uygulaması, Rest API Web Sunucusu, T-SQL veritabanı ve SQLite veritabanıdır. Uygulama'nın işlevi şu şekildedir:

Kullanıcının Mobil Uygulama ile Etkileşimi:

- Kullanıcı, Android uygulamaya etkileşime geçerek belirlenen işlevleri yerine getirebilir.

Android Uygulama Katmanı:

- HTTP İstekleri ile REST API İletişimi: Uygulama, sunucu ile iletişim kurmak için HTTP istemleri (GET, POST, PUT, DELETE) oluşturur.
- **Veri İşleme:** Kullanıcı girişini yönetir, yanıtları işler ve uygulamanın arayüzünde bilgileri sunar.

REST API Web Sunucu Katmanı:

- **İstek Alım:** Sunucu, uygulamadan HTTP isteklerini alır.
- **İş Mantığı Yürütme:** İstekleri işler, gerekli iş mantığını yürütür ve veritabanıyla etkileşime girer.
- **Yanıt Oluşturma:** Sunucu, istenen verileri veya işlemlerin sonuçlarını içeren uygun HTTP yanıtları oluşturur.

Veritabanı Katmanı:

- **SQLite Veritabanı:** Gereksinimler doğrultusunda kullanıcının uygulamayı kullandığı cihazda yerel olarak oluşturulmuş bir veritabanıdır.
- **T-SQL Veritabanı:** Yerel veritabanındaki verilere ek olarak ER Diyagramında gösterilen tüm verileri yönetmek için kullanılır.

7. Diğer Entegrasyonlar

Sosyal Medya İle Giriş Entegrasyonu:

Uygulama, kullanıcılarla Google hesapları üzerinden kolayca oturum açma imkanı sunacak şekilde sosyal medya entegrasyonunu başarıyla gerçekleştirmiştir.

Kullanıcılar, uygulamaya Google hesaplarıyla giriş yapabilmekte ve bu entegrasyon, arka planda Google API'sinin etkili bir şekilde kullanılmasıyla sağlanmıştır.

E-posta Entegrasyonu:

Kullanıcıların kayıt olurken kullandıkları e-posta hesaplarına doğrulama maili gönderilerek hesaplarının doğrulanması işlemi bu entegrasyon aracılığıyla gerçekleştirildi. Kullanıcılar, kayıt işleminden sonra e-posta doğrulama adımına yönlendirilir ve burada gönderilen doğrulama maili ile hesaplarını aktifleştirebilirler.

Veri Aktarımı:

Uygulama, sağlık durumu ve interrupt verilerini haftalık, aylık ve yıllık periyotlarla listeleme ve bu verileri dışa aktarma (export) özelliğine sahiptir. Bu veriler XLS formatlarında yani excel olarak dışa aktarılabilmektedir. Bu işlev, sunucuların performans analizlerini gerçekleştirmek ve geleceğe yönelik planlar yapabilmek için önemlidir. Örneğin, kesinti analiz sonuçlarına dayanarak geliştirmeler yapılabilir ve kesinti riskini minimuma indirmek için gerekli adımlar atılabilir. Bu dışa aktarma işlemi, verilerin analiz edilerek sunucuların daha iyi yönetilmesine yardımcı olur. Bu entegrasyonlar, kullanıcıların uygulamaya erişimini ve verilerin yönetimini kolaylaştırarak sunucu performansını analiz etme ve iyileştirme süreçlerini desteklemektedir. Bu özellikler sayesinde, kullanıcılar uygulamanın sağladığı veri analizi imkanlarıyla sunucuların daha etkin bir şekilde yönetimini gerçekleştirebilmektedirler.

Uygulama, sunucu veya hastane eklerken CSV formatında içe aktarma özelliğine sahiptir. Bu özellik, kullanıcılarla zaman kazandırarak mevcut veri setlerini hızlı ve toplu bir şekilde uygulamaya aktarma imkanı sunar.

	A	B	C	D	E	F	G	H	I	J
1	Health ID	Date	CPU Usage	RAM Usage	Storage Usage	Server Temp	Ambient Temp	Energy Usage	Heartbeat	Server Name
2	950	Jan 5 2024 10:19PM	73	3437	1477	20	20	890	0	Randevu Kayıt - 1
3	1422	Jan 5 2024 4:37AM	15	8987	1145	69	69	539	1	Randevu Kayıt - 1
4	698	Jan 3 2024 3:50AM	49	4389	989	67	29	813	1	Randevu Kayıt - 1
5	1284	Jan 2 2024 8:09AM	53	7826	1377	63	39	1151	0	Randevu Kayıt - 1
6	1915	Jan 2 2024 6:11AM	14	7373	936	58	52	1362	1	Randevu Kayıt - 1
7	1738	Dec 28 2023 10:49AM	73	6438	572	24	21	1226	1	Randevu Kayıt - 1
8	1535	Dec 28 2023 12:23AM	80	5379	775	41	29	563	1	Randevu Kayıt - 1
9	30	Dec 28 2023 6:03PM	13	8989	1018	63	42	539	1	Randevu Kayıt - 1
10	533	Dec 27 2023 10:38PM	19	6284	930	39	34	562	0	Randevu Kayıt - 1
11	1331	Dec 27 2023 3:38AM	79	8326	894	43	56	1131	1	Randevu Kayıt - 1
12	1124	Dec 26 2023 7:09PM	33	3187	1302	54	39	1155	1	Randevu Kayıt - 1
13	1485	Dec 25 2023 11:46PM	14	4262	809	62	38	701	1	Randevu Kayıt - 1
14	697	Dec 24 2023 3:07PM	35	3749	653	39	59	561	0	Randevu Kayıt - 1
15	1851	Dec 24 2023 7:26PM	83	2719	943	63	35	517	1	Randevu Kayıt - 1
16	867	Dec 23 2023 9:14AM	35	9311	968	29	29	1316	1	Randevu Kayıt - 1
17	917	Dec 22 2023 11:02PM	47	5249	881	53	20	678	0	Randevu Kayıt - 1
18	1990	Dec 19 2023 9:28AM	25	7352	912	31	50	791	0	Randevu Kayıt - 1
19	1820	Dec 18 2023 5:48PM	77	9179	873	69	33	629	1	Randevu Kayıt - 1
20	1359	Dec 17 2023 2:20AM	73	2687	1016	68	52	1316	1	Randevu Kayıt - 1
21	191	Dec 17 2023 6:56AM	85	7274	1159	33	21	997	1	Randevu Kayıt - 1
22	1862	Dec 16 2023 2:55PM	18	8552	1160	58	20	612	0	Randevu Kayıt - 1
23	649	Dec 13 2023 12:49AM	12	6272	1286	27	45	1213	1	Randevu Kayıt - 1
24	1946	Dec 12 2023 9:23AM	49	3071	662	59	45	640	1	Randevu Kayıt - 1
25	1244	Dec 10 2023 7:15AM	62	5603	668	60	44	502	0	Randevu Kayıt - 1
26	1430	Dec 9 2023 8:15AM	12	5221	1392	54	69	1245	1	Randevu Kayıt - 1
27	1479	Dec 8 2023 4:22AM	77	7539	1031	43	24	763	1	Randevu Kayıt - 1
28	596	Dec 7 2023 5:06PM	72	4224	1214	46	42	895	0	Randevu Kayıt - 1
29	1873	Dec 7 2023 12:59PM	23	3673	1460	54	53	1414	1	Randevu Kayıt - 1
30	1644	Dec 7 2023 1:29AM	87	5592	1090	34	21	1261	1	Randevu Kayıt - 1
31	341	Dec 5 2023 7:37PM	19	5708	1334	45	29	1446	1	Randevu Kayıt - 1
32	48	Dec 5 2023 5:49PM	57	9810	560	64	68	942	0	Randevu Kayıt - 1
33	869	Dec 5 2023 1:53PM	31	7561	740	26	65	892	0	Randevu Kayıt - 1
34	1434	Dec 4 2023 5:03PM	30	2345	1401	39	24	817	0	Randevu Kayıt - 1

8. Çalışma Adam-saat değerleri

Görev Dağılımı:

Hakan YAVAŞ:

1. Spring Boot Rest API Geliştirme.
2. Veritabanı ER diyagram tasarımları, SQL Server'a implementasyonu.
3. Stored Procedure, Trigger ve Error Handling işlemleri.
4. Google ile giriş entegrasyonu.
5. Excel Export ve CSV Import Operasyonları.
6. Giriş, Kayıt, Doğrulama, Sunucu Listeleme, Sunucu Detay, Sunucu Ekleme ve Profil sayfalarının API ile entegrasyonu, tasarımlarının düzeltilmesi.
7. Birim ve sistem testleri.
8. Raporlama.

Emre KART:

1. Gereksinim analizi yapılması.
2. Gereksinimlere göre veritabanı ER diyagram tasarımı.
3. Sunucu ekleme sayfasının entegrasyonu
4. Android studio üzerinden uygulama arayüzü giriş, kayıt, doğrulama ve profil prototip tasarımları.
5. Raporlama ve sunum.

Alperen GÜNEŞ:

1. Figma üzerinden gerekli tasarım prototipinin yapılması ve uygulama içerisinde kullanılan ikon, tema ve logo tasarımı.
2. Başlangıç ekranı, giriş, kayıt, doğrulama sayfalarının tasarımlarının Android Studio'ya aktarımı.
3. Sunucu Detay sayfasında sağlık verilerinin Retrofit aracılığıyla listelenmesi.
4. "Fragment" yapısı kullanarak uygulama menüsünün tasarlanması.
5. Tüm sayfaların ve butonların kontrol edilip belirli bir standart çerçevesinde düzenlenmesi ve basit arayüz hatalarının giderilmesi.
6. Raporlama.

Adam Saat Verileri

KİŞİ	ADAM/SAAT
Hakan YAVAŞ	143.55
Emre KART	106.67
Alperen GÜNEŞ	146.8
TOPLAM	397,02

VTYS - Sunucu Sağlık İzleme App. Add Task ▾

Overview List Board Calendar + View Search Hide Customize ▾

Group: Status Subtasks: Collapse all | Filters Me mode Assignees Show closed | Hide Search tasks...

COMPLETED 9 ... + Add Task

Name	Assignee	Due date	Priority	...
Kotlin'de Kullanılacak SQL Kararı ve Kullanımı	HY
Java Temelleri ve Kotlin Java Bağlantısı	EK HY AG
Gereksinim Analiz Raporu	HY EK AG
GY - OOP & Temel Eğitim	HY EK AG	...	Urgent	...
Kotlin üzerinde kullanılacak Docker vb. sanal su...	EK
Benzer uygulamalarda kullanılan teknolojiler ve ...	AG
Figma ile Arayüz Tasarımı ve Benzer Uygulama ...	EK HY AG	12/7/23
Gereksinim Analiz Raporu V2	EK
VTYS Proje Adım 1: Sunum 1	EK

+ Add Task

COMPLETE 6 ... + Add Task

Name	Assignee	Due date	Priority	...
Gereksinim Analizi V3	AG HY EK	12/27/23	Normal	...
Sosyal Medya Giriş Entegrasyonu	HY	...	Normal	...
Spring Boot API	HY	...	High	...
API'İN Uygulamaya Entegrasyonu	HY	...	High	...
Tasarımın Android Studio'ya Entegrasyonu	AG EK	Tomorrow	Urgent	...
Final Sunumlar & Rapor	HY AG EK

+ Add Task

9. Sonuç ve Değerlendirme

Uygulamada gereksinimler çerçevesinde değerlendirildiğinde eksikler mevcuttur. Google giriş ile alınan kullanıcıların veritabanına kaydı ve yetkilendirilmesi, bildirim gönderme, favoriler, özel olarak sunucu takibi, kullanıcıların yetkilendirilmeleri, profil ekranından bilgi değiştirebilme özellikleri uygulamamıza raporun teslim edildiği tarih itibarıyle entegre edilememiştir. Uygulama bir çok mimariyi iç içe kullanması bakımından geliştirilmeye açık durumdadır. Eksiklerin en kısa süre içerisinde tamamlanıp Mergen Yazılım A.Ş. firmasına teslimi planlanmaktadır.

Özdeğerlendirme Başlığı	% xx (0-100)
1-Proje başındaki amaç-hedeflerde değişme oranı	10
2- Mevcut veriyapısı (tasarım, oluşturulan tablo vb) nin proje ister ve uygulama ihtiyaçlarını karşılama oranı	80
3- Mevcut Karmaşık sorguların tasarım ve tam entegre uygulamaya alınma oranı;	100
4- Saklı Yordam, Hata Ayıklama ve Tetikleyicilerin tam entegre uygulamaya alınma oranı;	90
5- Arayüzlerin tamamlanması ve tam entegre uygulamaya alınma oranı	65
6- Diğer Entegrasyonların (sosyal medya, e-mail/gsm, export/import) tam entegre uygulamaya alınma oranı	80
7- Projenin tam entegre tamamlanma oranı	70
8-Uygulamanın alanda kullanılabilirlik oranı	75

9- Sunumun; genel akışı, teknik içerik aktarım yöntemleri, grup çalışanlarının kendi içinde dengeli şekilde konu anlatımı, zamanı etkin kullanım vb açısından değerlendirmesi	90
10- Proje raporunun proje çalışmalarını yansıtma oranı	85