AVM Yönetim Sistemi — Proje Raporu

Anisa Emin
Bilişim Sistemleri Mühendisliği
Teknoloji Fakültesi
Kocaeli Üniversitesi
Kocaeli, Türkiye
anisaaae04@gmail.com

Esma Nur Mantı
Bilişim Sistemleri Mühendisliği
Teknoloji Fakültesi
Kocaeli Üniversitesi
Kocaeli, Türkiye
esmanmanti@gmail.com

Nalan Kara
Bilişim Sistemleri Mühendisliği
Teknoloji Fakültesi
Kocaeli Üniversitesi
Kocaeli, Türkiye
nalankaraa3@gmail.com

Özet **- Bu proje, mağaza kiralamaları, çalışan sözleşmeleri ve ödemeler gibi alışveriş merkezi operasyonlarını verimli bir şekilde yönetmeye odaklanan tam kapsamlı bir uygulama olan Alışveriş Merkezi Yönetim Sistemi'ni tanıtıyor. Sistemin merkezinde yer alan ilişkisel MySQL veritabanı, tüm iş ile ilgili bilgilerin doğruluğunu sağlama, hızlı erişim ve güvenli depolama konularında hayati bir rol oynamaktadır.**

Veritabanı yapılandırılmış ve ölçeklenebilir veri yönetimini mümkün kılarak mağazalar, sözleşmeler, kullanıcılar ve ödemeler arasındaki ilişkileri sürdürmeyi kolaylaştırır. Bu entegrasyon sayesinde sistem, rol tabanlı erişim kontrolü (RBAC) uygulayarak her kullanıcının yalnızca kendi rolüne uygun verilerle etkileşimde bulunmasını sağlar — ister yönetici, ister müdür, ister muhasebeci olsun.

Next.js ve MySQL kullanılarak geliştirilen sistem, veritabanı verimliliğinden yararlanarak manuel takip ve evrak işleri gerektiren görevleri otomatikleştiriyor ve basitleştiriyor; böylece alışveriş merkezi operasyonları daha şeffaf, düzenli ve güvenilir hale geliyor.

Anahtar Kelimeler—Alışveriş Merkezi Yönetim Sistemi, Next.js, MySQL, Rol Tabanlı Erişim Kontrolü, Veritabanı Tasarımı

I. GİRİŞ

Günümüzün dinamik iş ortamında, ticari mülkleri, sözleşmeleri ve finansal işlemleri yönetmek yalnızca etkili bir organizasyon değil, aynı zamanda güvenilir veri depolama ve işlemeyi de gerektirir. Bu gereklilik, alışveriş merkezi yönetimindeki temel işlemleri otomatikleştiren bir bilgi sistemi geliştirmenin önemini vurgular.

Bu proje, mağazalar, kira sözleşmeleri, çalışan sözleşmeleri ve ödemelerle ilgili verileri merkezileştiren ve verimli bir şekilde yöneten MySQL ilişkisel veritabanından yararlanan web tabanlı tam kapsamlı bir uygulama olan Alışveriş Merkezi Yönetim Sistemi'ni sunmaktadır. İyi yapılandırılmış bir şema ve optimize edilmiş SQL sorguları sayesinde sistem, iş faaliyetlerine hızlı erişim, doğruluk ve izlenebilirlik sağlamaktadır.

Uygulama, hem frontend hem de backend mantığı için Next.js kullanılarak oluşturulmuştur ve yöneticiler, mağaza müdürleri ve muhasebeciler için güvenli ve düzenli erişimi sağlamak amacıyla rol tabanlı erişim denetimi (RBAC) uygulanmaktadır. Tüm kullanıcı eylemleri, MySQL veritabanı ile gerçek zamanlı etkileşimlerle desteklenmektedir ve bu da verilerin işlemleri etkinleştirme ve kolaylaştırmadaki merkezi rolünü göstermektedir.

Projenin temel amacı, ilişkisel veritabanlarının yalnızca teknik güçlü yönlerini değil, aynı zamanda alışveriş merkezleri gibi karmaşık iş yapılarının yönetimini nasıl basitleştirip geliştirdiğini de ortaya koymaktır.

II. PROBLEM TANIMI

A. *Alışveriş Merkezi Yönetiminde Operasyonel Zorluklar*

Sözleşmelerin, kira ödemelerinin, mağaza sahiplik bilgilerinin ve çalışan atamalarının yönetimi geleneksel olarak manuel yöntemlerle veya birbiriyle bağlantısız sistemler aracılığıyla yapılmaktadır. Bu durum, özellikle büyük miktarda verinin yönetilmesi gerektiğinde, verimsizliklere, gecikmelere ve insan hatası riskinin artmasına yol açmaktadır.

B. *Merkezi Veri ve Otomasyon Eksikliği*

Merkezi bir platform olmadan kira sözleşmelerini takip etmek, mağaza ile ilgili verilere erişmek veya geçmiş ödeme kayıtlarını görüntülemek zorlaşmaktadır. Finansal kayıtların ve sözleşme bitiş tarihlerinin manuel olarak takip edilmesi, karışıklığa ve potansiyel gelir kaybına neden olabilmektedir.

C. *Verimsiz Rol Tabanlı Kontrol*

Geleneksel yapılarda, roller (örneğin müdür, yönetici, muhasebeci) bazında erişimi sınırlamak, açıkça tanımlanmış bir yetkilendirme modeli olmadan zordur. Bu, istenmeyen veri düzenleme veya silme gibi yetkisiz eylemlere yol açabilir.

Ç. *Yapılandırılmış Veritabanının Önemi*

İlişkisel veritabanı (MySQL) kullanılan sistem şunları sağlar:

- Tüm veri varlıklarının (mağazalar, sözleşmeler, kullanıcılar, ödemeler) merkezi olarak depolanması
- SQL kullanarak hızlı ve doğru sorgulama yapılması
- Veriler arasındaki tutarlı ilişkilerin kurulması (örneğin, hangi kullanıcının hangi mağaza için hangi ödemeyi yaptığı)

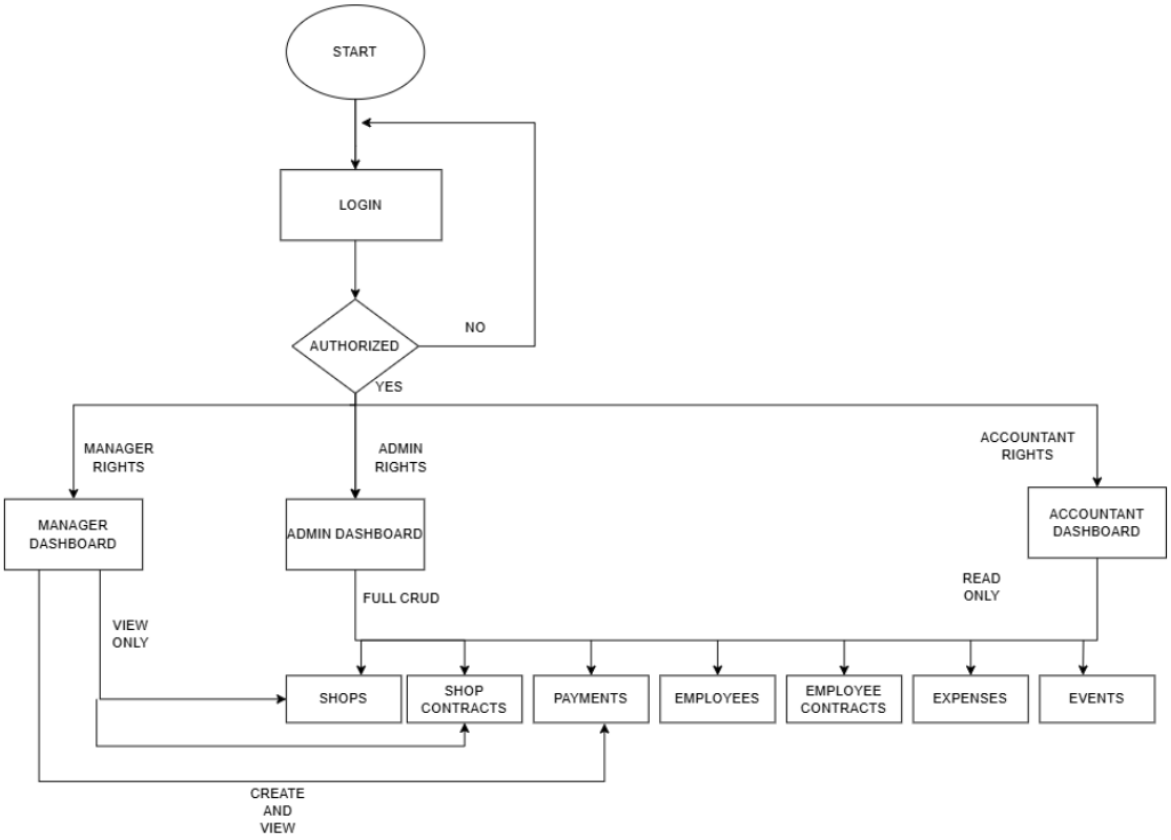
İlişkisel bir şema kullanımı, sistemin veri bütünlüğünü, ölçeklenebilirliğini ve uzun vadeli sürdürülebilirliğini artırır.

D. *Projenin Amacı*

Bu projede amaç, bu zorlukları çözen tam kapsamlı bir web uygulaması oluşturmaktır:

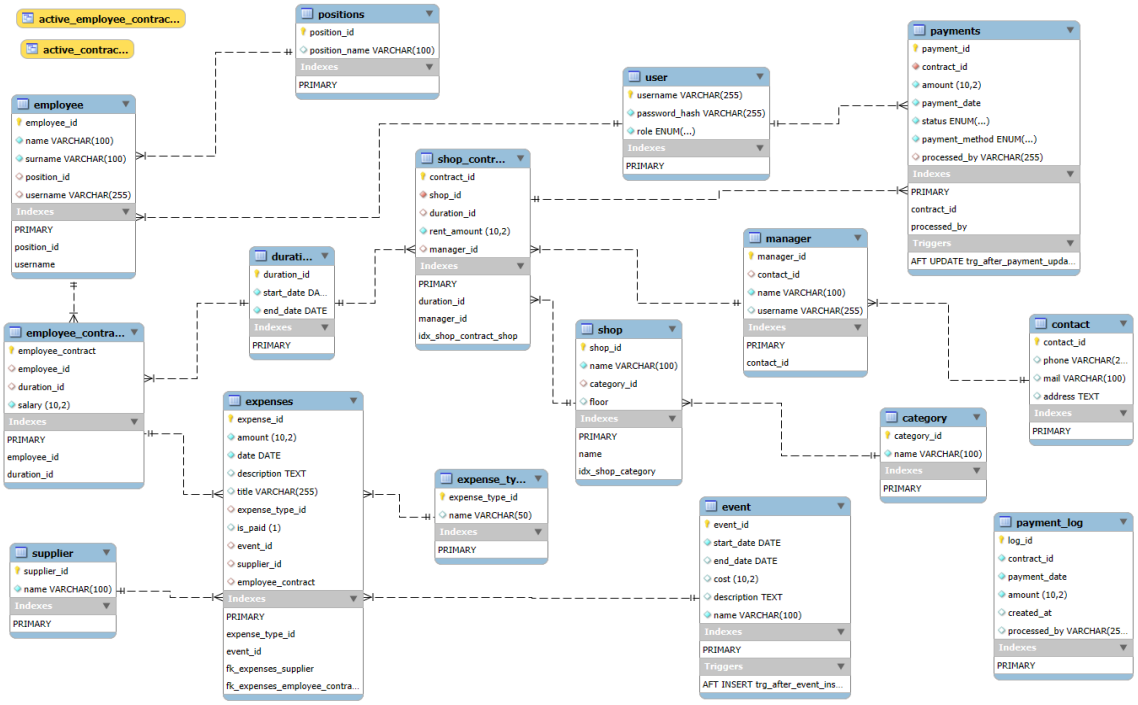
- Temiz ve sezgisel bir kullanıcı arayüzü
- Mantığı güvenli bir şekilde işleyen iyi tasarlanmış bir backend
- Alışveriş merkezi yönetimini basit, otomatik ve güvenilir hale getiren sağlam bir veritabanı

III. AKIŞ ŞEMASI



(Görsel 1.1 Akış şeması olarak verilmiştir.)

IV. ER DİYAGRAMI



(Görsel 1.2 ER diyagramı olarak verilmiştir.)

V. YAPILAN ARAŞTIRMALAR

A. Karşılaşılan Problemler

Alışveriş Merkezi Yönetim Sisteminin geliştirilmesi sırasında, öncelikle veritabanı şeması tasarımı ve rol tabanlı erişim kontrolü (RBAC) ile ilgili çeşitli teknik ve yapısal zorluklarla karşılaşıldı. Temel sorunlar şunları içeriyordu :

1. **Veritabanı Normalizasyonu** : Kullanıcılar, mağazalar, sözleşmeler ve ödemelerle ilgili tabloların ilişkisel yapısındaki tekrarları önlemek. Normalizasyon ve sorgu verimliliği arasında optimum dengeyi bulmak bir zorluktu.
2. **Karmaşık İlişkilerin Yönetimi :** Ödemeler ile mağazalar, sözleşmeler ile giderler arasındaki ilişkiler bire-çok ve çoktan-çoğa yapılar içerdiğinden, bu ilişkilerin veritabanı şemasında doğru şekilde temsil edilmesi gerekmiştir
3. **Sorgularda Erişim Kontrol Mantığı :** Müdürlerin yalnızca kendi mağazalarını, sözleşmelerini ve ödemelerini görebilmesi ve başka verileri görüntüleyip değiştirememesi sağlanırken, yöneticilerin tüm kayıtları görüntüleyebilmesi ancak düzenleme yapamaması gibi kurallar uygulanmıştır.
4. **Next.js'de API Rota Yapılandırması:** MySQL veritabanıyla doğru şekilde etkileşim kuracak ve backend'te erişim kontrolünü sağlayacak temiz ve güvenli RESTful endpoint'lerin oluşturulması gerekmiştir.

B. Araştırma ve Çözümler

Bu zorlukların üstesinden gelmek için aşağıdaki adımlar ve araştırma yaklaşımları izlendi:

1. **Rol Tabanlı Erişim Kontrolü (RBAC):** SQL tabanlı backend'te RBAC'yi uygulamada en iyi uygulamalar üzerine kapsamlı bir araştırma yürütüldü. Roller ayrı bir roller tablosuna ayrıldı ve her kullanıcıyı karşılık gelen rolüne eşlemek için kullanıcılar tablosunda yabancı anahtarlar kullanıldı. Backend'te her rota için erişim doğrulaması yapan middleware (ara yazılım) geliştirildi.
2. **Varlık-İlişki (ER) Diyagram Tasarımı:** Projenin erken aşamalarında, kullanıcılar, roller, mağazalar, sözleşmeler ve ödemeler arasındaki ilişkileri görsel olarak haritalamak amacıyla bir ER diyagramı oluşturuldu. Bu, veritabanının düzgün bir şekilde yapılandırılmasına yardımcı oldu ve hangi alanların hangi tabloya ait olduğunu netleştirdi.
3. **SQL JOIN ve Kısıtlamaların Kullanımı:** Yalnızca yetkili verilerin görüntülenmesini sağlamak amacıyla dikkatlice oluşturulmuş MySQL JOIN sorguları kullanılmıştır. Raporlama için LEFT JOIN, veri sahipliği ve rol eşleşmelerini sağlamak için INNER JOIN kullanımı tercih edilmiştir. Yabancı anahtar kısıtlamaları referans bütünlüğünün korunmasına yardımcı olmuştur.
4. **Backend Erişim Kontrol Ara Yazılımı:** Her isteği işlemeye başlamadan önce kullanıcının rolünü JWT üzerinden doğrulayan özel middleware (ara yazılım) geliştirilmiştir.
5. **Dokümantasyon, Topluluk Kaynakları ve Yapay Zeka Desteği:** Next.js'in resmi dokümantasyonu [1] ,MySQL öğretici kaynakları [2] kapsamlı bir şekilde kullanıldı. Ek olarak, açık kaynaklı proje yapıları ve forumları (örneğin, Stack Overflow, GitHub tartışmaları) en iyi uygulamalar hakkında fikir vermiştir. Ayrıca ChatGPT [3] gibi bir yapay zeka destekli Chatbot'tan yardım alınmıştır.
6. **Rol Senaryolarının Test Edilmesi:** Müdür ve yönetici kullanıcı rolleri için gerçek dünya senaryoları simüle edildi. Bunlar erişim kısıtlamalarının doğrulanmasına olanak sağladı ve hem frontend hem de backend katmanlarında eksik mantığın belirlenmesine yardımcı oldu.
7. **Veritabanı Normalizasyonu** : Kullanıcılara ilişkin tabloların ilişkisel yapısında veri tekrarını önlemek adına normalizasyon kuralları uygulanmıştır..

VI. YAZILIM MİMARİSİ

Sistemin yazılım mimarisi, hem frontend hem de backend işlevlerini tek bir çatı altında birleştiren Next.js 13+ ile monorepo yapısı kullanılarak tasarlanmıştır. Projede herhangi bir ORM katmanı kullanılmamaktadır; bunun yerine, tüm SQL sorguları doğrudan /app/api/ dizinindeki API route handler dosyaları içinde yazılmıştır.

A. Sistem genel bakışı

Frontend ve Backend Tek Uygulamada : Uygulama, aynı proje içinde sunucu tarafı işleme (SSR) ve API rotalarının aynı proje içinde ele alınmasını sağlayan Next.js App Router mimarisi kullanılarak yapılandırılmıştır. Tüm mantık /app dizini üzerinden yürütülmektedir.

Kimlik Doğrulama : Kullanıcı kimlik doğrulaması, Credentials Provider stratejisiyle NextAuth kütüphanesi kullanılarak sağlanmaktadır. Kullanıcı rolleri (admin, muhasebeci, müdür) içeren özel bir JWT token oluşturularak hem istemci hem de sunucu tarafında bu token üzerinden erişim yetkileri yönetilmektedir.

Veritabanı Katmanı : Sistem, ilişkisel ve MySQL uyumlu bir veritabanı kullanır. Tüm veritabanı işlemleri, ORM (örneğin Prisma veya Sequelize) kullanılmadan, doğrudan API route dosyaları içinde yazılmış ham SQL sorguları ile gerçekleştirilir (örneğin: /api/payments/route.ts). Bu yaklaşım, sorgu mantığı üzerinde tam kontrol, yüksek performans ve kolay optimizasyon imkânı sağlar.

Rol Tabanlı Erişim :

- Admin kullanıcılar, mağaza, sözleşme, çalışan, etkinlik ve gider oluşturma, düzenleme ve silme

işlemlerini gerçekleştirebilir. Ayrıca ödeme onaylama veya reddetme yetkisine sahiptir.
- Muhasebeciler, tüm verileri görüntüleme erişimine sahip olup, ödemeleri onaylayabilir veya reddedebilirler.
- Müdürler, sadece kendileriyle ilişkili mağaza ve sözleşmeleri görüntüleyebilir ve yeni ödeme girişleri başlatabilirler.

Uygulama, veritabanı ile etkileşim için Server Components ve API Routes yaklaşımlarını kullanım amacına göre birleştirerek kullanır:

- **Server Components** içinde, ham SQL sorguları doğrudan API rotaları üzerinden geçmeden çalıştırılır. Bu yöntem, özellikle sunucu tarafından işlenmiş paneller gibi güvenli ve rol-tabanlı verilerin alınmasında daha yüksek performans ve düşük karmaşıklık sağlar.

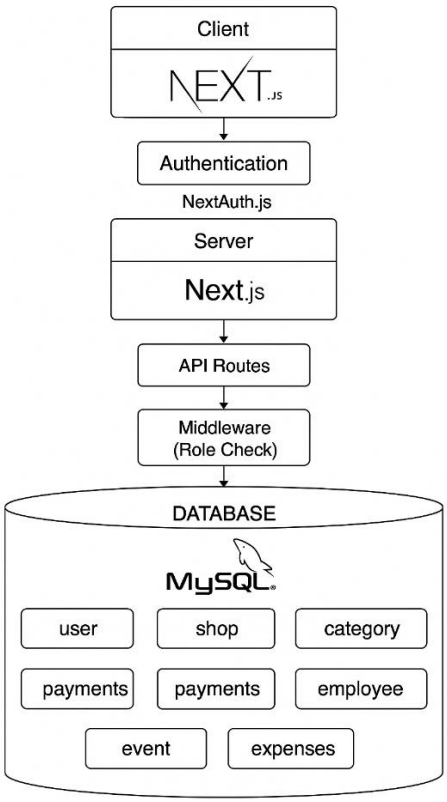
- **API Routes** (örneğin, /app/ api/payments/ route.ts), istemci tarafından tetiklenen işlemler veya yetkili kullanıcıların yaptığı değişiklikler (örneğin ödeme gönderme, sözleşme oluşturma) için kullanılır. Bu rotalarda da ham SQL sorguları yazılır ve rol bazlı izin kontrolleri yapılır.

Bu iki yaklaşımın birleştirilmesiyle sistem, performans (veri okumaları için Server Components) ve esneklik (istemci etkileşimleri için API Routes) arasında bir denge sağlar.

Oturum ve Rol Devamlılığı : Oturum açtıktan sonra, NextAuth kullanıcının rolünü ve kullanıcı adını içeren bir JWT oluşturur. Bu JWT, hem sunucuda hem de istemcide NextAuth oturum yardımcıları aracılığıyla erişilebilir olur ve güvenli ve kalıcı oturum işlemeyi garanti eder.

Güvenlik :
Her istekte kullanıcı rolü doğrulanır.
Sözleşme oluşturma/silme veya ödemeleri onaylama gibi hassas işlemler, hem kimlik doğrulamayı hem de yetkilendirmeyi JWT kullanarak kontrol eden sunucu tarafı middleware (ara yazılım) aracılığıyla korunur.

Triggers ve Views :
- Bir ödeme "Tamamlandı" olarak işaretlendiğinde payment_log tablosuna otomatik olarak kayıt ekleyen bir trigger vardır.
- Yeni bir etkinlik oluşturulduğunda gider (expense) kaydı oluşturan başka bir trigger vardır..
- active_contracts adlı bir view, geçerli tarih baz alınarak aktif durumda olan mağaza sözleşmelerini gösterir.



B. Yazılım Mimarisi Diyagramı

VII. GENEL YAPI

Bu proje, alışveriş merkezlerinde yer alan mağazalara ait kira sözleşmeleri, ödemeler, etkinlikler, çalışanlar ve giderlerin yönetimini sağlamak amacıyla geliştirilen web tabanlı bir yönetim sistemidir. Uygulama, rol tabanlı erişim kontrolünü destekler ve kullanıcıların (yönetici, muhasebeci, müdür) yalnızca kendilerine atanmış yetkiler doğrultusunda işlem yapmalarına izin verir.

Sistem, hem istemci hem de sunucu bileşenlerini kullanan Next.js 13+ sürümü ile geliştirilmiştir. Arka uç işlevselliği, app/api dizini altındaki API route'ları aracılığıyla sağlanmakta olup, tüm veritabanı işlemleri doğrudan yazılmış ham SQL sorguları ile gerçekleştirilir — herhangi bir ORM (Nesne-İlişkisel Eşleme) kullanılmamaktadır.

Kullanıcı kimlik doğrulaması, NextAuth kütüphanesi aracılığıyla gerçekleştirilir ve doğrulanan her kullanıcının rolü, bir JWT (JSON Web Token) içinde güvenli bir şekilde saklanarak oturum boyunca tutarlı erişim kontrolü sağlanır.

Backend'te, ilişkisel bir MySQL veritabanı kullanılır. Şema; mağaza (shop), mağaza sözleşmesi (shop_contract), ödeme (payments), etkinlik (event), çalışan (employee) ve gider (expenses) gibi temel varlıkları içerir. Veritabanında ayrıca, tamamlanan ödemeleri kaydetmek veya bir etkinlik oluşturulduğunda otomatik gider kaydı oluşturmak gibi işlemleri gerçekleştiren **görünümler (views)** ve **tetikleyiciler (triggers)** de kullanılmaktadır.

Kullanıcılar giriş yaptıktan sonra rollerine göre yönlendirilir ve yalnızca izinli oldukları işlevlere erişim hakkı kazanırlar. Örneğin, muhasebeciler ödeme ve gider işlemlerini yönetebilirken, müdürler mağaza sözleşmeleri ve çalışan kayıtlarını gözetebilir.

Bu mimari, esnek ve güvenli bir yetkilendirme modeli ile güçlü bir veri yapısı sunar. Doğrudan SQL sorguları kullanılması sayesinde sistem, veritabanı mantığı ve performans iyileştirmeleri üzerinde tam kontrol sağlar.

REFERANSLAR

[1] Next.js Documentation, Vercel. [Online]. Available: https://nextjs.org/docs

[2] MySQL Documentation, Oracle. [Online]. Available: https://dev.mysql.com/doc/

[3] ChatGPT , OpenAI. [Online]. Available: https://chat.openai.com