

MARMARA ÜNİVERSİTESİ TEKNOLOJİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ LİSANS PROGRAMI
2025-2026 GÜZ DÖNEMİ

PROJE SUNUM DOSYASI
JAVA PROGRAMLAMA DİLİ İLE
STOK OTOMASYON UYGULAMASI PROJESİ

Ders Kodu ve Adı

BLM2005.1 NESNE YÖNELİMLİ PROGRAMLAMA

Ders Sorumlusu

DOÇ. DR. KAZIM YILDIZ

Hazırlayan: Adı, Soyadı ve Numarası

ALİ TALHA YURTSEVEN 170424526

İBRAHİM ERDEM TOPÇU 171424013

İDİL ŞEN 170424067

Link: <https://github.com/alitalhq/stock-automation-system>

ARALIK 2025

Proje Detayları ve Kapsamı.....	3
1. Kullanılan Teknolojiler.....	3
2. Kullanıcı Arayüzü Tanıtımı	4
2.1. <i>Giriş – Kayıt Ekranı:</i>	4
2.2. <i>Kullanıcı Paneli:</i>	4
2.3. <i>Admin Paneli.....</i>	5
2.4. <i>Ürün Ekleme Paneli.....</i>	6
2.5. <i>Ürün Düzenleme Ekranı</i>	6
2.6. <i>İşlem Geçmiş Paneli</i>	7
3. Sistem mimarisi	7
3.1. <i>MainApp.java</i>	8
3.2. <i>controller/LoginController.java.....</i>	8
3.3. <i>controller/MainController.java.....</i>	8
3.4. <i>model/Stockable.java (Interface)</i>	8
3.5. <i>model/Loggable.java (Interface).....</i>	8
3.6. <i>models/rental_history.py.....</i>	8
3.7. <i>model/BaseProduct.java (Abstract)</i>	8
3.8. <i>model/Product.java.....</i>	8
3.9. <i>model/User.java</i>	8
3.10. <i>service/JsonService.java</i>	8
3.11. <i>resources/login-view.fxml.....</i>	9
3.12. <i>resources/main-view.fxml</i>	9
4. Örnek Kodlar ve UML Diyagramları	9
4.1. <i>Controller</i>	9
4.2. <i>Model.....</i>	10
4.3. <i>Service.....</i>	11
4.4. <i>Exception.....</i>	12
5. Kaynakça	13

Proje Detayları ve Kapsamı

Bu proje, işletmelerin envanter yönetim süreçlerini dijitalleştirerek ve stok hareketlerini gerçek zamanlı olarak takip etmek amacıyla tasarlanmış nesne yönelimli (OOP) bir otomasyon platformudur. Sistemin temel odağı, manuel veri takibinden kaynaklanan hataları minimize etmek ve kritik stok eşiklerini dinamik olarak yönetmektir. Proje kapsamında ürünlerin teknik detayları, fiyatlandırma bilgileri ve anlık stok durumları merkezi bir JSON veri tabanı üzerinden yönetilmektedir.

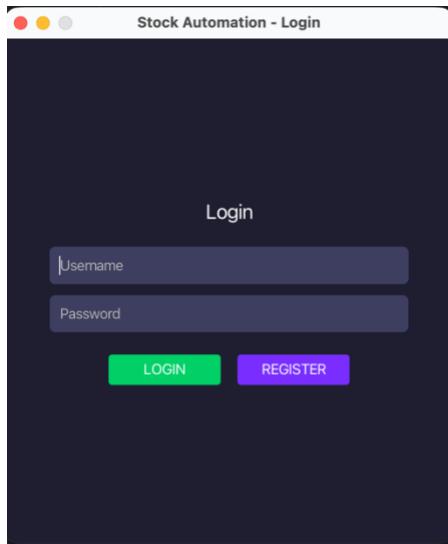
1. Kullanılan Teknolojiler

Tablo 1 Kullanılan Teknolojiler Tablosu

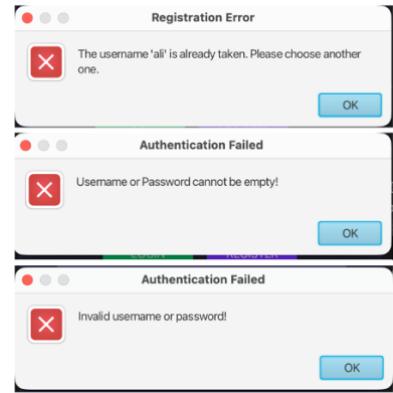
Kategori	Teknoloji	Seçilme Nedeni
Programlama Dili	Java 17 (LTS)	Güçlü OOP desteği, tip güvenliği ve kurumsal uygulama standartları.
Arayüz Teknolojisi	JavaFX	XML tabanlı (FXML) arayüz tasarımı ve CSS ile özelleştirilebilir modern görünüm.
Veri Yönetimi	JSON (Jackson)	Hafif, taşınabilir ve hiyerarşik veri saklama imkanı sunması.
Mimari Desen	Modüler (MVC)	Mantıksal katmanların (Model, View, Controller) birbirinden ayrılması ile yüksek bakım kolaylığı.
Geliştirme Ortamı	IntelliJ IDEA	Gelişmiş hata ayıklama ve moders arayüz sunması.

2. Kullanıcı Arayüzü Tanıtımı

2.1. Giriş – Kayıt Ekranı:



Şekil I Giriş - Kayıt Ekranı



Şekil II Giriş – Kayıt Ekranı Hata Yönetimi

- Sistemin ilk güvenlik katmanı
- Rol bazlı yetkilendirme (admin kullanıcı için şifre: *admin*)
- Hata yönetimi ile geçersiz giriş denemelerinin yakalanması
- Kullanıcıya anlık geri bildirim

2.2. Kullanıcı Paneli:

Name	Category	Price (TL)	Stock
Fiat Doblo	Automotive	850000.0	117
iPhone 15	Electronics	55000.0	10
Gaming Mouse	Electronics	450.0	9
Coffee Beans	Grocery	120.0	20
iPhone 16 Pro	Electronics	75000.0	10
Water 0.5L	Drink	10.0	100
Dining Table	Furniture	4500.0	2
MacBook Pro M3	Electronics	95000.0	6
Samsung S24 Ultra	Electronics	68000.0	15
AirPods Pro 2	Electronics	8500.0	40
Winter Tire Set	Automotive	14000.0	19
Engine Oil 5W-30	Automotive	1200.0	50
Brake Pad	Automotive	850.0	4
Tesla Wall Connector	Automotive	18000.0	6
Office Chair Ergonomic	Furniture	3200.0	12
Bookshelf Walnut	Furniture	1800.0	8
Queen Size Bed	Furniture	15000.0	5
Olive Oil 5L	Grocery	1800.0	25
Basmati Rice 5kg	Grocery	450.0	100
Black Tea 1kg	Grocery	280.0	150
Honey Organic	Grocery	350.0	4
Orange Juice 1L	Drink	45.0	80

Şekil III Kullanıcı Paneli

- Rol bazlı erişim ile kullanıcının sadece satın alma işlemlerine odaklanması sağlanması
- Sade arayüz tasarımlı

2.3. Admin Paneli

The screenshot shows the Stock Automation System Admin Panel. At the top, it says "Stock Automation System - Logged in as: admin". On the left, there's a sidebar titled "New Product" with fields for "Product Name", "Select Category", "Price (TL)", and "Stock Quantity", followed by a purple "+ ADD PRODUCT" button. At the bottom of the sidebar are buttons for "View System Logs", "BUY", "EDIT", and "DELETE". The main area is titled "Product List" and contains a table with columns: Name, Category, Price (TL), and Stock. The table lists various products like Fiat Doblo, iPhone 15, Gaming Mouse, etc., with their respective details. The last few rows show grocery items like Olive Oil 5L, Basmati Rice 5kg, Black Tea 1kg, Honey Organic, Orange Juice 1L, and Mineral Water 6-Pack.

Name	Category	Price (TL)	Stock
Fiat Doblo	Automotive	850000.0	117
iPhone 15	Electronics	55000.0	10
Gaming Mouse	Electronics	450.0	9
iPhone 16 Pro	Electronics	75000.0	10
Water 0.5L	Drink	10.0	100
Dining Table	Furniture	4500.0	1
MacBook Pro M3	Electronics	95000.0	6
Samsung S24 Ultra	Electronics	68000.0	16
AirPods Pro 2	Electronics	8500.0	40
Winter Tire Set	Automotive	14000.0	19
Engine Oil 5W-30	Automotive	1200.0	50
Brake Pad	Automotive	850.0	4
Tesla Wall Connector	Automotive	18000.0	6
Office Chair Ergonomic	Furniture	3200.0	12
Bookshelf Walnut	Furniture	1800.0	8
Queen Size Bed	Furniture	15000.0	5
Olive Oil 5L	Grocery	1800.0	25
Basmati Rice 5kg	Grocery	450.0	100
Black Tea 1kg	Grocery	280.0	150
Honey Organic	Grocery	350.0	4
Orange Juice 1L	Drink	45.0	80
Mineral Water 6-Pack	Drink	60.0	200

Şekil IV Admin Paneli

- Sisteme Admin rolü ile giriş yapıldığında, standart kullanıcılardan farklı olarak tam yetkili bir yönetim katmanı aktif hale gelir
- Admin rolüne özel “EDIT”, “DELETE” butonları ile verilere manuel müdahale imkanı
- Yeni ürün formu ile ürün ekleme yetkisi
- “View System Logs” butonu üzerinden erişilen sistem logları ile uygulamada yapılan tüm işlemlerin geçmişe dönük takip ve denetim imkanı

2.4. Ürün Ekleme Paneli

New Product

Product Name

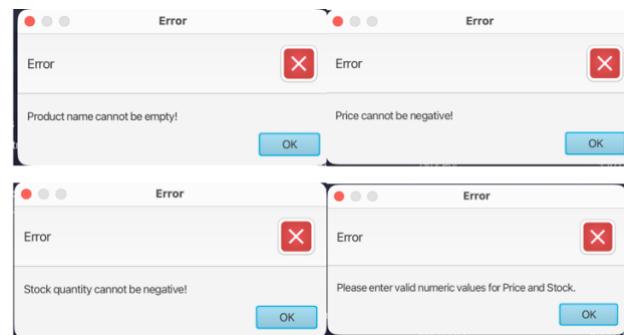
Select Category

Price (TL)

Stock Quantity

+ ADD PRODUCT

Şekil V Ürün Ekleme Paneli



Şekil VI Ürün Ekleme Paneli Hata Yönetimi

- Admin rolüne özel bir panel
- Ürün bilgilerinin doğruluğunu sağlamak için detaylı hata yakalama adımları

2.5. Ürün Düzenleme Ekranı

Edit Product

Name: Engine Oil 5W-30

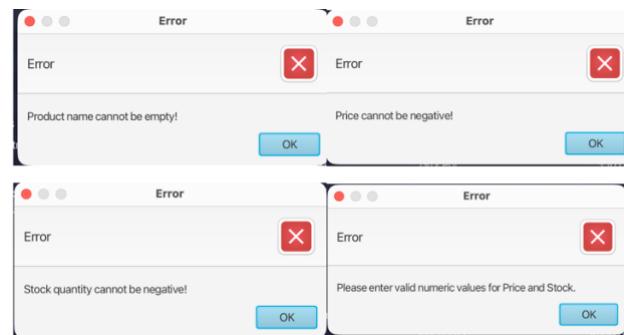
Category: Automotive

Price: 1200.0

Stock: 50

Cancel Save

Şekil VII Ürün Düzenleme Ekranı



Şekil VII Araç Düzenleme Ekranı Hata Yönetimi

- Admin rolüne özel bir panel
- Araç bilgilerinin doğruluğu sağlamak için detaylı hata yakalama adımları

2.6. İşlem Geçmişi Paneli

System Logs				
Filter by Type:		User	Action	Details
PURCHASE				
ADD	:47	admin	DELETE	Deleted: a
DELETE	:42	admin	ADD	Added: a
UPDATE	:38	admin	UPDATE	Updated: Water 0.5L to Wate...
2026-01-06 15:13:21		admin	DELETE	Deleted: Coffee Beans
2026-01-06 15:09:43		ali	PURCHASE	Bought 1 units of Dining Table
2025-12-20 03:03:55		admin	PURCHASE	Bought 1 of Winter Tire Set
2025-12-20 03:03:51		admin	DELETE	Deleted: L-Shaped Sofa
2025-12-20 03:03:47		admin	UPDATE	Updated: Water 0.5L to Wate...
2025-12-20 02:31:17		admin	UPDATE	Updated: iPhone 15 to iPhon...
2025-12-20 02:31:06		admin	PURCHASE	Bought 3 of iPhone 15
2025-12-20 02:28:00		admin	UPDATE	Updated: iPhone 16 to iPhon...
2025-12-20 02:27:22		ali	PURCHASE	Bought 1 of deneme ürün
2025-12-20 02:27:07		ali	PURCHASE	Bought 1 of Gaming Mouse
2025-12-20 02:04:01		ali	PURCHASE	Bought 1 of iPhone 15
2025-12-20 02:03:58		ali	PURCHASE	Bought 1 of deneme ürün

Şekil IX İşlem Geçmişi Paneli

- Başlangıç ve bitiş tarihlerini seçerek belirli dönemlere ait verileri anlık olarak görebilme
- Tarih, Kullanıcı adı, işlem ve detay gibi kritik bilgiler ile detaylı loglama
- İşlem kategorilerine göre filtreleme özelliği

3. Sistem mimarisi



Şekil X Sistem Mimarisi ve Dokümantasyonu

3.1. MainApp.java

Uygulamanın ana giriş noktasıdır; JavaFX çalışma zamanını başlatır, JsonService aracılığıyla ilk veri yüklemelerini gerçekleştirir ve kullanıcıyı karşılama ekranına (LoginView) yönlendirir.

3.2. controller/LoginController.java

Güvenlik katmanının ilk basamağıdır; kullanıcı giriş denemelerini doğrular, hatalı girişleri AuthenticationException ile yakalar ve yetki seviyesine göre (Admin/User) ilgili paneli yükler.

3.3. controller/MainController.java

Uygulamanın ana kontrol merkezidir; stok güncelleme, ürün ekleme/silme ve log görüntüleme gibi iş mantığı (business logic) süreçlerini koordine eder.

3.4. model/Stockable.java (Interface)

Stoklanabilir varlıklar için gerekli metod imzalarını (stok ekle/çıkar) içerir; iş mantığı katmanının farklı ürün türleri ile standart bir arayüz üzerinden konuşmasını sağlar.

3.5. model/Loggable.java (Interface)

Bir nesnenin loglanabilir olma yeteneğini tanımlar; polimorfizm sayesinde farklı sınıfların (Product, User vb.) ortak bir loglama mekanizmasına dahil edilmesini sağlar.

3.6. models/rental_history.py

Veritabanından çekilen geçmiş kiralama verilerini standart bir nesne formatına dönüştürerek, raporlama ve analiz modüllerinde tutarlı bir veri yapısı kullanılmasını sağlar.

3.7. model/BaseProduct.java (Abstract)

Tüm ürün nesneleri için ortak bir iskelet sunan soyut (abstract) sınıfı; kod tekrarını önleyerek sistemin temel veri şemasını belirler.

3.8. model/Product.java

BaseProduct sınıfından kalıtım alan nesne modelidir; stok düşürme, fiyat güncelleme ve log detaylarını oluşturma gibi nesneye özgü davranışları yönetir.

3.9. model/User.java

Sistem kullanıcılarını temsil eden modeldir; kullanıcı kimlik bilgilerini ve yetki kontrolü (admin/personel) için gerekli olan rol tanımlamalarını nesne tabanlı yapıda tutar.

3.10. service/JsonService.java

Uygulamanın Veri Erişim Katmanı (Data Access Layer) olarak görev yapar; Jackson kütüphanesini kullanarak ham JSON verileri ile Java nesne modelleri arasında çift yönlü köprü kurar.

3.11. resources/login-view.fxml

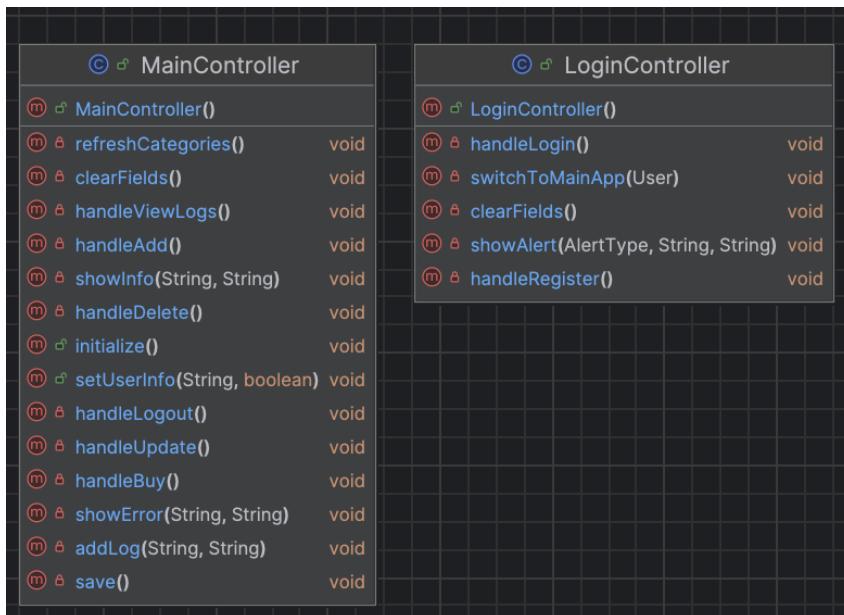
Uygulamanın giriş kapısıdır; LoginController ile bağlı olan bu dosya, kullanıcı adı ve şifre giriş alanlarını, kayıt ol butonunu ve kurumsal logoyu barındırır. Grid-pane yapısı kullanılarak bileşenlerin farklı ekran çözünürlüklerinde hizalı kalması sağlanmıştır.

3.12. resources/main-view.fxml

Uygulamanın ana dashboard (kontrol paneli) tasarımidır. Ürün tablosu (TableView), stok yönetim butonları ve admin araçlarını tek bir pencerede birleştirir. FXML mimarisi sayesinde arayüz tasarıımı kod mantığından (Controller) tamamen izole edilmiştir.

4. Örnek Kodlar ve UML Diyagramları

4.1. Controller



Şekil XI Controller UML Diyagramları

```
@FXML & @Inject
private void handleRegister() {
    String user = usernameField.getText().trim();
    String pass = passwordField.getText().trim();

    try {
        if (user.isEmpty() || pass.isEmpty()) {
            throw new InvalidProductException("Username or Password cannot be empty!");
        }
    } catch (InvalidProductException e) {
        // Verilen kullanıcı adı ve şifre geçerli değilse hata mesajı
        // verilecektir.
    }

    for (User u : registeredUsers) {
        if (u.getUsername().equalsIgnoreCase(user)) {
            // Eğer kullanıcı varsa hatalı fikret!
            throw new UserAlreadyExistsException(user);
        }
    }

    registeredUsers.add(new User(user, pass, isAdmin: false));
    showAlert(Alert.AlertType.INFORMATION, title: "Success", content: "Registration successful!");
    clearFields();
}

} catch (UserAlreadyExistsException | InvalidProductException e) {
    // Birde hatalı kullanıcı veya blokta yakalandı (Multi-catch)
    showAlert(Alert.AlertType.ERROR, title: "Registration Error", e.getMessage());
}
}
```

```
@FXML & @Inject
private void handleLogin() {
    String user = usernameField.getText().trim();
    String pass = passwordField.getText().trim();

    try {
        if (user.isEmpty() || pass.isEmpty()) {
            throw new AuthenticationException("Username or Password cannot be empty!");
        }
    } catch (AuthenticationException e) {
        // Kullanıcı adı ve şifresinden söz konusu hata mesajı gösterilecektir.
        showAlert(Alert.AlertType.ERROR, title: "Authentication Failed", e.getMessage());
    }

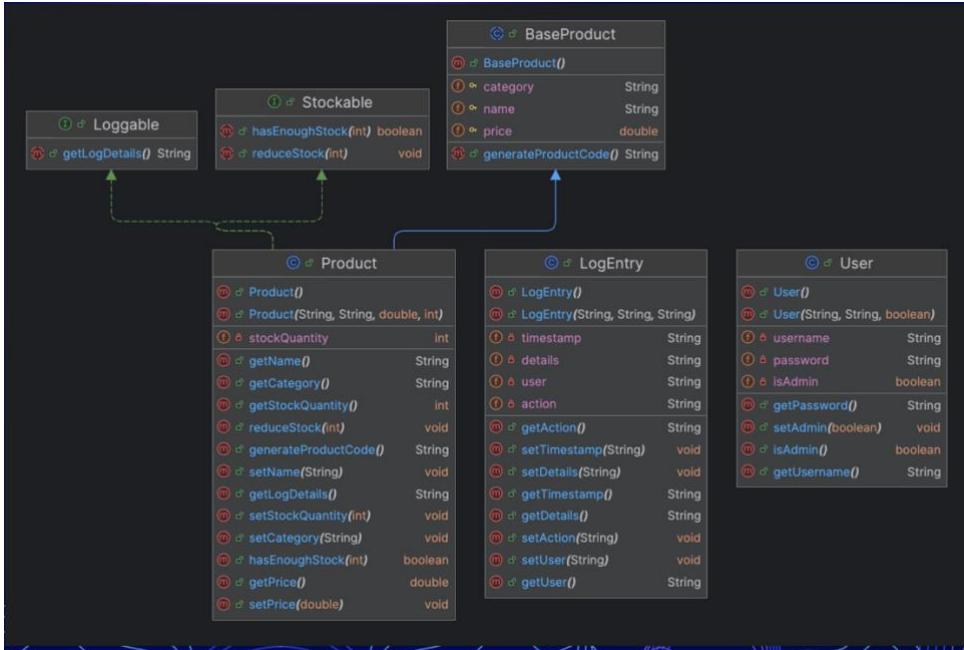
    boolean authenticated = false;
    for (User u : registeredUsers) {
        if (u.getUsername().equals(user) && u.getPassword().equals(pass)) {
            authenticated = true;
            switchToMainApp(u);
            break;
        }
    }

    if (!authenticated) {
        // Hatalı bilgiler! uygun mesaj hatalı fikretin!
        throw new AuthenticationException("Invalid username or password!");
    }
}

} catch (AuthenticationException e) {
    // Kullanıcı adı ve şifresinden söz konusu hata mesajı gösterilecektir.
    showAlert(Alert.AlertType.ERROR, title: "Authentication Failed", e.getMessage());
} catch (IOException e) {
    // Dosya yüklenmesi başarısız (Checked Exception)
    showAlert(Alert.AlertType.ERROR, title: "System Error", content: "Main screen could not be loaded!");
}
}
```

Şekil XII LoginController.java içerisindeki kayıt ve giriş işlemlerini yakalayan iki metot örneği

4.2. Model



Şekil XIII Model UML Diyagramları

```

public class LogEntry { 15 usages & alitalhq
    private String timestamp; 3 usages
    private String user; 3 usages
    private String action; 3 usages
    private String details; 3 usages

    public LogEntry(String user, String action, String details) { 2 usages & alitalhq

        this.timestamp = LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
        this.user = user;
        this.action = action;
        this.details = details;
    }

    public LogEntry() { 1 usage & alitalhq
    }

    public String getTimestamp() { return timestamp; }

    public String getUser() { return user; }

    public String getAction() { return action; }

    public String getDetails() { return details; }

    public void setTimestamp(String timestamp) { this.timestamp = timestamp; }

    public void setUser(String user) { this.user = user; }

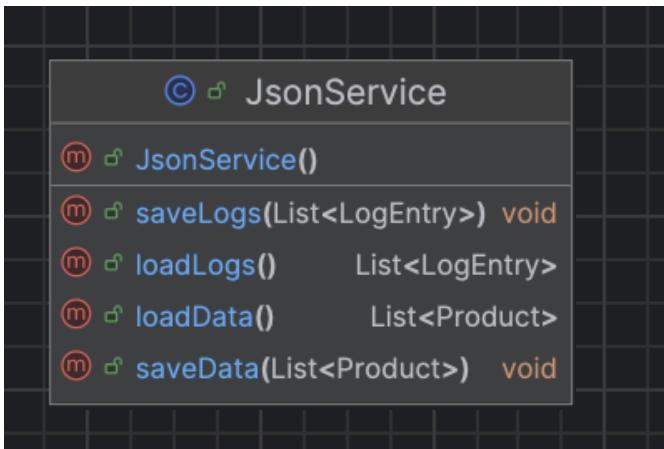
    public void setAction(String action) { this.action = action; }

    public void setDetails(String details) { this.details = details; }
}

```

Şekil XIV LogEntry.java modelinin içeriği

4.3. Service



Şekil XV Service UML Diyagramları

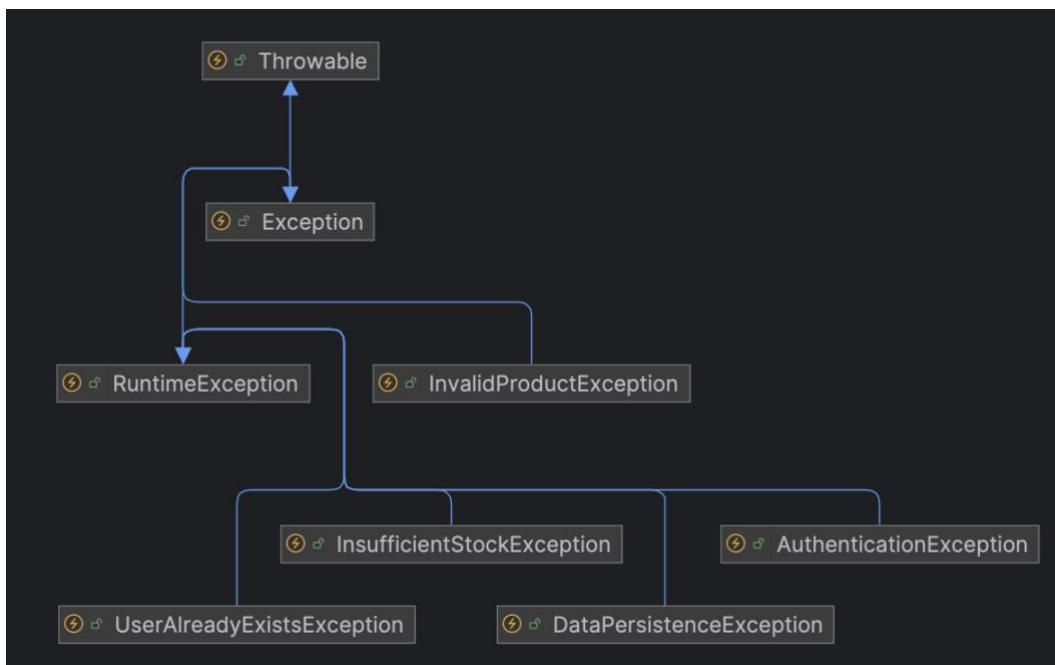
```
public List<Product> loadData() { 1 usage & alitalhq
    try {
        File file = new File(PRODUCT_FILE);
        if (!file.exists()) return new ArrayList<>();
        return mapper.readValue(file, new TypeReference<List<Product>>() {});
    } catch (IOException e) {
        e.printStackTrace();
        return new ArrayList<>();
    }
}

public void saveLogs(List<LogEntry> logs) { 1 usage & alitalhq
    try {
        mapper.writeValue(new File(LOG_FILE), logs);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public List<LogEntry> loadLogs() { 1 usage & alitalhq
    try {
        File file = new File(LOG_FILE);
        if (!file.exists()) return new ArrayList<>();
        return mapper.readValue(file, new TypeReference<List<LogEntry>>() {});
    } catch (IOException e) {
        e.printStackTrace();
        return new ArrayList<>();
    }
}
```

Şekil XVI JsonService.java içerisinde log yazmayı ve okumayı sağlayan ve ürün eklemeyi sağlayan metod örneği

4.4. Exception



Şekil XVII Exception UML Diyagramları

```
package com.example.stockautomationsystem.exception;
|
//yetersiz stok hatası
public class InsufficientStockException extends RuntimeException { 2 usages & alitalhq
    public InsufficientStockException(String productName, int requested, int available) { 2 usages & alitalhq
        super("Insufficient stock for " + productName +
              ". Requested: " + requested + ", Available: " + available);
    }
}
```

Şekil XVIII InsufficientStockException.java hatasının içeriği

5. Kaynakça

JavaFX Documentation & UI Design Principles. Oracle Corporation. <https://openjfx.io/openjfx-docs/>

Jackson Project Home & Documentation. FasterXML. <https://github.com/FasterXML/jackson-docs>

Apache Maven Project. The Apache Software Foundation. <https://maven.apache.org/>

Java Language Specification, Java SE 17 Edition. Oracle Corporation.

<https://docs.oracle.com/javase/specs/>