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

Bu proje, işletmelerin envanter yönetim süreçlerini dijitalleştirmek 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.

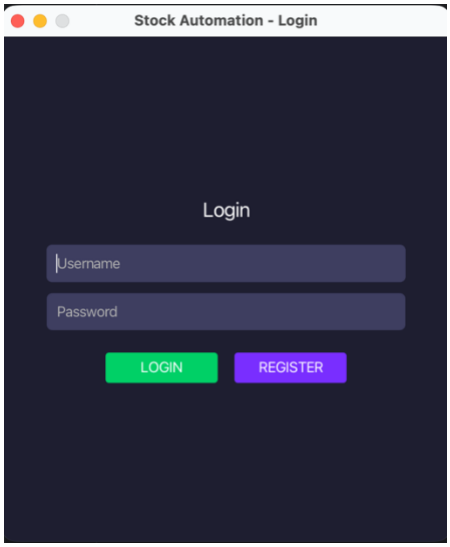
## 1. Kullanılan Teknolojiler

*Tablo 1 Kullanılan Teknolojiler Tablosu*

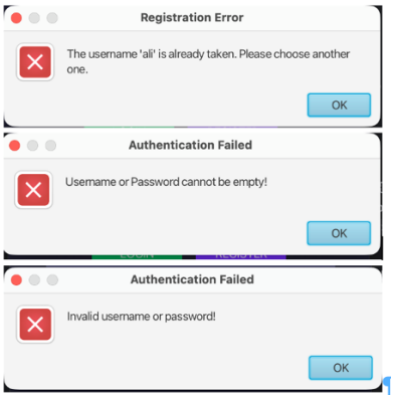
| Kategori | Teknoloji | Seçilme Nedeni |
|---|---|---|
| Programlama Dili | Java 17 (LTS) | Güçlü OOP desteği, tip güvenliği ve kurumsal uygulama standartları. |
| Arayüz Teknolojisi | JavaFX | XML tabanlı (FXML) arayüz tasarımı ve CSS ile özelleştirilebilir modern görünüm. |
| Veri Yönetimi | JSON (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:



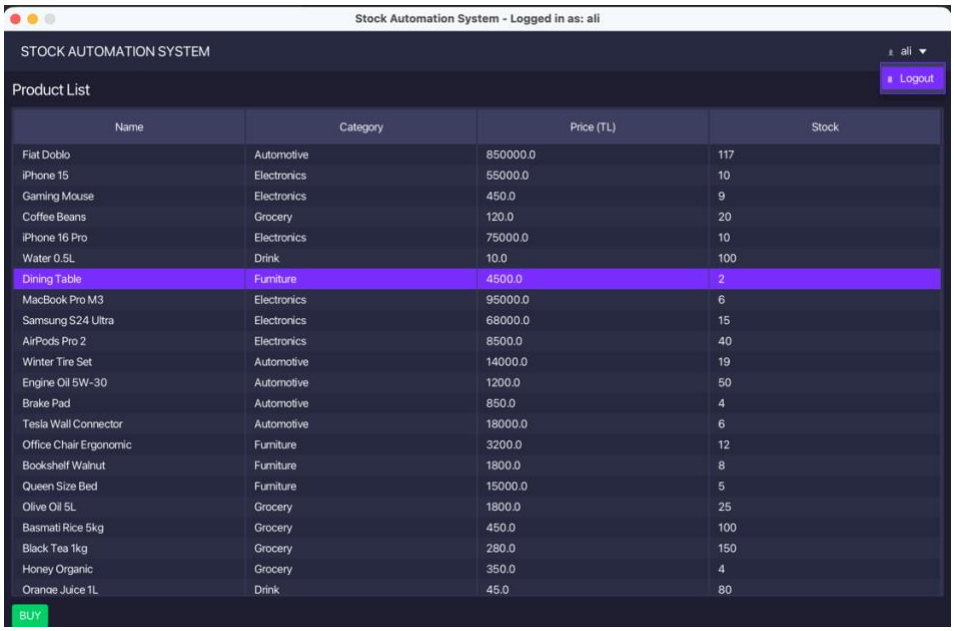

*Şekil I Giriş - Kayıt Ekranı*




*Şekil II Giriş – Kayıt Ekranı Hata Yönetimi*

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

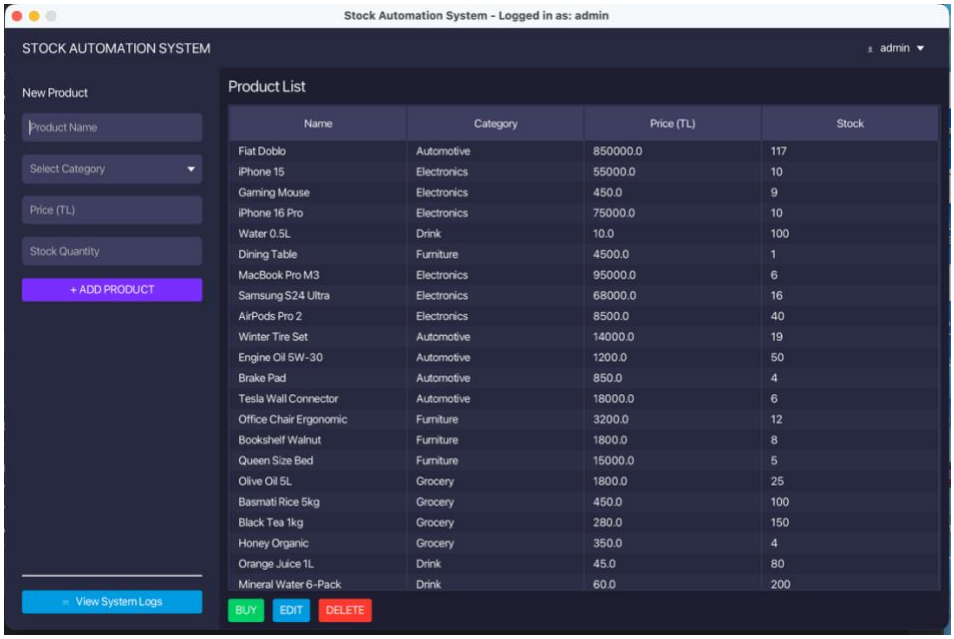
## 2.2. Kullanıcı Paneli:




*Şekil III Kullanıcı Paneli*

- Rol bazlı erişim ile kullanıcının sadece satın alma işlemlerine odaklanmasının sağlanması
- Sade arayüz tasarımı

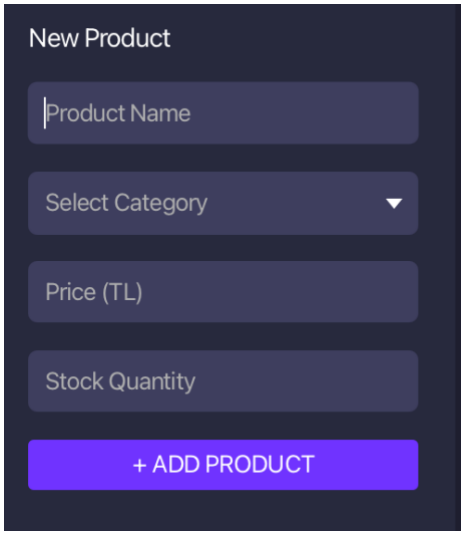
### 2.3. Admin Paneli



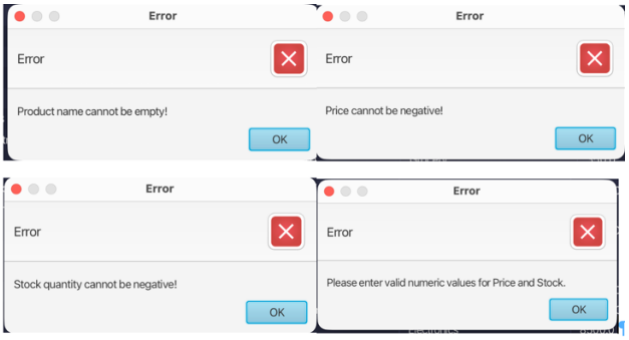

*Şekil IV Admin Paneli*

- 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



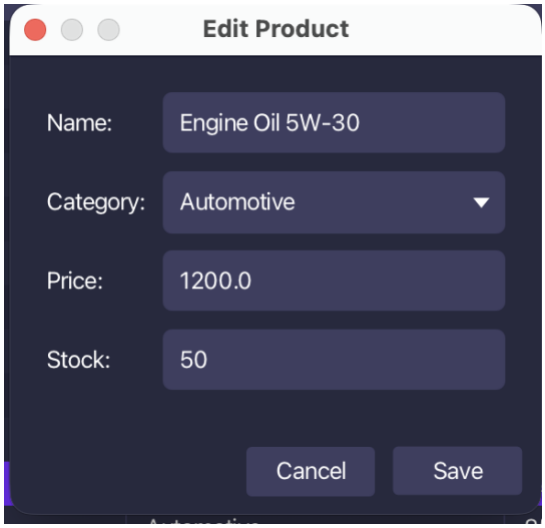

*Şekil V Ürün Ekleme Paneli*



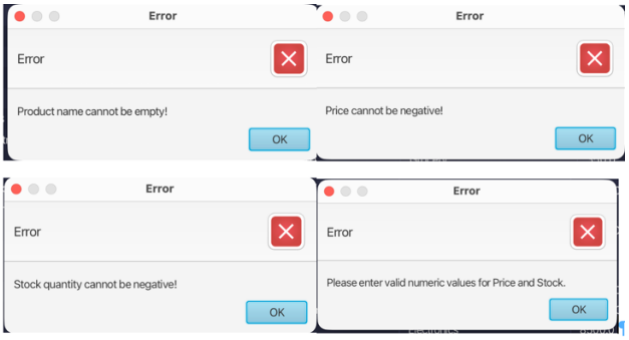

*Şekil VI Ürün Ekleme Paneli Hata Yönetimi*

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

### 2.5. Ürün Düzenleme Ekranı




*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


*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

```
src/
├── main/
│   ├── java/com/example/stockautomationsystem/
│   │   ├── controller/
│   │   │   ├── LoginController.java
│   │   │   └── MainController.java
│   │   ├── exception/
│   │   │   ├── AuthenticationException.java
│   │   │   ├── DataPersistenceException.java
│   │   │   ├── InsufficientStockException.java
│   │   │   ├── InvalidProductException.java
│   │   │   └── UserAlreadyExistsException.java
│   │   ├── model/
│   │   │   ├── BaseProduct.java
│   │   │   ├── LogEntry.java
│   │   │   ├── Loggable.java
│   │   │   ├── Product.java
│   │   │   ├── Stockable.java
│   │   │   └── User.java
│   │   ├── service/
│   │   │   └── JsonService.java
│   │   └── MainApp.java
│   └── resources/
│       ├── login-view.fxml
│       ├── main-view.fxml
│       └── style.css
```

*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) sınıftır; 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

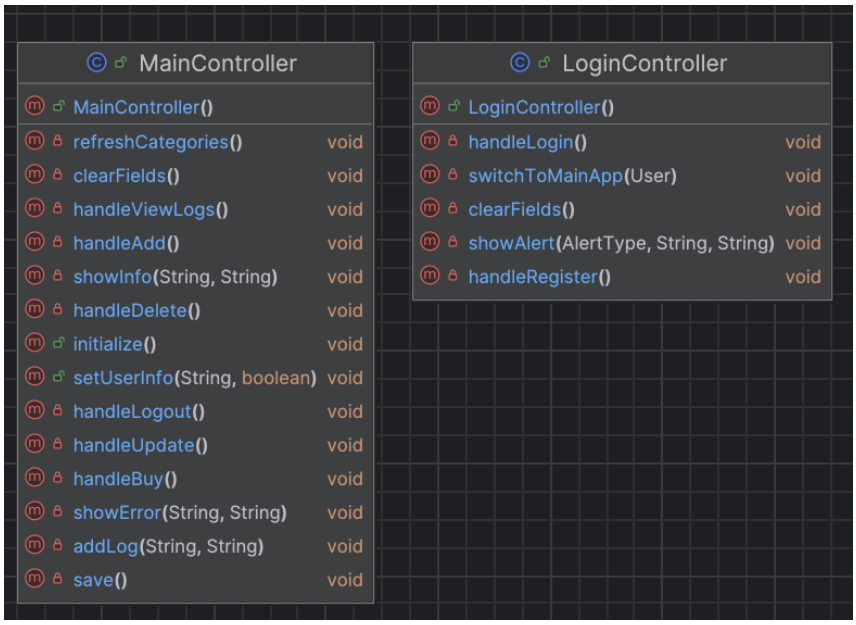
Uygulamanın giriş kapısıdır; LoginController ile bağlı olan bu dosya, kullanıcı adı ve şifre giriş alanlarını, kayıt ol butonunu ve kurumsal logoyu barındırır. Grid-pane yapısı kullanılarak bileşenlerin farklı ekran çözünürlüklerinde hizalı kalması sağlanmıştır.

### 3.12. resources/main-view.fxml

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

## 4. Örnek Kodlar ve UML Diyagramları

### 4.1. Controller



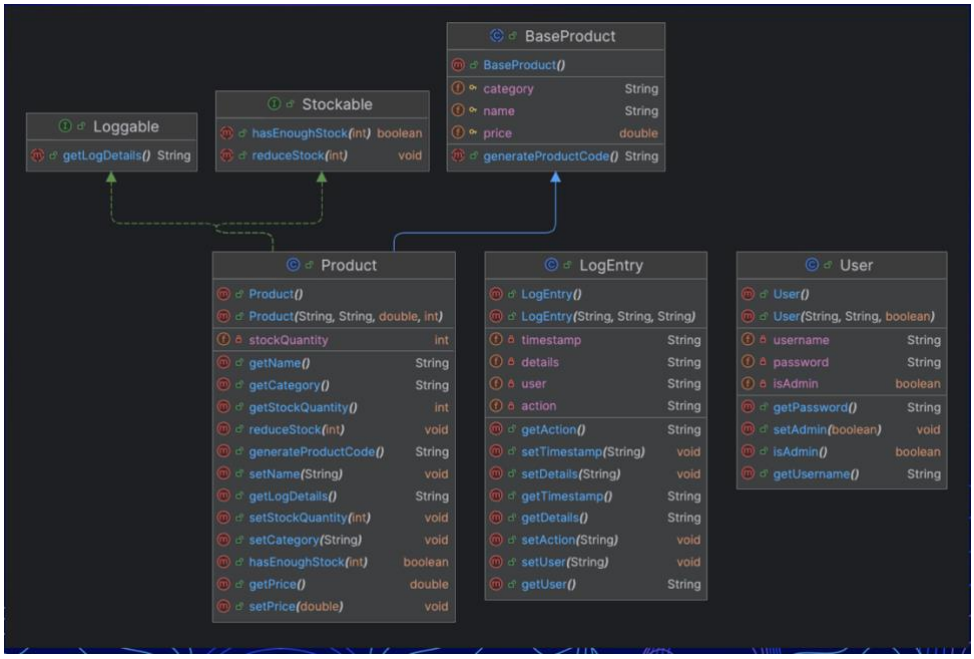

*Şekil XI Controller UML Diyagramları*

```
@FXML
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!");
            // Not: InvalidProduct yerine daha genel bir InvalidInputException da yazılabilir
        }

        for (User u : registeredUsers) {
            if (u.getUsername().equalsIgnoreCase(user)) {
                // Eğer kullanıcı varsa hatayı fırlatıyoruz
                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) {
        // Birden fazla hatayı aynı blokta yakalama (Multi-catch)
        showAlert(Alert.AlertType.ERROR, title: "Registration Error", e.getMessage());
    }
}
```

```
@FXML
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!");
        }

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

        if (!authenticated) {
            // Kimlik bilgileri uyuşmazsa hatayı fırlatıyoruz
            throw new AuthenticationException("Invalid username or password!");
        }

    } catch (AuthenticationException e) {
        // Kendi hata sınıfımızdan gelen mesajı gösteriyoruz
        showAlert(Alert.AlertType.ERROR, title: "Authentication Failed", e.getMessage());
    } catch (IOException e) {
        // Dosya yükleme hatası (Checked Exception)
        showAlert(Alert.AlertType.ERROR, title: "System Error", content: "Main screen could not be loaded!");
    }
}
```

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

## 4.2. Model




*Şekil XIII Model UML Diyagramları*

```java
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*

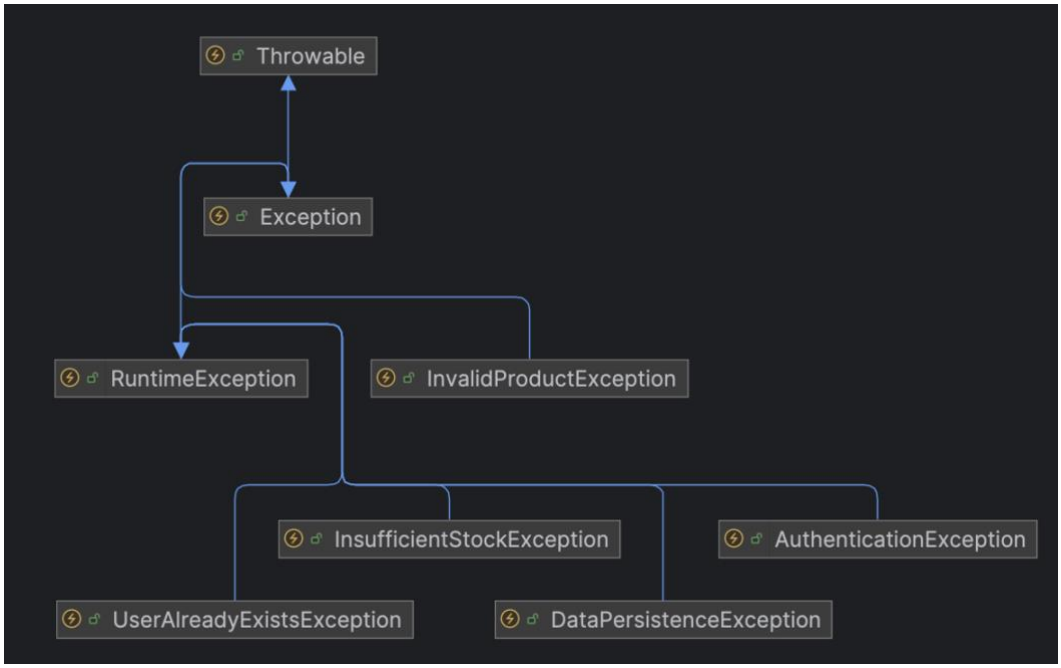
### 4.3. Service


*Şekil XV Service UML Diyagramları*

```java
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>>() {});  alitalhq
    } 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>>() {});  alitalhq
    } catch (IOException e) {
        e.printStackTrace();
        return new ArrayList<>();
    }
}
```

*Şekil XVI JsonService.java içerisinden log yazmayı ve okumayı sağlayan ve ürün eklemeyi sağlayan metot örneği*

## 4.4. Exception




*Şekil XVII Exception UML Diyagramları*

```java
package com.example.stockautomationsystem.exception;

//yetersiz stok hatasi
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/