문제1 : 예약결제와 환불처리에 대해, API 기능 정의하고 실패 시, 보상트랜잭션 정리

1. 예약결제

순서	동기/비동기	API		필요 보상 트랜잭션		트랜잭션 종류
		서비스	기능명	서비스	기능명	
1	동기	주문 서비스	주문 조회			
2	비동기	주문 서비스	주문 생성	주문 서비스	주문 취소	보상 트랜잭션
3	동기	포인트 서비스	포인트 조회			
4	비동기	포인트 서비스	포인트 사용	포인트 서비스	포인트 사용 취소	보상 트랜잭션
5	동기	결제 서비스	외부 업체 결제 요청			피봇 트랜잭션
6	비동기	포인트 서비스	포인트 적립			재시도 가능 트랜잭션
7	비동기	주문 서비스	주문 완료			재시도 가능 트랜잭션
8	비동기	공통 서비스	이메일 발송			재시도 가능 트랜잭션

문제1 : 예약결제와 환불처리에 대해, API 기능 정의하고 실패 시, 보상트랜잭션 정리

2. 환불처리

순서	동기/비동기	API		필요 보상 트랜잭션		트랜잭션 종류
		서비스	기능명	서비스	기능명	
1	비동기	환불 서비스	환불 요청	환불 서비스	환불 취소	보상 트랜잭션
2	비동기	주문 서비스	주문 취소			
3	비동기	포인트 서비스	포인트 사용 취소			
4	동기	결제 서비스	외부 업체 결제 요청(취소)			피봇 트랜잭션
5	비동기	포인트 서비스	포인트 적립 취소			재시도 가능 트랜잭션
6	비동기	주문 서비스	주문 취소 완료			재시도 가능 트랜잭션
7	비동기	공통 서비스	이메일 발송			재시도 가능 트랜잭션

■ 업무용 응용프로그램 개발 방안

- A고객사 시스템의 채널 확대 및 사용자 증가에 따라 발생하는 인증 및 세션 관리 문제 해결을 위한 인증 방식 개선 방안

1. OAuth 2.0과 OpenID Connect 도입
OAuth 2.0과 OpenID Connect(OIDC)를 도입하여 인증 및 권한 관리를 분리하는 방식은 사용자의 인증과 애플리케이션의 권한 부여를 더 안전하고 간편하게 관리할 수 있습니다.

OAuth 2.0: 권한 부여 프레임워크로, 클라이언트(애플리케이션)가 자원 소유자(사용자)를 대신하여 자원 서버(예: API 서버)에 접근할 수 있도록 합니다.
OpenID Connect: OAuth 2.0 위에 구축된 인증 프로토콜로, 사용자 인증을 처리하고 사용자 정보를 안전하게 공유할 수 있습니다.
2. 복수 인증 단계(Multi-Factor Authentication, MFA)
MFA를 추가하여 보안을 강화할 수 있습니다. 이는 특히 중요한 사용자나 민감한 데이터를 다룰 때 매우 유용합니다. 예를 들어, SMS, 이메일, 또는 전용 MFA 앱을 통한 코드를 추가 입력하게 할 수 있습니다.

3. 세션 관리 개선
세션 관리는 특히 사용자 증가 시 중요한 이슈입니다. 다음과 같은 방안을 고려할 수 있습니다:

세션 타임아웃(Timeouts): 적절한 세션 타임아웃을 설정하여 오래된 세션을 자동으로 종료하는 방식.
세션 스토리지: 분산 세션 저장소(예: Redis, Memcached)를 사용하여 세션 데이터를 중앙에서 관리하므로, 여러 서버 간에 세션 상태를 일관되게 유지할 수 있습니다.
4. 토큰 기반 인증
토큰 기반 인증 방법을 활용하여 세션 관리 문제를 해결합니다.

JWT(JSON Web Tokens): 사용자의 상태를 서버에 저장하지 않는 stateless한 방식으로, 토큰을 클라이언트 측에 저장하고 주고 받음으로써 인증 상태를 관리할 수 있습니다.
토큰 만료 및 갱신: 액세스 토큰과 리프레시 토큰을 사용하여 토큰의 유효 기간을 관리하고, 일정 시간 후 또는 액세스 토큰이 만료될 때 새로운 토큰으로 갱신.
5. 중앙 인증 서비스(Identity Provider, IdP)
SAML, OAuth 등 표준 프로토콜을 사용하는 중앙 인증 서비스를 도입하여, 여러 채널에서 일관된 인증 및 권한 부여를 관리할 수 있습니다.

사례: Keycloak, Okta, Auth0 등의 IdP를 활용하여 사회적 로그인(Social Login)과 같은 기능을 제공하고 사용자의 편의성을 높일 수 있습니다.
6. 비밀번호 정책 강화
비밀번호 정책을 강화하여 보안을 더욱 단단히 합니다.

비밀번호 복잡성 요구: 대문자, 소문자, 특수문자, 숫자를 포함한 복잡한 비밀번호 사용.

■ 업무용 응용프로그램 개발 방안

- A고객사 비즈니스 유연성과 성능 관점에서 상품을 관리하기 위한 데이터 모델을 새롭게 설계하고 설계 사유 제시

데이터 모델 설계
1. 제품 테이블 (Products)
sql

```
CREATE TABLE Products (
    ProductID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255) NOT NULL,
    Description TEXT,
    CategoryID INT,
    Price DECIMAL(10, 2) NOT NULL,
    Stock INT NOT NULL DEFAULT 0,
    CreatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UpdatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
);
```

2. 카테고리 테이블 (Categories)
sql

```
CREATE TABLE Categories (
    CategoryID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255) NOT NULL,
    Description TEXT
);
```

3. 가격 히스토리 테이블 (PriceHistory)
sql

```
CREATE TABLE PriceHistory (
```

■ 업무용 응용프로그램 개발 방안

- 예약 처리 프로세스와 데이터 모델에서 발생하고 있는 동시성 이슈의 해결 방안 제시

예약 처리 시스템에서 동시성(Concurrency) 이슈는 중요한 문제입니다. 이는 다수의 사용자가 동일한 자원을 동시에 예약하려고 시도할 때 발생할 수 있습니다. 이러한 문제를 효과적으로 해결하기 위한 방법을 몇 가지 제안합니다.

1. 데이터 모델 개선
동시성 문제를 해결하기 위해 데이터 모델을 적절히 설계하는 것이 필수적입니다. 이때 Locking Mechanism을 포함하여 동시성 제어를 강화할 수 있습니다.

테이블 구조 예시:
sql

```
CREATE TABLE Reservations (
    ReservationID INT PRIMARY KEY AUTO_INCREMENT,
    ResourceID INT NOT NULL,
    UserID INT NOT NULL,
    StartTime DATETIME NOT NULL,
    EndTime DATETIME NOT NULL,
    Status ENUM('PENDING', 'CONFIRMED', 'CANCELLED') DEFAULT 'PENDING',
    CreatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UpdatedAt TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    CONSTRAINT UC_ResourceBooking UNIQUE (ResourceID, StartTime, EndTime)
);
```

이 구조에서 ResourceID, StartTime, EndTime의 고유 제약조건(Unique Constraint)이 겹치는 예약을 방지합니다.

2. 동시성 제어 기법
2.1. 비관적 잠금(Pessimistic Locking)
비관적 잠금은 한 트랜잭션이 자원을 잠글 때 다른 트랜잭션이 해당 자원에 접근하지 못하도록 막는 방식입니다. 이는 동시성 문제가 자주 발생할 가능성이 높을 때 유효합니