```java
package com.apex.eqp.inventory.repositories;

import com.apex.eqp.inventory.entities.Product;
import com.apex.eqp.inventory.entities.RecalledProduct;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import javax.persistence.EntityManager;
import javax.persistence.criteria.*;
import java.util.ArrayList;
import java.util.List;

@Repository
public class InventoryRepositoryCustomImpl implements InventoryRepositoryCustom {
    EntityManager entityManager;

    @Autowired
    public InventoryRepositoryCustomImpl(EntityManager entityManager) {
        this.entityManager = entityManager;
    }

    @Override
    public List<Product> findByName(String name) {
        CriteriaBuilder criteriaBuilder = entityManager.getCriteriaBuilder();
        CriteriaQuery<Product> query = criteriaBuilder.createQuery(Product.class);

        Root<Product> root = query.from(Product.class);
        List<Predicate> predicates = new ArrayList<>();

        if (name != null) {
            predicates.add(criteriaBuilder.like(root.get("name"), "%" + name.strip() +
"%"));
        }
        query.where(predicates.toArray(new Predicate[0]));

        return entityManager.createQuery(query).getResultList();
    }

    @Override
    public List<Product> findAllActive() {
        CriteriaBuilder criteriaBuilder = entityManager.getCriteriaBuilder();
        CriteriaQuery<Product> query = criteriaBuilder.createQuery(Product.class);

        Root<Product> productRoot = query.from(Product.class);

        List<Predicate> predicates = new ArrayList<>();

        Subquery<RecalledProduct> subquery = query.subquery(RecalledProduct.class);
        Root<RecalledProduct> recalledProductRoot = subquery.from(RecalledProduct.class
);
        subquery.select(recalledProductRoot);

        predicates.add(
                criteriaBuilder.equal(
                        recalledProductRoot.get("name"),
                        productRoot.get("name")
                )
        );
```

```
        subquery.where(predicates.toArray(new Predicate[0]));
        query.where(criteriaBuilder.not(criteriaBuilder.exists(subquery)));

        return entityManager.createQuery(query).getResultList();
    }
}
```