Repository:

_____

1) CheckoutRepository:

```
package com.ArtGalleryManagement.Backend.Repository;

import com.ArtGalleryManagement.Backend.Entity.Checkout;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import java.util.List;

public interface CheckoutRepository extends JpaRepository<Checkout, Long> {

    Logger logger = LoggerFactory.getLogger(CheckoutRepository.class);

    Checkout findByUserEmailAndProductId(String userEmail, Long productId);

    List<Checkout> findProductsByUserEmail(String userEmail);

    @Modifying
    @Query("delete from Checkout where product_id in :product_id")
    void deleteAllByProductId(@Param("product_id") Long productId);

    // Add logging statements
    default Checkout findByUserEmailAndProductIdWithLogging(String userEmail, Long productId) {
        logger.info("Searching for Checkout by userEmail: {} and productId: {}", userEmail, productId);
        return findByUserEmailAndProductId(userEmail, productId);
    }

    // Add logging statements
    default List<Checkout> findProductsByUserEmailWithLogging(String userEmail) {
        logger.info("Searching for Checkouts by userEmail: {}", userEmail);
        return findProductsByUserEmail(userEmail);
    }

    // Add logging statements
    @Modifying
    @Query("delete from Checkout where product_id in :product_id")
    void deleteAllByProductIdWithLogging(@Param("product_id") Long productId) {
        logger.info("Deleting Checkout by productId: {}", productId);
        deleteAllByProductId(productId);
    }
}
```

_____

2) History Repository:

```
package com.ArtGalleryManagement.Backend.Repository;

import com.ArtGalleryManagement.Backend.Entity.History;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.web.bind.annotation.RequestParam;

public interface HistoryRepository extends JpaRepository<History, Long> {

    Logger logger = LoggerFactory.getLogger(HistoryRepository.class);

    Page<History> findProductsByUserEmail(@RequestParam("email") String userEmail, Pageable pagea
ble);

    default Page<History> findProductsByUserEmailWithLogging(@RequestParam("email") String userEm
ail, Pageable pageable) {
        logger.info("Searching for History by userEmail: {}", userEmail);
        return findProductsByUserEmail(userEmail, pageable);
    }
}
```

_____

3) Message Repository :

```
package com.ArtGalleryManagement.Backend.Repository;

import com.ArtGalleryManagement.Backend.Entity.Message;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.web.bind.annotation.RequestParam;

public interface MessageRepository extends JpaRepository<Message, Long> {

    Logger logger = LoggerFactory.getLogger(MessageRepository.class);

    Page<Message> findByUserEmail(@RequestParam("user_email") String userEmail, Pageable pageab
le);

    Page<Message> findByClosed(@RequestParam("closed") boolean closed, Pageable pageable);
```

```
    // Add logging statements
    default Page<Message> findByUserEmailWithLogging(@RequestParam("user_email") String userEma
il, Pageable pageable) {
        logger.info("Searching for Messages by userEmail: {}", userEmail);
        return findByUserEmail(userEmail, pageable);
    }

    // Add logging statements
    default Page<Message> findByClosedWithLogging(@RequestParam("closed") boolean closed, Pagea
ble pageable) {
        logger.info("Searching for Messages by closed status: {}", closed);
        return findByClosed(closed, pageable);
    }
}
```

---

4)Payment Repository:

```
package com.ArtGalleryManagement.Backend.Repository;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.data.jpa.repository.JpaRepository;

import com.ArtGalleryManagement.Backend.Entity.Payment;

public interface PaymentRepository extends JpaRepository<Payment, Long> {

    Logger logger = LoggerFactory.getLogger(PaymentRepository.class);

    Payment findByUserEmail(String userEmail);

    // Add logging statements
    default Payment findByUserEmailWithLogging(String userEmail) {
        logger.info("Searching for Payment by userEmail: {}", userEmail);
        return findByUserEmail(userEmail);
    }
}
```

---

5)Product Repository:

```
package com.ArtGalleryManagement.Backend.Repository;

import com.ArtGalleryManagement.Backend.Entity.Product;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
```

```
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.web.bind.annotation.RequestParam;

import java.util.List;

public interface ProductRepository extends JpaRepository<Product, Long> {

    Logger logger = LoggerFactory.getLogger(ProductRepository.class);

    Page<Product> findByTitleContaining(@RequestParam("title") String title, Pageable pageable);

    Page<Product> findByCategory(@RequestParam("category") String category, Pageable pageable);

    @Query("select o from Product o where id in :product_ids")
    List<Product> findProductsByProductIds(@Param("product_ids") List<Long> productId);

    // Add logging statements
    default Page<Product> findByTitleContainingWithLogging(String title, Pageable pageable) {
        logger.info("Searching for Products by title containing: {}", title);
        return findByTitleContaining(title, pageable);
    }

    // Add logging statements
    default Page<Product> findByCategoryWithLogging(String category, Pageable pageable) {
        logger.info("Searching for Products by category: {}", category);
        return findByCategory(category, pageable);
    }

    // Add logging statements
    default List<Product> findProductsByProductIdsWithLogging(List<Long> productIds) {
        logger.info("Searching for Products by product IDs: {}", productIds);
        return findProductsByProductIds(productIds);
    }
}
```

---

6)

```
package com.ArtGalleryManagement.Backend.Repository;

import com.ArtGalleryManagement.Backend.Entity.Review;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.web.bind.annotation.RequestParam;

public interface ReviewRepository extends JpaRepository<Review, Long> {

    Logger logger = LoggerFactory.getLogger(ReviewRepository.class);
```

```
    Page<Review> findByProductId(@RequestParam("product_id") Long productId, Pageable pageable);

    Review findByUserEmailAndProductId(String userEmail, Long productId);

    @Modifying
    @Query("delete from Review where product_id in :product_id")
    void deleteAllByProductId(@Param("product_id") Long productId);

    // Add logging statements
    default Page<Review> findByProductIdWithLogging(Long productId, Pageable pageable) {
        logger.info("Searching for Reviews by product ID: {}", productId);
        return findByProductId(productId, pageable);
    }

    // Add logging statements
    default Review findByUserEmailAndProductIdWithLogging(String userEmail, Long productId) {
        logger.info("Searching for Review by user email: {} and product ID: {}", userEmail, productId);
        return findByUserEmailAndProductId(userEmail, productId);
    }
}
```

--LOVE YOU JAAANU --