Controllers:

1.Admin Controller:

-->pom.xml:

```
<dependencies>
    <!-- Spring Boot Starter -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter</artifactId>
    </dependency>

    <!-- SLF4J Logging API -->
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
        <version>1.7.32</version>
    </dependency>

    <!-- Logback Logging Implementation -->
    <dependency>
        <groupId>ch.qos.logback</groupId>
        <artifactId>logback-classic</artifactId>
        <version>1.2.6</version>
    </dependency>

    <!-- Other Dependencies -->
    <!-- ... -->
</dependencies>
```

--> build.gradle:

```
dependencies {
    // Spring Boot Starter
    implementation ’org.springframework.boot:spring-boot-starter’

    // SLF4J Logging API
    implementation ’org.slf4j:slf4j-api:1.7.32’

    // Logback Logging Implementation
    implementation ’ch.qos.logback:logback-classic:1.2.6’

    // Other Dependencies
    // ...
}
```

-->art-gallery-logback.xml:

```
<configuration>
```

```
    <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>
        </encoder>
    </appender>

    <root level="info">
        <appender-ref ref="CONSOLE" />
    </root>
</configuration>
```

-->AdminController:

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

// ...

@RestController
@RequestMapping("/admin")
public class AdminController {

    private static final Logger logger = LoggerFactory.getLogger(AdminController.class);

    private AdminServiceImpl adminService;

    @Autowired
    public AdminController(AdminServiceImpl adminService) {
        this.adminService = adminService;
    }

    @PutMapping("/secure/increase/product/quantity")
    public void increaseProductQuantity(@RequestParam Long productId) throws Exception {
        String admin = "admin";
        if (admin == null || !admin.equals("admin")) {
            logger.warn("Unauthorized access to increaseProductQuantity endpoint");
            throw new MethodNotAllowedException();
        }
        logger.info("Increasing product quantity for productId: {}", productId);
        adminService.increaseProductQuantity(productId);
    }

    @PutMapping("/secure/decrease/product/quantity")
    public void decreaseProductQuantity(@RequestParam Long productId) throws Exception {
        String admin ="admin";
        if (admin == null || !admin.equals("admin")) {
            logger.warn("Unauthorized access to decreaseProductQuantity endpoint");
            throw new MethodNotAllowedException();
        }
        logger.info("Decreasing product quantity for productId: {}", productId);
        adminService.decreaseProductQuantity(productId);
    }

    @PostMapping("/secure/add/product")
    public void postProduct(@RequestBody AddProductRequest addBookRequest) throws Exception {
        String admin = "admin";
        if (admin == null || !admin.equals("admin")) {
            logger.warn("Unauthorized access to postProduct endpoint");
            throw new MethodNotAllowedException();
        }
        logger.info("Adding a new product: {}", addBookRequest);
        adminService.postProduct(addBookRequest);
    }

    @DeleteMapping("/secure/delete/product")
    public void deleteProduct(@RequestParam Long productId) throws Exception {
        String admin = "admin";
        if (admin == null || !admin.equals("admin")) {
            logger.warn("Unauthorized access to deleteProduct endpoint");
            throw new MethodNotAllowedException();
        }
        logger.info("Deleting product with productId: {}", productId);
        adminService.deleteProduct(productId);
    }
}
```

2.Message Controller:

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

@RestController
@RequestMapping("/messages")
public class MessagesController {

    private static final Logger logger = LoggerFactory.getLogger(MessagesController.class);

    private MessagesServiceImpl messagesService;
    @Autowired
    public MessagesController(MessagesServiceImpl messagesService) {
        this.messagesService = messagesService;
    }

    @PostMapping("/secure/add/message")
    public void postMessage(@RequestBody Message messageRequest) {
        String userEmail = null;
        logger.info("Received a new message: {}", messageRequest);
        messagesService.postMessage(messageRequest, userEmail);
    }

    @PutMapping("/secure/admin/message")
    public void putMessage(@RequestBody AdminQuestionRequest adminQuestionRequest) throws Exce
ption {
        String userEmail = "admin@gmail.com";
        String admin = "admin";
        if (admin == null || !admin.equals("admin")) {
            logger.warn("Unauthorized access to putMessage endpoint");
            throw new MethodNotAllowedException();
        }
        logger.info("Processing admin question: {}", adminQuestionRequest);
        messagesService.putMessage(adminQuestionRequest, userEmail);
    }
}
```

3.Payment Controller:

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import com.ArtGalleryManagement.Backend.GlobalExceptionsHandler.UserEmailNotFoundException;
import com.ArtGalleryManagement.Backend.RequestModels.PaymentInfoRequest;
import com.ArtGalleryManagement.Backend.Service.PaymentService;
import com.stripe.exception.StripeException;
import com.stripe.model.PaymentIntent;

@RestController
@CrossOrigin("https://localhost:3000")
@RequestMapping("/payment/secure")
public class PaymentController {

    private static final Logger logger = LoggerFactory.getLogger(PaymentController.class);
    private PaymentService paymentService;

    @Autowired
    public PaymentController(PaymentService paymentService) {
        this.paymentService = paymentService;
    }

    @PostMapping("/payment-intent")
    public ResponseEntity<String> createPaymentIntent(@RequestBody PaymentInfoRequest paymentInf
oRequest) throws StripeException {
        logger.info("Creating payment intent for request: {}", paymentInfoRequest);
        PaymentIntent paymentIntent = paymentService.createPaymentIntent(paymentInfoRequest);
        String paymentStr = paymentIntent.toJson();
        return new ResponseEntity<>(paymentStr, HttpStatus.OK);
    }

    @PutMapping("/payment-complete")
    public ResponseEntity<String> stripePaymentComplete() throws Exception {
        String userEmail = "";
        if (userEmail == null) {
            logger.warn("User email not found");
            throw new UserEmailNotFoundException();
        }
        logger.info("Completing stripe payment for user email: {}", userEmail);
        return paymentService.stripePayment(userEmail);
    }
}
```

---

4) ProductController:

```
package com.ArtGalleryManagement.Backend.Controller;

import com.ArtGalleryManagement.Backend.Entity.*;
import com.ArtGalleryManagement.Backend.ResponseModels.*;
import com.ArtGalleryManagement.Backend.Service.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@CrossOrigin("https://localhost:3000")
@RestController
@RequestMapping("/products")
public class ProductController {

 private static final Logger logger = LoggerFactory.getLogger(ProductController.class);

 private ProductServiceImpl productService;
 @Autowired
 public ProductController(ProductServiceImpl productService) {
  this.productService = productService;
 }

 @GetMapping("/secure/currentloans")
 public List<ShelfCurrentLoansResponse> currentLoans() throws Exception {
  String userEmail = "mohit@gmail.com";
  logger.info("Fetching current loans for user: {}", userEmail);
  return productService.currentLoans(userEmail);
 }

 @GetMapping("/secure/currentloans/count")
 public int currentLoansCount() {
  String userEmail = "mohit@gmail.com";
  logger.info("Fetching current loans count for user: {}", userEmail);
  return productService.currentLoansCount(userEmail);
 }

 @GetMapping("/secure/ischeckedout/byuser")
 public Boolean checkoutProductByUser(@RequestParam Long productId) {
  String userEmail = "mohit@gmail.com";
  logger.info("Checking if product with ID {} is checked out by user: {}", productId, userEmail);
  return productService.checkoutProductByUser(userEmail, productId);
 }

 @PutMapping("/secure/checkout")
 public Product checkoutProduct(@RequestParam Long productId) throws Exception {
  String userEmail = "mohit@gmail.com";
  logger.info("Checking out product with ID {} for user: {}", productId, userEmail);
  return productService.checkoutProduct(userEmail, productId);
 }

 @PutMapping("/secure/return")
 public void returnProduct(@RequestParam Long productId) throws Exception {
  String userEmail = "mohit@gmail.com";
  logger.info("Returning product with ID {} for user: {}", productId, userEmail);
  productService.returnProduct(userEmail, productId);
 }

 @PutMapping("/secure/renew/loan")
 public void renewLoan(@RequestParam Long productId) throws Exception {
  String userEmail = "mohit@gmail.com";
  logger.info("Renewing loan for product with ID {} for user: {}", productId, userEmail);
  productService.renewLoan(userEmail, productId);
 }
}
```

---

5) ReviewController:

```
package com.ArtGalleryManagement.Backend.Controller;
```

```
import com.ArtGalleryManagement.Backend.GlobalExceptionsHandler.UserEmailNotFoundException;
import com.ArtGalleryManagement.Backend.RequestModels.*;
import com.ArtGalleryManagement.Backend.Service.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.*;

@CrossOrigin("https://localhost:3000")
@RestController
@RequestMapping("/reviews")
public class ReviewController {

 private static final Logger logger = LoggerFactory.getLogger(ReviewController.class);

 private ReviewServiceImpl reviewService;

 public ReviewController(ReviewServiceImpl reviewService) {
  this.reviewService = reviewService;
 }

 @GetMapping("/secure/user/product")
 public Boolean reviewProductByUser(@RequestParam Long productId) throws Exception {
  String userEmail = "arsh@gmail.com";

  if (userEmail == null) {
   logger.error("User email not found");
   throw new UserEmailNotFoundException();
  }

  logger.info("Checking if user {} reviewed product with ID: {}", userEmail, productId);
  return reviewService.userReviewListed(userEmail, productId);
 }

 @PostMapping("/secure")
 public void postReview(@RequestBody ReviewRequest reviewRequest) throws Exception {
  String userEmail = "ankita@gmail.com";
  if (userEmail == null) {
   logger.error("User email not found");
   throw new UserEmailNotFoundException();
  }

  logger.info("Posting review for user {}: {}", userEmail, reviewRequest);
  reviewService.postReview(userEmail, reviewRequest);
 }
}
```

"Love_You_Anki_Jaan"