UE20CS352- OBJECT ORIENTED ANALYSIS AND DESIGN WITH JAVA

MINI PROJECT REPORT

TITLE: LIBRARY MANAGEMENT SYSTEM

SUBMITTED BY:

ABHISHEK PATIL – PES1UG20CS531
BHUVAN A J – PES1UG20CS541
DARSHAN V – PES1UG20CS546
G RANJITHA RANI – PES1UG20CS549

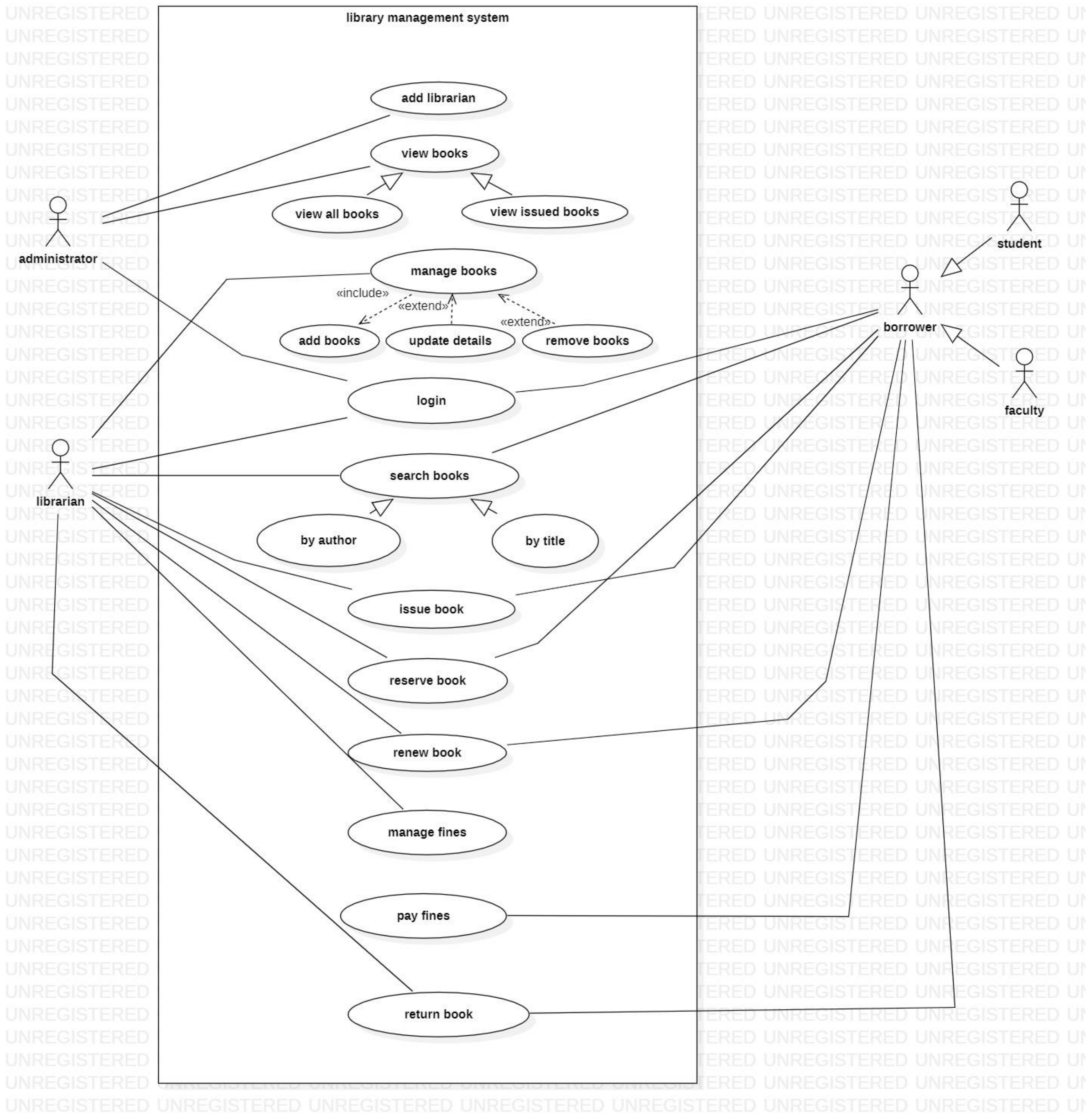
SECTION: I

INDEX

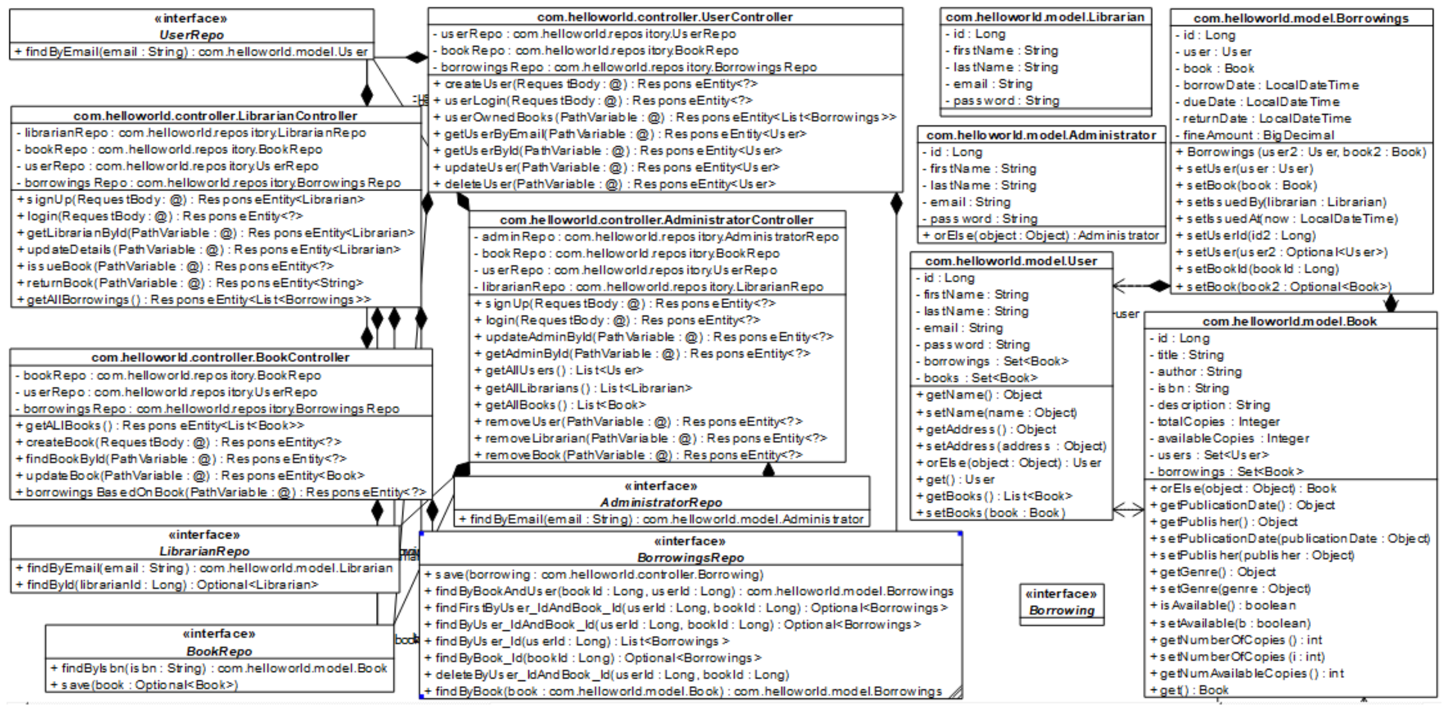
I. ABSTRACT

The project is a library management system developed using Spring Boot, Next.js, and PostgreSQL as the database. It allows admin to manage books, librarians and students. It allows the librarian to manage books and students. It allows the students to borrow and return books. The Spring Boot framework is used for the backend, while Next.js is used for the frontend. PostgreSQL is used as the database to store the data.

II. USE-CASE DIAGRAM



III. CLASS-DIAGRAM


«interface»
UserRepo
+ findByEmail(email : String) : com.helloworld.model.User
com.helloworld.controller.UserController
- userRepo : com.helloworld.repository.UserRepo
- bookRepo : com.helloworld.repository.BookRepo
- borrowingsRepo : com.helloworld.repository.BorrowingsRepo
+ createUser(RequestBody : @) : ResponseEntity<?>
+ userLogin(RequestBody : @) : ResponseEntity<?>
+ userOwnedBooks(PathVariable : @) : ResponseEntity<List<Borrowings>>
+ getUserByEmail(PathVariable : @) : ResponseEntity<User>
+ getUserById(PathVariable : @) : ResponseEntity<User>
+ updateUser(PathVariable : @) : ResponseEntity<User>
+ deleteUser(PathVariable : @) : ResponseEntity<User>
com.helloworld.model.Librarian
- id : Long
- firstName : String
- lastName : String
- email : String
- password : String
com.helloworld.model.Borrowings
- id : Long
- user : User
- book : Book
- borrowDate : LocalDateTime
- dueDate : LocalDateTime
- returnDate : LocalDateTime
- fineAmount : BigDecimal
+ Borrowings (user2 : User, book2 : Book)
+ setUser(user : User)
+ setBook(book : Book)
+ setIssuedBy(librarian : Librarian)
+ setIssuedAt(now : LocalDateTime)
+ setUserId(id2 : Long)
+ setUser(user2 : Optional<User>)
+ setBookId(bookId : Long)
+ setBook(book2 : Optional<Book>)
com.helloworld.controller.LibrarianController
- librarianRepo : com.helloworld.repository.LibrarianRepo
- bookRepo : com.helloworld.repository.BookRepo
- userRepo : com.helloworld.repository.UserRepo
- borrowingsRepo : com.helloworld.repository.BorrowingsRepo
+ signUp(RequestBody : @) : ResponseEntity<Librarian>
+ login(RequestBody : @) : ResponseEntity<?>
+ getLibrarianById(PathVariable : @) : ResponseEntity<Librarian>
+ updateDetails(PathVariable : @) : ResponseEntity<Librarian>
+ issueBook(PathVariable : @) : ResponseEntity<?>
+ returnBook(PathVariable : @) : ResponseEntity<String>
+ getAllBorrowings() : ResponseEntity<List<Borrowings>>
com.helloworld.model.Administrator
- id : Long
- firstName : String
- lastName : String
- email : String
- password : String
+ orElse(object : Object) : Administrator
com.helloworld.controller.AdministratorController
- adminRepo : com.helloworld.repository.AdministratorRepo
- bookRepo : com.helloworld.repository.BookRepo
- userRepo : com.helloworld.repository.UserRepo
- librarianRepo : com.helloworld.repository.LibrarianRepo
+ signUp(RequestBody : @) : ResponseEntity<?>
+ login(RequestBody : @) : ResponseEntity<?>
+ updateAdminById(PathVariable : @) : ResponseEntity<?>
+ getAdminById(PathVariable : @) : ResponseEntity<?>
+ getAllUsers() : List<User>
+ getAllLibrarians() : List<Librarian>
+ getAllBooks() : List<Book>
+ removeUser(PathVariable : @) : ResponseEntity<?>
+ removeLibrarian(PathVariable : @) : ResponseEntity<?>
+ removeBook(PathVariable : @) : ResponseEntity<?>
com.helloworld.model.User
- id : Long
- firstName : String
- lastName : String
- email : String
- password : String
- borrowings : Set<Book>
- books : Set<Book>
+ getName() : Object
+ setName(name : Object)
+ getAddress() : Object
+ setAddress(address : Object)
+ orElse(object : Object) : User
+ get() : User
+ getBooks() : List<Book>
+ setBooks(book : Book)
user
com.helloworld.model.Book
- id : Long
- title : String
- author : String
- isbn : String
- description : String
- totalCopies : Integer
- availableCopies : Integer
- users : Set<User>
- borrowings : Set<Book>
+ orElse(object : Object) : Book
+ getPublicationDate() : Object
+ getPublisher() : Object
+ setPublicationDate(publicationDate : Object)
+ setPublisher(publisher : Object)
+ getGenre() : Object
+ setGenre(genre : Object)
+ isAvailable() : boolean
+ setAvailable(b : boolean)
+ getNumberOfCopies() : int
+ setNumberOfCopies(i : int)
+ getNumAvailableCopies() : int
+ get() : Book
com.helloworld.controller.BookController
- bookRepo : com.helloworld.repository.BookRepo
- userRepo : com.helloworld.repository.UserRepo
- borrowingsRepo : com.helloworld.repository.BorrowingsRepo
+ getALlBooks() : ResponseEntity<List<Book>>
+ createBook(RequestBody : @) : ResponseEntity<?>
+ findBookById(PathVariable : @) : ResponseEntity<?>
+ updateBook(PathVariable : @) : ResponseEntity<Book>
+ borrowingsBasedOnBook(PathVariable : @) : ResponseEntity<?>
«interface»
AdministratorRepo
+ findByEmail(email : String) : com.helloworld.model.Administrator
«interface»
LibrarianRepo
+ findByEmail(email : String) : com.helloworld.model.Librarian
+ findById(librarianId : Long) : Optional<Librarian>
«interface»
BorrowingsRepo
+ save(borrowing : com.helloworld.controller.Borrowing)
+ findByBookAndUser(bookId : Long, userId : Long) : com.helloworld.model.Borrowings
+ findFirstByUser_IdAndBook_Id(userId : Long, bookId : Long) : Optional<Borrowings>
+ findByUser_IdAndBook_Id(userId : Long, bookId : Long) : Optional<Borrowings>
+ findByUser_Id(userId : Long) : List<Borrowings>
+ findByBook_Id(bookId : Long) : Optional<Borrowings>
+ deleteByUser_IdAndBook_Id(userId : Long, bookId : Long)
+ findByBook(book : com.helloworld.model.Book) : com.helloworld.model.Borrowings
«interface»
Borrowing
«interface»
BookRepo
+ findByIsbn(isbn : String) : com.helloworld.model.Book
+ save(book : Optional<Book>)
book

IV. DESIGN PRINCIPLES AND PATTERNS:

a. MVC Architecture:

MVC stands for Model-View-Controller, which is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. The model represents the data and business logic, the view represents the user interface, and the controller handles user input and updates the model and view accordingly.

b. Singleton pattern:

The Singleton pattern is a design pattern that restricts the instantiation of a class to one object. It is a creational pattern that provides a way to ensure that a class has only one instance and provides a global point of access to that instance.

c. Single responsibility principle:

The Single Responsibility Principle (SRP) is a programming principle that states that a class should have only one reason to change, or in other words, only one responsibility. This principle suggests that a class should have only one job or purpose, and that job should be encapsulated within the class.

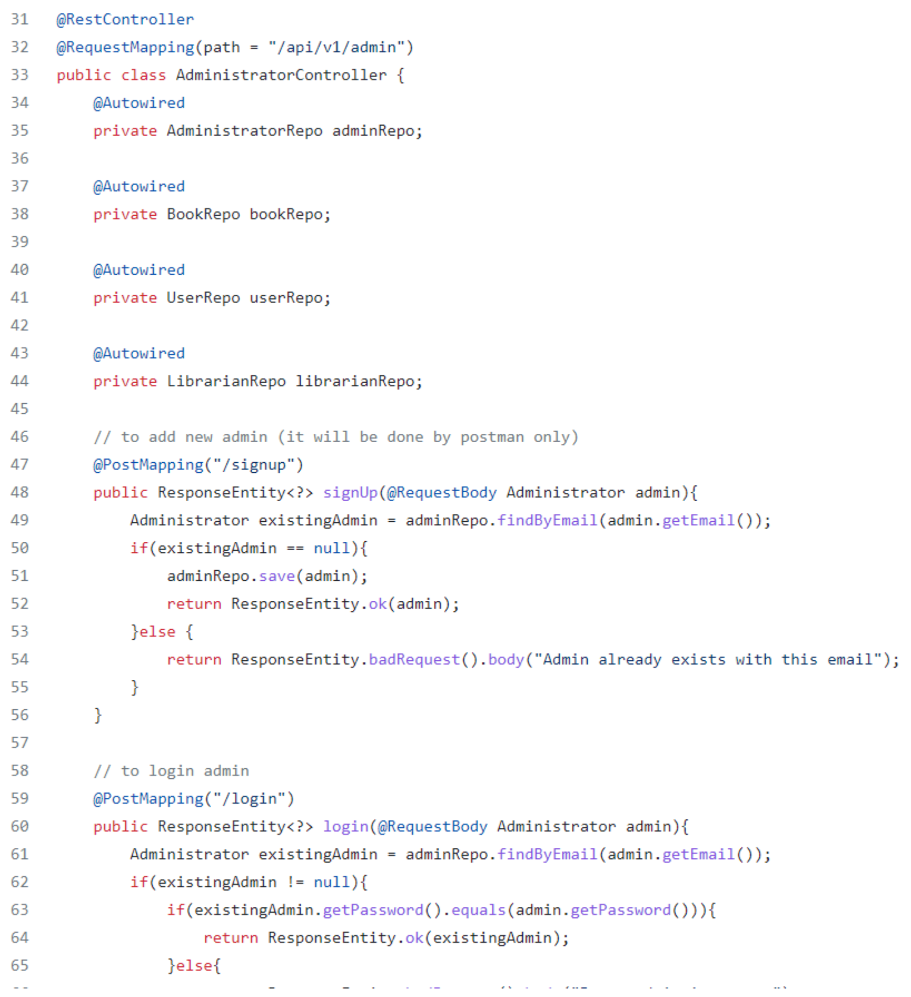
d. Open/Close principle:

The Open-Closed Principle (OCP) is a programming principle that states that software entities (classes, modules, functions, etc.) should be open for extension but closed for modification. This means that a class should be designed in such a way that it can be easily extended to add new functionality without modifying its source code.

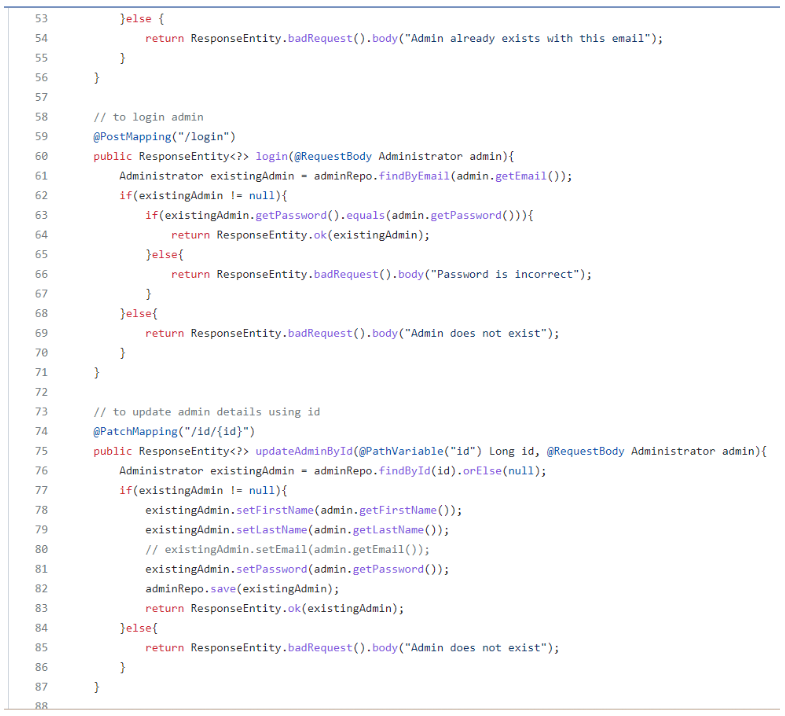
V. CODE

a. CONTROLLER

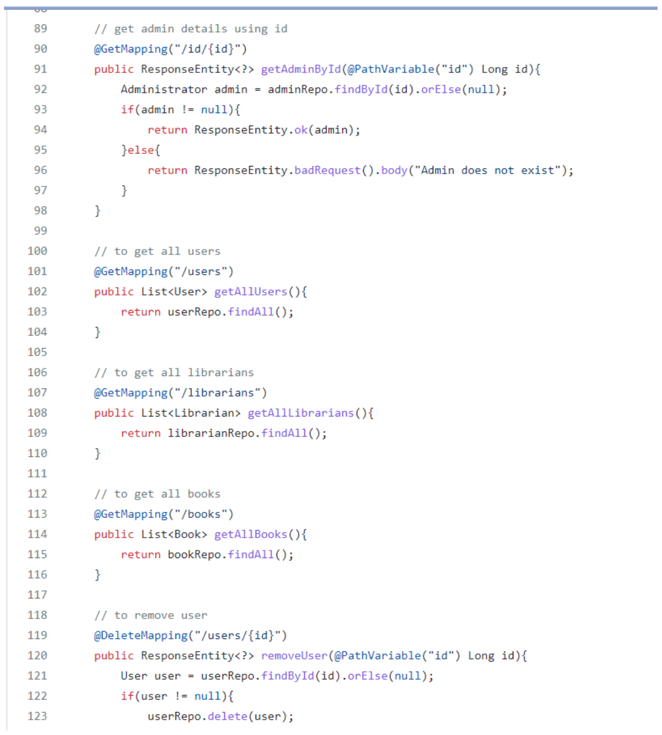
i. AdministratorController.java



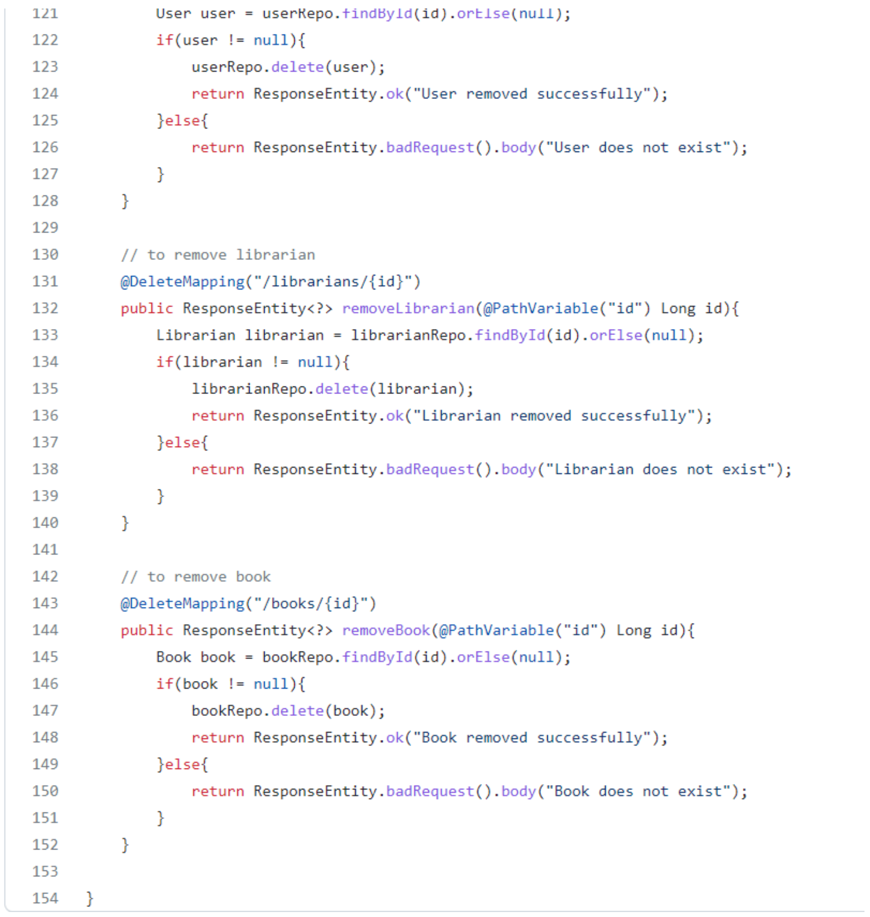
```java
@RestController
@RequestMapping(path = "/api/v1/admin")
public class AdministratorController {
    @Autowired
    private AdministratorRepo adminRepo;

    @Autowired
    private BookRepo bookRepo;

    @Autowired
    private UserRepo userRepo;

    @Autowired
    private LibrarianRepo librarianRepo;

    // to add new admin (it will be done by postman only)
    @PostMapping("/signup")
    public ResponseEntity<?> signUp(@RequestBody Administrator admin){
        Administrator existingAdmin = adminRepo.findByEmail(admin.getEmail());
        if(existingAdmin == null){
            adminRepo.save(admin);
            return ResponseEntity.ok(admin);
        }else {
            return ResponseEntity.badRequest().body("Admin already exists with this email");
        }
    }

    // to login admin
    @PostMapping("/login")
    public ResponseEntity<?> login(@RequestBody Administrator admin){
        Administrator existingAdmin = adminRepo.findByEmail(admin.getEmail());
        if(existingAdmin != null){
            if(existingAdmin.getPassword().equals(admin.getPassword())){
                return ResponseEntity.ok(existingAdmin);
            }else{
```



```java
        }else {
            return ResponseEntity.badRequest().body("Admin already exists with this email");
        }
    }

    // to login admin
    @PostMapping("/login")
    public ResponseEntity<?> login(@RequestBody Administrator admin){
        Administrator existingAdmin = adminRepo.findByEmail(admin.getEmail());
        if(existingAdmin != null){
            if(existingAdmin.getPassword().equals(admin.getPassword())){
                return ResponseEntity.ok(existingAdmin);
            }else{
                return ResponseEntity.badRequest().body("Password is incorrect");
            }
        }else{
            return ResponseEntity.badRequest().body("Admin does not exist");
        }
    }

    // to update admin details using id
    @PatchMapping("/id/{id}")
    public ResponseEntity<?> updateAdminById(@PathVariable("id") Long id, @RequestBody Administrator admin){
        Administrator existingAdmin = adminRepo.findById(id).orElse(null);
        if(existingAdmin != null){
            existingAdmin.setFirstName(admin.getFirstName());
            existingAdmin.setLastName(admin.getLastName());
            // existingAdmin.setEmail(admin.getEmail());
            existingAdmin.setPassword(admin.getPassword());
            adminRepo.save(existingAdmin);
            return ResponseEntity.ok(existingAdmin);
        }else{
            return ResponseEntity.badRequest().body("Admin does not exist");
        }
    }
```

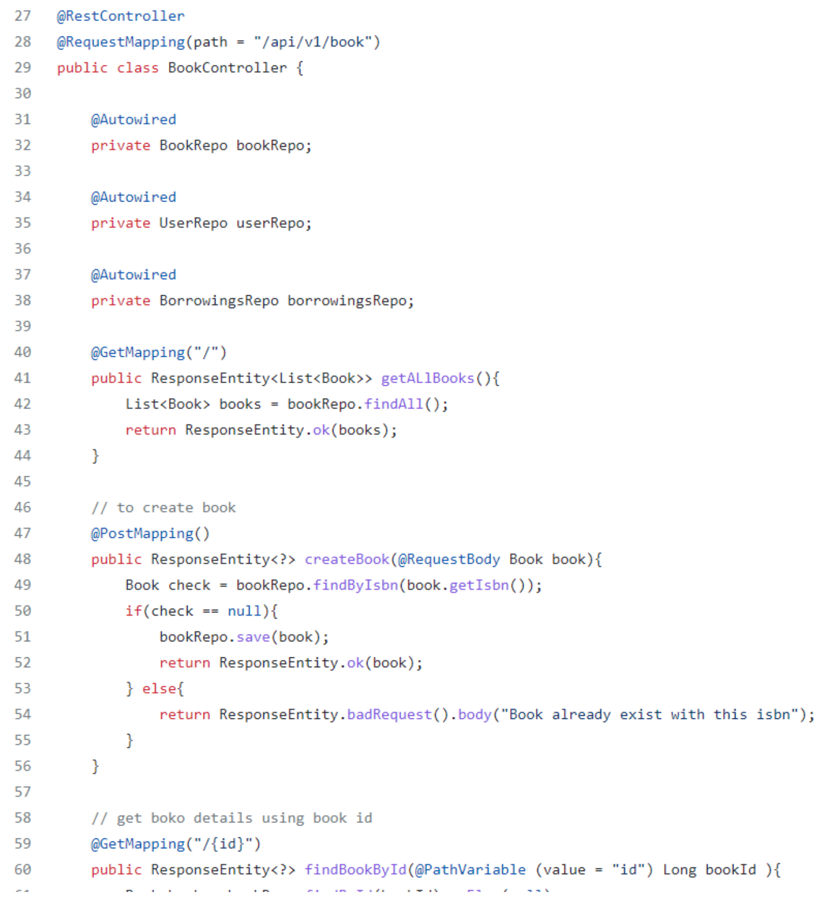


```java
    // get admin details using id
    @GetMapping("/id/{id}")
    public ResponseEntity<?> getAdminById(@PathVariable("id") Long id){
        Administrator admin = adminRepo.findById(id).orElse(null);
        if(admin != null){
            return ResponseEntity.ok(admin);
        }else{
            return ResponseEntity.badRequest().body("Admin does not exist");
        }
    }

    // to get all users
    @GetMapping("/users")
    public List<User> getAllUsers(){
        return userRepo.findAll();
    }

    // to get all librarians
    @GetMapping("/librarians")
    public List<Librarian> getAllLibrarians(){
        return librarianRepo.findAll();
    }

    // to get all books
    @GetMapping("/books")
    public List<Book> getAllBooks(){
        return bookRepo.findAll();
    }

    // to remove user
    @DeleteMapping("/users/{id}")
    public ResponseEntity<?> removeUser(@PathVariable("id") Long id){
        User user = userRepo.findById(id).orElse(null);
        if(user != null){
            userRepo.delete(user);
```

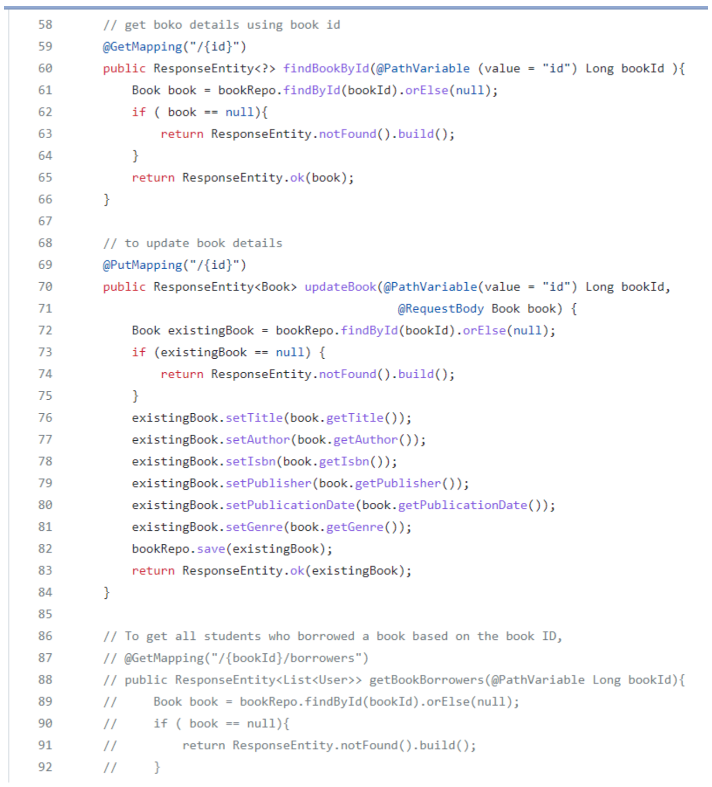


```java
        User user = userRepo.findById(id).orElse(null);
        if(user != null){
            userRepo.delete(user);
            return ResponseEntity.ok("User removed successfully");
        }else{
            return ResponseEntity.badRequest().body("User does not exist");
        }
    }

    // to remove librarian
    @DeleteMapping("/librarians/{id}")
    public ResponseEntity<?> removeLibrarian(@PathVariable("id") Long id){
        Librarian librarian = librarianRepo.findById(id).orElse(null);
        if(librarian != null){
            librarianRepo.delete(librarian);
            return ResponseEntity.ok("Librarian removed successfully");
        }else{
            return ResponseEntity.badRequest().body("Librarian does not exist");
        }
    }

    // to remove book
    @DeleteMapping("/books/{id}")
    public ResponseEntity<?> removeBook(@PathVariable("id") Long id){
        Book book = bookRepo.findById(id).orElse(null);
        if(book != null){
            bookRepo.delete(book);
            return ResponseEntity.ok("Book removed successfully");
        }else{
            return ResponseEntity.badRequest().body("Book does not exist");
        }
    }

}
```

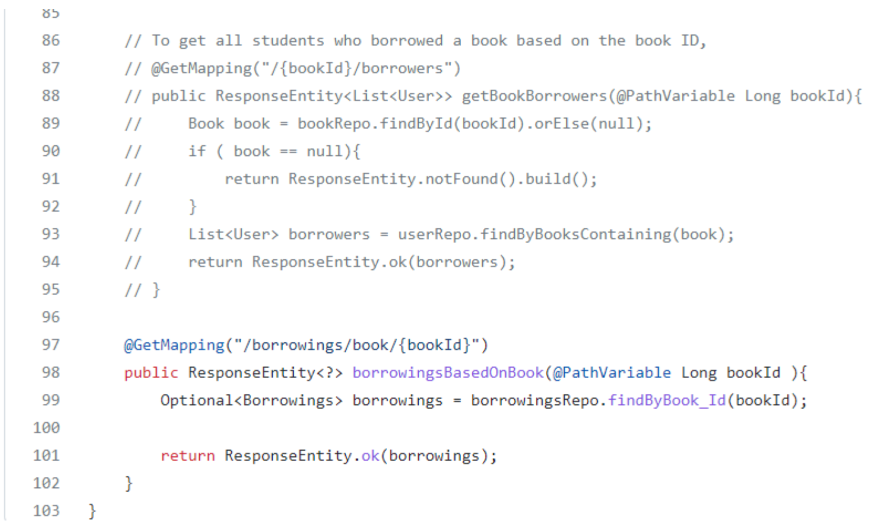
ii. BookController.java



```java
@RestController
@RequestMapping(path = "/api/v1/book")
public class BookController {

    @Autowired
    private BookRepo bookRepo;

    @Autowired
    private UserRepo userRepo;

    @Autowired
    private BorrowingsRepo borrowingsRepo;

    @GetMapping("/")
    public ResponseEntity<List<Book>> getALlBooks(){
        List<Book> books = bookRepo.findAll();
        return ResponseEntity.ok(books);
    }

    // to create book
    @PostMapping()
    public ResponseEntity<?> createBook(@RequestBody Book book){
        Book check = bookRepo.findByIsbn(book.getIsbn());
        if(check == null){
            bookRepo.save(book);
            return ResponseEntity.ok(book);
        } else{
            return ResponseEntity.badRequest().body("Book already exist with this isbn");
        }
    }

    // get boko details using book id
    @GetMapping("/{id}")
    public ResponseEntity<?> findBookById(@PathVariable (value = "id") Long bookId ){
```

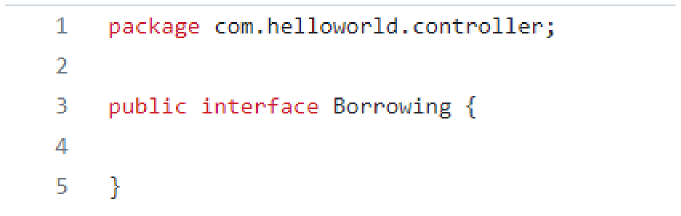


```
    // get boko details using book id
    @GetMapping("/{id}")
    public ResponseEntity<?> findBookById(@PathVariable (value = "id") Long bookId ){
        Book book = bookRepo.findById(bookId).orElse(null);
        if ( book == null){
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(book);
    }

    // to update book details
    @PutMapping("/{id}")
    public ResponseEntity<Book> updateBook(@PathVariable(value = "id") Long bookId,
                                            @RequestBody Book book) {
        Book existingBook = bookRepo.findById(bookId).orElse(null);
        if (existingBook == null) {
            return ResponseEntity.notFound().build();
        }
        existingBook.setTitle(book.getTitle());
        existingBook.setAuthor(book.getAuthor());
        existingBook.setIsbn(book.getIsbn());
        existingBook.setPublisher(book.getPublisher());
        existingBook.setPublicationDate(book.getPublicationDate());
        existingBook.setGenre(book.getGenre());
        bookRepo.save(existingBook);
        return ResponseEntity.ok(existingBook);
    }

    // To get all students who borrowed a book based on the book ID,
    // @GetMapping("/{bookId}/borrowers")
    // public ResponseEntity<List<User>> getBookBorrowers(@PathVariable Long bookId){
    //     Book book = bookRepo.findById(bookId).orElse(null);
    //     if ( book == null){
    //         return ResponseEntity.notFound().build();
    //     }
```

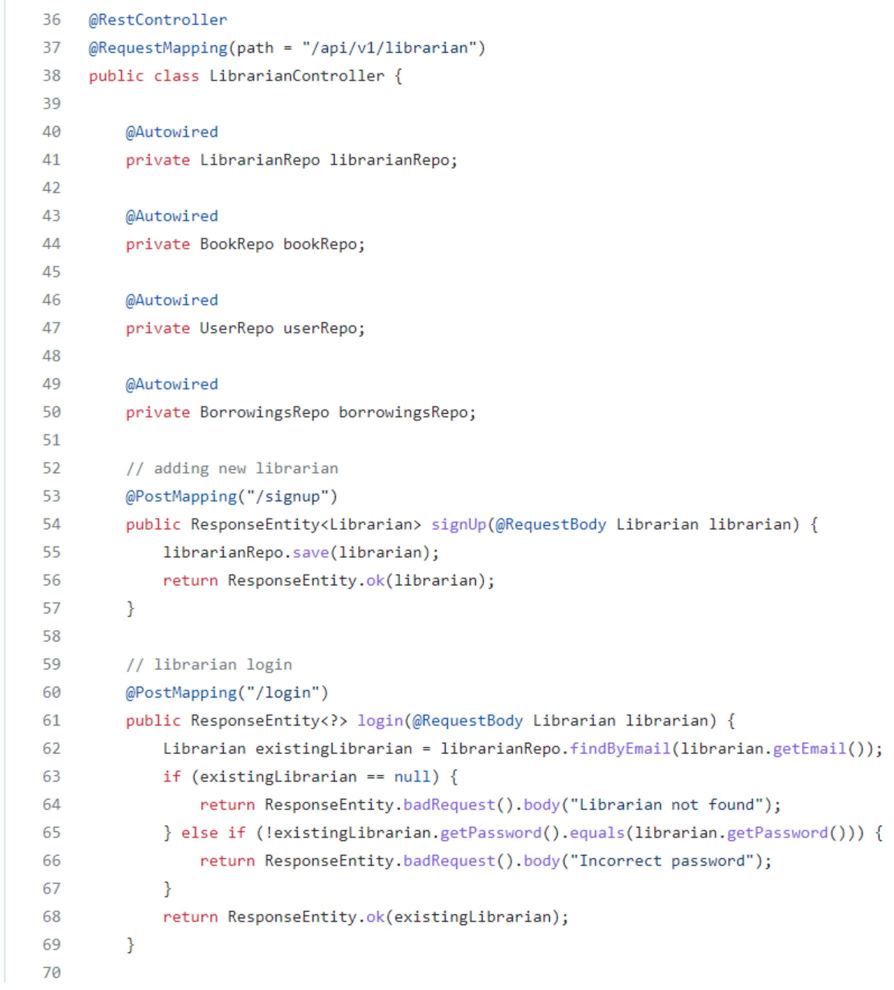


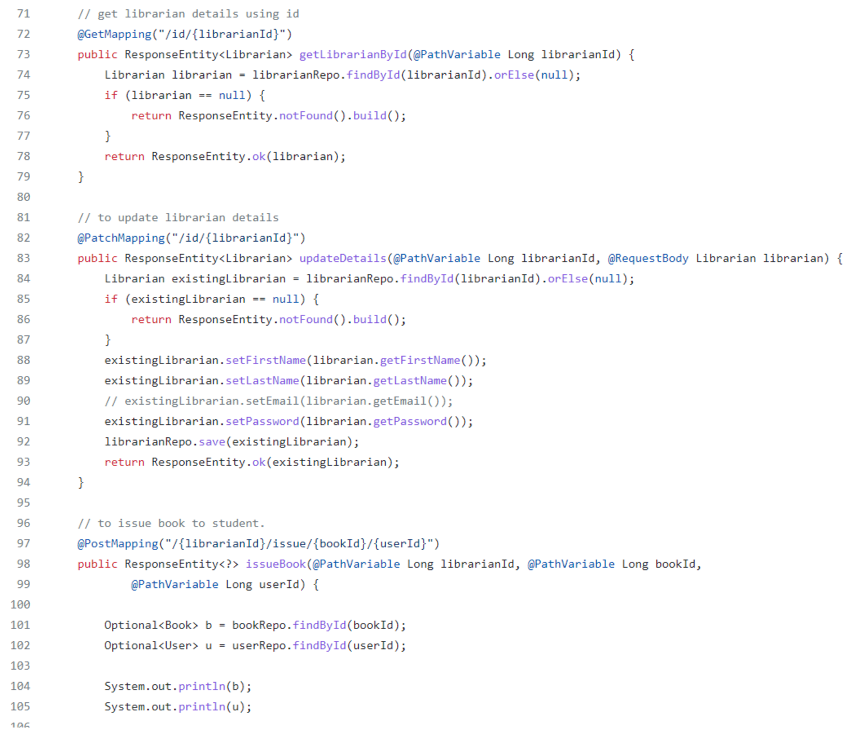
```
    // To get all students who borrowed a book based on the book ID,
    // @GetMapping("/{bookId}/borrowers")
    // public ResponseEntity<List<User>> getBookBorrowers(@PathVariable Long bookId){
    //     Book book = bookRepo.findById(bookId).orElse(null);
    //     if ( book == null){
    //         return ResponseEntity.notFound().build();
    //     }
    //     List<User> borrowers = userRepo.findByBooksContaining(book);
    //     return ResponseEntity.ok(borrowers);
    // }

    @GetMapping("/borrowings/book/{bookId}")
    public ResponseEntity<?> borrowingsBasedOnBook(@PathVariable Long bookId ){
        Optional<Borrowings> borrowings = borrowingsRepo.findByBook_Id(bookId);

        return ResponseEntity.ok(borrowings);
    }
}
```

iii. Borrowing.java

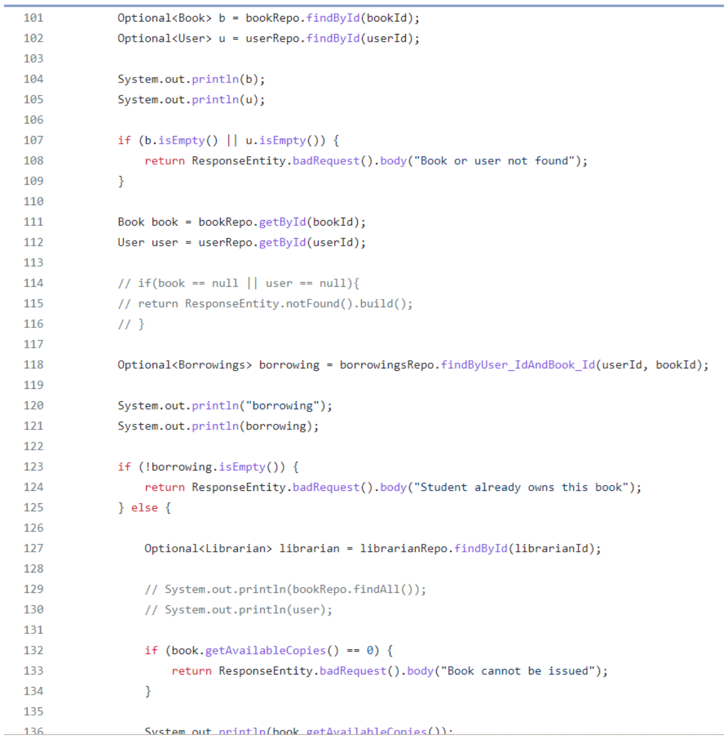

```java
package com.helloworld.controller;

public interface Borrowing {

}
```

iv. LibrarianController.java

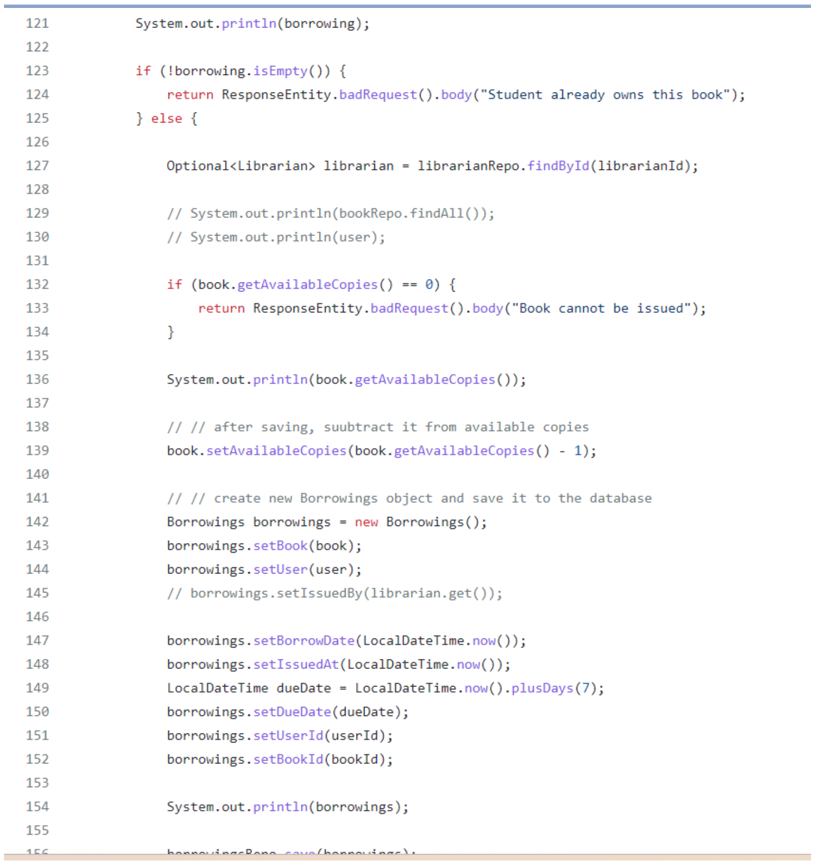

```java
@RestController
@RequestMapping(path = "/api/v1/librarian")
public class LibrarianController {

    @Autowired
    private LibrarianRepo librarianRepo;

    @Autowired
    private BookRepo bookRepo;

    @Autowired
    private UserRepo userRepo;

    @Autowired
    private BorrowingsRepo borrowingsRepo;

    // adding new librarian
    @PostMapping("/signup")
    public ResponseEntity<Librarian> signUp(@RequestBody Librarian librarian) {
        librarianRepo.save(librarian);
        return ResponseEntity.ok(librarian);
    }

    // librarian login
    @PostMapping("/login")
    public ResponseEntity<?> login(@RequestBody Librarian librarian) {
        Librarian existingLibrarian = librarianRepo.findByEmail(librarian.getEmail());
        if (existingLibrarian == null) {
            return ResponseEntity.badRequest().body("Librarian not found");
        } else if (!existingLibrarian.getPassword().equals(librarian.getPassword())) {
            return ResponseEntity.badRequest().body("Incorrect password");
        }
        return ResponseEntity.ok(existingLibrarian);
    }

```



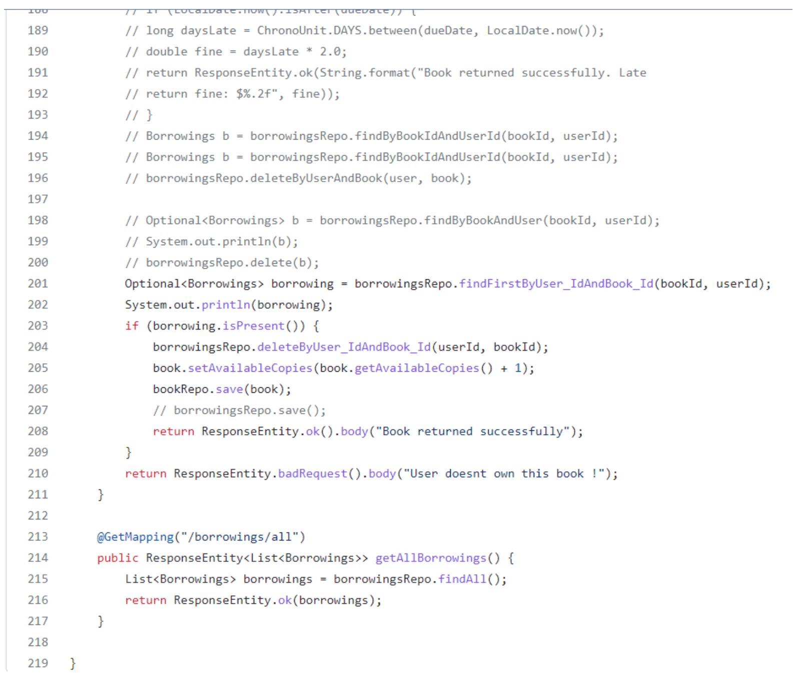
```java
    // get librarian details using id
    @GetMapping("/id/{librarianId}")
    public ResponseEntity<Librarian> getLibrarianById(@PathVariable Long librarianId) {
        Librarian librarian = librarianRepo.findById(librarianId).orElse(null);
        if (librarian == null) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(librarian);
    }

    // to update librarian details
    @PatchMapping("/id/{librarianId}")
    public ResponseEntity<Librarian> updateDetails(@PathVariable Long librarianId, @RequestBody Librarian librarian) {
        Librarian existingLibrarian = librarianRepo.findById(librarianId).orElse(null);
        if (existingLibrarian == null) {
            return ResponseEntity.notFound().build();
        }
        existingLibrarian.setFirstName(librarian.getFirstName());
        existingLibrarian.setLastName(librarian.getLastName());
        // existingLibrarian.setEmail(librarian.getEmail());
        existingLibrarian.setPassword(librarian.getPassword());
        librarianRepo.save(existingLibrarian);
        return ResponseEntity.ok(existingLibrarian);
    }

    // to issue book to student.
    @PostMapping("/{librarianId}/issue/{bookId}/{userId}")
    public ResponseEntity<?> issueBook(@PathVariable Long librarianId, @PathVariable Long bookId,
            @PathVariable Long userId) {

        Optional<Book> b = bookRepo.findById(bookId);
        Optional<User> u = userRepo.findById(userId);

        System.out.println(b);
        System.out.println(u);
```



```java
        Optional<Book> b = bookRepo.findById(bookId);
        Optional<User> u = userRepo.findById(userId);

        System.out.println(b);
        System.out.println(u);

        if (b.isEmpty() || u.isEmpty()) {
            return ResponseEntity.badRequest().body("Book or user not found");
        }

        Book book = bookRepo.getById(bookId);
        User user = userRepo.getById(userId);

        // if(book == null || user == null){
        // return ResponseEntity.notFound().build();
        // }

        Optional<Borrowings> borrowing = borrowingsRepo.findByUser_IdAndBook_Id(userId, bookId);

        System.out.println("borrowing");
        System.out.println(borrowing);

        if (!borrowing.isEmpty()) {
            return ResponseEntity.badRequest().body("Student already owns this book");
        } else {

            Optional<Librarian> librarian = librarianRepo.findById(librarianId);

            // System.out.println(bookRepo.findAll());
            // System.out.println(user);

            if (book.getAvailableCopies() == 0) {
                return ResponseEntity.badRequest().body("Book cannot be issued");
            }

            System.out.println(book.getAvailableCopies());
```

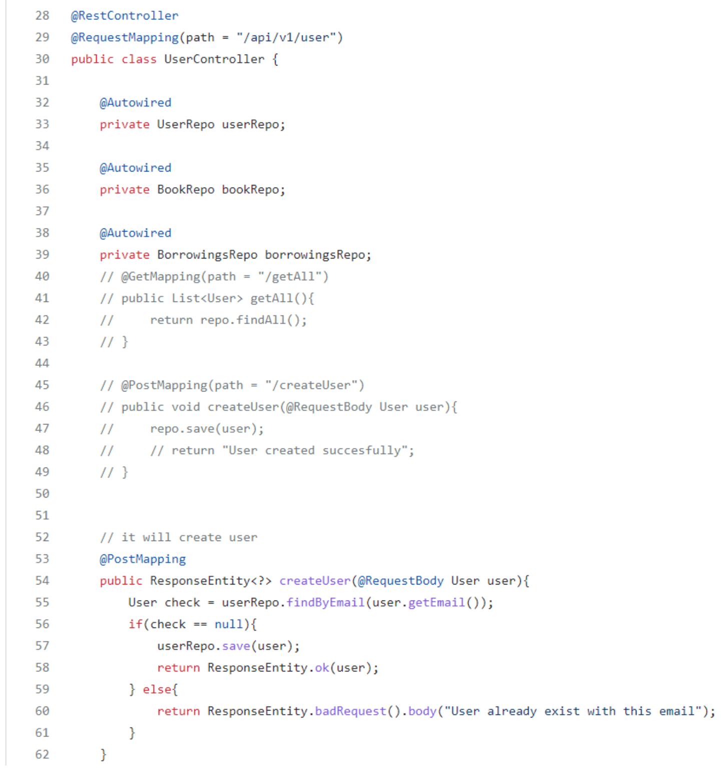


```
        System.out.println(borrowing);

        if (!borrowing.isEmpty()) {
            return ResponseEntity.badRequest().body("Student already owns this book");
        } else {

            Optional<Librarian> librarian = librarianRepo.findById(librarianId);

            // System.out.println(bookRepo.findAll());
            // System.out.println(user);

            if (book.getAvailableCopies() == 0) {
                return ResponseEntity.badRequest().body("Book cannot be issued");
            }

            System.out.println(book.getAvailableCopies());

            // // after saving, suubtract it from available copies
            book.setAvailableCopies(book.getAvailableCopies() - 1);

            // // create new Borrowings object and save it to the database
            Borrowings borrowings = new Borrowings();
            borrowings.setBook(book);
            borrowings.setUser(user);
            // borrowings.setIssuedBy(librarian.get());

            borrowings.setBorrowDate(LocalDateTime.now());
            borrowings.setIssuedAt(LocalDateTime.now());
            LocalDateTime dueDate = LocalDateTime.now().plusDays(7);
            borrowings.setDueDate(dueDate);
            borrowings.setUserId(userId);
            borrowings.setBookId(bookId);

            System.out.println(borrowings);

            borrowingsRepo.save(borrowings);
```

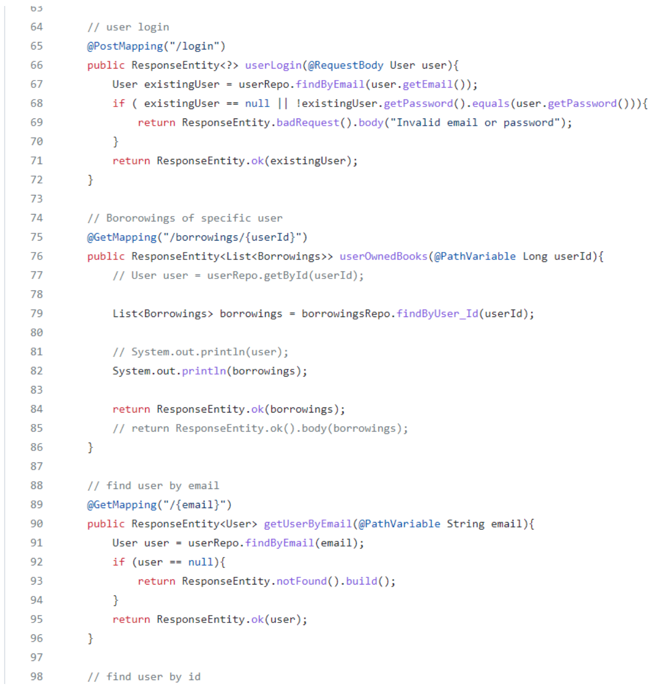


```
        // if (LocalDate.now().isAfter(dueDate)) {
        // long daysLate = ChronoUnit.DAYS.between(dueDate, LocalDate.now());
        // double fine = daysLate * 2.0;
        // return ResponseEntity.ok(String.format("Book returned successfully. Late
        // return fine: $%.2f", fine));
        // }
        // Borrowings b = borrowingsRepo.findByBookIdAndUserId(bookId, userId);
        // Borrowings b = borrowingsRepo.findByBookIdAndUserId(bookId, userId);
        // borrowingsRepo.deleteByUserAndBook(user, book);

        // Optional<Borrowings> b = borrowingsRepo.findByBookAndUser(bookId, userId);
        // System.out.println(b);
        // borrowingsRepo.delete(b);
        Optional<Borrowings> borrowing = borrowingsRepo.findFirstByUser_IdAndBook_Id(bookId, userId);
        System.out.println(borrowing);
        if (borrowing.isPresent()) {
            borrowingsRepo.deleteByUser_IdAndBook_Id(userId, bookId);
            book.setAvailableCopies(book.getAvailableCopies() + 1);
            bookRepo.save(book);
            // borrowingsRepo.save();
            return ResponseEntity.ok().body("Book returned successfully");
        }
        return ResponseEntity.badRequest().body("User doesnt own this book !");
    }

    @GetMapping("/borrowings/all")
    public ResponseEntity<List<Borrowings>> getAllBorrowings() {
        List<Borrowings> borrowings = borrowingsRepo.findAll();
        return ResponseEntity.ok(borrowings);
    }

}
```

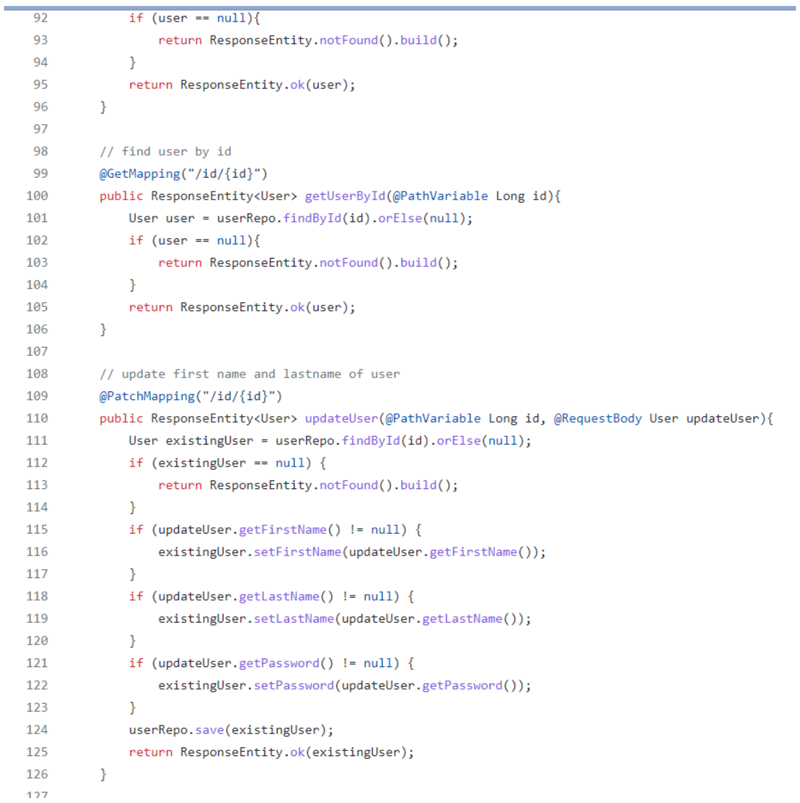
v. UserController.java



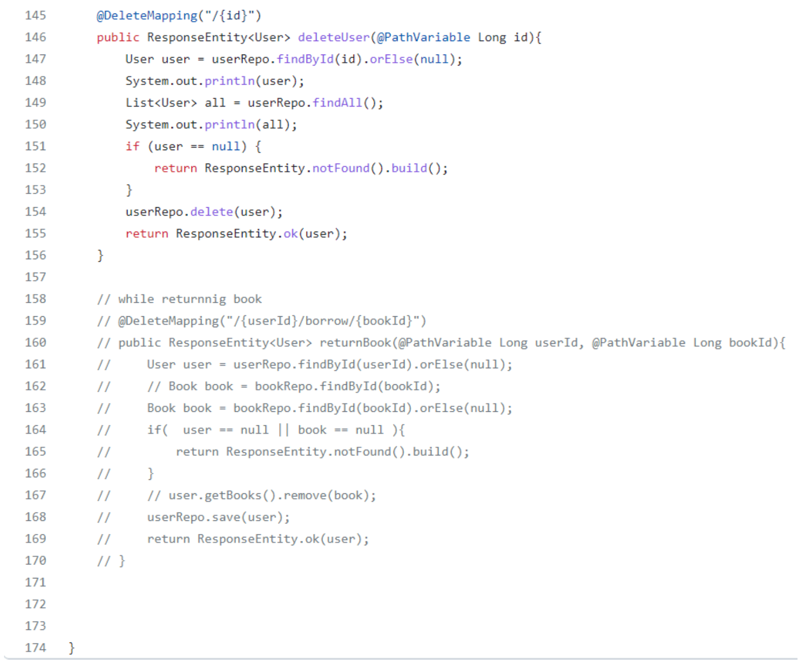
```java
@RestController
@RequestMapping(path = "/api/v1/user")
public class UserController {

    @Autowired
    private UserRepo userRepo;

    @Autowired
    private BookRepo bookRepo;

    @Autowired
    private BorrowingsRepo borrowingsRepo;
    // @GetMapping(path = "/getAll")
    // public List<User> getAll(){
    //     return repo.findAll();
    // }

    // @PostMapping(path = "/createUser")
    // public void createUser(@RequestBody User user){
    //     repo.save(user);
    //     // return "User created succesfully";
    // }


    // it will create user
    @PostMapping
    public ResponseEntity<?> createUser(@RequestBody User user){
        User check = userRepo.findByEmail(user.getEmail());
        if(check == null){
            userRepo.save(user);
            return ResponseEntity.ok(user);
        } else{
            return ResponseEntity.badRequest().body("User already exist with this email");
        }
    }
```



```java

    // user login
    @PostMapping("/login")
    public ResponseEntity<?> userLogin(@RequestBody User user){
        User existingUser = userRepo.findByEmail(user.getEmail());
        if ( existingUser == null || !existingUser.getPassword().equals(user.getPassword())){
            return ResponseEntity.badRequest().body("Invalid email or password");
        }
        return ResponseEntity.ok(existingUser);
    }

    // Bororowings of specific user
    @GetMapping("/borrowings/{userId}")
    public ResponseEntity<List<Borrowings>> userOwnedBooks(@PathVariable Long userId){
        // User user = userRepo.getById(userId);

        List<Borrowings> borrowings = borrowingsRepo.findByUser_Id(userId);

        // System.out.println(user);
        System.out.println(borrowings);

        return ResponseEntity.ok(borrowings);
        // return ResponseEntity.ok().body(borrowings);
    }

    // find user by email
    @GetMapping("/{email}")
    public ResponseEntity<User> getUserByEmail(@PathVariable String email){
        User user = userRepo.findByEmail(email);
        if (user == null){
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(user);
    }

    // find user by id
```



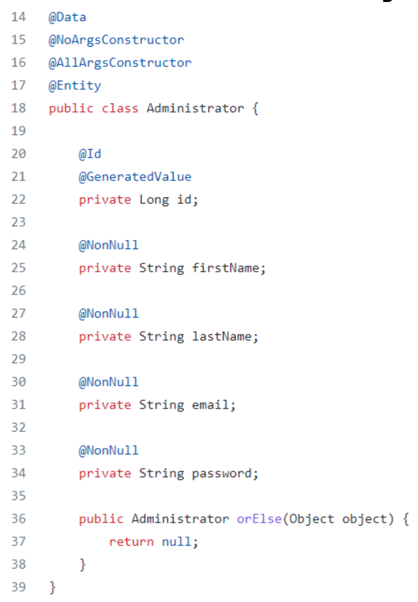
```
        if (user == null){
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(user);
    }

    // find user by id
    @GetMapping("/id/{id}")
    public ResponseEntity<User> getUserById(@PathVariable Long id){
        User user = userRepo.findById(id).orElse(null);
        if (user == null){
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(user);
    }

    // update first name and lastname of user
    @PatchMapping("/id/{id}")
    public ResponseEntity<User> updateUser(@PathVariable Long id, @RequestBody User updateUser){
        User existingUser = userRepo.findById(id).orElse(null);
        if (existingUser == null) {
            return ResponseEntity.notFound().build();
        }
        if (updateUser.getFirstName() != null) {
            existingUser.setFirstName(updateUser.getFirstName());
        }
        if (updateUser.getLastName() != null) {
            existingUser.setLastName(updateUser.getLastName());
        }
        if (updateUser.getPassword() != null) {
            existingUser.setPassword(updateUser.getPassword());
        }
        userRepo.save(existingUser);
        return ResponseEntity.ok(existingUser);
    }
```



```
    @DeleteMapping("/{id}")
    public ResponseEntity<User> deleteUser(@PathVariable Long id){
        User user = userRepo.findById(id).orElse(null);
        System.out.println(user);
        List<User> all = userRepo.findAll();
        System.out.println(all);
        if (user == null) {
            return ResponseEntity.notFound().build();
        }
        userRepo.delete(user);
        return ResponseEntity.ok(user);
    }

    // while returnnig book
    // @DeleteMapping("/{userId}/borrow/{bookId}")
    // public ResponseEntity<User> returnBook(@PathVariable Long userId, @PathVariable Long bookId){
    //     User user = userRepo.findById(userId).orElse(null);
    //     // Book book = bookRepo.findById(bookId);
    //     Book book = bookRepo.findById(bookId).orElse(null);
    //     if(  user == null || book == null ){
    //         return ResponseEntity.notFound().build();
    //     }
    //     // user.getBooks().remove(book);
    //     userRepo.save(user);
    //     return ResponseEntity.ok(user);
    // }



}
```

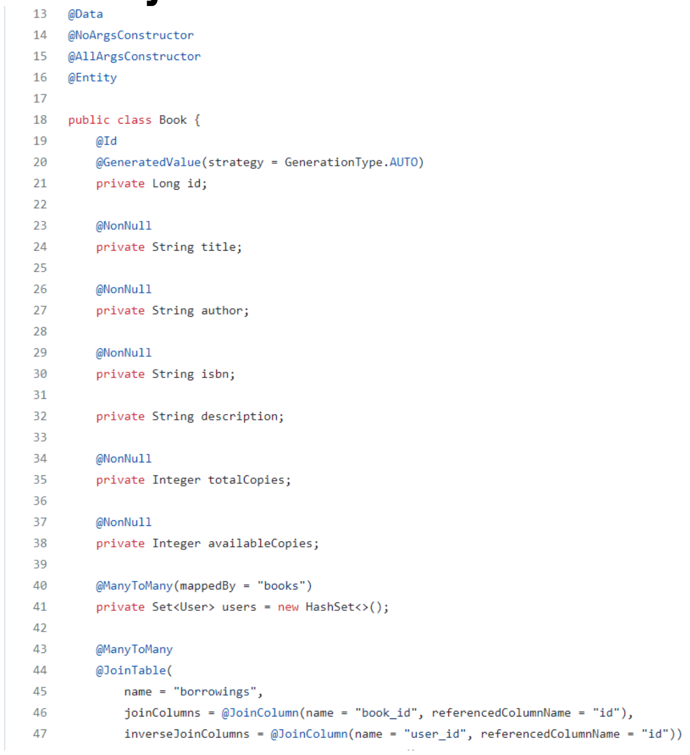
b.MODEL

i. Administrator.java



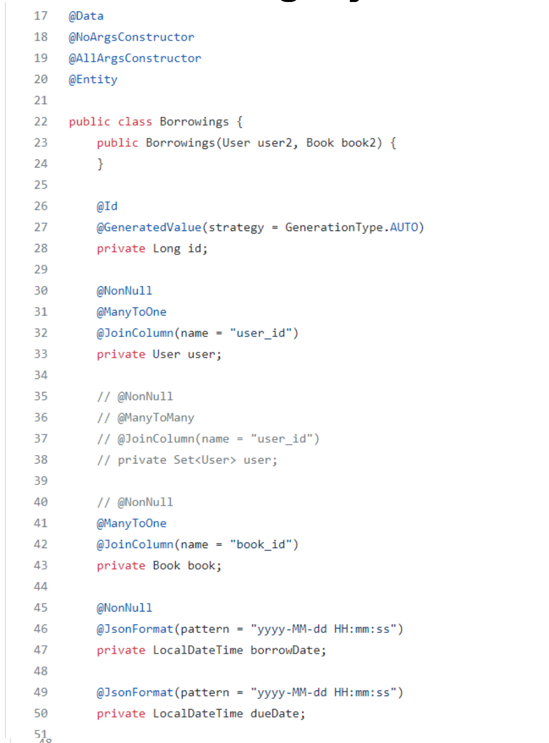
```java
@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
public class Administrator {

    @Id
    @GeneratedValue
    private Long id;

    @NonNull
    private String firstName;

    @NonNull
    private String lastName;

    @NonNull
    private String email;

    @NonNull
    private String password;

    public Administrator orElse(Object object) {
        return null;
    }
}
```

ii. Book.java

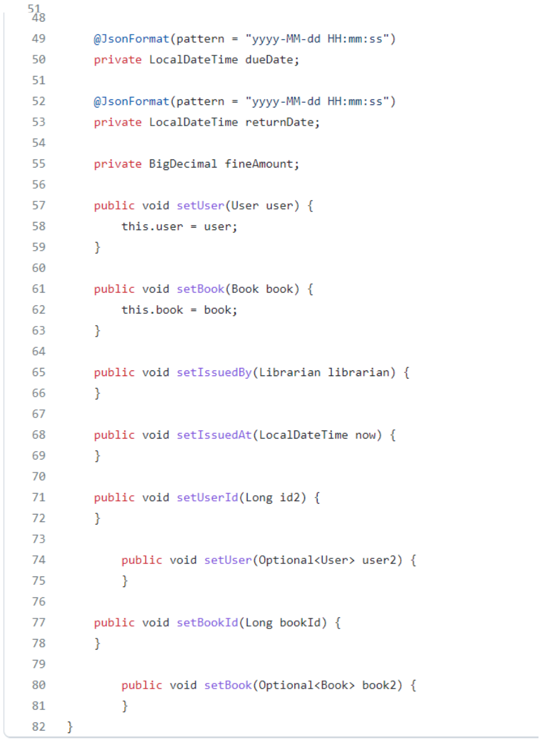


```java
@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity

public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @NonNull
    private String title;

    @NonNull
    private String author;

    @NonNull
    private String isbn;

    private String description;

    @NonNull
    private Integer totalCopies;

    @NonNull
    private Integer availableCopies;

    @ManyToMany(mappedBy = "books")
    private Set<User> users = new HashSet<>();

    @ManyToMany
    @JoinTable(
        name = "borrowings",
        joinColumns = @JoinColumn(name = "book_id", referencedColumnName = "id"),
        inverseJoinColumns = @JoinColumn(name = "user_id", referencedColumnName = "id"))
```

iii. Borrowings.java

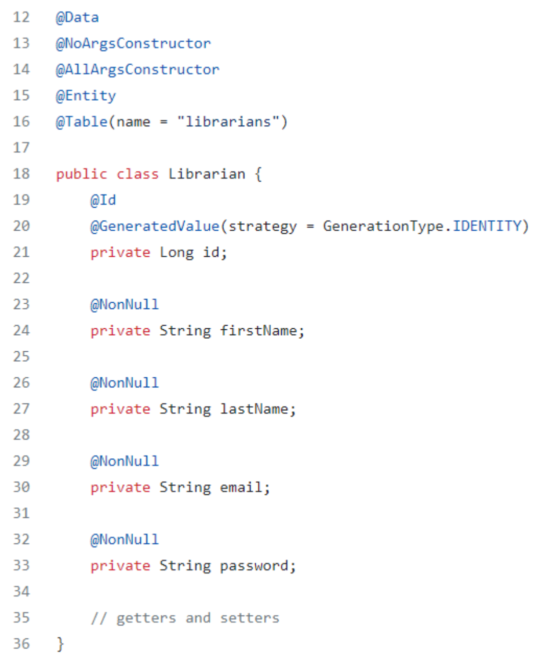


```java
@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity

public class Borrowings {
    public Borrowings(User user2, Book book2) {
    }

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @NonNull
    @ManyToOne
    @JoinColumn(name = "user_id")
    private User user;

    // @NonNull
    // @ManyToMany
    // @JoinColumn(name = "user_id")
    // private Set<User> user;

    // @NonNull
    @ManyToOne
    @JoinColumn(name = "book_id")
    private Book book;

    @NonNull
    @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")
    private LocalDateTime borrowDate;

    @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")
    private LocalDateTime dueDate;
```



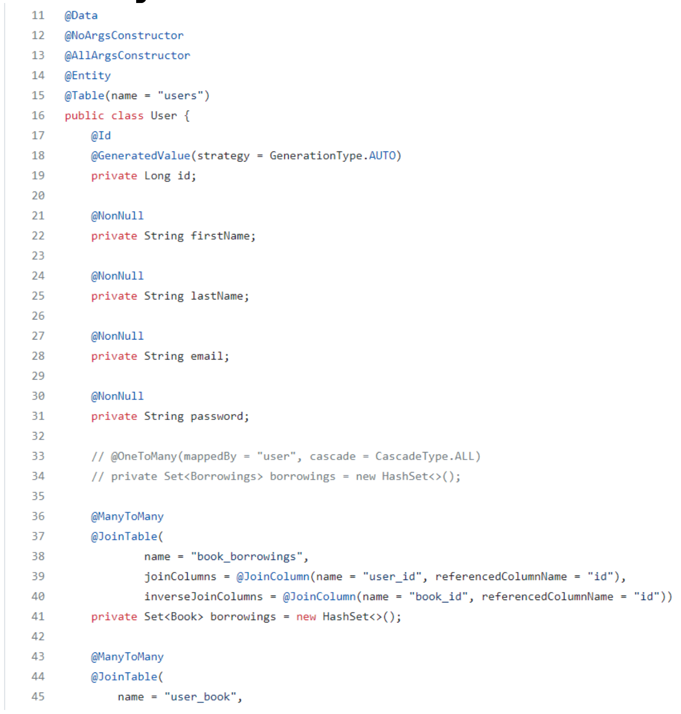
```java

    @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")
    private LocalDateTime dueDate;

    @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")
    private LocalDateTime returnDate;

    private BigDecimal fineAmount;

    public void setUser(User user) {
        this.user = user;
    }

    public void setBook(Book book) {
        this.book = book;
    }

    public void setIssuedBy(Librarian librarian) {
    }

    public void setIssuedAt(LocalDateTime now) {
    }

    public void setUserId(Long id2) {
    }

        public void setUser(Optional<User> user2) {
        }

    public void setBookId(Long bookId) {
    }

        public void setBook(Optional<Book> book2) {
        }
}
```

iv. Librarian.java

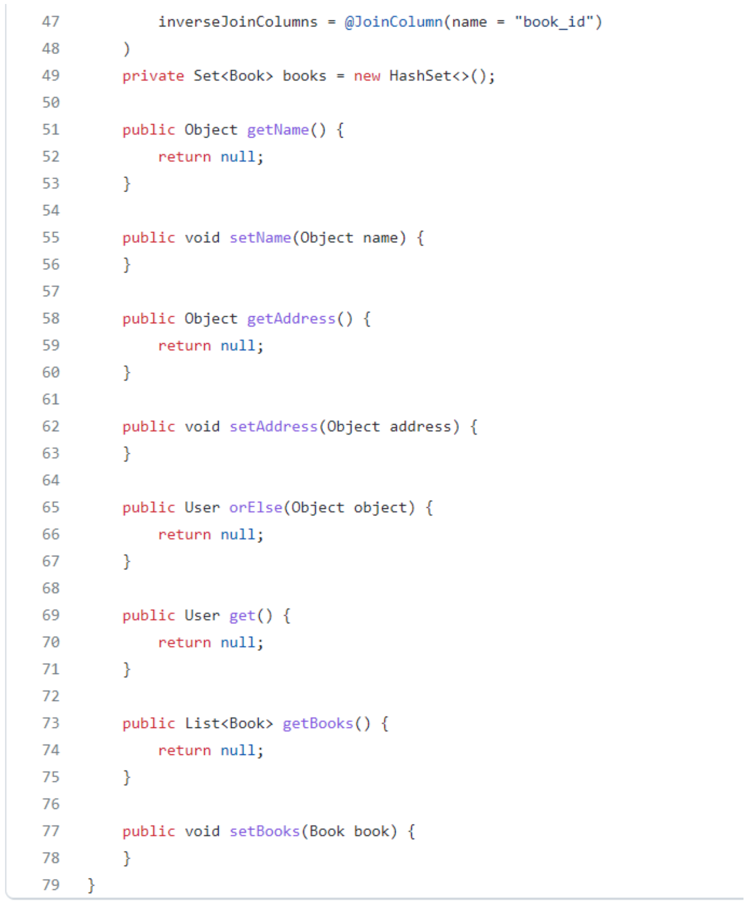


```java
@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "librarians")

public class Librarian {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NonNull
    private String firstName;

    @NonNull
    private String lastName;

    @NonNull
    private String email;

    @NonNull
    private String password;

    // getters and setters
}
```

v. User.java



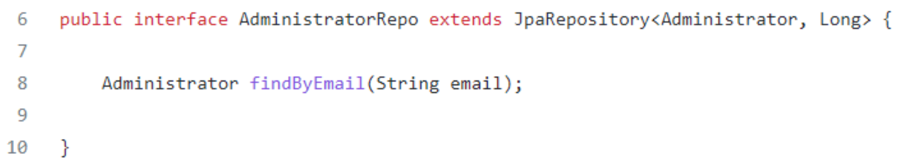
```java
@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "users")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @NonNull
    private String firstName;

    @NonNull
    private String lastName;

    @NonNull
    private String email;

    @NonNull
    private String password;

    // @OneToMany(mappedBy = "user", cascade = CascadeType.ALL)
    // private Set<Borrowings> borrowings = new HashSet<>();

    @ManyToMany
    @JoinTable(
            name = "book_borrowings",
            joinColumns = @JoinColumn(name = "user_id", referencedColumnName = "id"),
            inverseJoinColumns = @JoinColumn(name = "book_id", referencedColumnName = "id"))
    private Set<Book> borrowings = new HashSet<>();

    @ManyToMany
    @JoinTable(
        name = "user_book",
```



```
            inverseJoinColumns = @JoinColumn(name = "book_id")
        )
        private Set<Book> books = new HashSet<>();

        public Object getName() {
            return null;
        }

        public void setName(Object name) {
        }

        public Object getAddress() {
            return null;
        }

        public void setAddress(Object address) {
        }

        public User orElse(Object object) {
            return null;
        }

        public User get() {
            return null;
        }

        public List<Book> getBooks() {
            return null;
        }

        public void setBooks(Book book) {
        }
    }
```

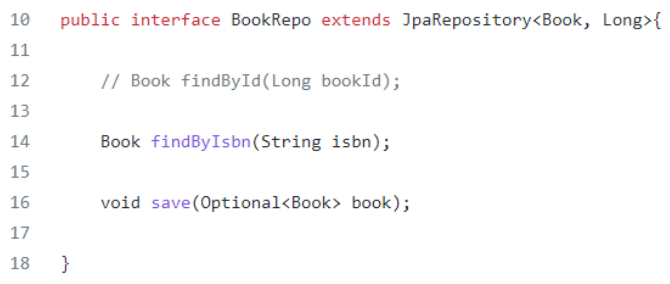
c. REPOSITORY

i. AdministratorRepo.java



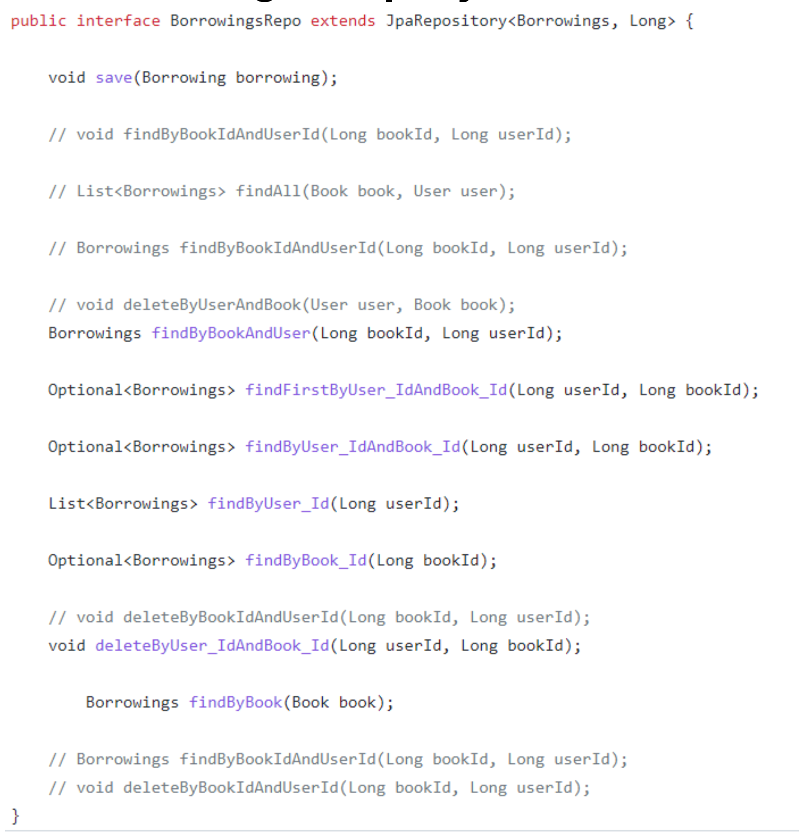
```
public interface AdministratorRepo extends JpaRepository<Administrator, Long> {

    Administrator findByEmail(String email);

}
```

ii. BookRepo.java



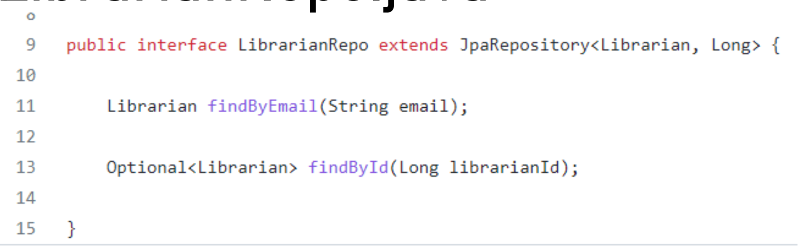
```
public interface BookRepo extends JpaRepository<Book, Long>{

    // Book findById(Long bookId);

    Book findByIsbn(String isbn);

    void save(Optional<Book> book);

}
```

iii. BorrowingsRepo.java



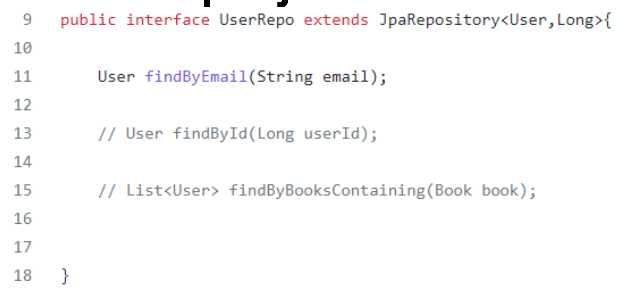
```java
public interface BorrowingsRepo extends JpaRepository<Borrowings, Long> {

    void save(Borrowing borrowing);

    // void findByBookIdAndUserId(Long bookId, Long userId);

    // List<Borrowings> findAll(Book book, User user);

    // Borrowings findByBookIdAndUserId(Long bookId, Long userId);

    // void deleteByUserAndBook(User user, Book book);
    Borrowings findByBookAndUser(Long bookId, Long userId);

    Optional<Borrowings> findFirstByUser_IdAndBook_Id(Long userId, Long bookId);

    Optional<Borrowings> findByUser_IdAndBook_Id(Long userId, Long bookId);

    List<Borrowings> findByUser_Id(Long userId);

    Optional<Borrowings> findByBook_Id(Long bookId);

    // void deleteByBookIdAndUserId(Long bookId, Long userId);
    void deleteByUser_IdAndBook_Id(Long userId, Long bookId);

        Borrowings findByBook(Book book);

    // Borrowings findByBookIdAndUserId(Long bookId, Long userId);
    // void deleteByBookIdAndUserId(Long bookId, Long userId);
}
```

iv. LibrarianRepo.java



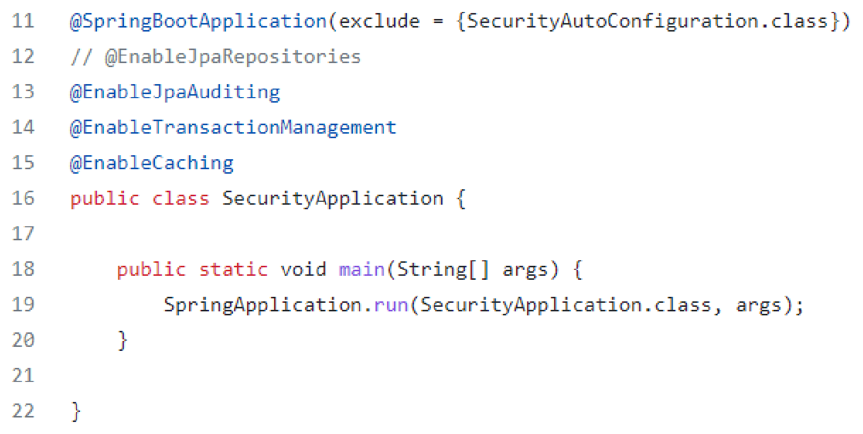
```java
public interface LibrarianRepo extends JpaRepository<Librarian, Long> {

    Librarian findByEmail(String email);

    Optional<Librarian> findById(Long librarianId);

}
```

v. UserRepo.java



```java
public interface UserRepo extends JpaRepository<User,Long>{

    User findByEmail(String email);

    // User findById(Long userId);

    // List<User> findByBooksContaining(Book book);


}
```

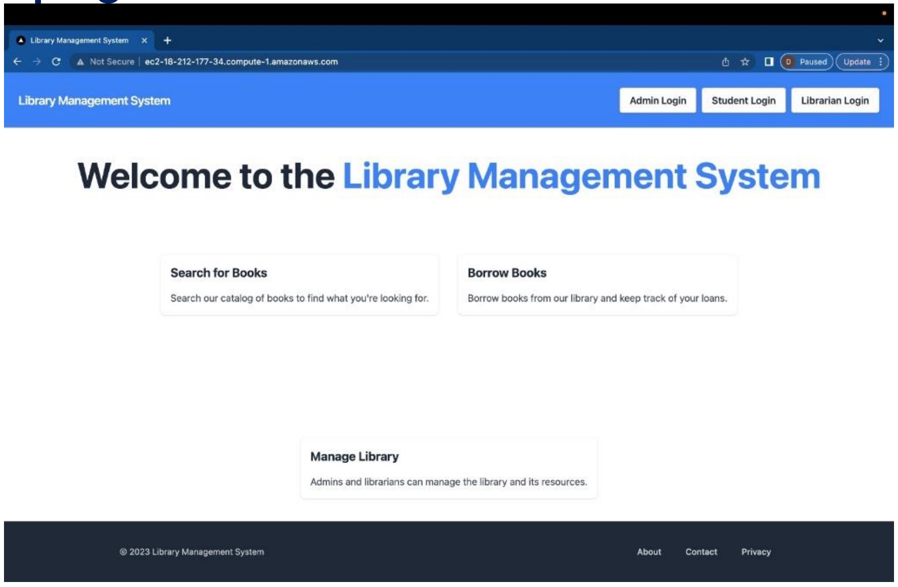
d.MAIN APPLICATION



```java
@SpringBootApplication(exclude = {SecurityAutoConfiguration.class})
// @EnableJpaRepositories
@EnableJpaAuditing
@EnableTransactionManagement
@EnableCaching
public class SecurityApplication {

    public static void main(String[] args) {
        SpringApplication.run(SecurityApplication.class, args);
    }

}
```

VI. DEMO

Home page:


Library Management System

Admin Login

Student Login

Librarian Login

Welcome to the Library Management System

Search for Books

Search our catalog of books to find what you're looking for.

Borrow Books

Borrow books from our library and keep track of your loans.

Manage Library

Admins and librarians can manage the library and its resources.

© 2023 Library Management System

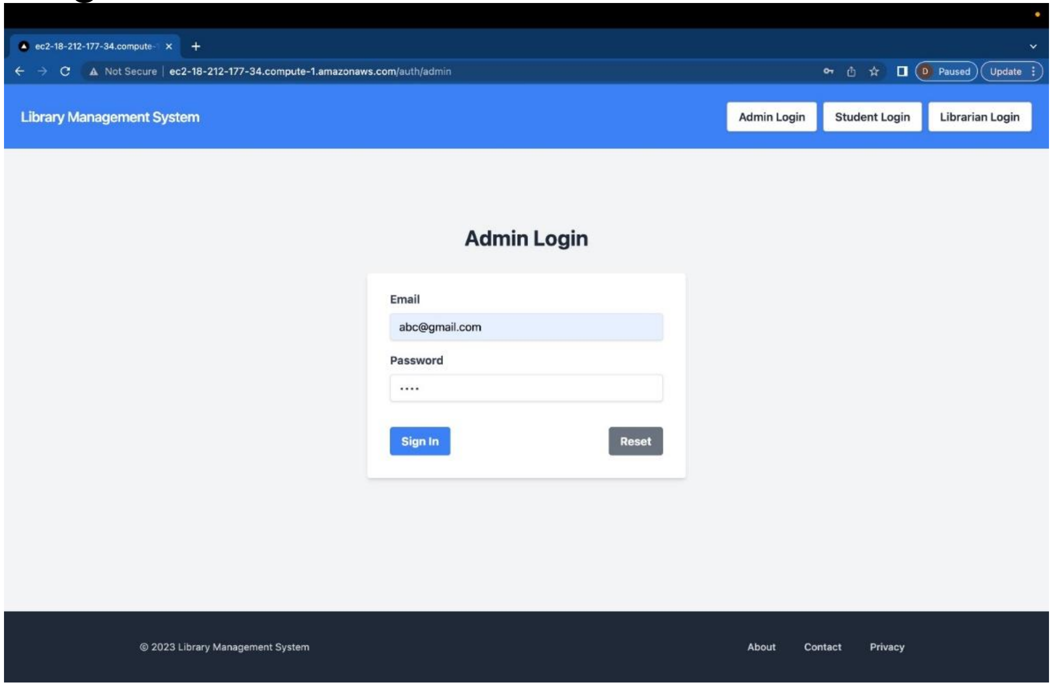
About

Contact

Privacy

a. ADMIN

i. Login


Library Management System

Admin Login

Student Login

Librarian Login

Admin Login

Email

abc@gmail.com

Password

••••

Sign In

Reset

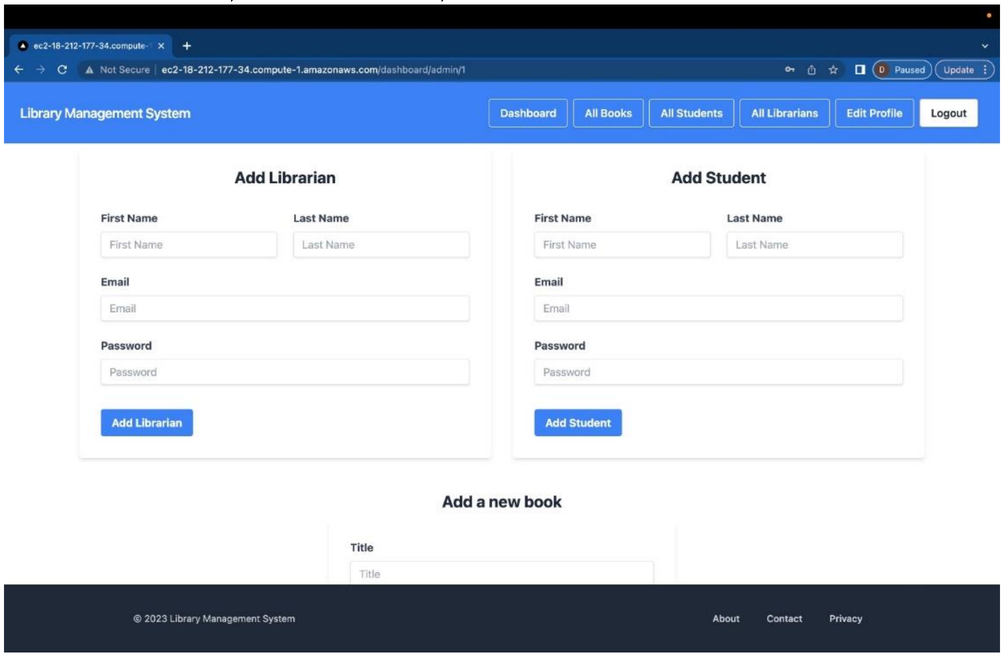
© 2023 Library Management System

About

Contact

Privacy

ii. Dashboard-Add librarian,student,book



ec2-18-212-177-34.compute-1.amazonaws.com/dashboard/admin/1

Library Management System

Dashboard | All Books | All Students | All Librarians | Edit Profile | Logout

Add Librarian

First Name: First Name
Last Name: Last Name
Email: Email
Password: Password

Add Librarian

Add Student

First Name: First Name
Last Name: Last Name
Email: Email
Password: Password

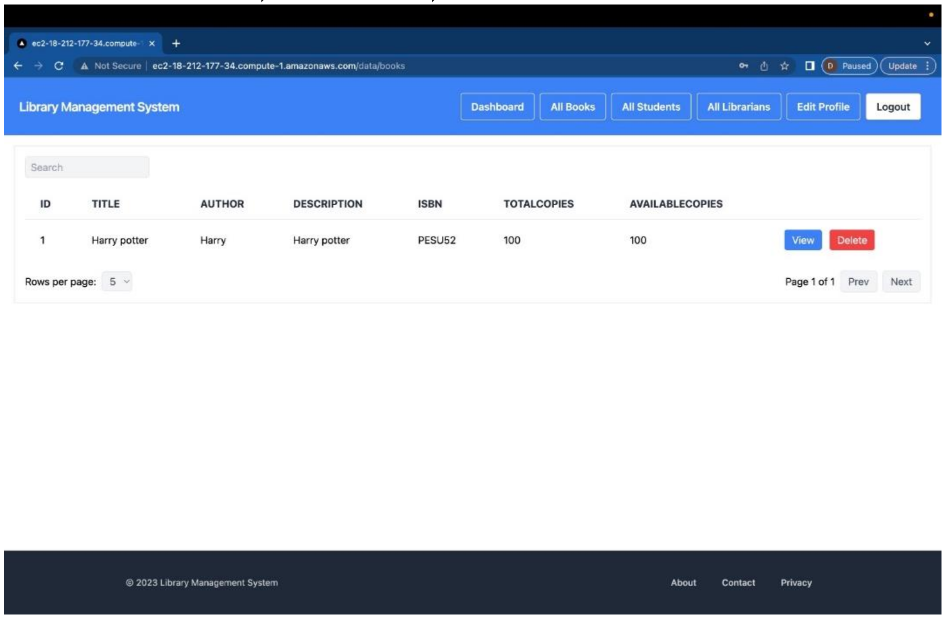
Add Student

Add a new book

Title: Title

© 2023 Library Management System — About | Contact | Privacy

iii. Admin can view- all librarians,books,students



ec2-18-212-177-34.compute-1.amazonaws.com/data/books

Library Management System

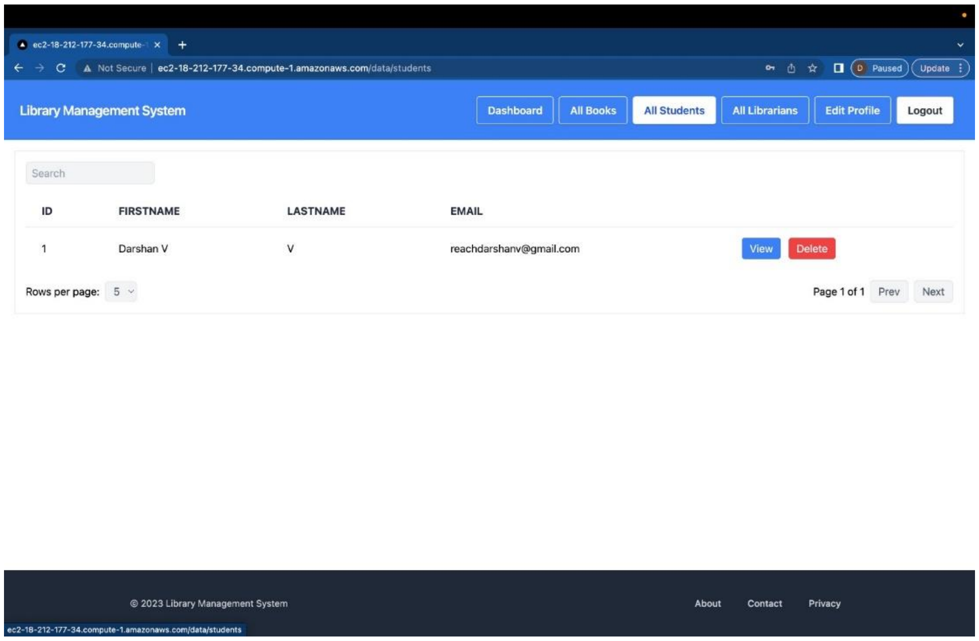
Dashboard | All Books | All Students | All Librarians | Edit Profile | Logout

Search

ID	TITLE	AUTHOR	DESCRIPTION	ISBN	TOTALCOPIES	AVAILABLECOPIES	
1	Harry potter	Harry	Harry potter	PESU52	100	100	View Delete

Rows per page: 5 — Page 1 of 1 Prev Next

© 2023 Library Management System — About | Contact | Privacy



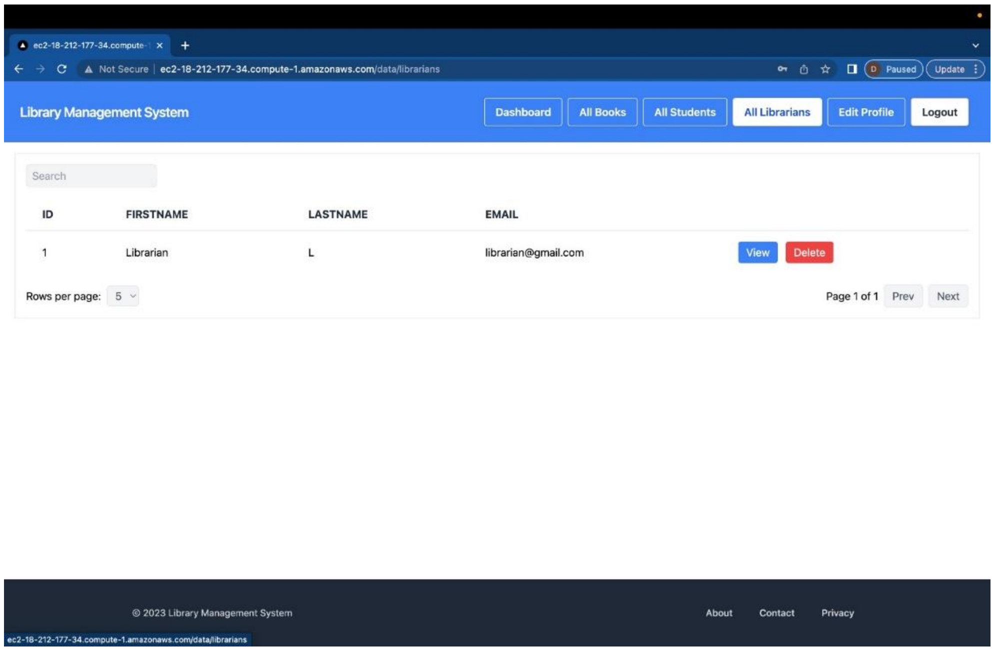
ec2-18-212-177-34.compute-1.amazonaws.com/data/students

Library Management System

ID	FIRSTNAME	LASTNAME	EMAIL	
1	Darshan V	V	reachdarshanv@gmail.com	View Delete

Rows per page: 5 Page 1 of 1 Prev Next

© 2023 Library Management System About Contact Privacy



ec2-18-212-177-34.compute-1.amazonaws.com/data/librarians

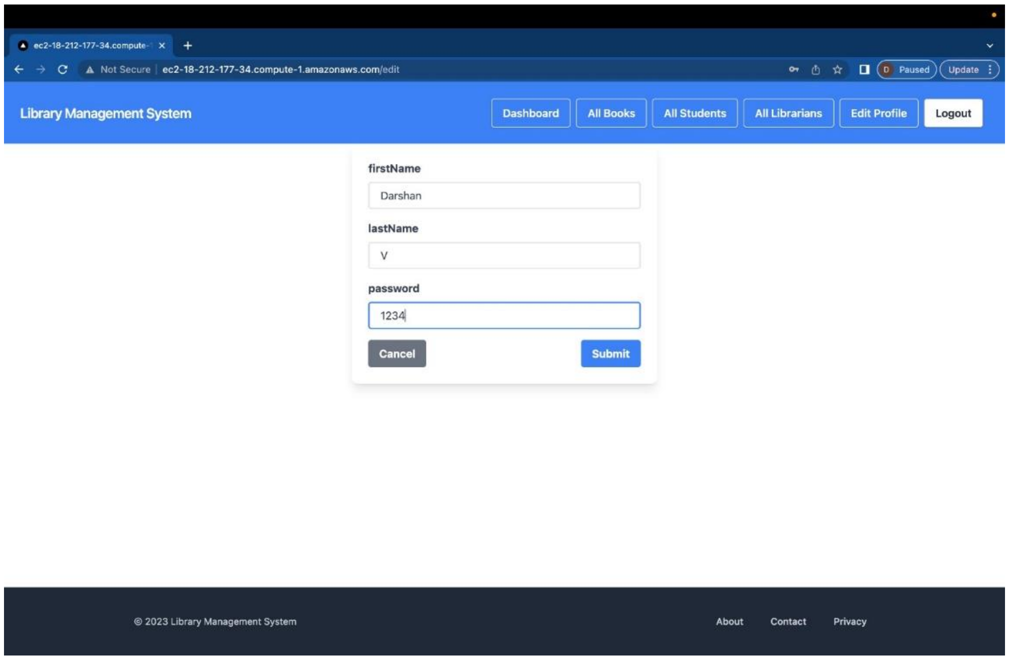
Library Management System

ID	FIRSTNAME	LASTNAME	EMAIL	
1	Librarian	L	librarian@gmail.com	View Delete

Rows per page: 5 Page 1 of 1 Prev Next

© 2023 Library Management System About Contact Privacy

iv. Edit details


ec2-18-212-177-34.compute-1.amazonaws.com/edit
Library Management System
Dashboard
All Books
All Students
All Librarians
Edit Profile
Logout
Cancel
Submit
© 2023 Library Management System
About
Contact
Privacy

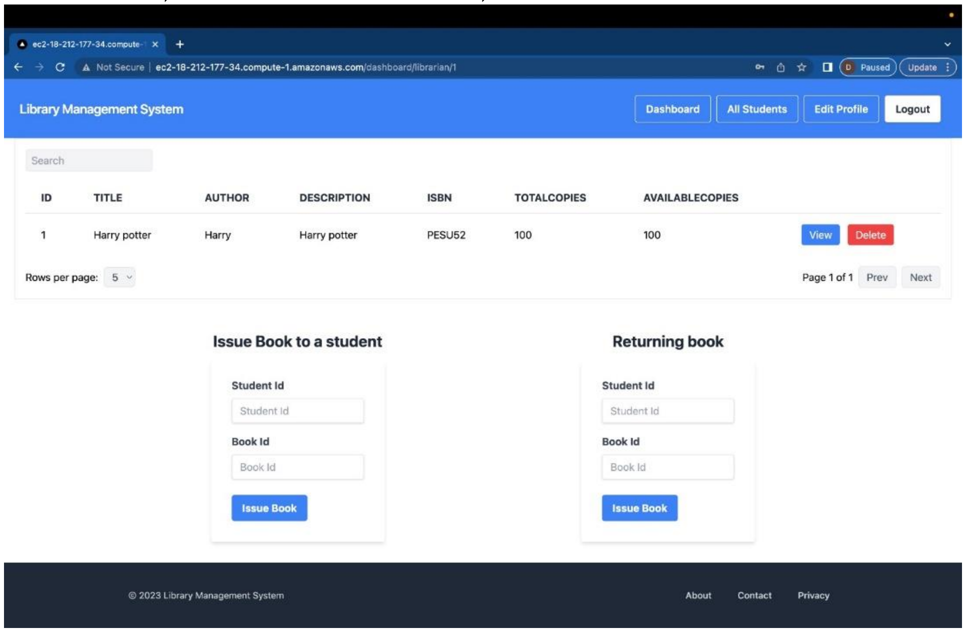
firstName: Darshan

lastName: V

password: 1234

b.LIBRARIAN

i. Dashboard-Add books,IssueBooks,Return book


ec2-18-212-177-34.compute-1.amazonaws.com/dashboard/librarian/1
Library Management System
Dashboard
All Students
Edit Profile
Logout
Search
Rows per page: 5
Page 1 of 1
Prev
Next
© 2023 Library Management System
About
Contact
Privacy

ID	TITLE	AUTHOR	DESCRIPTION	ISBN	TOTALCOPIES	AVAILABLECOPIES	
1	Harry potter	Harry	Harry potter	PESU52	100	100	View Delete

Issue Book to a student

Student Id

Book Id

Issue Book

Returning book

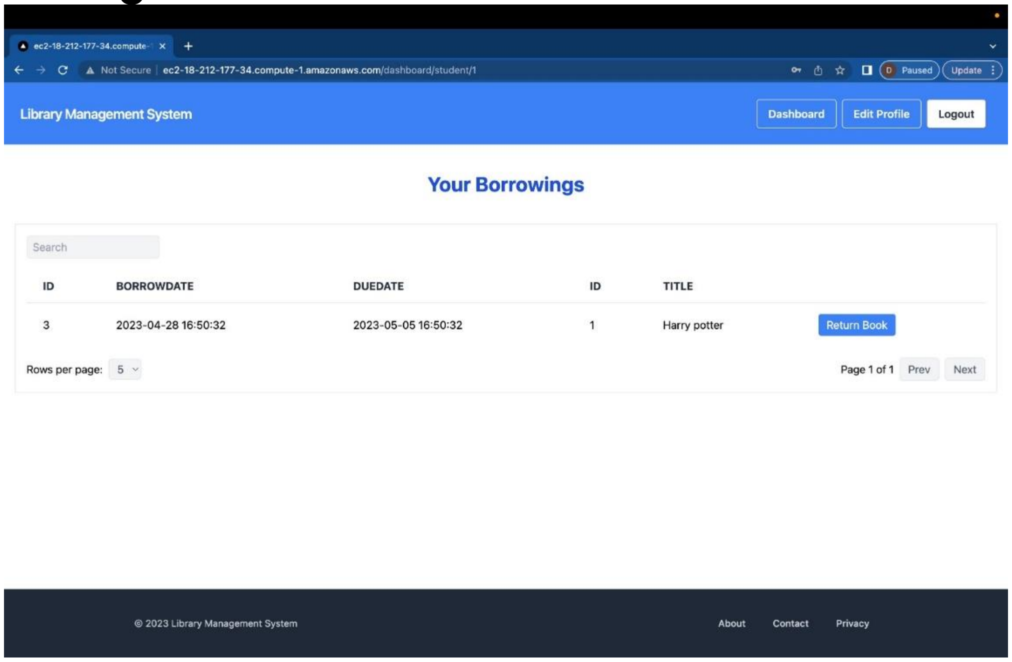
Student Id

Book Id

Issue Book

c. STUDENT

i. Dashboard – See all borrowings along with due date and return book


ec2-18-212-177-34.compute-
Not Secure | ec2-18-212-177-34.compute-1.amazonaws.com/dashboard/student/1
Paused
Update
Library Management System
Dashboard
Edit Profile
Logout

Your Borrowings

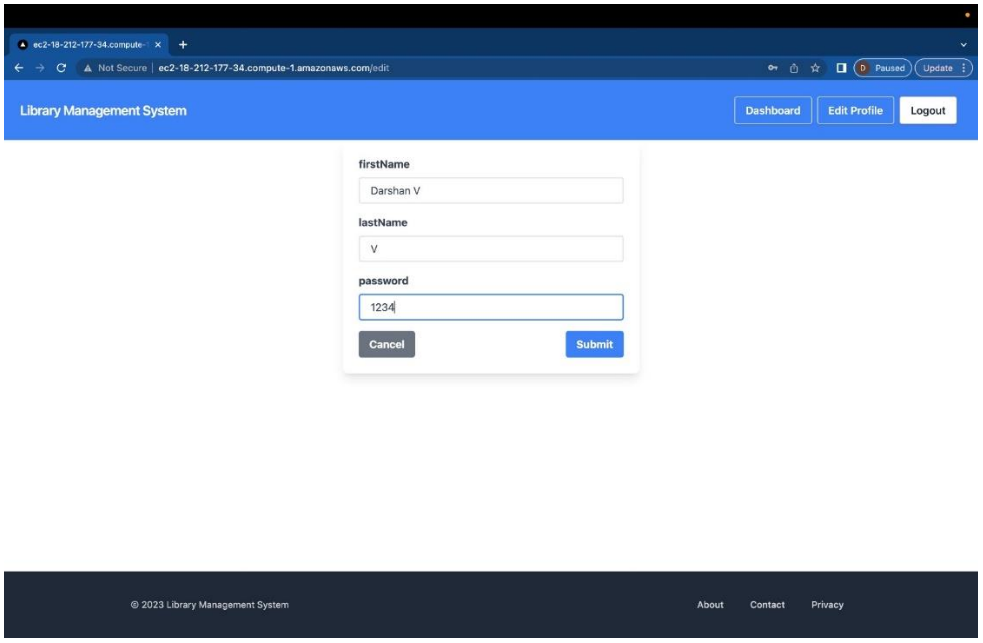
Search

ID	BORROWDATE	DUEDATE	ID	TITLE	
3	2023-04-28 16:50:32	2023-05-05 16:50:32	1	Harry potter	Return Book

Rows per page: 5 Page 1 of 1 Prev Next

© 2023 Library Management System About Contact Privacy

ii. Edit details


ec2-18-212-177-34.compute-
Not Secure | ec2-18-212-177-34.compute-1.amazonaws.com/edit
Paused
Update
Library Management System
Dashboard
Edit Profile
Logout

firstName: Darshan V

lastName: V

password: 1234

Cancel Submit

© 2023 Library Management System About Contact Privacy