

***Mini project report on***

Library Management System

*Submitted in partial fulfilment of the requirements for the award of degree of*

**Bachelor of Technology**

**in**

**Computer Science & Engineering**

**UE20CS352 –OOADJ Project**

***Submitted by:***

|  |  |
| --- | --- |
| **Pranav M**  **Raghavendra A K**  **Rudresh S Patil**  **Harsha N** | **PES2UG20CS535**  **PES2UG20CS537**  **PES2UG20CS540**  **PES2UG20CS580** |

Under the guidance of

**Prof. Nivedita Kasturi**

Assistant Professor

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

FACULTY OF ENGINEERING

**PES UNIVERSITY**



# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Title** | **Page No.** |
|  | **INTRODUCTION** | **3** |
|  | **PROBLEM DEFINITION** | **4** |
|  | **USECASE MODELING** | **5** |
|  | **CLASS MODELING** | **6** |
|  | **ACTIVITY MODELING** | **7** |
|  | **IMPLEMENTATION** | **8** |
|  | **RESULTS SCREENSHOTS** | **18** |

NOTE: Please add appropriate description for all diagrams where ever required. Only important class implementation needs to be added to IMPLEMNTATION SECTION.

**INTRODUCTION**

The library management project is a comprehensive software solution that aims to provide a streamlined and efficient system for managing the day-to-day operations of a library. The project was designed and developed using Java programming language with the help of Model-View-Controller (MVC) architecture, and implemented with the help of various design patterns including creational, structural, and behavioural patterns.

The main objective of this project is to provide a user-friendly interface for library staff to manage the library's resources, including books, and borrowings. The system also allows for tracking the availability and status of books, as well as keeping track of borrowing history and due dates.

The project's design was developed using MVC architecture, which separates the system's components into three distinct layers: Model, View, and Controller. The project also implemented several design patterns to ensure its scalability, reusability, and maintainability. These include the Singleton pattern, which was used to ensure a single instance of the database connection throughout the project's runtime, as well as the Factory pattern, which was used to create instances of database access objects.

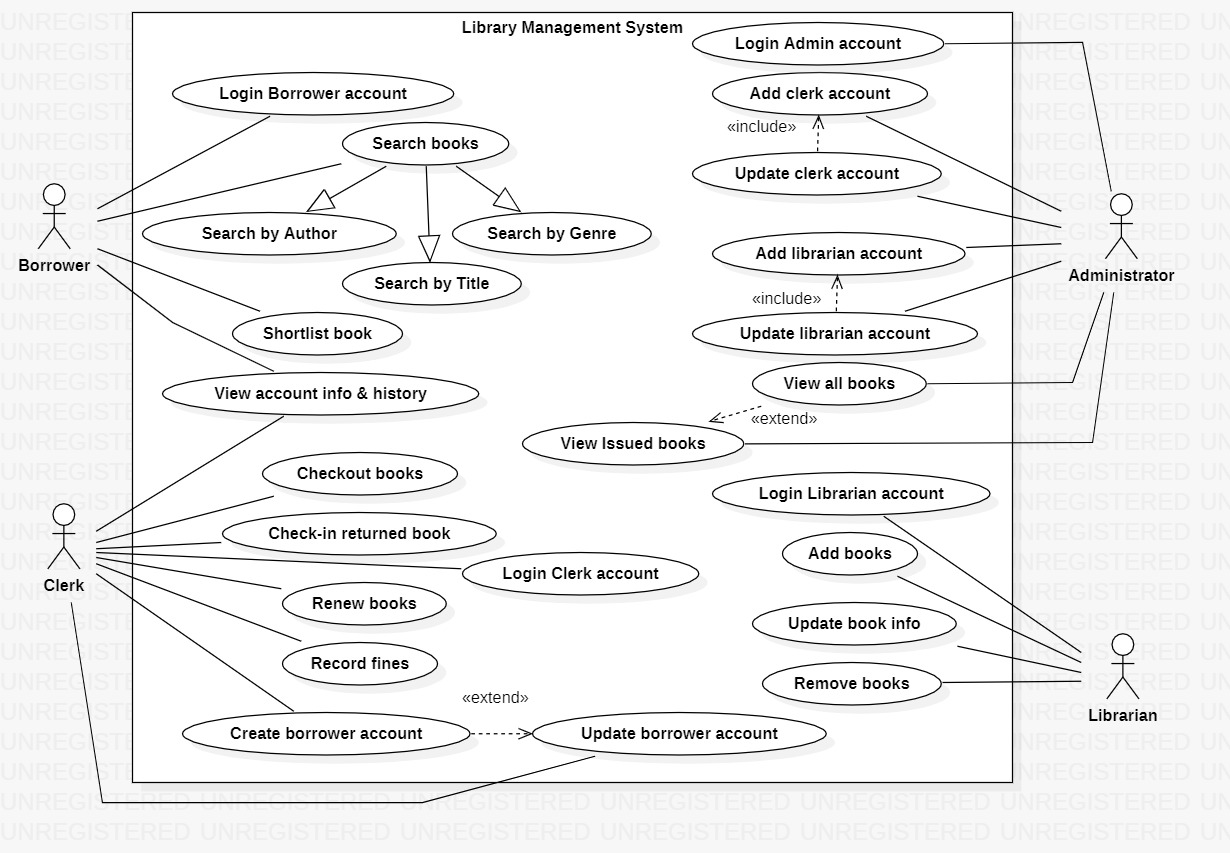
**PROBLEM DEFINITION**

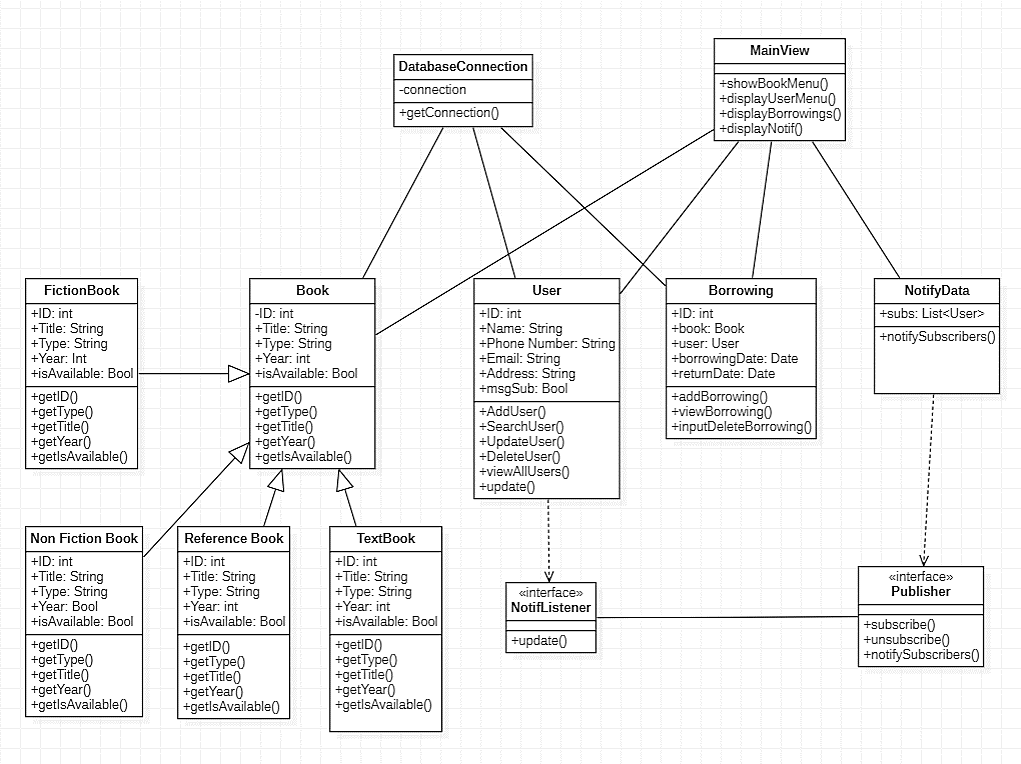
Libraries are an essential part of educational and cultural institutions, providing access to information and resources for learning, research, and leisure. However, managing a library's resources can be a complex and time-consuming task, especially for larger libraries with extensive collections.

One of the significant challenges faced by libraries is keeping track of their resources, including books, journals, and other materials. Libraries must also manage patrons' borrowing activity, including loan periods, renewals, and overdue fines.

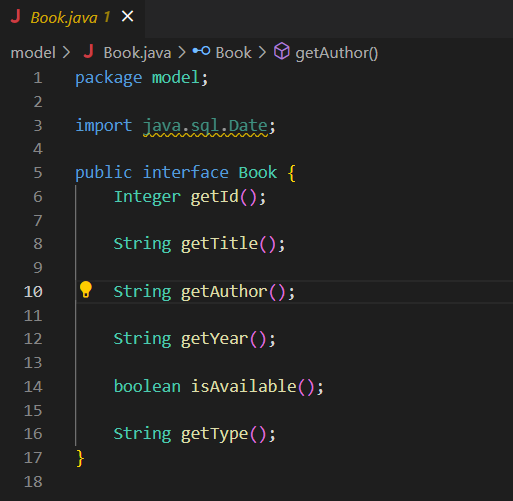
Traditional methods of managing library resources, such as manual record-keeping and paper-based systems, can be inefficient and error-prone, leading to lost or misplaced books and inaccurate records. Additionally, such systems can make it challenging to keep track of overdue books, resulting in lost revenue and reduced user satisfaction.

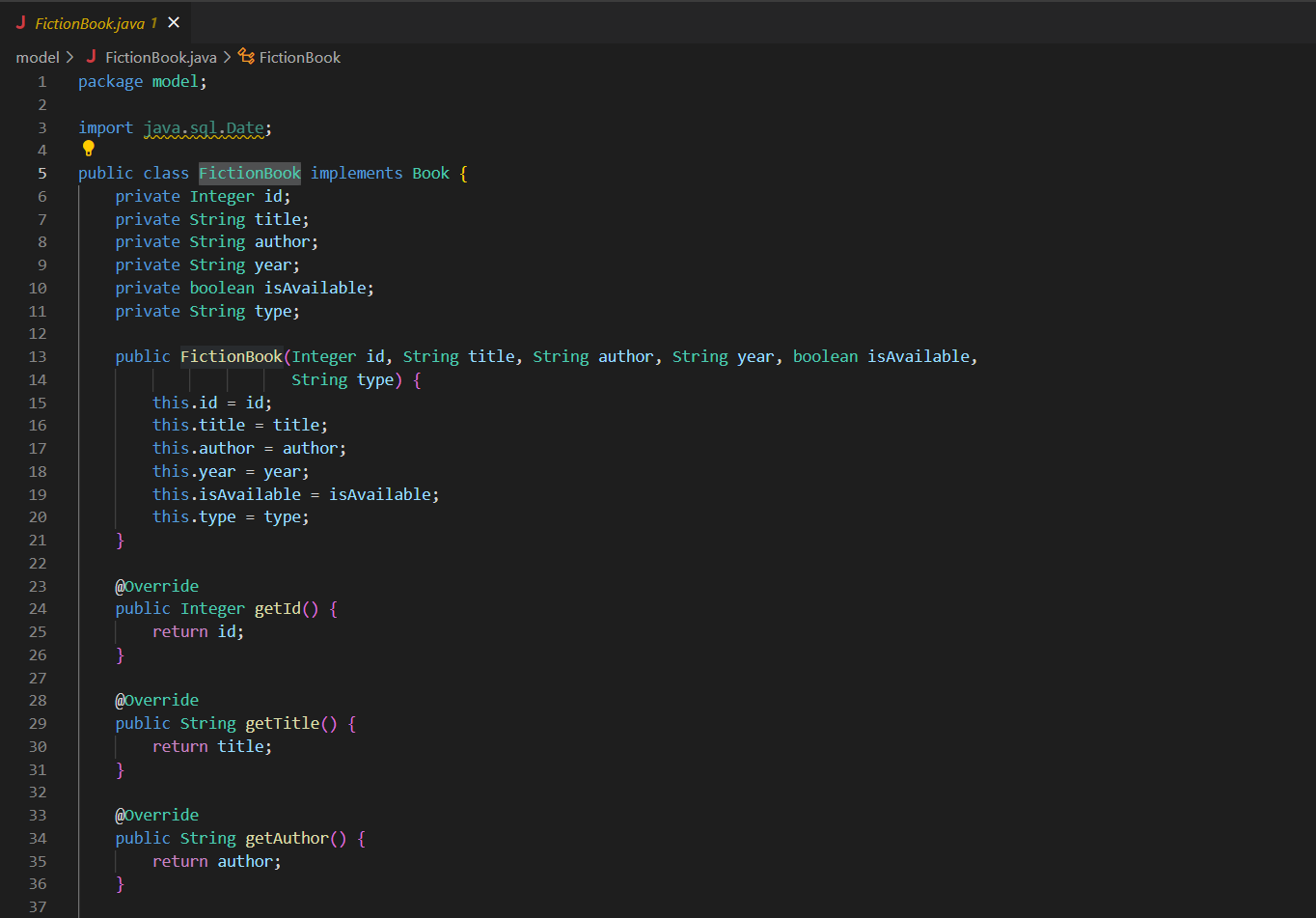
Therefore, there is a need for a comprehensive and efficient software solution to manage library resources, including books, patrons, and borrowing activity. This project aims to provide a solution to this problem by implementing a library management system using Java programming language with the help of Model-View-Controller (MVC) architecture, and implemented with the help of various design patterns including creational, structural, and behavioral patterns.

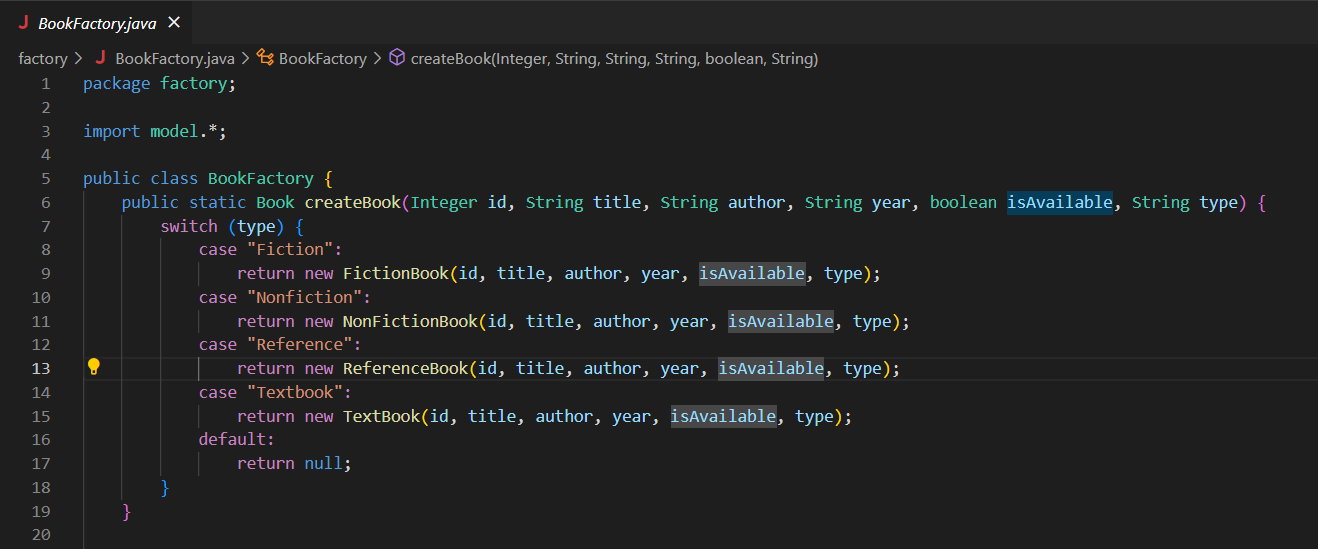
**USECASE MODELING**

**CLASS MODELING **

**ACTIVITY MODELING**

**IMPLEMENTATION**

****

****

**NotifyData.java**

package observer;

import java.util.ArrayList;

import java.util.List;

import dao.UserDAO;

import model.User;

public class NotifyData implements Publisher {

    private static List<notifListener> subs;

    UserDAO userDao = new UserDAO();

    public NotifyData(){

        subs = new ArrayList<>();

    }

    // Observer

    public void subscribe(notifListener e) {

        subs.add(e);

        System.out.println("User successfully subscribed!");

    }

    public void unsubscribe(notifListener e) {

        subs.remove(e);

        System.out.println("User successfully unsubscribed!");

    }

    public void notifySubs(){

        System.out.println(subs);

        List<User> subs = userDao.FetchSubs();

        System.out.printf("+--------+------------------+----------------------+------------------------+\n");

        System.out.printf("| %6s | %16s | %20s | %22s |\n","ID","Name","Email","Phone Number");

        System.out.printf("+--------+------------------+----------------------+------------------------+\n");

        for(User user : subs){

            user.update();

            System.out.printf("| %6d | %16s | %20s | %22s |\n",

                    user.getID(), user.getName(), user.getEmail(), user.getPhoneNumber());

        }

        System.out.printf("+--------+------------------+----------------------+------------------------+\n");

        System.out.println("Users successfully notified!");

    };

    public void displayNotif(){

        System.out.println("Showing notifications");

        notifySubs();

    }

}

**DatabaseConnection.java**

package dao;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DatabaseConnection {

    private static Connection connection;

    private static final String url = "jdbc:mysql://localhost:3306/librarydb";

    private static final String username = "root";

    private static final String password = "password";

    private DatabaseConnection() {}

    public static Connection getConnection() {

        if (connection == null) {

            try {

                connection = DriverManager.getConnection(url, username, password);

            } catch (SQLException e) {

                System.out.println("Error connecting to MySQL database");

                e.printStackTrace();

            }

        }

        return connection;

    }

}

**Main.java**

import java.sql.SQLException;

import java.text.ParseException;

import java.util.Scanner;

import controller.BookController;

import controller.UserController;

import view.BookView;

import observer.NotifyData;

import view.BorrowingView;

import view.UserView;

public class Main {

    public static void main(String[] args) throws ParseException, SQLException {

        UserController userController = new UserController();

        UserView userView = new UserView(userController);

        BookController bookController = new BookController();

        BookView bookView = new BookView(bookController);

        BorrowingView borrowingView = new BorrowingView();

        NotifyData notifyData = new NotifyData();

        Scanner scanner = new Scanner(System.in);

        boolean exit = false;

        while (!exit) {

            System.out.println("Choose an option:");

            System.out.println("1. Manage Books");

            System.out.println("2. Manage users");

            System.out.println("3. Manage borrowings");

            System.out.println("4. Show Notifications");

            System.out.println("5. Exit");

            int choice = scanner.nextInt();

            scanner.nextLine();

            switch (choice) {

                case 1:

                    bookView.showBookMenu();

                    break;

                case 2:

                    userView.displayUserMenu();

                    break;

                case 3:

                    borrowingView.displayBorrowings();

                    break;

                case 4:

                    notifyData.displayNotif();

                    break;

                case 5:

                    exit = true;

                    break;

                default:

                    System.out.println("Invalid choice");

                    break;

            }

        }

        scanner.close();

    }

}

**BookView.java**

public class BookView {

    private BookController bookController;

    private Scanner scanner;

    public BookView(BookController bookController) {

        this.bookController = bookController;

        scanner = new Scanner(System.in);

    }

    public void showBookMenu() throws SQLException {

        while (true) {

            System.out.println("1. Add Book");

            System.out.println("2. View All Books");

            System.out.println("3. Delete Book");

            System.out.println("4. Update Book");

            System.out.println("5. Search Book");

            System.out.println("0. Exit");

            System.out.print("Enter your choice: ");

            int choice = scanner.nextInt();

            scanner.nextLine();

            switch (choice) {

                case 1:

                    addBook();

                    break;

                case 2:

                    viewAllBooks();

                    break;

                case 3:

                    deleteBook();

                    break;

                case 4:

                    updateBook();

                    break;

                case 5:

                    searchBooks();

                    break;

                case 0:

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

                    System.exit(0);

                    break;

                default:

                    System.out.println("Invalid choice!");

            }

        }

    }

**UserView.java**

public class UserView {

    private UserController userController;

    private Scanner scanner;

    public UserView(UserController userController) {

        this.userController = userController;

        scanner = new Scanner(System.in);

    }

    public void displayUserMenu() {

        System.out.println("Select an option:");

        System.out.println("1. Add new user");

        System.out.println("2. View all users");

        System.out.println("3. Search user by ID");

        System.out.println("4. Update user");

        System.out.println("5. Delete user");

        System.out.println("6. Exit");

        int choice = scanner.nextInt();

        scanner.nextLine(); // consume leftover newline character

        switch (choice) {

            case 1:

                addUser();

                break;

            case 2:

                viewAllUsers();

                break;

            case 3:

                searchUserById();

                break;

            case 4:

                updateUser();

                break;

            case 5:

                deleteUser();

                break;

            case 6:

                System.exit(0);

            default:

                System.out.println("Invalid choice. Please try again.");

        }

        displayUserMenu(); // continue displaying menu until user exits

    }

**BorrowingView.java**

public class BorrowingView {

    private BorrowingController borrowingController;

    public BorrowingView() {

        borrowingController = new BorrowingController();

    }

    public void displayBorrowings() throws ParseException, SQLException {

        Scanner scanner = new Scanner(System.in);

        System.out.println("==== Borrowings ====");

        System.out.println("1. View All Borrowings");

        System.out.println("2. Add New Borrowing");

        System.out.println("3. Return a Borrowed Book");

        System.out.println("4. View Borrowings by User ID");

        System.out.println("5. Exit");

        while (true) {

            System.out.print("Enter your choice: ");

            int choice = scanner.nextInt();

            switch (choice) {

                case 1:

                    // View all borrowings

                    List<Borrowing> borrowings = borrowingController.getAllBorrowings();

                    displayBorrowingList(borrowings);

                    break;

                case 2:

                    // Add new borrowing

                    Borrowing newBorrowing = readBorrowingData();

                    borrowingController.addBorrowing(newBorrowing);

                    System.out.println("New borrowing added successfully!");

                    break;

                case 3:

                    // Return a borrowed book

                    System.out.print("Enter the ID of the borrowing to return: ");

                    int borrowingID = scanner.nextInt();

                    borrowingController.deleteBorrowing(borrowingID);

                    System.out.println("Borrowed book returned successfully!");

                    break;

                case 4:

                    // Exit

                    System.out.println("Goodbye!");

                    return;

                default:

                    System.out.println("Invalid choice. Please try again.");

            }

        }

    }

**BookController.java**

public class BookController {

    private BookDAO bookDAO;

    private Scanner scanner;

    private BookView bookview;

    public BookController() throws SQLException {

        this.bookDAO = new BookDAOImpl();

        this.scanner = new Scanner(System.in);

        this.bookview = new BookView(this);

    }

    public boolean addBook(Integer id, String title, String author, String year, boolean isAvailable, String type) throws SQLException {

        Book book = BookFactory.createBook(id, title, author, year, isAvailable, type);

        if(book!=null) {

            this.bookDAO.addBook(book);

            return true;

        } else {

            return false;

        }

    }

    public List<Book> getAllBooks() throws SQLException {

        return bookDAO.getAllBooks();

    }

    public void deleteBook(int id) throws SQLException {

        bookDAO.deleteBook(id);

    }

    public void searchBooks(String keyword) throws SQLException {

        try {

            List<Book> books = bookDAO.searchBooks(keyword);

            if (books.isEmpty()) {

                bookview.displayMessage("No books found!");

            } else {

                bookview.displayMessage("Search Results:");

                bookview.displayMessage("ID\tTitle\tAuthor\tYear\tAvailable");

                for (Book book : books) {

                    bookview.displayMessage(book.getId() + "\t" + book.getTitle() + "\t" + book.getAuthor() + "\t" +

                            book.getYear() + "\t" + book.isAvailable());

                }

            }

        } catch (SQLException e) {

            bookview.displayMessage("An error occurred while searching books: " + e.getMessage());

        }

    }

    public void updateBook(int id) {

        try {

            Book book = bookDAO.getBookById(id);

            if (book == null) {

                bookview.displayMessage("Book not found!");

                return;

            }

            bookview.displayMessage("Current book details:");

            bookview.displayBookDetails(book);

            String[] newValues = bookview.getBookUpdates();

            String title = newValues[0].isEmpty() ? book.getTitle() : newValues[0];

            String author = newValues[1].isEmpty() ? book.getAuthor() : newValues[1];

            String year = newValues[2].isEmpty() ? book.getYear() : newValues[2];

            boolean isAvailable = newValues[3].isEmpty() ? book.isAvailable() : Boolean.parseBoolean(newValues[3]);

            String type = newValues[4].isEmpty() ? book.getType() : newValues[4];

            Book newBook = BookFactory.createBook(id, title, author, year, isAvailable, type);

            bookDAO.updateBook(newBook);

            bookview.displayMessage("Book updated successfully!");

        } catch (SQLException e) {

            bookview.displayMessage("An error occurred while updating book: " + e.getMessage());

        }

    }

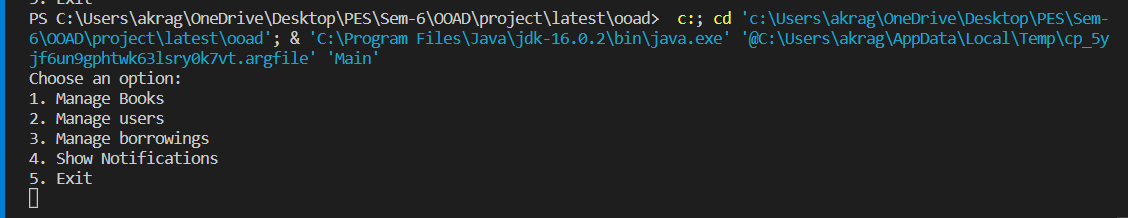
    public Book getBookById(int bookID) throws SQLException {

        return bookDAO.getBookById(bookID);

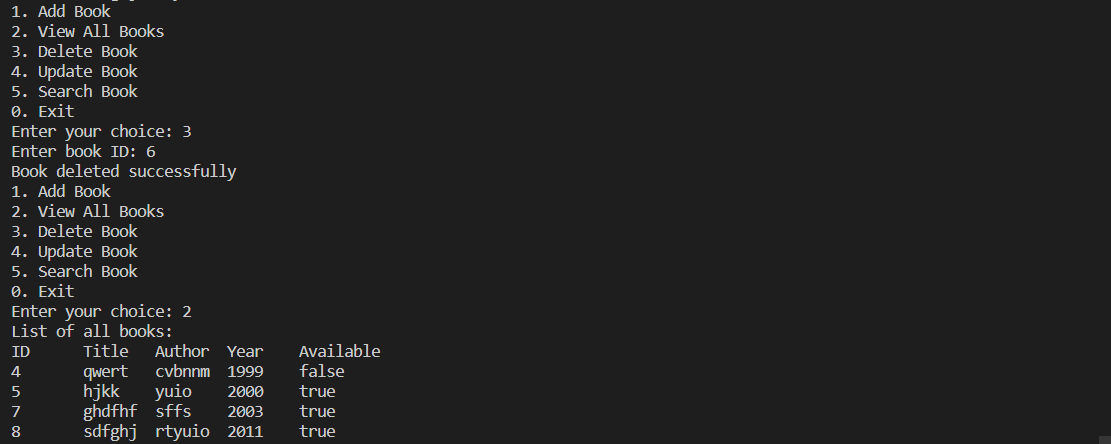
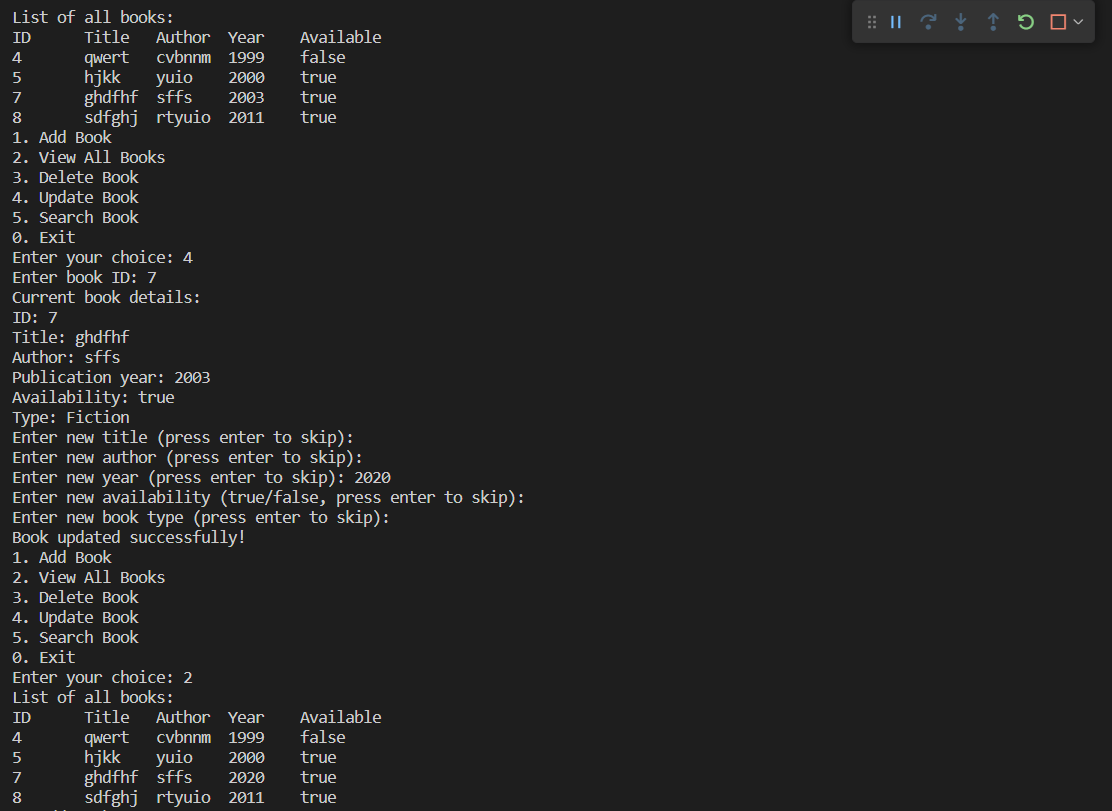
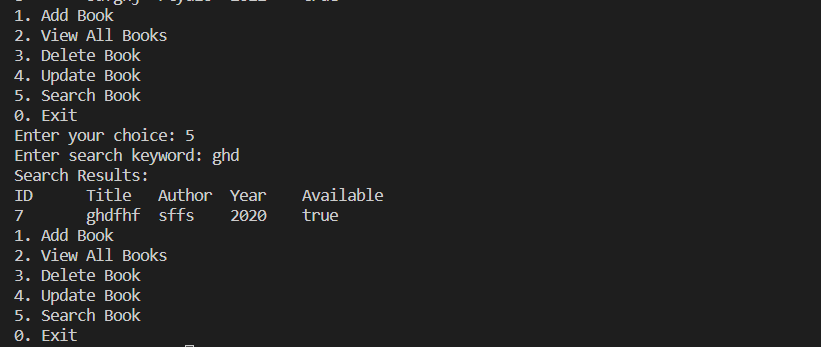
    }

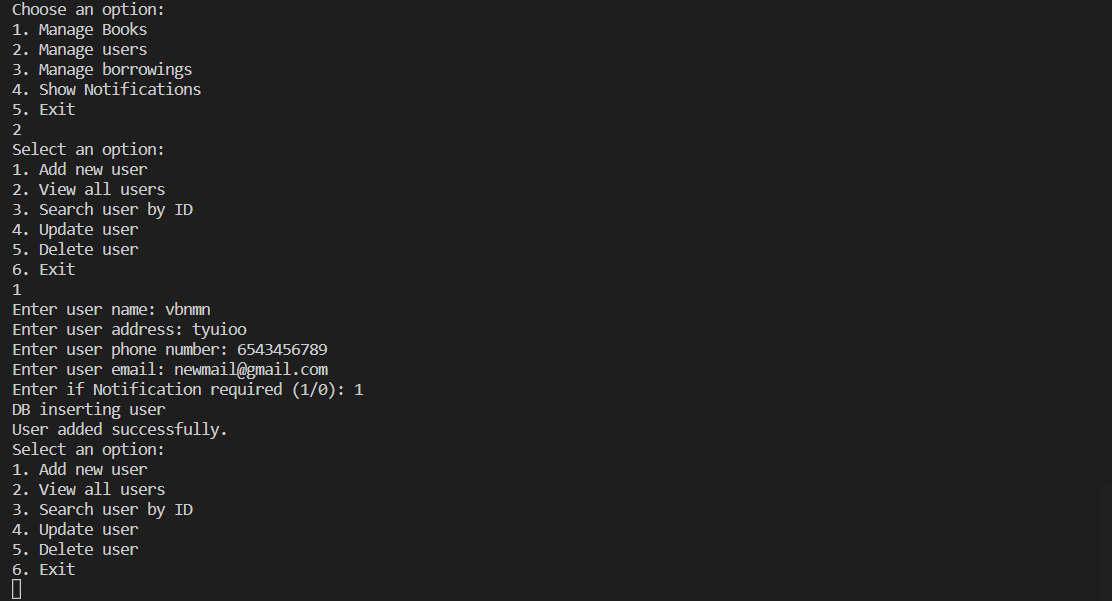
}

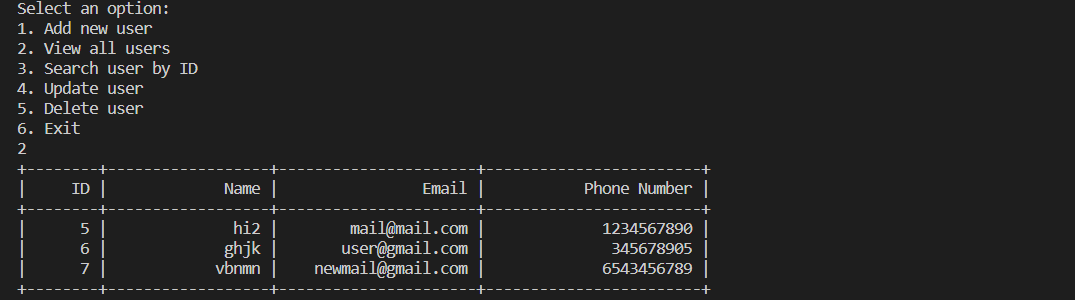
**RESULTS SCREENSHOTS**

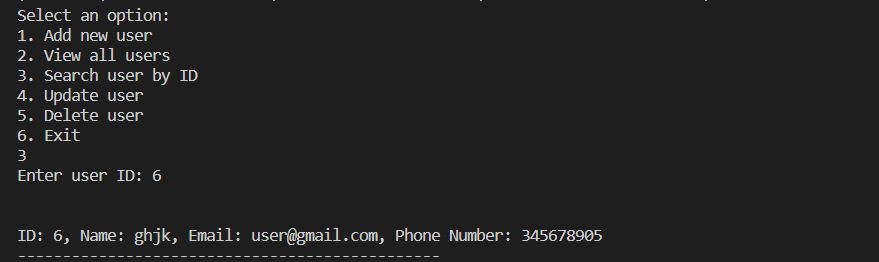
**** ****

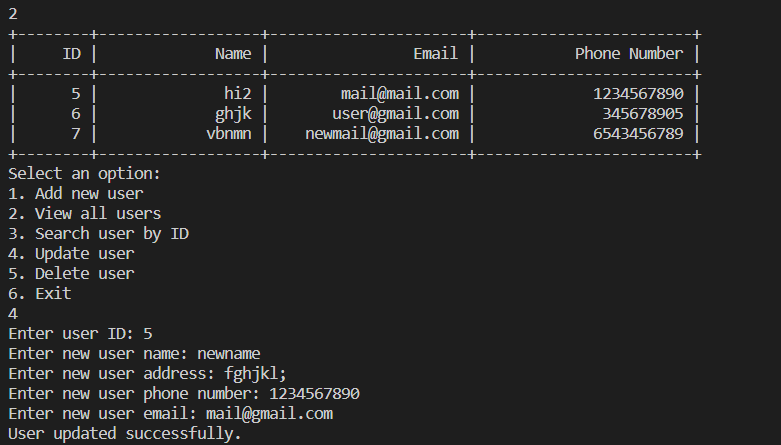
****

**** **** 

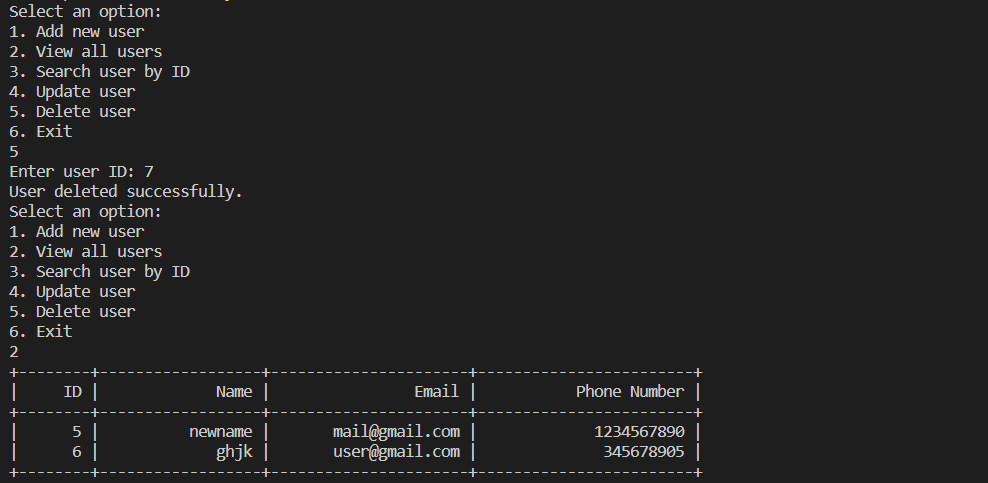
****

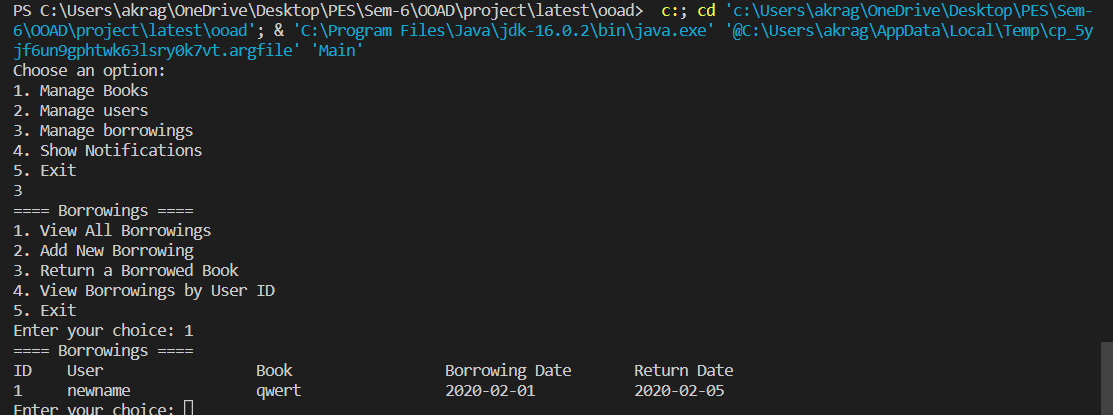
****

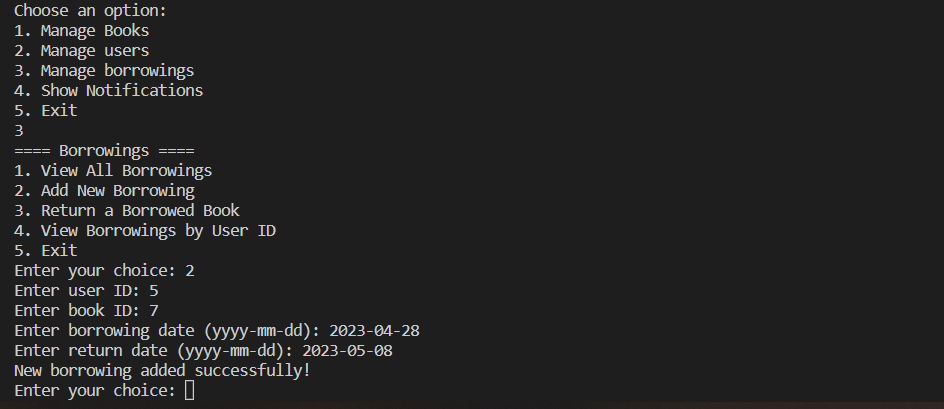
****

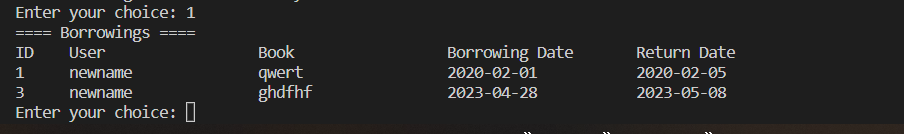
****

****

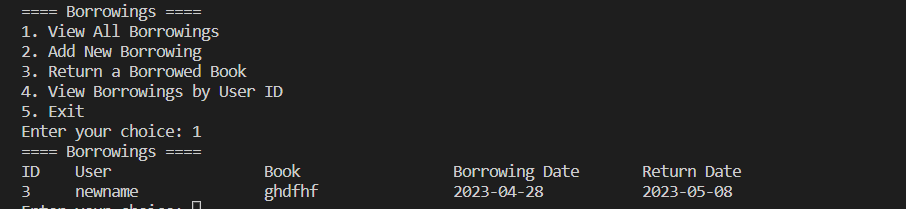
****

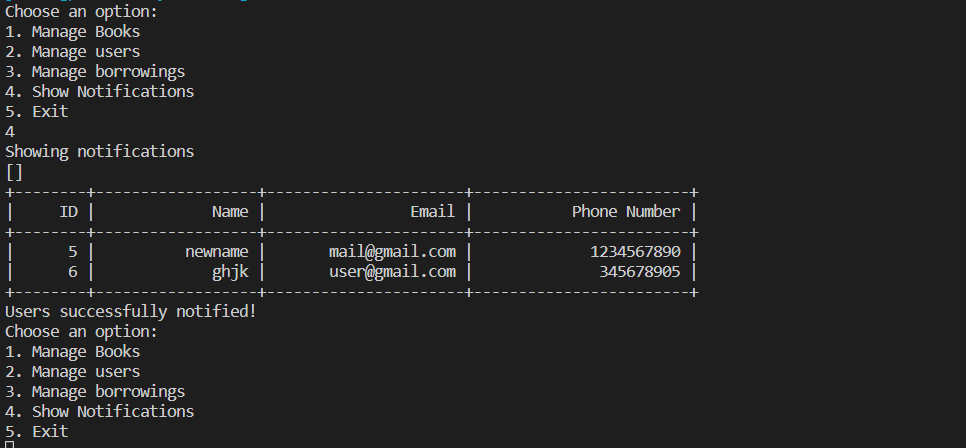
****

****

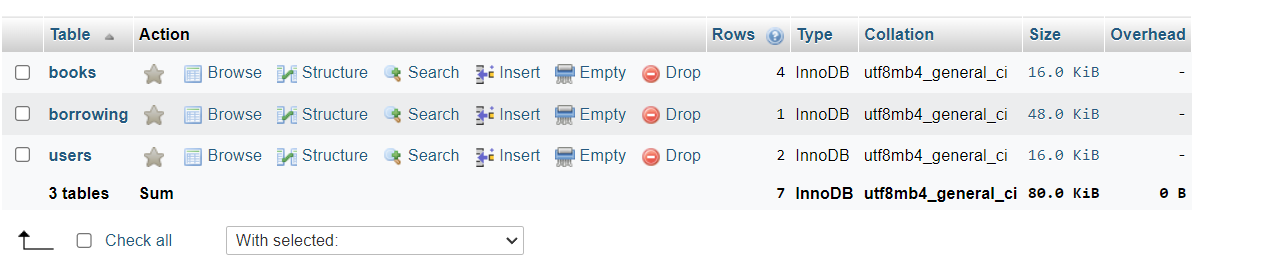
****

****

****

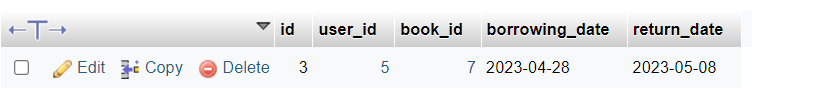
****

**TABLES:**

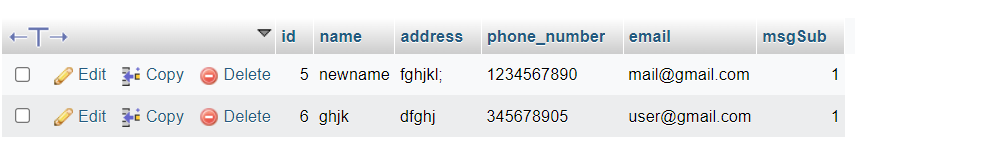
****

**booksS**

**borrowing**

****

**users**

****