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	**PES2UG20CS535**
Raghavendra A K	**PES2UG20CS537**
Rudresh S Patil	**PES2UG20CS540**
Harsha N	**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.
1.	**INTRODUCTION**	**3**
2.	**PROBLEM DEFINITION**	**4**
3.	**USECASE MODELING**	**5**
4.	**CLASS MODELING**	**6**
5.	**ACTIVITY MODELING**	**7**
6.	**IMPLEMENTATION**	**8**
7.	**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

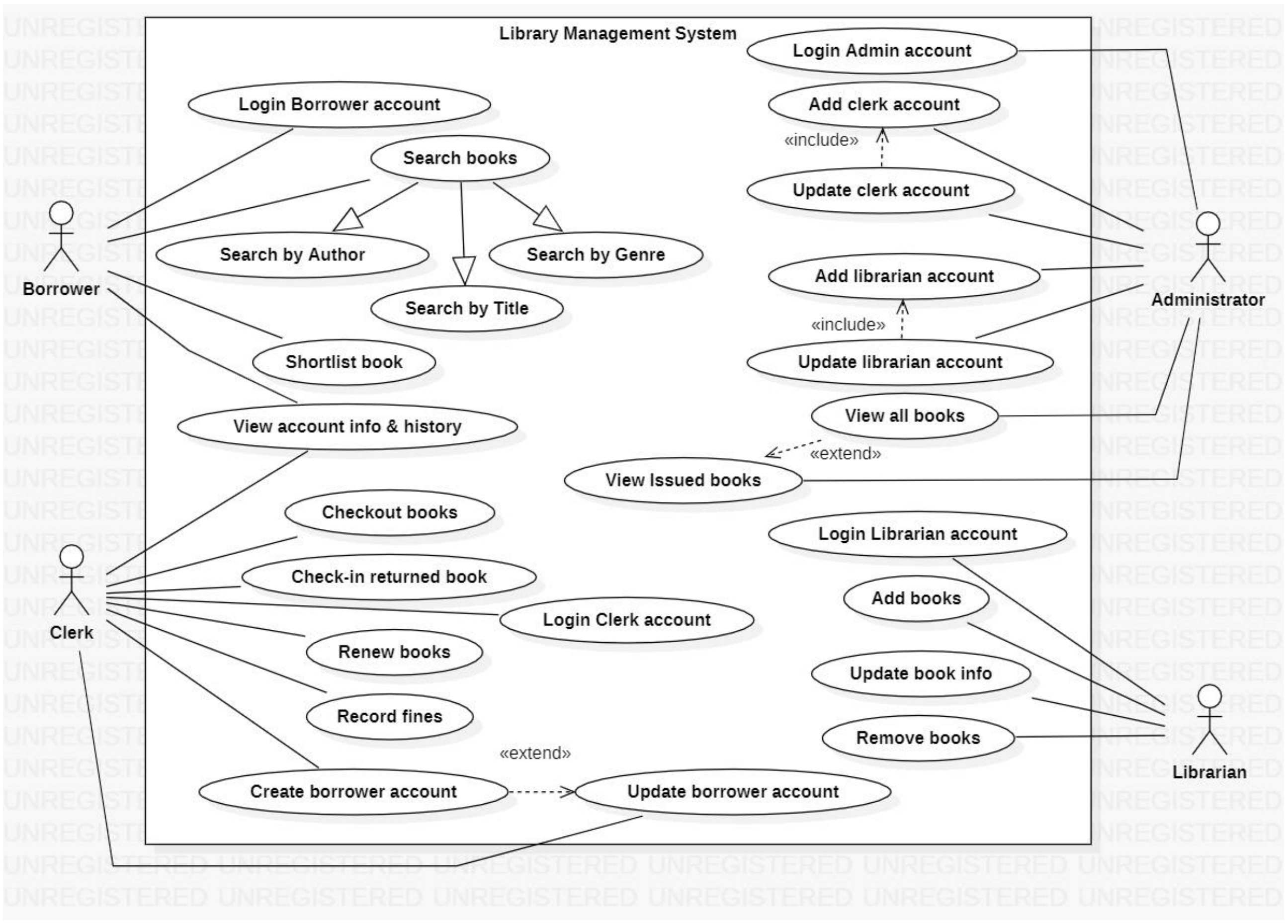
Libraries are an essential part of educational and cultural institutions, providing access to information and resources for learning, research, and leisure. However, managing a library's resources can be a complex and time-consuming task, especially for larger libraries with extensive collections.

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.

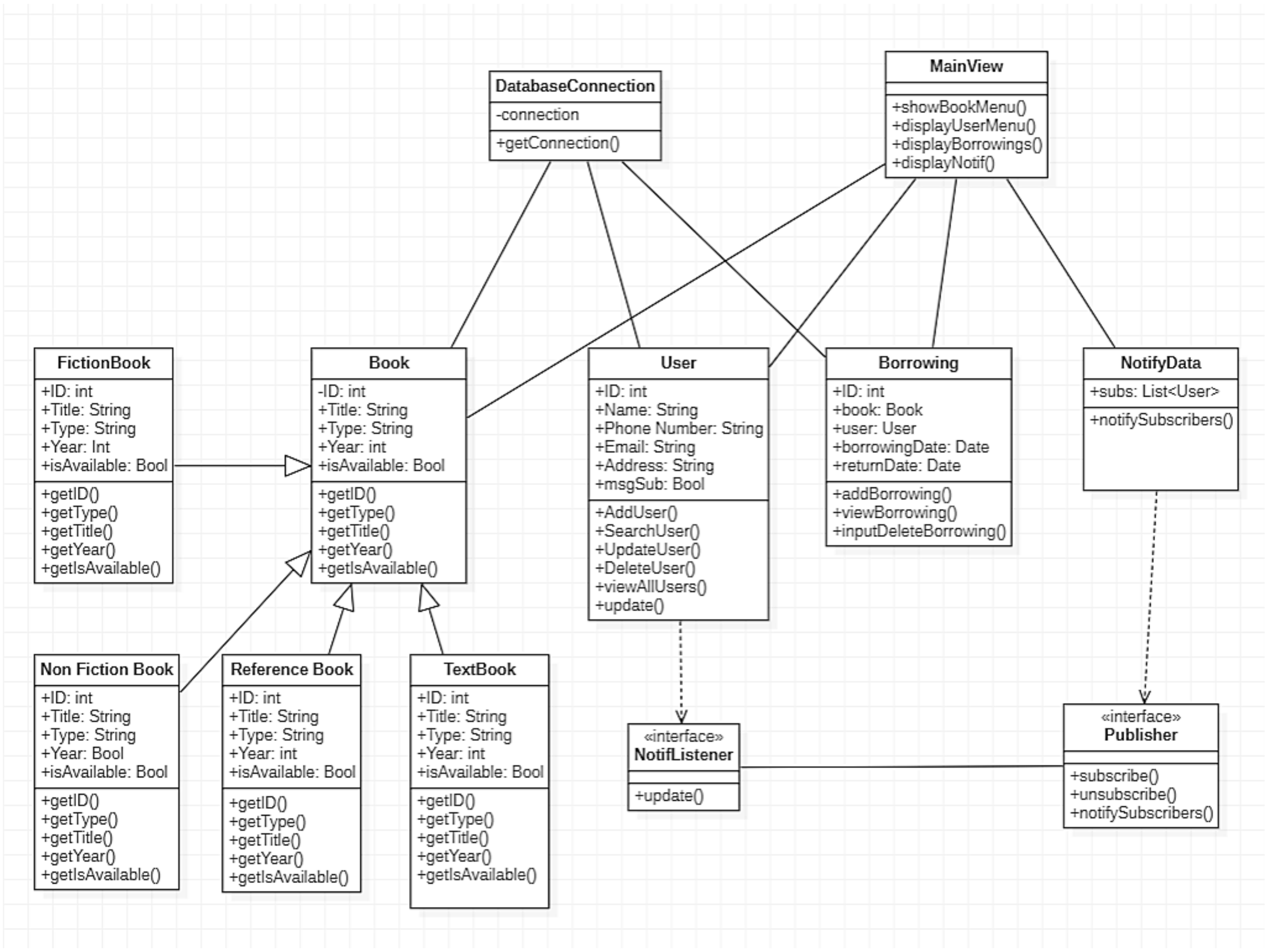
Traditional methods of managing library resources, such as manual record-keeping and paper-based systems, can be inefficient and error-prone, leading to lost or misplaced books and inaccurate records. Additionally, such systems can make it challenging to keep track of overdue books, resulting in lost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

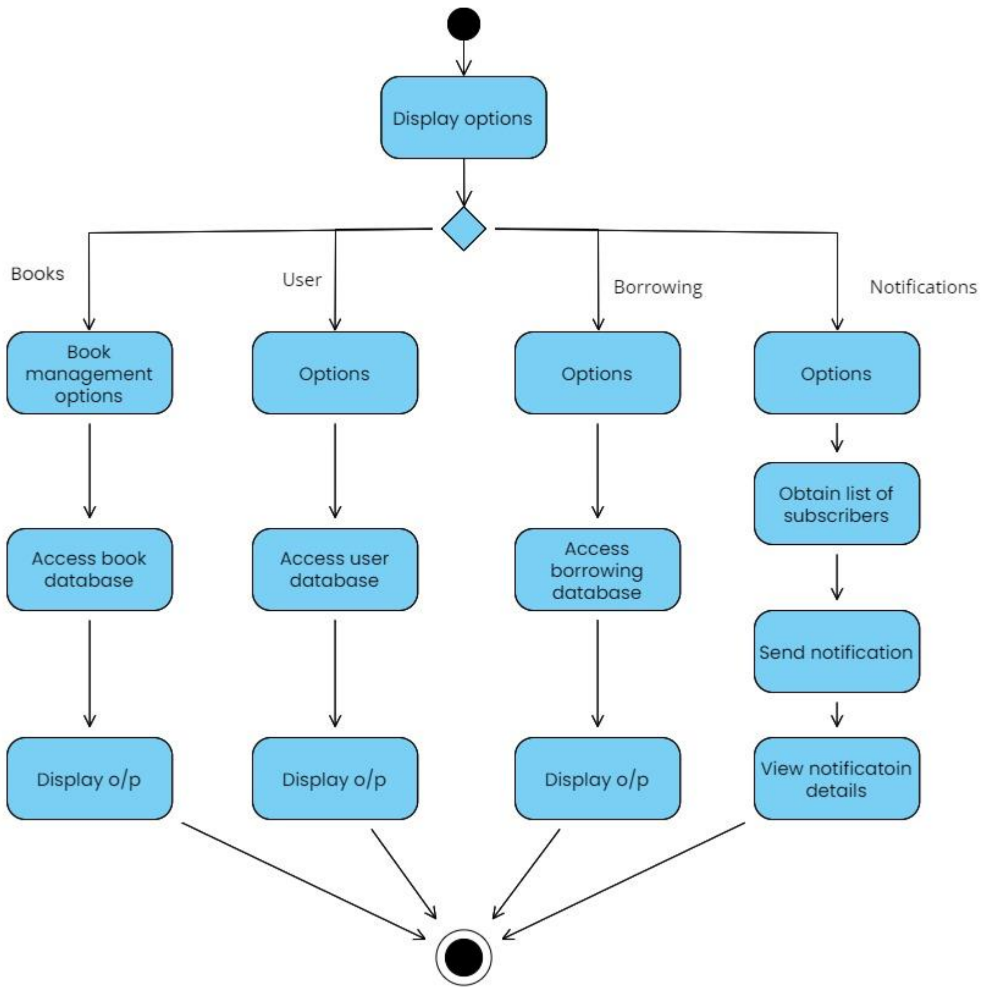
USECASE MODELING



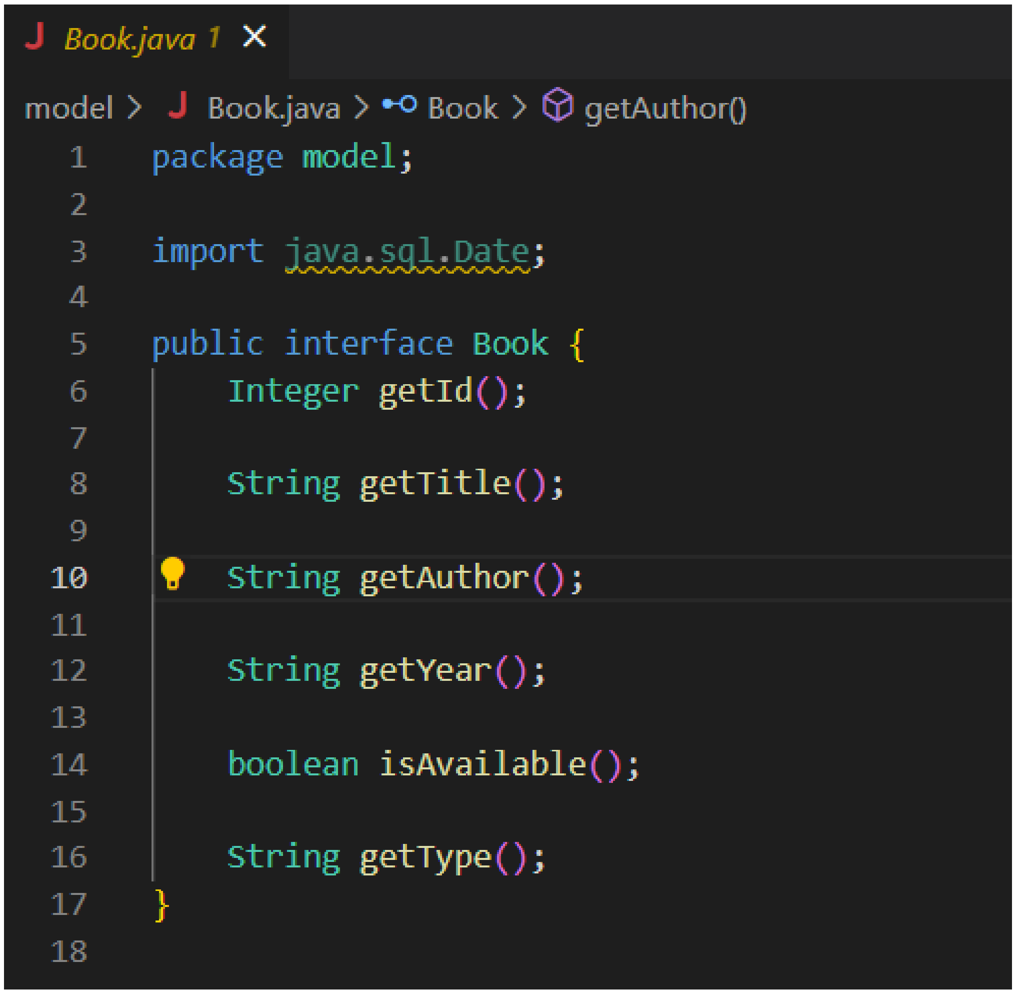
CLASS MODELING



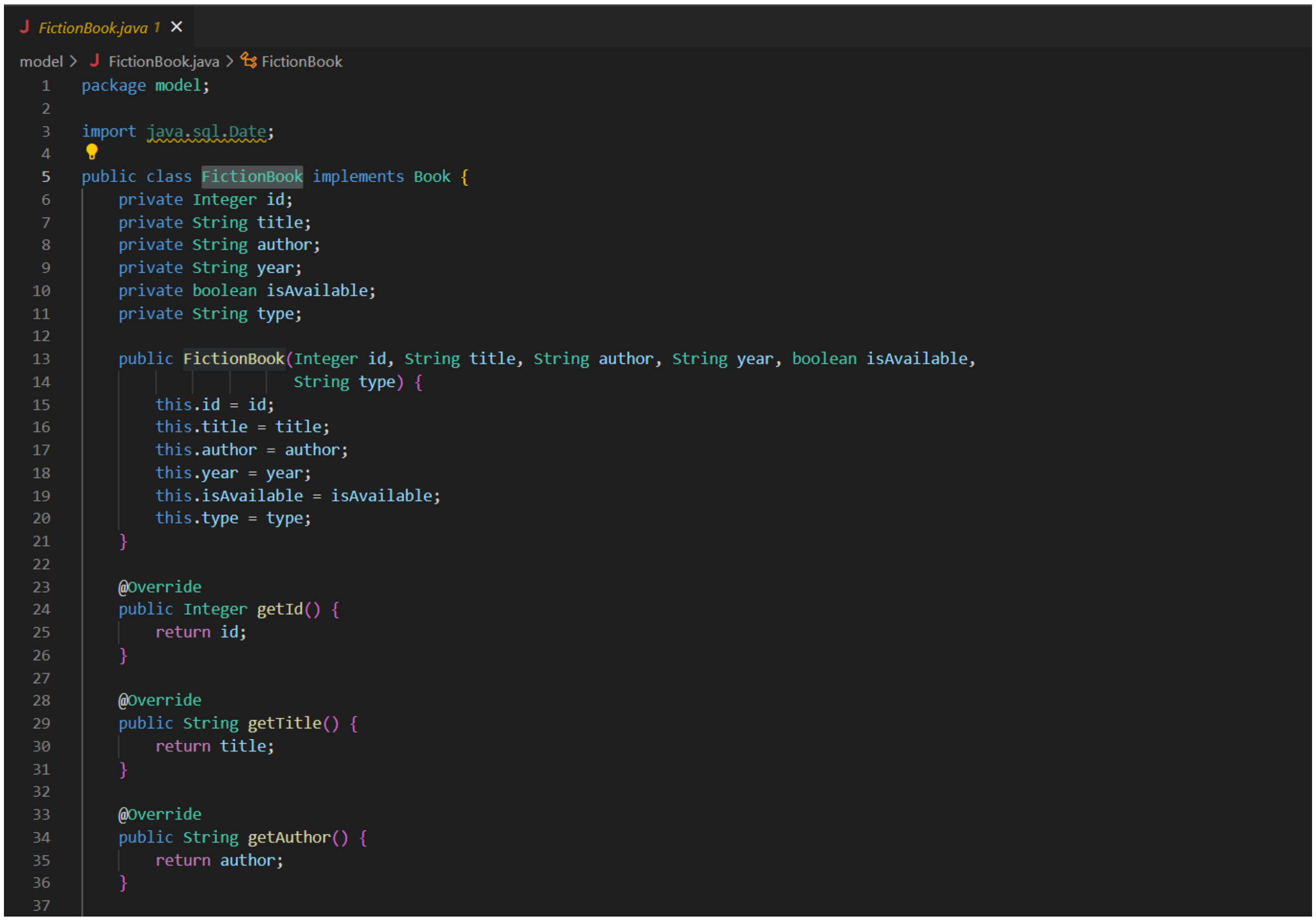
ACTIVITY MODELING



IMPLEMENTATION

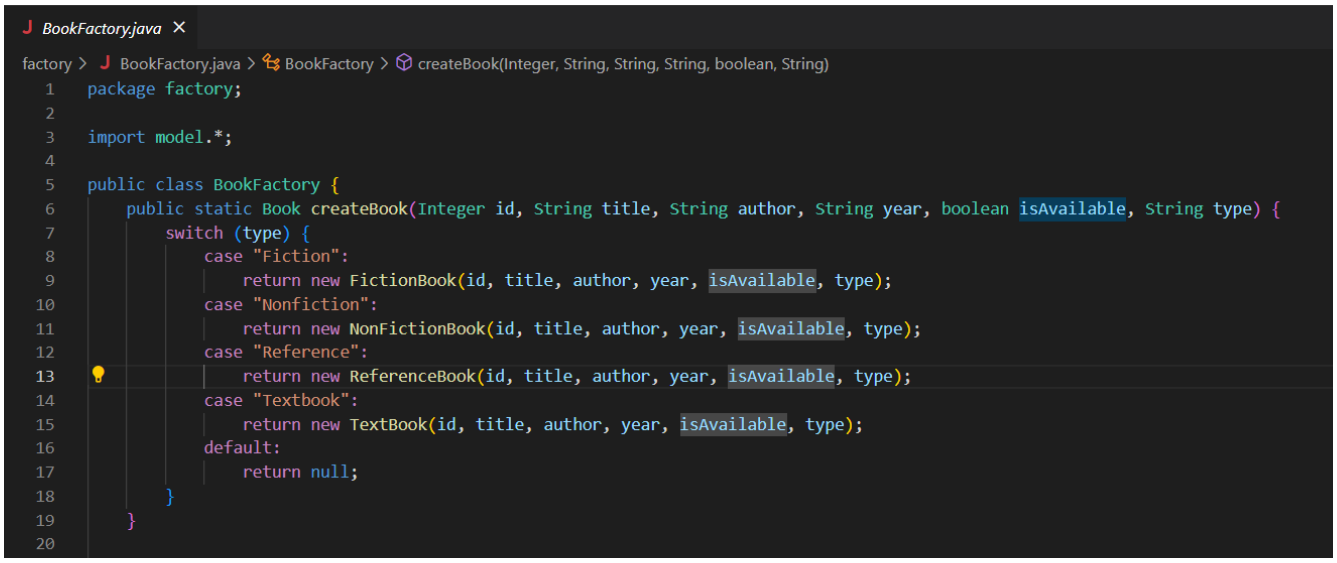


```java
package model;

import java.sql.Date;

public interface Book {
    Integer getId();

    String getTitle();

    String getAuthor();

    String getYear();

    boolean isAvailable();

    String getType();
}

```


FictionBook.java 1 ×

model > FictionBook.java > FictionBook

```java
package model;

import java.sql.Date;

public class FictionBook implements Book {
    private Integer id;
    private String title;
    private String author;
    private String year;
    private boolean isAvailable;
    private String type;

    public FictionBook(Integer id, String title, String author, String year, boolean isAvailable,
                        String type) {
        this.id = id;
        this.title = title;
        this.author = author;
        this.year = year;
        this.isAvailable = isAvailable;
        this.type = type;
    }

    @Override
    public Integer getId() {
        return id;
    }

    @Override
    public String getTitle() {
        return title;
    }

    @Override
    public String getAuthor() {
        return author;
    }

```


BookFactory.java ×

factory > BookFactory.java > BookFactory > createBook(Integer, String, String, String, boolean, String)

```java
package factory;

import model.*;

public class BookFactory {
    public static Book createBook(Integer id, String title, String author, String year, boolean isAvailable, String type) {
        switch (type) {
            case "Fiction":
                return new FictionBook(id, title, author, year, isAvailable, type);
            case "Nonfiction":
                return new NonFictionBook(id, title, author, year, isAvailable, type);
            case "Reference":
                return new ReferenceBook(id, title, author, year, isAvailable, type);
            case "Textbook":
                return new TextBook(id, title, author, year, isAvailable, type);
            default:
                return null;
        }
    }

```

NotifyData.java

```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

```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

```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;
```

```java
                default:
                    System.out.println("Invalid choice!");
            }
        }
    }
```

UserView.java

```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
    }
```

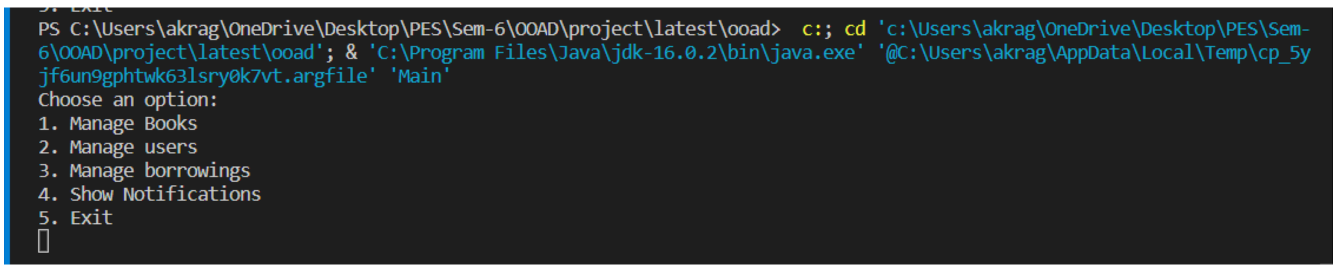
BorrowingView.java

```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.");
            }
        }
    }
```

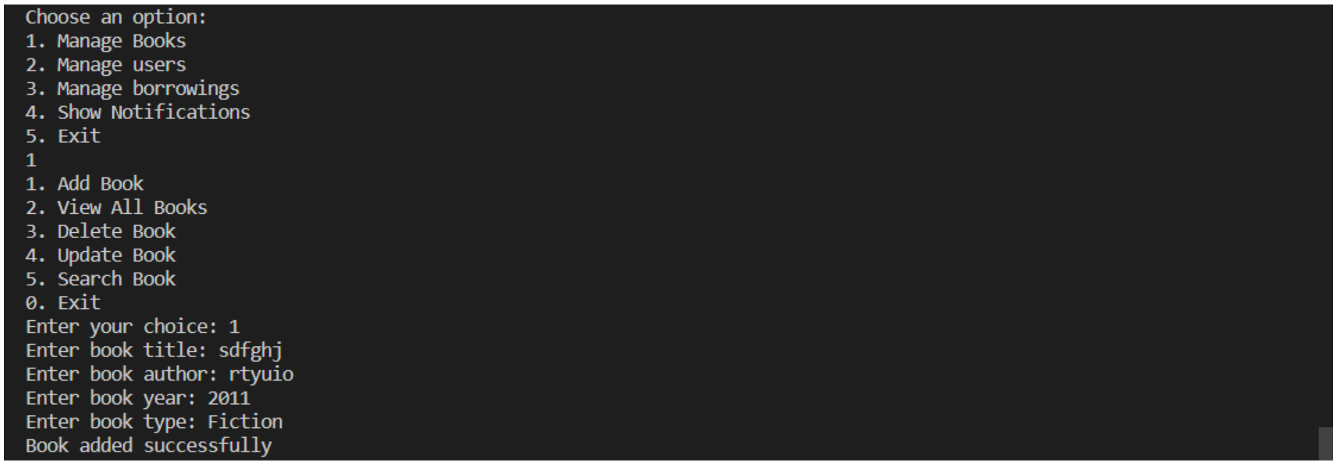
BookController.java

```
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);
    }
}
```

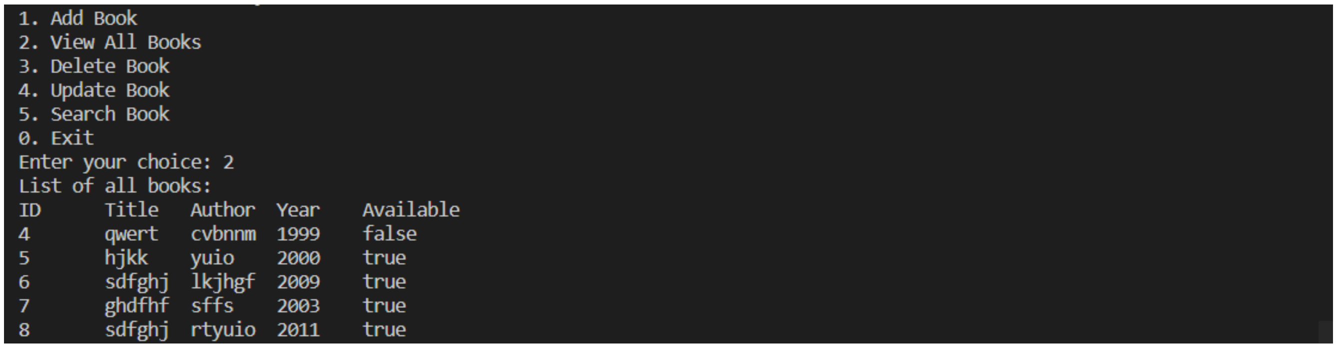
RESULTS SCREENSHOTS



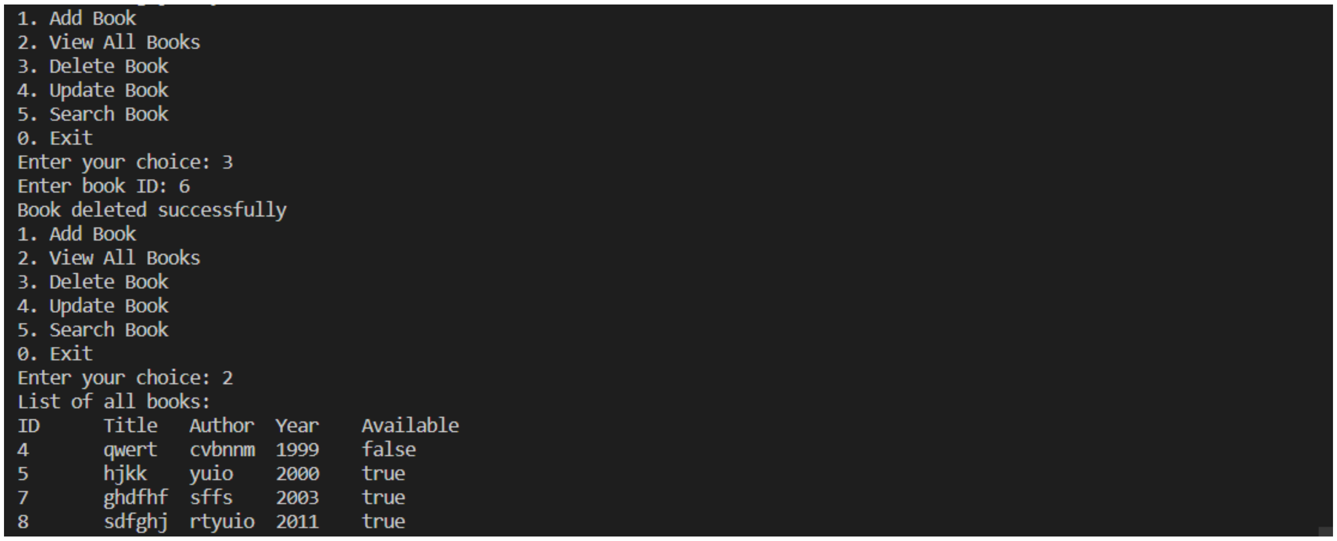
```
PS C:\Users\akrag\OneDrive\Desktop\PES\Sem-6\OOAD\project\latest\ooad>  c:; cd 'c:\Users\akrag\OneDrive\Desktop\PES\Sem-
6\OOAD\project\latest\ooad'; & 'C:\Program Files\Java\jdk-16.0.2\bin\java.exe' '@C:\Users\akrag\AppData\Local\Temp\cp_5y
jf6un9gphtwk63lsry0k7vt.argfile' 'Main'
Choose an option:
1. Manage Books
2. Manage users
3. Manage borrowings
4. Show Notifications
5. Exit
```



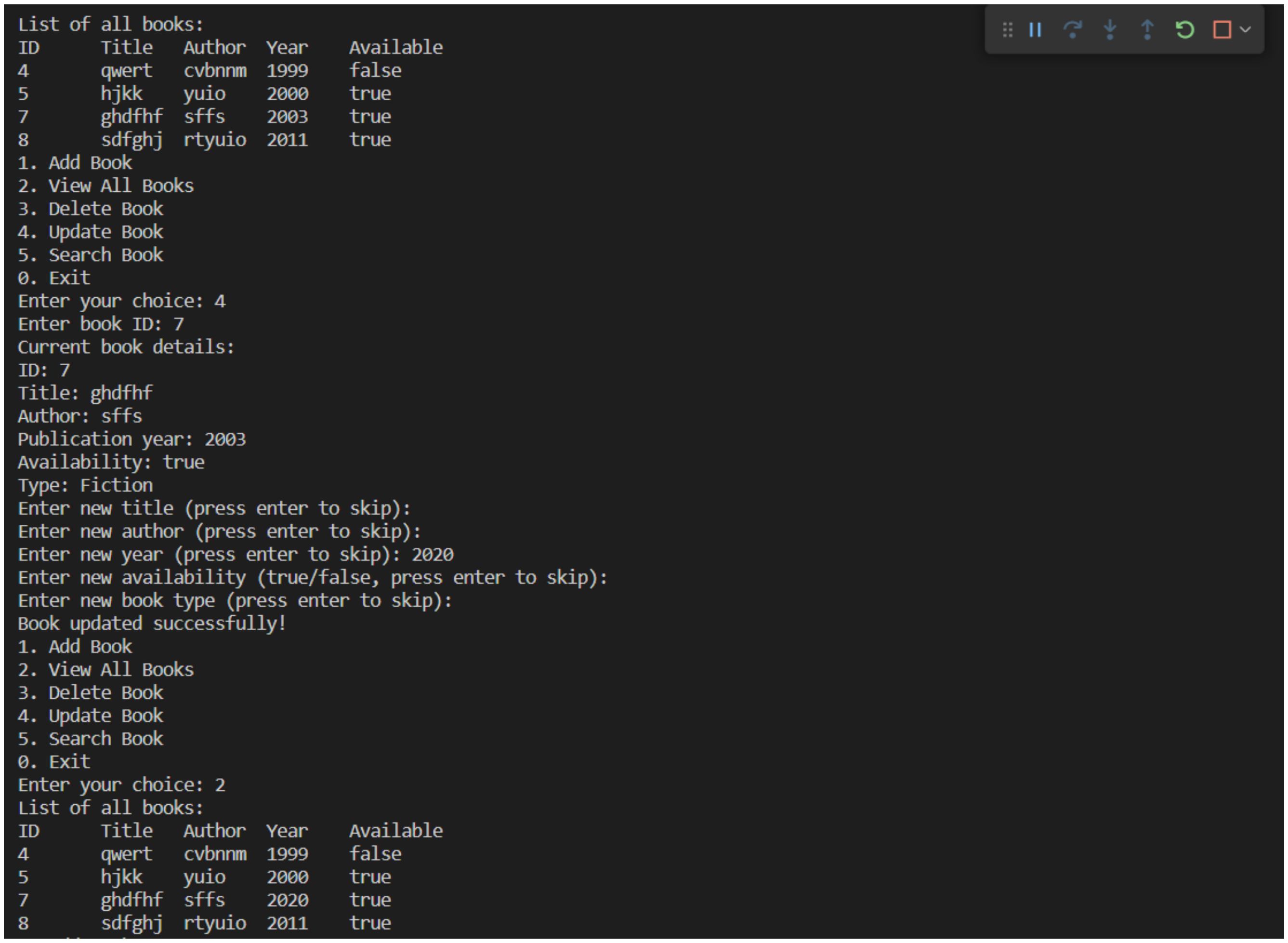
```
Choose an option:
1. Manage Books
2. Manage users
3. Manage borrowings
4. Show Notifications
5. Exit
1
1. Add Book
2. View All Books
3. Delete Book
4. Update Book
5. Search Book
0. Exit
Enter your choice: 1
Enter book title: sdfghj
Enter book author: rtyuio
Enter book year: 2011
Enter book type: Fiction
Book added successfully
```



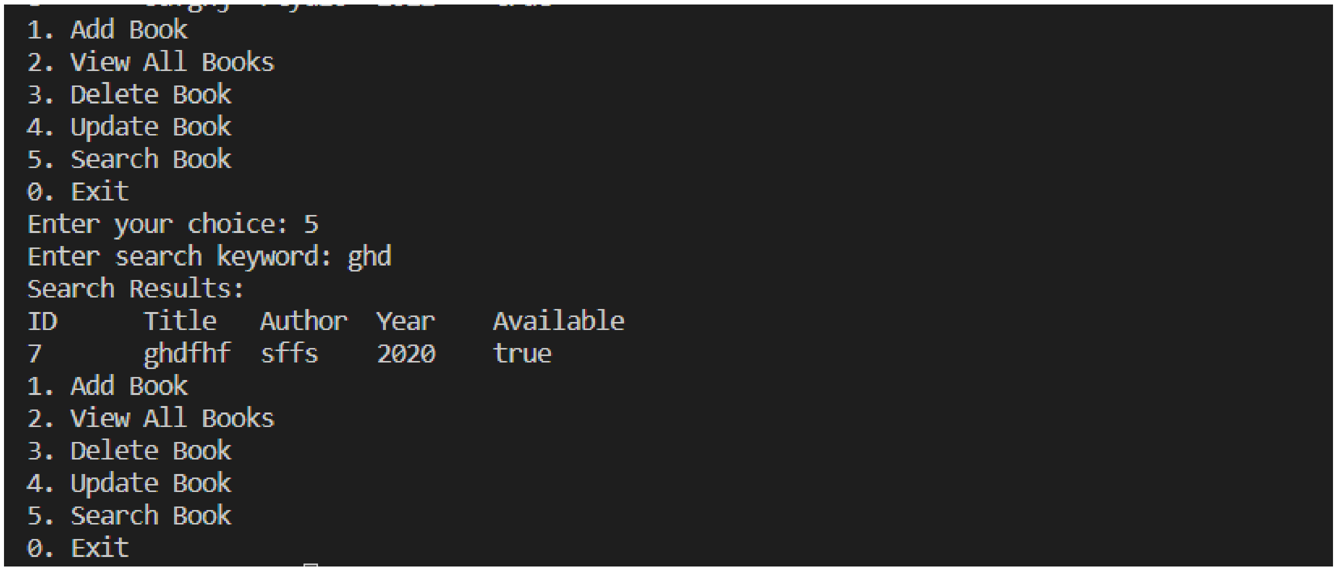
```
1. Add Book
2. View All Books
3. Delete Book
4. Update Book
5. Search Book
0. Exit
Enter your choice: 2
List of all books:
ID      Title   Author  Year    Available
4       qwert   cvbnnm  1999    false
5       hjkk    yuio    2000    true
6       sdfghj  lkjhgf  2009    true
7       ghdfhf  sffs    2003    true
8       sdfghj  rtyuio  2011    true
```



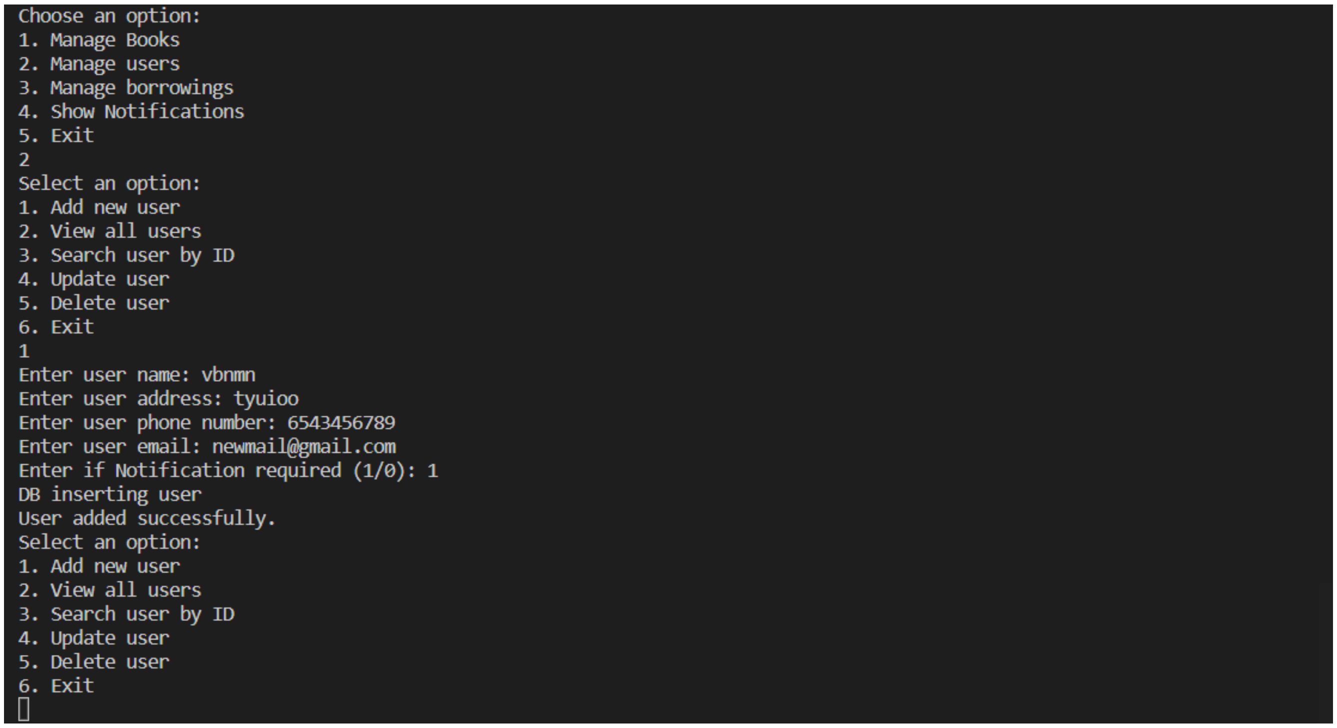
```
1. Add Book
2. View All Books
3. Delete Book
4. Update Book
5. Search Book
0. Exit
Enter your choice: 3
Enter book ID: 6
Book deleted successfully
1. Add Book
2. View All Books
3. Delete Book
4. Update Book
5. Search Book
0. Exit
Enter your choice: 2
List of all books:
ID      Title   Author  Year    Available
4       qwert   cvbnnm  1999    false
5       hjkk    yuio    2000    true
7       ghdfhf  sffs    2003    true
8       sdfghj  rtyuio  2011    true
```



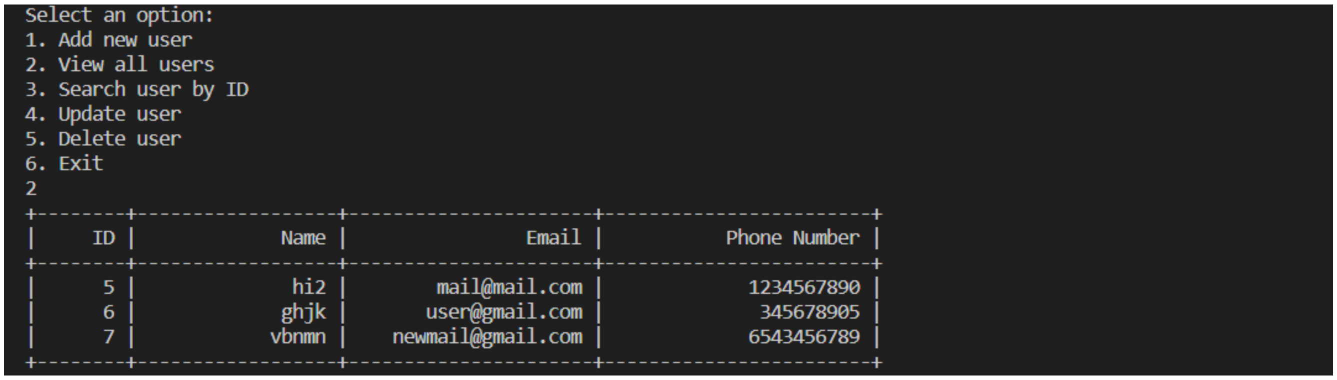
```
List of all books:
ID      Title   Author  Year    Available
4       qwert   cvbnnm  1999    false
5       hjkk    yuio    2000    true
7       ghdfhf  sffs    2003    true
8       sdfghj  rtyuio  2011    true
1. Add Book
2. View All Books
3. Delete Book
4. Update Book
5. Search Book
0. Exit
Enter your choice: 4
Enter book ID: 7
Current book details:
ID: 7
Title: ghdfhf
Author: sffs
Publication year: 2003
Availability: true
Type: Fiction
Enter new title (press enter to skip):
Enter new author (press enter to skip):
Enter new year (press enter to skip): 2020
Enter new availability (true/false, press enter to skip):
Enter new book type (press enter to skip):
Book updated successfully!
1. Add Book
2. View All Books
3. Delete Book
4. Update Book
5. Search Book
0. Exit
Enter your choice: 2
List of all books:
ID      Title   Author  Year    Available
4       qwert   cvbnnm  1999    false
5       hjkk    yuio    2000    true
7       ghdfhf  sffs    2020    true
8       sdfghj  rtyuio  2011    true
```



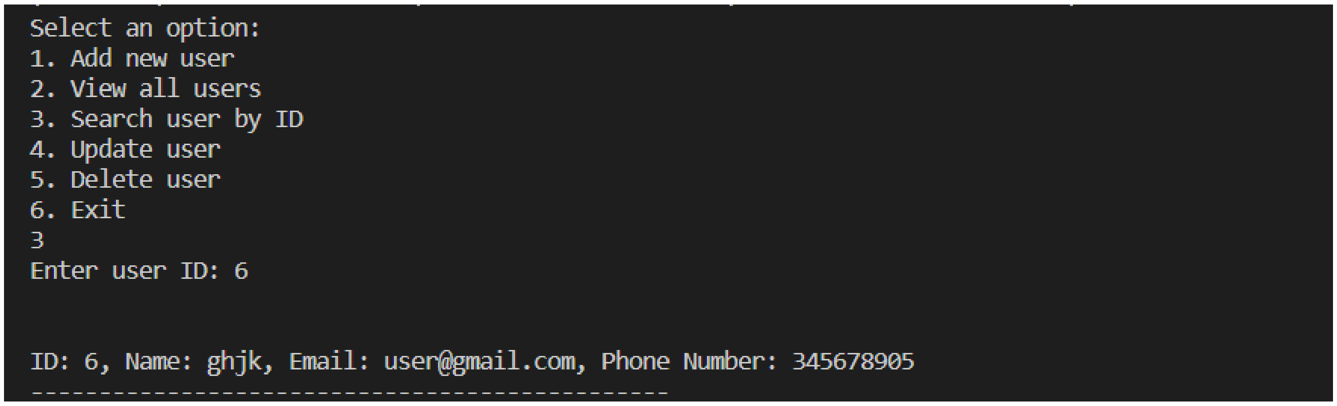
```
1. Add Book
2. View All Books
3. Delete Book
4. Update Book
5. Search Book
0. Exit
Enter your choice: 5
Enter search keyword: ghd
Search Results:
ID      Title   Author  Year    Available
7       ghdfhf  sffs    2020    true
1. Add Book
2. View All Books
3. Delete Book
4. Update Book
5. Search Book
0. Exit
```



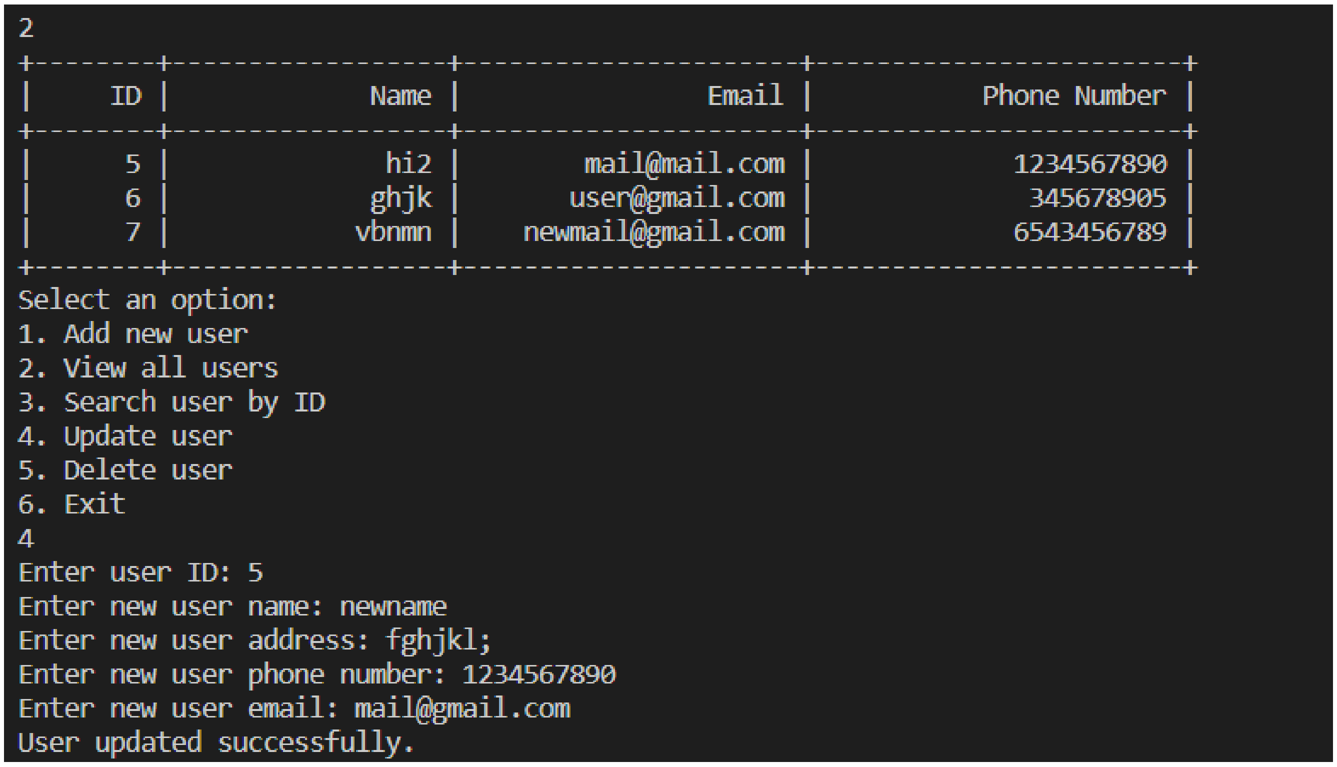
```
Choose an option:
1. Manage Books
2. Manage users
3. Manage borrowings
4. Show Notifications
5. Exit
2
Select an option:
1. Add new user
2. View all users
3. Search user by ID
4. Update user
5. Delete user
6. Exit
1
Enter user name: vbnmn
Enter user address: tyuioo
Enter user phone number: 6543456789
Enter user email: newmail@gmail.com
Enter if Notification required (1/0): 1
DB inserting user
User added successfully.
Select an option:
1. Add new user
2. View all users
3. Search user by ID
4. Update user
5. Delete user
6. Exit
```



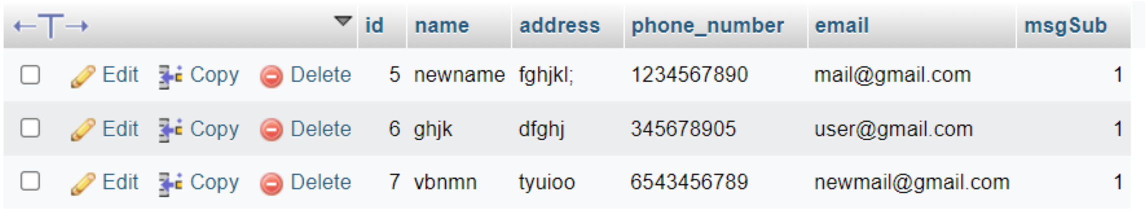
```
Select an option:
1. Add new user
2. View all users
3. Search user by ID
4. Update user
5. Delete user
6. Exit
2
+--------+-----------------+--------------------+----------------------+
|     ID |            Name |              Email |         Phone Number |
+--------+-----------------+--------------------+----------------------+
|      5 |             hi2 |      mail@mail.com |           1234567890 |
|      6 |            ghjk |     user@gmail.com |            345678905 |
|      7 |           vbnmn |  newmail@gmail.com |           6543456789 |
+--------+-----------------+--------------------+----------------------+
```



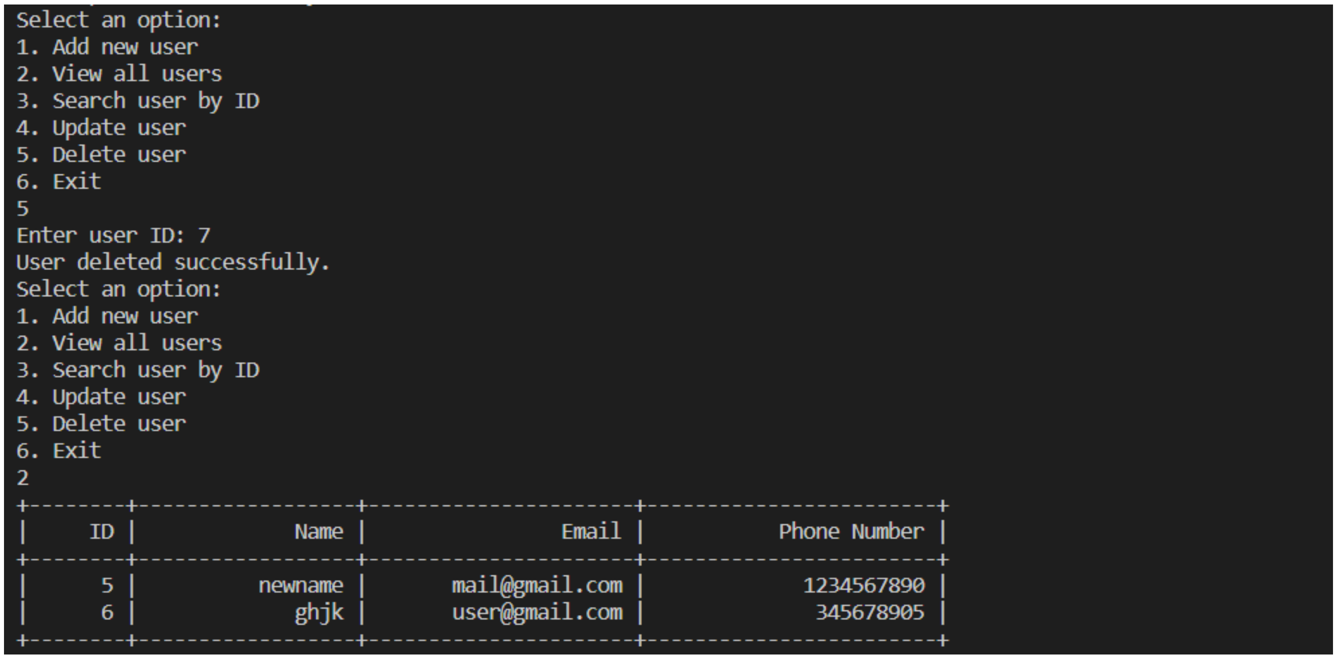
```
Select an option:
1. Add new user
2. View all users
3. Search user by ID
4. Update user
5. Delete user
6. Exit
3
Enter user ID: 6


ID: 6, Name: ghjk, Email: user@gmail.com, Phone Number: 345678905
-----------------------------------------------
```



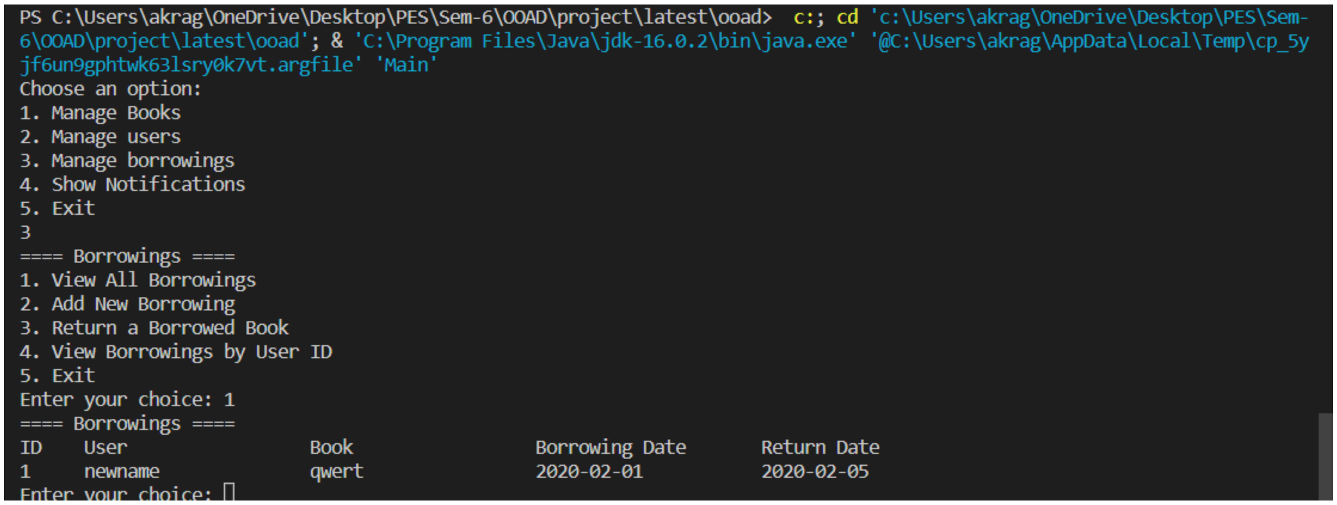
```
2
+--------+-----------------+--------------------+-----------------------+
|     ID |            Name |              Email |          Phone Number |
+--------+-----------------+--------------------+-----------------------+
|      5 |             hi2 |      mail@mail.com |            1234567890 |
|      6 |            ghjk |     user@gmail.com |             345678905 |
|      7 |          vbnmn |  newmail@gmail.com |            6543456789 |
+--------+-----------------+--------------------+-----------------------+
Select an option:
1. Add new user
2. View all users
3. Search user by ID
4. Update user
5. Delete user
6. Exit
4
Enter user ID: 5
Enter new user name: newname
Enter new user address: fghjkl;
Enter new user phone number: 1234567890
Enter new user email: mail@gmail.com
User updated successfully.
```



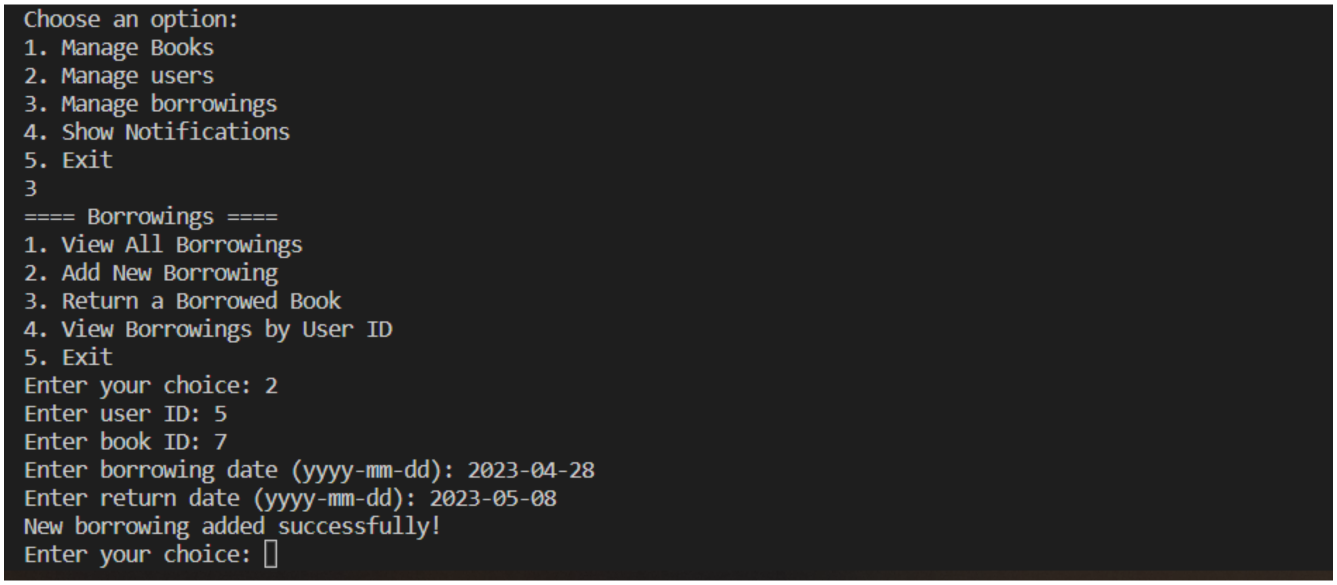
	id	name	address	phone_number	email	msgSub
Edit Copy Delete	5	newname	fghjkl;	1234567890	mail@gmail.com	1
Edit Copy Delete	6	ghjk	dfghj	345678905	user@gmail.com	1
Edit Copy Delete	7	vbnmn	tyuioo	6543456789	newmail@gmail.com	1



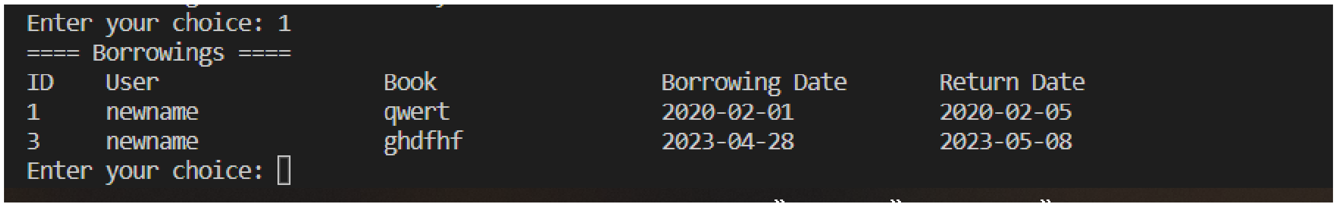
```
Select an option:
1. Add new user
2. View all users
3. Search user by ID
4. Update user
5. Delete user
6. Exit
5
Enter user ID: 7
User deleted successfully.
Select an option:
1. Add new user
2. View all users
3. Search user by ID
4. Update user
5. Delete user
6. Exit
2
+--------+-------------------+----------------------+-------------------------+
|     ID |              Name |                Email |            Phone Number |
+--------+-------------------+----------------------+-------------------------+
|      5 |           newname |       mail@gmail.com |              1234567890 |
|      6 |              ghjk |       user@gmail.com |               345678905 |
+--------+-------------------+----------------------+-------------------------+
```



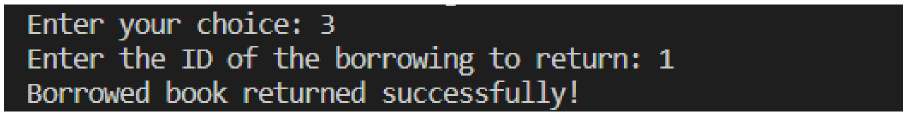
```
PS C:\Users\akrag\OneDrive\Desktop\PES\Sem-6\OOAD\project\latest\ood>  c:; cd 'c:\Users\akrag\OneDrive\Desktop\PES\Sem-6\OOAD\project\latest\ood'; & 'C:\Program Files\Java\jdk-16.0.2\bin\java.exe' '@C:\Users\akrag\AppData\Local\Temp\cp_5yjf6un9gphtwk63lsry0k7vt.argfile' 'Main'
Choose an option:
1. Manage Books
2. Manage users
3. Manage borrowings
4. Show Notifications
5. Exit
3
==== Borrowings ====
1. View All Borrowings
2. Add New Borrowing
3. Return a Borrowed Book
4. View Borrowings by User ID
5. Exit
Enter your choice: 1
==== Borrowings ====
ID     User                Book                Borrowing Date      Return Date
1      newname             qwert               2020-02-01          2020-02-05
Enter your choice: 
```



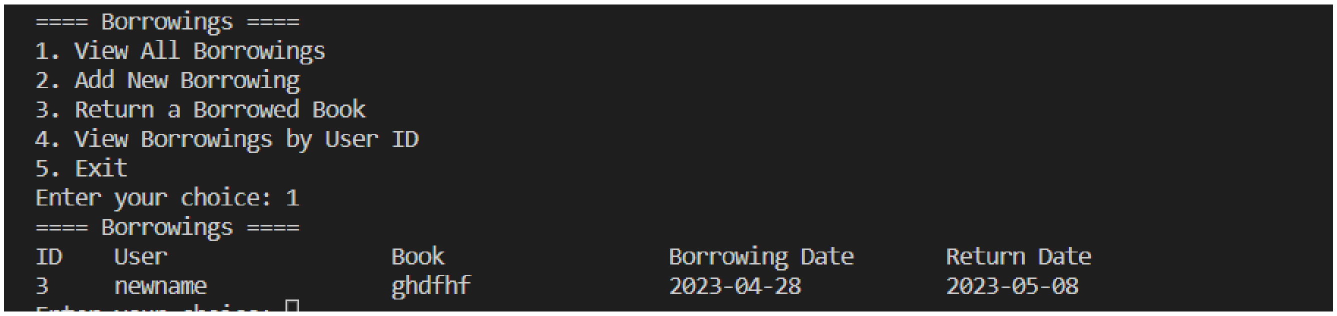
```
Choose an option:
1. Manage Books
2. Manage users
3. Manage borrowings
4. Show Notifications
5. Exit
3
==== Borrowings ====
1. View All Borrowings
2. Add New Borrowing
3. Return a Borrowed Book
4. View Borrowings by User ID
5. Exit
Enter your choice: 2
Enter user ID: 5
Enter book ID: 7
Enter borrowing date (yyyy-mm-dd): 2023-04-28
Enter return date (yyyy-mm-dd): 2023-05-08
New borrowing added successfully!
Enter your choice: 
```



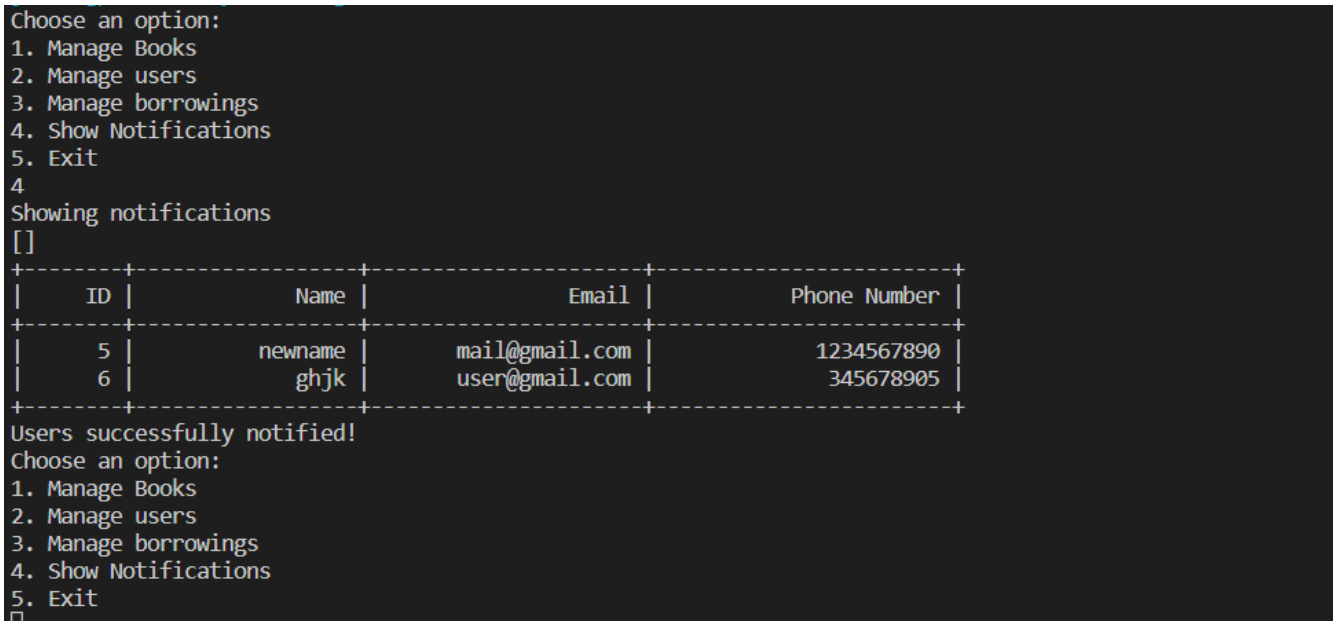
```
Enter your choice: 1
==== Borrowings ====
ID     User                Book                Borrowing Date      Return Date
1      newname             qwert               2020-02-01          2020-02-05
3      newname             ghdfhf              2023-04-28          2023-05-08
Enter your choice: 
```



```
Enter your choice: 3
Enter the ID of the borrowing to return: 1
Borrowed book returned successfully!
```



```
==== Borrowings ====
1. View All Borrowings
2. Add New Borrowing
3. Return a Borrowed Book
4. View Borrowings by User ID
5. Exit
Enter your choice: 1
==== Borrowings ====
ID     User                Book                Borrowing Date      Return Date
3      newname             ghdfhf              2023-04-28          2023-05-08
```



```
Choose an option:
1. Manage Books
2. Manage users
3. Manage borrowings
4. Show Notifications
5. Exit
4
Showing notifications
[]
+--------+-----------------+--------------------+----------------------+
|     ID |            Name |              Email |         Phone Number |
+--------+-----------------+--------------------+----------------------+
|      5 |         newname |     mail@gmail.com |           1234567890 |
|      6 |            ghjk |     user@gmail.com |            345678905 |
+--------+-----------------+--------------------+----------------------+
Users successfully notified!
Choose an option:
1. Manage Books
2. Manage users
3. Manage borrowings
4. Show Notifications
5. Exit
```

TABLES:

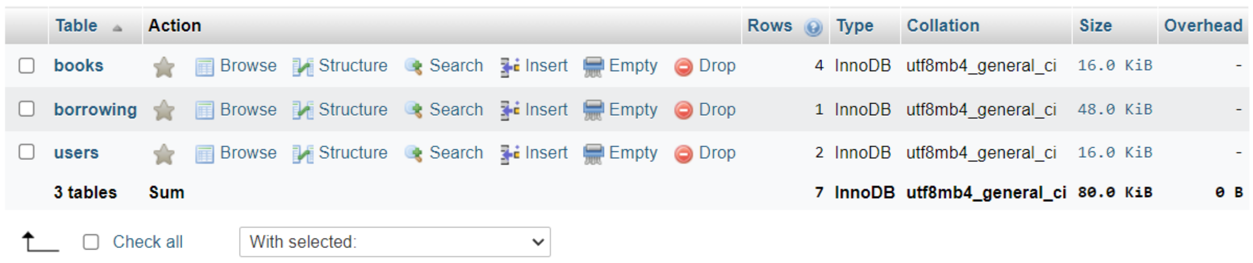
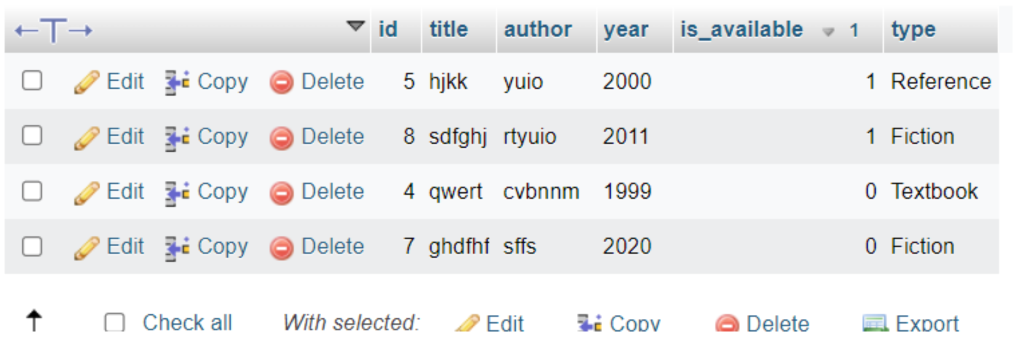


Table	Action	Rows	Type	Collation	Size	Overhead
books	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 KiB	-
borrowing	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_general_ci	48.0 KiB	-
users	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	16.0 KiB	-
3 tables	**Sum**	**7**	**InnoDB**	**utf8mb4_general_ci**	**80.0 KiB**	**0 B**

Check all   With selected:

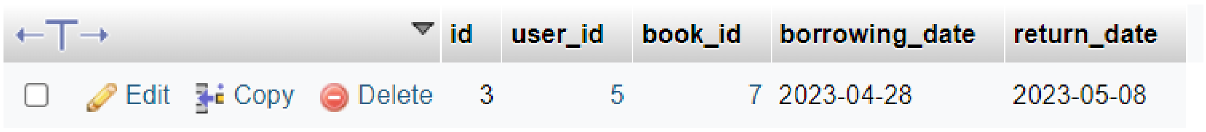
books



id	title	author	year	is_available	type
5	hjkk	yuio	2000	1	Reference
8	sdfghj	rtyuio	2011	1	Fiction
4	qwert	cvbnnm	1999	0	Textbook
7	ghdfhf	sffs	2020	0	Fiction

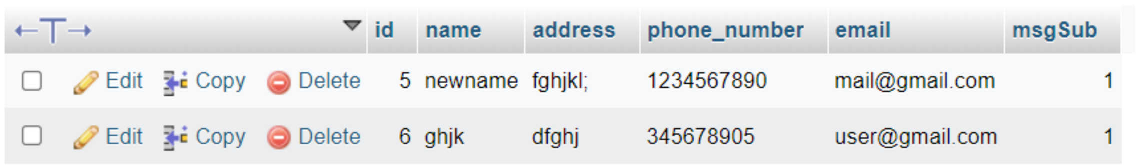
Check all With selected: Edit Copy Delete Export

borrowing



id	user_id	book_id	borrowing_date	return_date
3	5	7	2023-04-28	2023-05-08

users



id	name	address	phone_number	email	msgSub
5	newname	fghjkl;	1234567890	mail@gmail.com	1
6	ghjk	dfghj	345678905	user@gmail.com	1