



**NEW HORIZON  
COLLEGE OF ENGINEERING**

Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC  
Accredited by NAAC with 'A' Grade, Accredited by NBA

**A PROJECT REPORT**

*for*

**Mini Project using Java (21CSE56)**

*on*

**Book Hub**

*Submitted by*

**DEEPIKA SINGH N**

**USN: 1NH21CS068, Sem-Sec: 5-B**

*In partial fulfillment for the award of the degree of*

**BACHELOR OF ENGINEERING**

*in*

**COMPUTER SCIENCE AND ENGINEERING**

**Academic Year: 2023-24**



# NEW HORIZON COLLEGE OF ENGINEERING

Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC  
Accredited by NAAC with 'A' Grade, Accredited by NBA

## CERTIFICATE

This is to certify that the mini project work titled

### Book Hub

Submitted in partial fulfillment of the degree of Bachelor of Engineering in  
Computer Science and Engineering by

**DEEPIKA SINGH N**

**USN: 1NH21CS068**

*DURING*

*ODD SEMESTER 2023-2024*

*for*

*Course: Mini Project using Java-21CSE56*

Signature of Reviewer

Signature of HOD

### SEMESTER END EXAMINATION

*Name of the Examiner(s)*

*Signature with date*

1. \_\_\_\_\_

\_\_\_\_\_

2. \_\_\_\_\_

\_\_\_\_\_

---

## ABSTRACT

The Book Hub introduces an innovative online bookstore system developed using Java, JDBC, and MySQL, with a primary focus on user convenience and expanded options. The web-based interface allows customers to easily search for books by author or genre (e.g., horror, fiction, fantasy) and place orders. The system seamlessly integrates with two real commercial online bookstores, providing users with the ability to explore external inventories and receive comprehensive information on titles, prices, and availability.

The graphical user interface (GUI) facilitates a user-friendly experience, with customer inputs processed by a Java-based control function. This function manages the search process, interacting with the database to retrieve relevant information and present it back to the user interface. A notable feature of this Book Hub is the software bridge connecting the system to external online bookstores. This integration broadens the range of available options for customers, fostering a richer and more diverse shopping experience. The architecture leverages Java Swing for the GUI, JDBC for database connectivity, and MySQL for effective data management, ensuring a robust and scalable solution.

This Book Hub delivers an advanced online bookstore that goes beyond conventional offerings by seamlessly integrating with external bookstores. Through Java, JDBC, and MySQL, the system provides a user-friendly interface, efficient order processing, and an expanded inventory search capability, ultimately enhancing the overall online book shopping experience.

---

## ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be impossible without the mention of the people who made it possible, whose constant guidance and encouragement crowned our efforts with success.

I am delighted to express my gratitude to **Dr. Mohan Manghnani**, Chairman, New Horizon Educational Institutions, for furnishing the essential infrastructure and fostering a positive environment.

I would like to take this chance to express my deep gratitude to **Dr. Manjunatha, Principal**, New Horizon College of Engineering, for consistently offering support and encouragement.

I wish to convey my gratitude to **Dr. Anandhi R J**, Professor and Dean-Academics at NHCE, for providing indispensable guidance and unwavering support.

I want to express my heartfelt gratitude to **Dr. B. Rajalakshmi**, Professor and Head of the Department, Computer Science and Engineering, for the steadfast support that has remarkably shaped my academic journey.

I would like to extend my thanks to **Ms. Jayshree**, Assistant Professor, Department of Computer Science and Engineering, who served as the reviewer for my mini project.

**Deepika Singh N**

**USN: 1NH21CS068**

---

# CONTENTS

<b>ABSTRACT</b>	<b>I</b>
<b>ACKNOWLEDGEMENT</b>	<b>II</b>
<b>1. INTRODUCTION</b>	
1.1. PROBLEM DEFINITION	1
1.2. OBJECTIVES	2
1.3. METHODOLOGY TO BE FOLLOWED	3
1.4. EXPECTED OUTCOMES	5
1.5. HARDWARE AND SOFTWARE REQUIREMENTS	5
<b>2. FUNDAMENTALS OF JAVA</b>	
2.1. INTRODUCTION TO JAVA	6
2.2. ADVANTAGES OF JAVA	7
2.3. DATA TYPES	8
2.4. CONTROL FLOW	8
2.5. METHODS	9
2.6. OBJECT ORIENTED CONCEPTS	10
2.7. EXCEPTION HANDLING	11
2.8. FILE HANDLING	12
2.9. PACKAGES AND IMPORT	13
2.10. INTERFACES	13
2.11. CONCURRENCY	14
<b>3. JAVA COLLECTIONS GUIDE</b>	
3.1. OVERVIEW OF JAVA COLLECTIONS	16
3.2. IMPORTANTS IN JAVA DEVELOPMENT	17

---

3.3.	BENEFITS AND USECASES	18
3.4.	CORE COLLECTION INTERFACES	18
3.5.	COMMON COLLECTION IMPLEMENTATIONS	20
3.6.	ITERATORS AND COLLECTIONS API	21
3.7.	CUSTOM COLLECTIONS AND GENERICS	22
4.	<b>FUNDAMENTALS OF DBMS</b>	
4.1.	INTRODUCTION	24
4.2.	CHARACTERISTICS OF A DBMS	25
4.3.	DATA MODEL	27
4.4.	THREE - SCHEMA ARCHITECTURE	28
4.5.	DBMS COMPONENT MODULES	29
4.6.	ENTITY-RELATIONSHIP (ER) MODEL	30
4.7.	RELATIONAL SCHEMA	31
5.	<b>FUNDAMENTALS OF SQL</b>	
5.1.	INTRODUCTION	32
5.2.	SQL COMMANDS	32
5.3.	DATA DEFINITION LANGUAGE	34
5.4.	DATA MANIPULATION LANGUAGE	35
5.5.	DATA CONTROL LANGUAGE	36
5.6.	TRANSACTION CONTROL LANGUAGE	37
5.7.	DATA QUERY LANGUAGE	38
6.	<b>DESIGN AND ARCHITECTURE</b>	
6.1.	DESIGN GOALS	39
6.2.	DATABASE STRUCTURE	41
6.3.	HIGH LEVEL ARCHITECTURE	42

---

---

6.4. CLASS DIAGRAM	43
<b>7. IMPLEMENTATION</b>	
7.1. APPLICATION OF THE PROJECT	44
7.2. CONNECTING THE DATABASE TO THE APPLICATION	46
<b>8. TESTING</b>	
8.1. UNIT TESTING	48
8.2. INTEGRATION TESTING	50
8.3. SYSTEM TESTING	51
<b>9. RESULTS</b>	
9.1. HOME PAGE	55
9.2. SIGNUP PAGE	55
9.3. LOGIN PAGE	56
9.4. INCASE OF WRONG LOGIN	56
9.5. SEARCH BOOK	57
9.6. SEARCH BOOK BY AUTHOR	57
9.7. SEARCH BOOK BY TYPE	58
9.8. LIST OF BOOKS	58
9.9. BOOK DETAILS	59
9.10. BILL DETAILS	59
9.11. USER INFORMATION FROM DATABASE	60
9.12. BOOK INFORMATION FROM DATABASE	60
9.13. ORDER NOTED	61
<b>10. CONCLUSION</b>	62
<b>REFERENCES</b>	63

---

---

## LIST OF FIGURES

Figure No	Figure Description	Page No
2.1	JAVA ARCHITECTURE	6
6.1	USE CASE DESIGN	39
6.31	DATAFLOW DIAGRAM	41
6.32	HIGH LEVEL ARCHITECTURE	42
2.2	CLASS DIAGRAM	43
9.1	HOME PAGE	55
9.2	SIGNUP PAGE	55
9.3	LOGIN PAGE	56
9.4	IN CASE OF WRONG LOGIN	56
9.5	SEARCH BOOK	57
9.6	SEARCH BOOK BY AUTHOR	57
9.7	SEARCH BOOK BY TYPE	58
9.8	LIST OF BOOKS	58
9.9	BOOK DETAILS	59
9.10	BILL DETAILS	60
9.11	USER INFORMATION FROM DATABASE	61
9.12	BOOK INFORMATION FROM DATABASE	61
9.13	ORDER NOTED	62



## CHAPTER 1

# INTRODUCTION

### 1.1 PROBLEM DEFINITION

The project focuses on addressing the need for a convenient and efficient online book shopping platform, named "Book Hub," to cater to the evolving preferences of customers. In the current landscape, where digital experiences play a crucial role, there is a requirement for a user-friendly web-based system that enables users to effortlessly explore and purchase a wide variety of books from the comfort of their homes.

1. **Authentication and Accessibility:** The system necessitates users to create an account using a unique email and password, ensuring secure access and personalization. This eliminates the necessity for physical visits to stores, making book shopping accessible to a broader audience.
2. **Efficient Online Shopping:** Book Hub streamlines the book shopping process, offering an intuitive interface for users to easily navigate through a vast collection of books categorized by genre, author, or other relevant criteria. The goal is to eliminate the hassles associated with traditional shopping and provide a seamless online alternative.
3. **Order Processing and Dispatch:** Once users log in, they can add desired books to their virtual shopping cart and proceed to checkout. The system facilitates secure payment transactions and ensures timely dispatch of orders, providing customers with a convenient and efficient book purchasing experience.
4. **Inclusivity and Accessibility:** The design of Book Hub aims to be inclusive, accommodating various user preferences and technological proficiencies. The user interface is crafted to be intuitive, ensuring a positive experience for users of diverse backgrounds.

5. Scalability and Adaptability: Book Hub's architecture is constructed with scalability in mind, allowing for future enhancements and updates. The system can adapt to evolving user expectations and technological advancements, ensuring its relevance and efficiency over time.

The problem at hand involves developing an online book shopping platform that prioritizes user authentication, accessibility, efficient shopping processes, timely order dispatch, transparent communication, inclusivity, and scalability. Book Hub seeks to redefine the online book shopping experience, providing a solution that aligns with the preferences and demands of modern-day customers.

## 1.2 OBJECTIVES

The primary objective of "Book Hub" is to establish a user-friendly and efficient online book shopping platform that addresses the challenges and inconveniences associated with traditional book purchasing. The key goals include:

### 1. User Registration:

- Allow users to register and maintain personalized profiles within the application.
- Facilitate both existing users to log in and new users to sign up if no registered account exists.

### 2. Book Category Search:

- Enable users to search for books based on predefined static categories.
- Display a list of books under the selected category, providing users with the option to choose from the available titles.

### 3. Place Order and Billing:

- Allow users to add selected books to their cart and proceed to place orders.
- Generate a bill from the cart, summarizing the cost of the selected items.
- Implement a confirmation mail system to send order details to the registered email address upon successful order placement.

#### 4. Admin Login and Order Tracking:

- Provide an administrative login for the admin to access a dedicated interface.
- Allow the admin to view a list of orders placed by customers, including date and time details.
- Facilitate effective order tracking and management for the administrative user.

The online bookstore application aims to deliver a user-friendly experience for customers, streamlining the process from book selection to order placement. Additionally, the admin interface ensures efficient order tracking and management, contributing to the overall success and functionality of the application.

### **1.3 METHODOLOGY TO BE FOLLOWED**

The development of the online bookstore application involves a systematic and phased approach to ensure the successful implementation of the specified features. The methodology can be outlined as follows:

#### 1. Requirement Analysis:

- Conduct a detailed analysis of the requirements outlined in the problem definition.
- Identify and prioritize key functionalities, including user registration, book category search, order placement, and admin functionalities.

#### 2. Design:

- Design the user interface for both customers and administrators, ensuring a user-friendly experience.
- Develop database schemas to store user profiles, book information, and order details.
- Create wireframes or mockups to visualize the application's layout and flow.

#### 3. Database Implementation:

- Implement a relational database using MySQL to store user data, book details, and order information.
- Establish relationships between different entities such as users, books, and orders.
- Database Interaction (JDBC):
- Use JDBC (Java Database Connectivity) to connect to the MySQL database.
- Implement functions to fetch data from the database based on user queries and return it to the GUI.

#### 4. User Authentication and Profile Management:

- Develop the authentication system, allowing users to register, log in, and manage their profiles.
- Implement secure password storage and retrieval mechanisms.

#### 5. Book Category Search:

- Define and categorize books based on static categories.
- Implement a search mechanism to display books under selected categories.
- Ensure the availability of a user-friendly interface for browsing and selecting books.

#### 6. Shopping Cart and Order Placement:

- Create a shopping cart feature to add and manage selected books.
- Develop the order placement process, including the generation of bills.
- Implement email confirmation to send order details to the registered email address.

#### 7. Admin Interface:

- Design a separate admin login interface with secure access controls.
- Develop functionalities for the admin to view and track orders placed by customers.

## 1.4 EXPECTED OUTCOMES

- Login and Registration : A user can register in the application and maintain his profile. After login user resume shopping.
- Existing user can log in in the application and new user can sign up if there is no registered account.
- Book category: The user can search a book according to the category. There is some static category. When user will search a category, all the books under the category will be listed on the resulted page. The user can select any book
- Place order: Once place order is done, bill is generated from the cart and a confirmation mail is sent to registered mail address.
- Admin login: For admin login, admin can view the list of orders placed by the customer along with date and time.

## 1.5 HARDWARE AND SOFTWARE REQUIREMENTS

### Hardware Specifications:-

- RAM: 256MB
- Processor: Pentium IV or above
- Speed: 2.50 GHz

### Software Specifications:-

- Operating System: Windows
- Developing language used: Java, MySQL
- Tool used: Eclipse IDE(Neon 3), MySQL

## CHAPTER 2

# FUNDAMENTALS OF JAVA

## 2.1 INTRODUCTION TO JAVA

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers “write once, run anywhere” (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.

For example, you can write and compile a Java program on UNIX and run it on Microsoft Windows, Macintosh, or UNIX machine without any modifications to the source code. WORA is achieved by compiling a Java program into an intermediate language called byte code. The format of byte code is platform-independent. A virtual machine, called the Java Virtual Machine (JVM), is used to run the byte code on each platform.

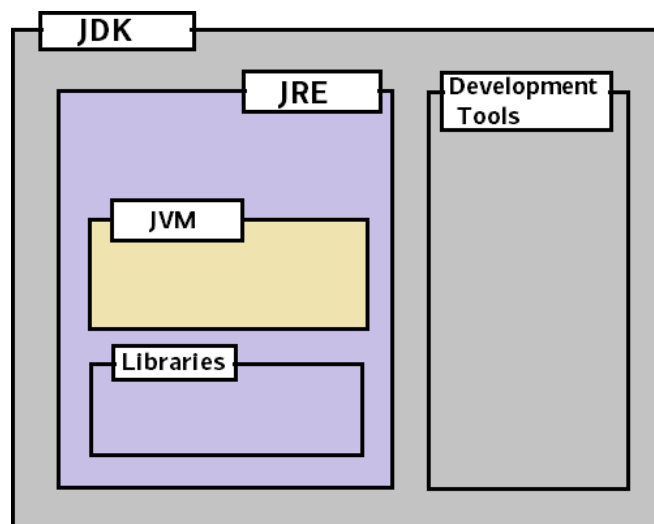


Figure 2.1 Java Architecture

## 2.2 ADVANTAGES OF JAVA

1. Platform Independence (Write Once, Run Anywhere): Java programs can run on any device that has the Java Virtual Machine (JVM). This allows Java applications to be platform-independent.
2. Object-Oriented: Java follows an object-oriented programming (OOP) paradigm, which encourages the use of objects, classes, and encapsulation for creating modular and reusable code.
3. Simple and Easy to Learn: Java was designed to be easy to learn and use, with a syntax similar to C++. It eliminates complex features such as pointers and operator overloading, making it more straightforward for beginners.
4. Robust and Secure: Java includes features like automatic memory management (garbage collection), exception handling, and type checking, which contribute to the language's robustness and security. The Java Virtual Machine adds an additional layer of security by running Java bytecode in a controlled environment.
5. Distributed Computing: Java supports distributed computing through its Remote Method Invocation (RMI) and Java Naming and Directory Interface (JNDI) features, making it suitable for building networked applications.
6. Multi-threading: Java provides built-in support for multithreading, allowing developers to create concurrent applications that can execute multiple tasks simultaneously. This is crucial for building efficient and responsive software.
7. Dynamic and Extensible: Java supports dynamic loading of classes, which allows classes to be loaded on-demand. Additionally, Java's reflection capabilities enable inspection and manipulation of classes at runtime.
8. High Performance: Java achieves high performance through the use of Just-In-Time (JIT) compilation, which translates Java bytecode into machine code at runtime. This helps in optimizing the execution speed of Java programs.

9. Large Standard Library: Java comes with a comprehensive standard library (Java API) that provides pre-built, reusable classes and packages, covering a wide range of functionalities such as networking, I/O, data structures, and more.
10. Community Support: Java has a large and active community of developers, which means abundant resources like documentation, forums, and third-party libraries. This community support contributes to the longevity and sustainability of the language.

## 2.3 DATA TYPES

### Primitive Data Types:

1. int: Represents integer values, e.g., 10, -5, 1000.
2. double: Represents floating-point numbers, e.g., 3.14, -0.5, 2.0.
3. char: Represents a single character, e.g., 'A', '5', '\$'.
4. boolean: Represents true or false values.

### Reference Data Types:

1. String: Represents a sequence of characters, e.g., "Hello, World!".
2. Arrays: Represents a collection of similar data types, e.g., int[], String[].
3. Classes: Represents user-defined data types through classes.

## 2.4 CONTROL FLOW

Control flow in programming refers to the sequence in which statements are executed within a program. It involves determining the logical order of code execution, deciding which code block to execute based on specific conditions. In Java, control flow is facilitated through diverse structures:



1. Conditional Statements (if, else if, else): These statements enable the execution of specific code blocks depending on given conditions. For instance, an "if" statement  
2. executes a block of code when a specified condition is true.
  2. Switch Statements: This construct provides an alternative for handling multiple conditions. It evaluates an expression and executes a corresponding code block based on the evaluated value.
  3. Loops (for, while, do-while): Loops allow the repetitive execution of a code block. A "for" loop is suitable when the number of iterations is known, a "while" loop checks the condition before each iteration, and a "do-while" loop checks the condition after each iteration.
  4. Branching Statements (break, continue, return): These statements modify the standard control flow. "Break" exits a loop or switch statement prematurely, "continue" skips the remaining loop content and proceeds to the next iteration, while "return" exits a method.
- Comprehending and managing control flow is crucial for crafting efficient and responsive programs, empowering developers to design logic that appropriately responds to diverse scenarios and user inputs.

## 2.5 METHODS

Methods in Java play a crucial role in structuring code by encapsulating specific functionalities, fostering code reusability, maintainability, and readability for an efficient program. A typical Java method includes a method signature and body.

1. Method Signature: It consists of the method name and parameters, defining the necessary input values. The return type specifies the type of value returned, with "void" indicating no return.
2. Method Body: Enclosed in curly braces {}, it contains the actual code determining the method's behavior.

Java accommodates static methods (linked to the class), instance methods (associated with an instance of the class), getter and setter methods (for private instance variable access), and constructor methods (special methods initializing objects).

To invoke a method, its name is called, followed by parentheses with any required arguments. A solid grasp of methods is vital for developing well-organized and modular Java programs, elevating the software development process.

## 2.6 OBJECT ORIENTED CONCEPTS

Object-Oriented Programming (OOP) is a programming paradigm based on the concept of "objects," which encapsulate data and behaviour. Java is a fully object-oriented programming language, and it incorporates the fundamental principles of OOP. Here are the key OOP concepts in Java:

### 1. **Classes and Objects:**

- **Class:** A class is a blueprint or template that defines the attributes and behaviours of objects. It is a user-defined data type.
- **Object:** An object is an instance of a class. It represents a real-world entity and has both state (attributes) and behaviour (methods).

### 2. **Encapsulation:**

- Encapsulation involves bundling the data (attributes) and methods (behavior) that operate on the data into a single unit, i.e., a class. It helps in data hiding and protects the integrity of the object.

### 3. **Inheritance:**

- Inheritance is a mechanism that allows a class (subclass or derived class) to inherit properties and behaviors from another class (superclass or base class). It promotes code reuse and establishes an "is-a" relationship.

### 4. **Polymorphism:**

- Polymorphism allows objects of different classes to be treated as objects of a common super class. It includes method overloading (compile-time polymorphism) and method overriding (run-time polymorphism).

5. **Abstraction:**

- Abstraction is the process of hiding the implementation details and showing only the essential features of an object. Abstract classes and interfaces are used to achieve abstraction in Java.

## 2.7 EXCEPTION HANDLING

1. Try-Catch Blocks:

- Used to handle exceptions.
- try block contains code that might throw an exception.
- catch block catches and handles the exception.

2. Multiple Catch Blocks:

- Allows handling different types of exceptions.
- Blocks are evaluated in order; the first matching block is executed.

3. Finally Block:

- Contains code that always executes, regardless of whether an exception occurs.
- Used for cleanup operations.

4. Throw Statement:

- Used to manually throw an exception.
- Helpful for signaling errors in specific conditions.

5. Custom Exceptions:

- Extend the Exception class to create custom exceptions.
- Enables application-specific exception handling.

6. Exception Hierarchy:

- Hierarchical structure with Throwable as the base class.

- Exception for recoverable conditions, Error for unrecoverable conditions.

7. Checked and Unchecked Exceptions:

- Checked exceptions are checked at compile-time, requiring handling or declaration.
- Unchecked exceptions (RuntimeExceptions) not checked at compile-time.

8. Exception Propagation:

- Uncaught exceptions propagate up the call stack until caught or program terminates.
- Methods can declare exceptions they might throw using throws clause.

9. Suppressed Exceptions:

- Introduced in Java 7.
- Used in try-with-resources to handle exceptions during resource cleanup.

10. Handling Multiple Exceptions:

- Single try block can have multiple catch blocks for different exceptions.
- 

## 2.8 FILE HANDLING

File handling in Java is a crucial aspect that allows the manipulation and management of files in a program. It provides functionality to read from and write to files, creating a seamless interaction between the program and external data storage. The process involves several key steps:

1. **File Classes:** Java utilizes classes like File, FileReader, FileWriter, BufferedReader, and BufferedWriter to handle different file operations.

2. **File Reading:** The FileReader and BufferedReader classes enable reading data from a file. It involves opening the file, reading its contents line by line, and closing the file after processing.

**3. File Writing:** Conversely, the `FileWriter` and `BufferedWriter` classes facilitate writing data to a file. The steps include opening the file, writing data to it, and closing the file to save changes.

**4. Exception Handling:** Since file operations involve external resources, proper exception handling is crucial to address potential errors, ensuring robust and error-resistant programs.

## 2.9 PACKAGES AND IMPORT

Packages and imports in Java serve as essential organizational tools for code management and reusability. A package is a container that holds related classes, preventing naming conflicts and providing a hierarchical structure. It enables categorizing classes into meaningful groups, enhancing code organization. Import statements facilitate the utilization of classes from external packages, avoiding fully qualified names. By importing specific classes or entire packages, developers streamline code readability and reduce redundancy. Java offers a standard set of packages (e.g., `java.lang`) that are implicitly imported, ensuring fundamental functionalities are readily available. Custom packages can be created to encapsulate related classes, promoting modularity. The use of packages and imports fosters a systematic approach to coding, simplifying maintenance and collaboration in large-scale projects. Overall, these features contribute to the maintainability, scalability, and clarity of Java code, aligning with the principles of structured and modular programming.

## 2.10 INTERFACES

Interfaces in Java provide a blueprint for defining a set of abstract methods that concrete classes must implement. They play a vital role in achieving abstraction and facilitating multiple inheritances, allowing a class to implement multiple interfaces. An interface

declares method signatures without providing their implementation details. Concrete classes that implement an interface must provide actual code for the declared methods.

Interfaces support the development of loosely coupled and highly cohesive code, as they enable classes to interact based on shared behaviors without specifying their underlying structures. This enhances code flexibility, reusability, and adaptability. Additionally, interfaces support the creation of APIs, promoting a standardized way for classes to communicate.

Java interfaces are integral to the implementation of design patterns, such as the Strategy Pattern, enabling dynamic behavior changes. They contribute to code organization by grouping related functionalities under a common contract.

## **2.11 CONCURRENCY**

Concurrency in Java refers to the ability of a program to execute multiple tasks simultaneously, allowing for efficient utilization of resources and improved application responsiveness. Key concepts and mechanisms in Java support concurrent programming, enhancing the development of multi-threaded applications.

### **1. Thread Class and Runnable Interface:**

- Java provides the Thread class and the Runnable interface to create and manage threads.
- Threads represent the smallest unit of execution within a process.

### **2. Synchronization:**

- Synchronization ensures that only one thread can access a shared resource at a time, preventing data corruption.
- Keywords like synchronized and methods like wait() and notify() facilitate synchronization.

### **3. Executor Framework:**

- The Executor framework simplifies the management of threads and the execution of asynchronous tasks.

#### **4. Concurrency Utilities:**

- Java provides utilities in the `java.util.concurrent` package, including `Lock` interface, `Semaphore`, and `CountDownLatch`.

#### **5. Atomic Variables:**

- Classes in the `java.util.concurrent.atomic` package offer atomic operations, ensuring thread-safe updates.

## CHAPTER 3

# JAVA COLLECTIONS GUIDE

### 3.1 OVERVIEW OF JAVA COLLECTIONS

Java Collections Framework is a set of classes and interfaces in Java that provide an architecture to store, organize, and manipulate a group of objects. It includes commonly used data structures like List, Set, Map, Queue, and their implementations. The Collections Framework simplifies data manipulation and makes it easy to perform common operations on collections of objects.

1. **Interfaces:** The framework includes core interfaces like Collection, List, Set, Queue, and Map that define common behaviors for different types of collections.
2. **Classes:** Implementation classes such as ArrayList, LinkedList, HashSet, TreeSet, HashMap, and TreeMap provide concrete implementations of the collection interfaces.
3. **Dynamic Sizing:** Dynamic sizing is a feature of many collection classes like ArrayList, allowing them to grow or shrink dynamically based on the number of elements.
4. **Iterators:** Iterators provide a uniform way to traverse elements in a collection. The enhanced for loop (for-each) simplifies iteration.
5. **Algorithms:** The Collections class offers static methods for common algorithms like sorting, shuffling, and searching, providing consistency across different collections.
6. **Concurrency Collections:** The `java.util.concurrent` package extends the collections framework to support concurrent programming, offering thread-safe alternatives like `ConcurrentHashMap`.



7. **Generics:** Generics allow the specification of the type of elements in a collection, enhancing type safety and eliminating the need for casting.
8. **Autoboxing:** Autoboxing automatically converts primitive types to their corresponding wrapper classes, simplifying the handling of primitive data types in collections.
9. **Comparable and Comparator:** Objects in collections can be sorted using the natural order (Comparable interface) or a custom order (Comparator interface).
10. **Java Streams:** Introduced in Java 8, the Stream API provides a functional-style approach for processing collections, enabling concise and expressive code.

## **3.2 IMPORTANCE IN JAVA DEVELOPMENT**

Java Collections are vital in Java development, simplifying how data is managed. They bring numerous benefits to programs, making them more efficient and reliable. Collections make it easier to handle complex tasks like searching, sorting, and iterating over elements, contributing to the robustness and scalability of Java applications. They adapt well to different data sizes, ensuring programs perform optimally even with extensive data manipulation needs.

Beyond their basic functions, Java Collections offer versatility for implementing advanced algorithms and data structures, enabling developers to handle intricate scenarios effortlessly. Their dynamic nature allows them to scale according to the changing requirements of diverse applications. This adaptability is crucial for maintaining peak performance in various scenarios.

Moreover, Java Collections improve code readability and maintainability. By providing a consistent way to handle data structures, they encourage clean and understandable code. This clarity in code promotes collaboration among developers and supports the ongoing maintenance and evolution of Java applications.

## **3.3 BENEFITS AND USECASES**

1. Efficient Data Management: Collections provide efficient data structures and algorithms for managing and manipulating data, facilitating seamless operations like insertion, deletion, and retrieval.
2. Code Reusability: Developers can reuse collection classes, promoting modular and reusable code. This simplifies the development process and accelerates project timelines.
3. Enhanced Performance: Collections are designed to deliver optimal performance in terms of time and space complexity, ensuring efficient handling of data even in large-scale applications.
4. Versatility in Data Handling: Collections cater to diverse use cases, accommodating different types of data structures and allowing developers to choose the most suitable collection based on specific requirements.
5. Simplified Iteration: Collections provide interfaces and classes that simplify the iteration process, making it easier for developers to traverse and process elements within a collection.
6. Facilitates Sorting and Searching: Collections offer built-in methods for sorting and searching elements, streamlining common tasks, and enhancing the overall efficiency of Java programs.

### **3.4 CORE COLLECTIONS INTERFACES**

Java's core collection interfaces form the foundation of the Java Collections Framework, defining fundamental behaviors and structures that various collection classes implement. These interfaces provide a standardized way to interact with different types of collections, offering consistency and ease of use. Here are the key core collection interfaces:

1. Collection Interface:

- Represents the root interface of the collection hierarchy.
- Specifies basic operations like add, remove, and contains.
- Subinterfaces include List and Set.

2. List Interface:

- Extends the Collection interface and represents an ordered collection.
- Allows duplicate elements and maintains the order of insertion.
- Key implementations include ArrayList and LinkedList.

3. Set Interface:

- Also extends the Collection interface but does not allow duplicate elements.
- No specific order is maintained for elements.
- Notable implementations include HashSet and TreeSet.

4. Queue Interface:

- Extends Collection and represents a collection designed for holding elements before processing.
- Follows the FIFO (First-In-First-Out) order.
- Key implementation is LinkedList.

5. Map Interface:

- Represents a collection of key-value pairs.
- Does not extend Collection as it deals with pairs rather than individual elements.
- Common implementations include HashMap and TreeMap.

6. Deque Interface:

- Stands for "double-ended queue."
- Extends Queue and allows elements to be inserted or removed from both ends.
- Implemented by LinkedList.

## 3.5 COMMON COLLECTION IMPLEMENTATIONS

In Java, common collection implementations refer to the concrete classes that realize the core collection interfaces, providing specific implementations for various data structures. These classes are crucial for developers as they offer practical solutions to diverse programming needs. Here are some key common collection implementations in Java:

### 1. Array List:

- Implements the List interface.
- Represents a dynamically resizable array.
- Provides fast random access and efficient element manipulation.
- Suitable for scenarios where frequent access and modification are required.

### 2. LinkedList:

- Also implements the List interface.
- Represents a doubly-linked list.
- Supports fast sequential access and efficient insertion/deletion at both ends.
- Well-suited for scenarios requiring frequent insertion or removal.

### 3. HashSet:

- Implements the Set interface.
- Stores elements in a hash table, offering constant-time performance for basic operations.
- Does not guarantee order.
- Ideal for scenarios where uniqueness is essential.

### 4. TreeSet:

- Also implements the Set interface.

- Stores elements in a sorted tree structure.
- Maintains order based on the natural order or a provided comparator.
- Useful when a sorted set is required.

5. HashMap:

- Implements the Map interface.
- Stores key-value pairs in a hash table.
- Offers constant-time performance for basic operations.
- Efficient for scenarios requiring key-based data retrieval.

6. TreeMap:

- Also implements the Map interface.
- Stores key-value pairs in a sorted tree structure.
- Maintains order based on the natural order or a provided comparator.
- Suitable for scenarios where a sorted map is needed.

7. PriorityQueue:

- Implements the Queue interface.
- Represents a priority queue based on the heap data structure.
- Elements are retrieved based on their priority.
- Useful for applications involving prioritized processing.

## 3.6 ITERATORS AND COLLECTIONS API

In Java, iterators and the Collections API form a powerful combination for efficiently traversing and manipulating collections. Here's an overview of these concepts:

Iterators:

- Iterators provide a uniform way to access elements of a collection without exposing its underlying implementation.
- The Iterator interface offers methods like `hasNext()` and `next()` for sequential access to elements.
- Iterators are applicable to various collection types, including lists, sets, and maps.
- They enable safe and efficient traversal, supporting removal of elements during iteration.

#### Collections API:

- The Collections API in Java encompasses a set of interfaces and classes to represent and manipulate collections.
- Key interfaces include `Collection`, `List`, `Set`, `Queue`, and `Map`, each tailored for specific use cases.
- The `Collections` class provides utility methods for working with collections, offering functionalities like sorting, searching, and synchronizing.
- This API promotes consistency and ease of use across different collection types.

### **3.7 CUSTOM COLLECTIONS AND GENERICS**

Custom collections and generics offer powerful features for creating specialized data structures with enhanced type safety and reusability.

#### Custom Collections:

- Developers can design custom collections tailored to specific needs by implementing collection interfaces such as `List`, `Set`, or `Map`.
- Custom collections are useful when standard collections do not precisely meet the requirements of an application.

- Creating custom collections involves defining classes that encapsulate desired behaviors and functionalities.

Generics:

- Generics enable the creation of classes, interfaces, and methods that operate on parameterized types, allowing developers to write code that works with various data types.
- They enhance type safety by providing compile-time checks and eliminating the need for explicit type casting.
- Common use cases for generics include creating generic classes like `ArrayList<T>` and writing methods that can handle different types.

## CHAPTER 4

# FUNDAMENTALS OF DBMS

### 4.1 INTRODUCTION

A Database Management System (DBMS) is software that provides an interface for interacting with databases and managing the storage, retrieval, and manipulation of data. The primary goal of a DBMS is to provide an efficient and secure way to organize and access large volumes of data.

Key components and concepts of a DBMS include:

1. Database Examples:

- MySQL, PostgreSQL, Oracle Database, and Microsoft SQL Server are common examples.

2. Structured Query Language (SQL):

- Standard language for interacting with relational databases.
- Used for defining and manipulating data in the database.

3. Table:

- Fundamental structure in a relational database.
- Stores data in rows and columns.
- Each row represents a record, and each column represents an attribute.

4. Query:

- Request for information from the database.
- Written in SQL for retrieving, updating, or deleting data.

5. Normalization:

- Process of organizing data to reduce redundancy and improve data integrity.



- Involves breaking down large tables into smaller, related tables.

6. ACID Properties:

- Set of properties ensuring transaction reliability in a database.
- ACID stands for Atomicity, Consistency, Isolation, and Durability.

7. Transaction:

- Unit of work on a database with operations like insert, update, delete.
- Follows ACID properties for data consistency.

8. Index:

- Data structure improving data retrieval speed in a database table.
- Created on specific columns for faster query processing.

9. Concurrency Control:

- Mechanism ensuring multiple transactions can execute concurrently without data inconsistency.
- Techniques include locking, timestamping, and optimistic concurrency control.

10. Data Security:

- Measures to protect data from unauthorized access, modification, or destruction.
- Includes user authentication, authorization, and encryption.

## **4.2 CHARACTERISTICS OF DBMS**

1. Data Independence:

- Ensures changes in logical or physical structure do not impact application programs.

2. Data Integrity:

- Maintains accuracy, consistency, and reliability through constraints.

3. Data Security:

- Controls access with authentication, authorization, and encryption.

4. Data Recovery:

- Recovers the database after system failures using features like transaction logging.

5. Concurrency Control:

- Manages simultaneous access, preventing interference and ensuring data consistency.

6. Query Language Support:

- Provides SQL for defining and manipulating data.

7. Data Abstraction:

- Presents a simplified view, hiding complexities through layers.

8. Multiuser and Concurrent Access:

- Supports multiple users concurrently with concurrency control mechanisms.

9. Scalability:

- Handles growing data and user loads by adding resources.

10. Backup and Recovery:

- Prevents data loss through backups and restores consistency after failures.

11. Transaction Management:

- Manages transactions as single units, adhering to ACID properties.

12. Data Dictionary Management:

- Maintains a repository with information about the database structure.

13. Centralized Control:

- Provides centralized control over data management.

14. Ad Hoc Query Support:

- Allows flexible ad-hoc queries for data retrieval and analysis.

15. Application Programming Interface (API):

- Offers APIs for integration, enabling programmatic interaction with the database.

## 4.3 DATA MODEL

A data model in the context of a Database Management System (DBMS) serves as a blueprint for structuring and organizing data within a database. It defines how data elements are related to each other and how they can be accessed and manipulated. There are various types of data models, with the relational data model being one of the most widely used.

1. Relational Data Model: This model represents data as tables with rows and columns, emphasizing relationships between tables. Entities and their attributes are organized to ensure data integrity and consistency. The use of primary and foreign keys establishes connections between tables.

2. Entity-Relationship Model (ER Model): This model focuses on entities, their attributes, and the relationships between them. It is particularly useful in visualizing the overall structure of a database, making it a popular choice during the initial design phase.

3. Object-Oriented Data Model: This model extends the principles of object-oriented programming to the database realm. It involves the representation of real-world entities as objects, complete with attributes and methods, fostering a more natural representation of complex structures.

4. Hierarchical Data Model: In this model, data is organized in a tree-like structure, with a parent-child relationship between records. Each record, except the root, has a single parent, leading to a hierarchical arrangement.

## 4.4 THREE – SCHEMA ARCHITECTURE

The Three-Schema Architecture, also known as the three-schema approach, is a framework used in Database Management Systems (DBMS) to separate the user interface, logical schema, and physical schema. This architecture enhances database management by providing a clear and organized structure for data representation.

### 1. User Schema (External Schema):

This top layer of the architecture focuses on the user interface and represents how data is viewed by different user groups. Each user or application interacts with the database through their specific user schema, which includes only the relevant data and structures needed for their tasks. This layer promotes data independence, allowing modifications to the logical or physical schema without affecting the user interface.

### 2. Logical Schema (Conceptual Schema):

Positioned in the middle layer, the logical schema defines the overall structure and organization of the data. It describes the relationships and constraints between different data elements, providing a conceptual understanding of the entire database. The logical schema serves as a bridge between the user schema and the physical schema, facilitating communication between different user views and the underlying data storage.

### 3. Physical Schema (Internal Schema):

At the bottom layer, the physical schema focuses on the actual storage and retrieval of data. It deals with aspects such as data storage structures, indexing mechanisms, and access paths. The physical schema is concerned with optimizing data storage and retrieval for efficient performance. Changes made at this layer, such as modifications to storage structures or indexing, do not impact the logical or user schema.

## 4.5 DBMS COMPONENT MODULES

The components of a Database Management System (DBMS) play crucial roles in facilitating the storage, retrieval, and management of data. These components work together to ensure the integrity, security, and efficient operation of a database. Here's an overview of key DBMS components:

1. Query Processor: Responsible for translating user queries into a series of instructions that the database can understand and execute. It includes components for query optimization to enhance performance.
2. Database Engine: Manages the core functionality of the DBMS, including data storage, retrieval, and indexing. It interprets and executes commands issued by the query processor.
3. Transaction Manager: Ensures the ACID properties (Atomicity, Consistency, Isolation, Durability) of database transactions. It oversees the execution of multiple operations as a single, atomic unit.
4. Data Dictionary: Stores metadata about the database, including information about tables, relationships, and constraints. It provides a centralized repository for data definitions.
5. Storage Manager: Handles the physical organization of data on storage media, optimizing access and retrieval. It manages tasks such as data storage, indexing, and buffering.
6. Security Component: Enforces access controls to ensure that only authorized users can perform specific operations on the database. It includes authentication and authorization mechanisms.
7. Backup and Recovery Manager: Manages the creation of database backups and handles recovery in case of system failures. This component ensures data durability and availability.

8. Concurrency Control Manager: Coordinates access to the database by multiple users or transactions simultaneously, preventing conflicts and ensuring data consistency.

9. Database Utilities: Provides tools for database administration tasks, such as loading data, monitoring performance, and optimizing database structures.

## **4.6 ENTITY – RELATIONSHIP (ER) MODEL**

The ER Model provides a visual and systematic way to design databases, ensuring a clear understanding of the data structure and relationships. It serves as a foundation for creating the relational schema that will be implemented in a database management system.

1. Entities: Entities are objects or concepts in the real world that are represented in the database. Each entity has attributes that describe its properties.

2. Attributes: Attributes are properties or characteristics of entities.

3. Relationships: Relationships illustrate connections between entities. They define how entities interact with each other.

4. Key Attributes: Each entity has a key attribute that uniquely identifies instances of that entity.

5. Cardinality: Cardinality defines the number of instances of one entity that can be associated with the number of instances of another entity. It helps specify the nature of relationships, such as one-to-one or one-to-many.

6. Weak Entities: A weak entity is an entity that cannot be uniquely identified by its attributes alone and relies on a related entity for identification. They are often represented with double rectangles.

## 4.7 RELATIONAL SCHEMA

A relational schema in database management represents the structure of the tables, their attributes, and the relationships between them. It serves as a blueprint for organizing and storing data in a relational database.

1. Tables: A relational schema consists of tables, each representing a specific entity or concept.
2. Attributes: Tables contain attributes, which are properties or characteristics of the entities they represent. Each column in a table corresponds to an attribute.
3. Primary Key: Each table has a primary key, which is a unique identifier for each record in the table. It ensures that each row can be uniquely identified.
4. Foreign Key: Relationships between tables are established using foreign keys. A foreign key is a column in a table that refers to the primary key in another table.
5. Relationships: Relationships define how tables are related to each other. Common relationship types include one-to-one, one-to-many, and many-to-many.
6. Normalization: The process of normalization is applied to ensure that the relational schema minimizes redundancy and dependency. It involves organizing tables to eliminate data anomalies and improve data integrity.
7. Denormalization: In some cases, denormalization may be applied to optimize query performance. It involves introducing redundancy to simplify and speed up data retrieval.

## CHAPTER 5

# FUNDAMENTALS OF SQL

### 5.1 INTRODUCTION

MySQL is a popular open-source relational database management system (RDBMS) that plays a fundamental role in data storage and retrieval for numerous applications. Developed by Oracle Corporation, MySQL is known for its reliability, ease of use, and scalability. As an RDBMS, it organizes data into structured tables, allowing for efficient querying and management. MySQL supports SQL (Structured Query Language), enabling users to interact with databases seamlessly. Its versatility makes it suitable for various applications, from small-scale projects to large-scale enterprises.

One of MySQL's strengths is its compatibility with different operating systems, including Windows, Linux, and macOS. It offers robust features such as transaction support, indexing, and security mechanisms to ensure data integrity. MySQL is commonly utilized in web development scenarios, often paired with scripting languages like PHP. It has a vibrant community and extensive documentation, making it accessible for developers worldwide.

### 5.2 SQL COMMANDS

SQL (Structured Query Language) commands are essential for interacting with relational databases. Here are some fundamental SQL commands:

1. SELECT: Retrieves data from one or more tables.

```
>sql
```

```
SELECT column1, column2 FROM table WHERE condition;
```

2. INSERT: Adds new records to a table.



```
>sql
```

```
INSERT INTO table (column1, column2) VALUES (value1, value2);
```

3. UPDATE: Modifies existing records in a table.

```
>sql
```

```
UPDATE table SET column1 = value1 WHERE condition;
```

4. DELETE: Removes records from a table.

```
>sql
```

```
DELETE FROM table WHERE condition;
```

5. CREATE TABLE: Creates a new table with specified columns and data types.

```
>sql
```

```
CREATE TABLE table (  
    column1 datatype1,  
    column2 datatype2,  
    ...  
);
```

6. ALTER TABLE: Modifies an existing table (e.g., adds or drops columns).

```
>sql
```

```
ALTER TABLE table ADD column datatype;
```

7. DROP TABLE: Deletes an entire table.

```
>sql
```

```
DROP TABLE table;
```

8. CREATE DATABASE: Creates a new database.

```
>sql  
CREATE DATABASE database;
```

10. CREATE INDEX: Creates an index on one or more columns to improve query performance.

```
>sql  
CREATE INDEX index_name ON table (column1, column2);
```

These commands provide the foundational operations for managing and manipulating data within a relational database using SQL.

## 5.3 DATA DEFINITION LANGUAGE

Data Definition Language (DDL) in SQL is a subset of SQL commands used for defining and managing the structure of a database, including tables, relationships, and constraints. Here are some key DDL commands:

1. CREATE TABLE: Defines a new table with its columns, data types, and constraints.

```
sql  
CREATE TABLE table_name (  
    column1 datatype1 constraint1,  
    column2 datatype2 constraint2,  
    ...  
);
```

2. ALTER TABLE: Modifies an existing table, allowing actions like adding, modifying, or dropping columns.

```
sql  
ALTER TABLE table_name
```

ADD column\_name datatype constraint;

3. DROP TABLE: Deletes an existing table and all its data.

sql

DROP TABLE table\_name;

4. CREATE INDEX: Adds an index to one or more columns to enhance search performance.

sql

CREATE INDEX index\_name

ON table\_name (column1, column2, ...);

5. DROP INDEX: Removes an existing index from a table.

sql

DROP INDEX index\_name

ON table\_name;

6. CREATE DATABASE: Establishes a new database.

sql

CREATE DATABASE database\_name;

7. DROP DATABASE: Deletes an entire database and all its tables.

sql

DROP DATABASE database\_name;

These DDL commands are crucial for defining and managing the structure of a database, providing the foundation for data organization and storage.

## 5.4 DATA MANIPULATION LANGUAGE

Data Manipulation Language (DML) in SQL is a set of commands used to manage and manipulate data stored in a database. DML operations primarily involve querying,

inserting, updating, and deleting data within database tables.

1. SELECT Retrieves data from one or more tables based on specified criteria.

```
sql
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

2. INSERT: Adds new records into a table.

```
sql
INSERT INTO table_name (column1, column2 ...)
VALUES (value1, value2, ...);
```

3. UPDATE: Modifies existing records in a table based on a specified condition.

```
sql
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

4. DELETE: Removes records from a table based on a specified condition.

```
sql
DELETE FROM table_name
WHERE condition;
```

## 5.5 DATA CONTROL LANGUAGE

Data Control Language (DCL) in SQL is a set of commands that manage permissions and access control within a database. DCL commands primarily deal with defining and controlling the rights and privileges of database users. Two key DCL commands are:

1. GRANT: Provides specific privileges or permissions to a user or a group of users.

```
sql
GRANT privilege_type
```

ON object\_name

TO {user\_name | PUBLIC | role\_name};

- privilege\_type: The type of permission (e.g., SELECT, INSERT, UPDATE, DELETE).

- object\_name: The database object (e.g., table, view) to which the permission is granted.

- user\_name: The user or users to whom the permission is granted.

- PUBLIC: Grants the specified permission to all users.

- role\_name: The role to which the permission is granted.

2. **REVOKE**: Removes specific privileges from a user or a group of users.

sql

REVOKE privilege\_type

ON object\_name

FROM {user\_name | PUBLIC | role\_name};

- Similar parameters as in the GRANT command.

## 5.6 TRANSACTION CONTROL LANGUAGE

Transaction Control Language (TCL) in SQL consists of commands that manage transactions within a database. Two primary TCL commands are:

1. **COMMIT**: This command is used to permanently save any changes made during the current transaction. Once committed, the changes become a permanent part of the database.

sql

COMMIT;

2. **ROLLBACK**: This command is used to undo any changes made during the current transaction. It restores the database to its state before the transaction began.

sql

ROLLBACK;

TCL commands are essential for managing the durability and atomicity properties of transactions.

## 5.7 DATA QUERY LANGUAGE

Data Query Language (DQL) is a subset of SQL (Structured Query Language) specifically designed for querying and retrieving data from a database.

DQL commands focus on extracting information without modifying the database structure or content. The primary DQL command is **SELECT**, which retrieves data from one or more tables based on specified criteria. It allows users to filter, sort, and group data, providing a versatile mechanism for data retrieval. The **SELECT** statement is fundamental for generating reports, obtaining insights, and analyzing data stored in a relational database. DQL empowers users to interact with the database to gain meaningful information without affecting the underlying data.

## CHAPTER 6

# DESIGN AND ARCHITECTURE

### 6.1 DESIGN GOALS

A use case diagram is a dynamic or behavior diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform.

The project commences with a comprehensive analysis of user requirements to grasp the needs of job seekers and employers. This insight directs the creation of a user-friendly interface. Development focuses on building a modular, maintainable codebase using Java and JDBC for seamless database connectivity.

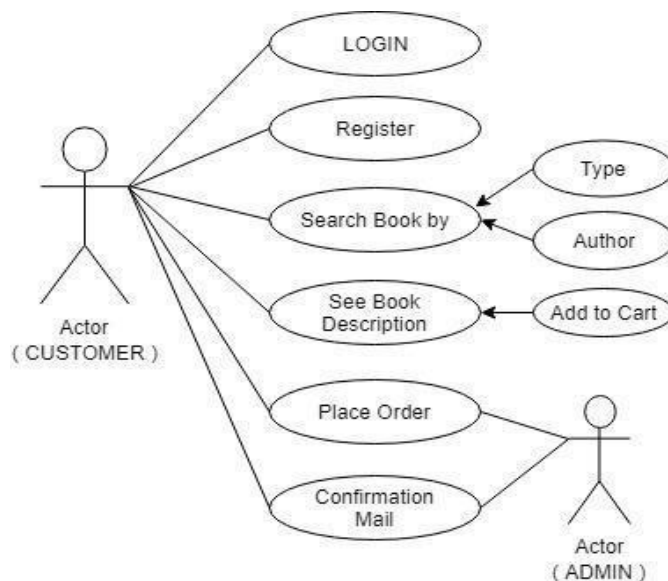


FIGURE 6.1 USE CASE DESIGN

## 6.2 DATABASE STRUCTURE

Based on the provided code, it appears that the application is interacting with a MySQL database named "Book Hub." Below is the database structure for the tables used in this code:

```
>sql
```

```
1.Create database book_hub;
```

```
>Use book_hub;
```

```
2. Table book
```

```
>CREATE TABLE book (
```

- title VARCHAR(255),
- author VARCHAR(255),
- type VARCHAR(255),
- isbn VARCHAR(13),
- cost DECIMAL(10, 2),
- stock INT,
- descp TEXT

```
);
```

```
3.Table info
```

```
>CREATE TABLE info (
```

- un VARCHAR(255),
- psswd VARCHAR(255),
- cont VARCHAR(20),
- ei VARCHAR(255),
- addr VARCHAR(255)

```
);
```



## 4. orders

```
>CREATE TABLE orders (
```

- un VARCHAR(255),
- ei VARCHAR(255),
- title VARCHAR(255),
- cost DECIMAL(10, 2),
- qu INT,
- time VARCHAR(255)

```
);
```

### 6.3 HIGH LEVEL ARCHITECTURE

A data-flow diagram (DFD) is a way of representing a flow of a data of a process or a system (usually an information system) The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops.

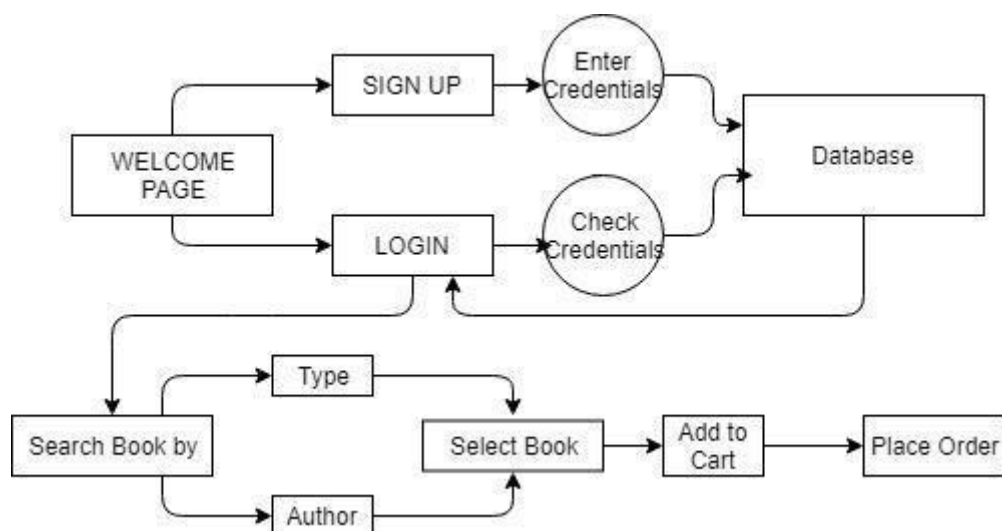


FIGURE 6.31 DATAFLOW DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development.

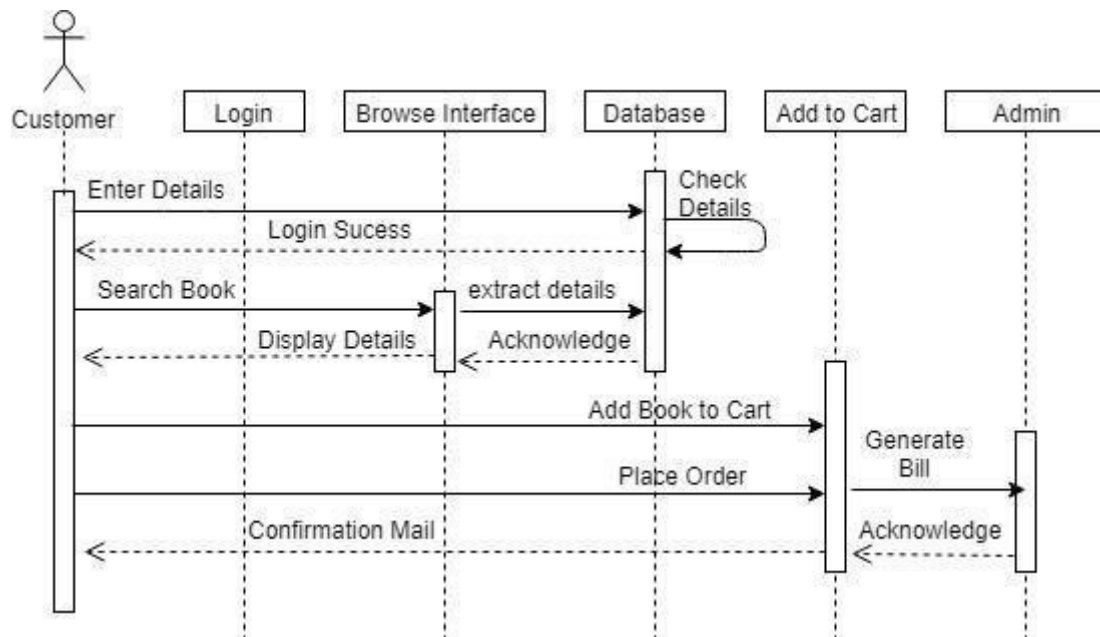


FIGURE 6.32 HIGH LEVEL ARCHITECTURE

## 6.4 CLASS DIAGRAM

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for translating the models into programming code. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

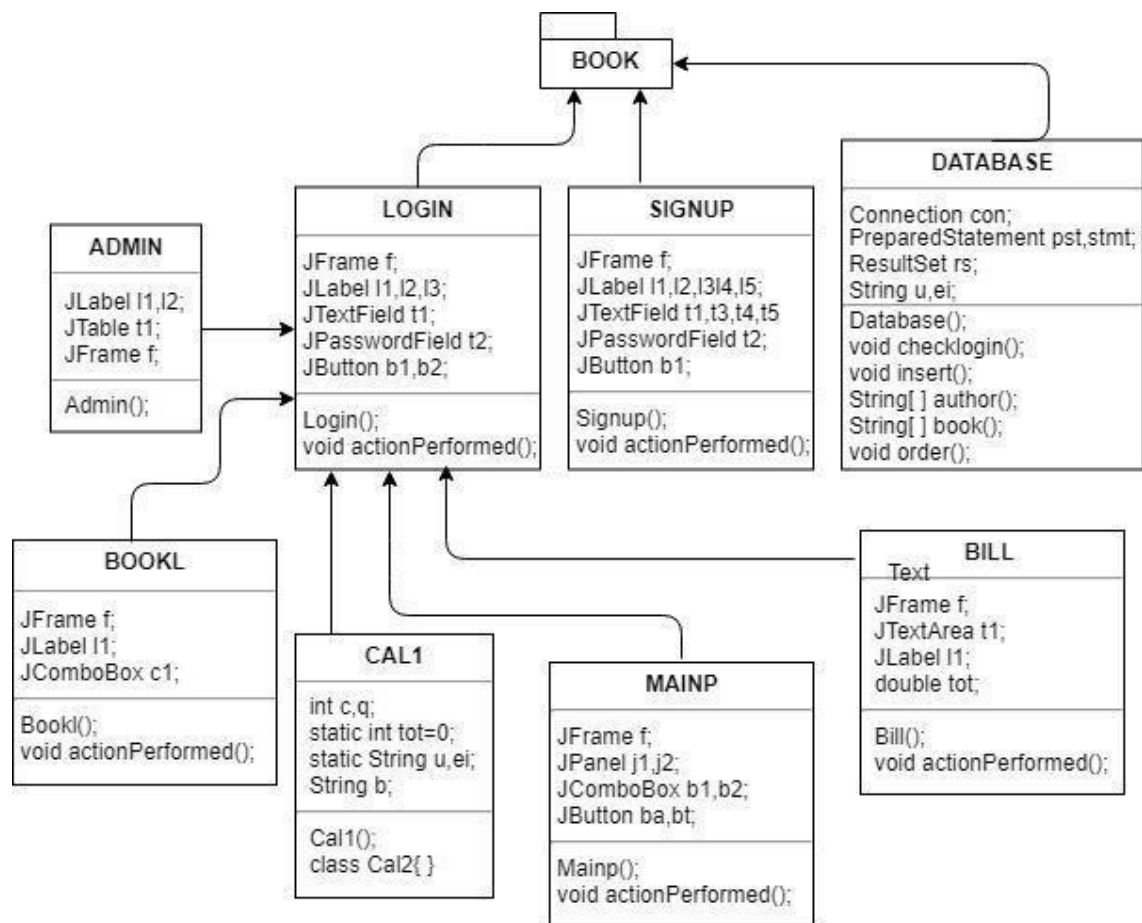


FIGURE 6.4 CLASS DIAGRAM

## CHAPTER 7

# IMPLEMENTATION

### 7.1 CREATING THE APPLICATION

#### Java Swing:

- Developed for bookstores and shops to computerize book purchasing.
- A lightweight GUI toolkit providing a sophisticated set of components.
- Platform-independent, built on Java Abstract Widget Toolkit (AWT).
- Three types of containers: Panel (organizes components), Frame (fully functioning window), Dialog (pop-up window).

#### Java JDBC:

- Java Database Connectivity API for connecting and executing queries with databases.
- Part of Java Standard Edition platform, using JDBC drivers.
- Provides a Java-based data access technology for Java database connectivity.

#### Classes Used:

##### a) **Abc:**

- Displays welcome page, redirects to login frame on 'CONTINUE' button click.

##### b) **Login:**

- User login with validation, shows success or failure message.
- Allows signup for new users.
- Admin login ('admin'/'admin123') opens admin page.

##### c) **Admin:**

- Lists orders placed with details fetched from the MySQL database.

d) **Signup:**

- Enables new user registration, inserts details into the database.

e) **Database:**

- Establishes connection to MySQL database using JDBC.
- Contains functions like **checklogin()**, **insert\_signup()**.

f) **Mainp:**

- Main page after successful login.
- Allows searching books by author or type.

g) **Bookl:**

- Displays a list of books for the selected author or type.

h) **Display:**

- Displays book details (author, type, cost, description).
- Allows adding to the cart or returning.

i) **Cal:**

- Contains an ArrayList for tracking added books to the cart.

j) **Bill:**

- Generates the bill for items in the cart.
- Places order and adds details to the order table in the database.

k) **Final:**

- The last frame/page.
- Sends a confirmation email to the registered email address upon successful order placement.

## 7.2 CONNECTING THE DATABASE TO THE APPLICATION

To connect Java code in Eclipse with MySQL:

### Step 1: Download MySQL Connector

#### 1. Download Connector:

- Download the MySQL Connector/J JAR file from the official MySQL website

### Step 2: Create a Java Project in Eclipse

#### 1. Open Eclipse:

- Launch Eclipse IDE.

### Step 3: Add MySQL Connector/J to the Project

#### 1. Add External JAR:

- Copy the downloaded `mysql-connector-java-x.x.xx.jar` file to a location on your machine.

#### 2. Add JAR to Build Path:

- In Eclipse, right-click on your project in the `Package Explorer`.
- Go to `Build Path -> Configure Build Path...`.
- In the `Libraries` tab, click `Add External JARs...` and select the `mysql-connector-java-x.x.xx.jar` file.

### Step 4: Write Java Code for Database Connection

#### 1. Create a Java Class:

- Right-click on the `src` folder in your project.
- Go to `New -> Class`.
- Enter a class name (e.g., `DatabaseConnection`) and check the option to include the `public static void main(String[] args)` method.

#### 2. Write Code for Database Connection:

- In the `DatabaseConnection` class, write code to establish a connection to your MySQL database.

Step 5: Run the Java Program

1. Run the Program:

- Right-click on the `DatabaseConnection` class.
- Go to `Run As -> Java Application`

## CHAPTER 8

# TESTING

### 8.1 UNIT TESTING

#### 1. User Management Module:

- Test user registration with valid information.
- Verify that registration fails with invalid or missing information.
- Test password encryption and storage.
- Test user login with correct credentials.
- Verify login failure with incorrect username/password.
- Test login session persistence.

#### 2. Book Category Module:

- Check if books are correctly categorized and displayed.
- Ensure that the correct books are shown when a specific category is selected.
- Test the handling of invalid category searches.

#### 3. Book Selection and Cart Module:

- Verify that users can select books to add to their shopping cart.
- Check if selected books are accurately added to the user's cart.
- Test the removal of books from the cart.
- Test the calculation of the total bill when items are added or removed.
- Ensure that the cart state is maintained between different pages or sessions.
- Verify that the cart is empty after completing a purchase.



**4. Order Placement Module:**

- Test the successful placement of an order.
- Verify that a confirmation email is sent to the registered email address.
- Check for errors in the order placement process.

**5. Admin Module:**

- Test successful login for admin users.
- Verify that admin users can view the list of orders along with dates and times.
- Check for security measures to ensure only admins can access this information.

**6. Security Module:**

- Test the system's resistance to common security vulnerabilities (e.g., SQL injection, XSS).
- Verify that passwords are securely stored and encrypted.
- Test session management and authorization mechanisms.

**7. Error Handling Module:**

- Test the system's response to unexpected inputs and errors.
- Ensure appropriate error messages are displayed to users.
- Check how the system recovers from unexpected failures.

**8. User Profile Module:**

- Test the ability to update user profile information.
- Verify that changes to the user profile are reflected correctly.
- Test handling of profile-related errors.

## 8.2 INTEGRATION TESTING

### 1. User Registration and Login Integration:

- Verify that user registration data seamlessly integrates with the user login functionality.
- Test the flow from user registration to successful login, ensuring proper data transfer and validation.

### 2. User Profile and Login Integration:

- Test the integration between user profile management and the login module.
- Verify that user profile data is correctly retrieved and displayed after a successful login.

### 3. Book Category and Book Selection Integration:

- Test the integration between the book category module and the book selection functionality.
- Ensure that selected books belong to the chosen category, and the display is consistent.

### 4. Cart and Order Placement Integration:

- Verify that the shopping cart and order placement modules integrate seamlessly.
- Test the flow from adding items to the cart to successfully placing an order, ensuring accurate billing.

### 5. Admin and Order Management Integration:

- Test the integration between the admin module and order management.
- Verify that the admin can view the list of orders, including relevant details such as date and time.

**6. Security and User Authentication Integration:**

- Verify that security measures integrate well with user authentication.
- Test the system's response to unauthorized access attempts and ensure proper authentication mechanisms.

**7. Book Category and Search Integration:**

- Test the integration between book categories and the search functionality.
- Ensure that searching within a category returns accurate and relevant results.

**8. Error Handling and User Feedback Integration:**

- Test the integration of error handling mechanisms with user feedback.
- Verify that appropriate error messages are displayed to users, and the system gracefully handles unexpected errors.

**9. User Profile and Order History Integration:**

- Test the integration between user profiles and order history.
- Verify that the user's order history is correctly displayed in the user profile.

**10. Book Selection and Cart Persistence Integration:**

- Test the integration between book selection and cart persistence.
- Ensure that the selected books remain in the cart during the user's session and across pages.

## **8.2 SYSTEM TESTING**

System testing is a crucial phase in software development, encompassing various tests to ensure the application's overall quality and functionality. Key aspects include:

1. User Scenarios Testing: Evaluate end-to-end user actions from login to specific interactions.

2. Usability Testing: Assess UI/UX for clarity, consistency, and ease of navigation.
3. Performance Testing: Test application under expected and peak loads, including load and stress.
4. Security Testing: Verify authentication, authorization, and data security measures.
5. Compatibility Testing: Check app across browsers and devices for consistent functionality.
6. Database Integrity Testing: Confirm data accuracy, checking for corruption or discrepancies.
7. Error Handling and Recovery Testing: Test error scenarios and recovery mechanisms.
8. Installation and Configuration Testing: Test installation process for successful deployment on user systems.

## CHAPTER 9

## RESULTS

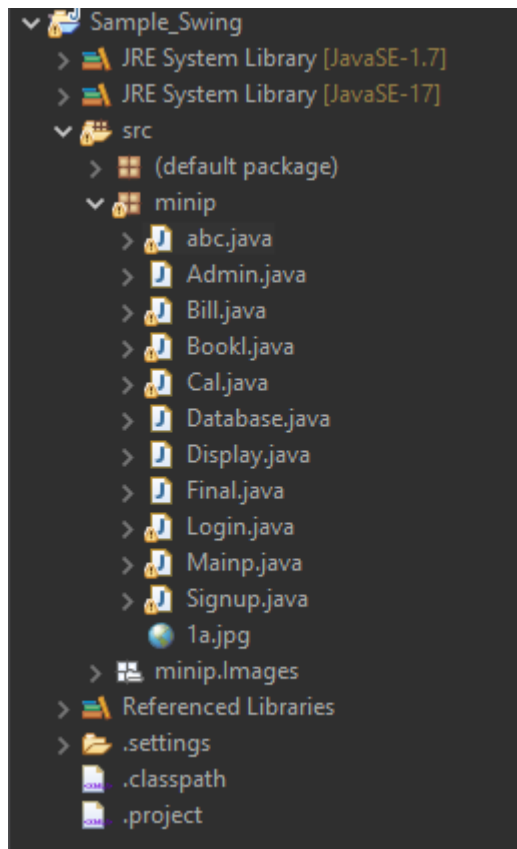
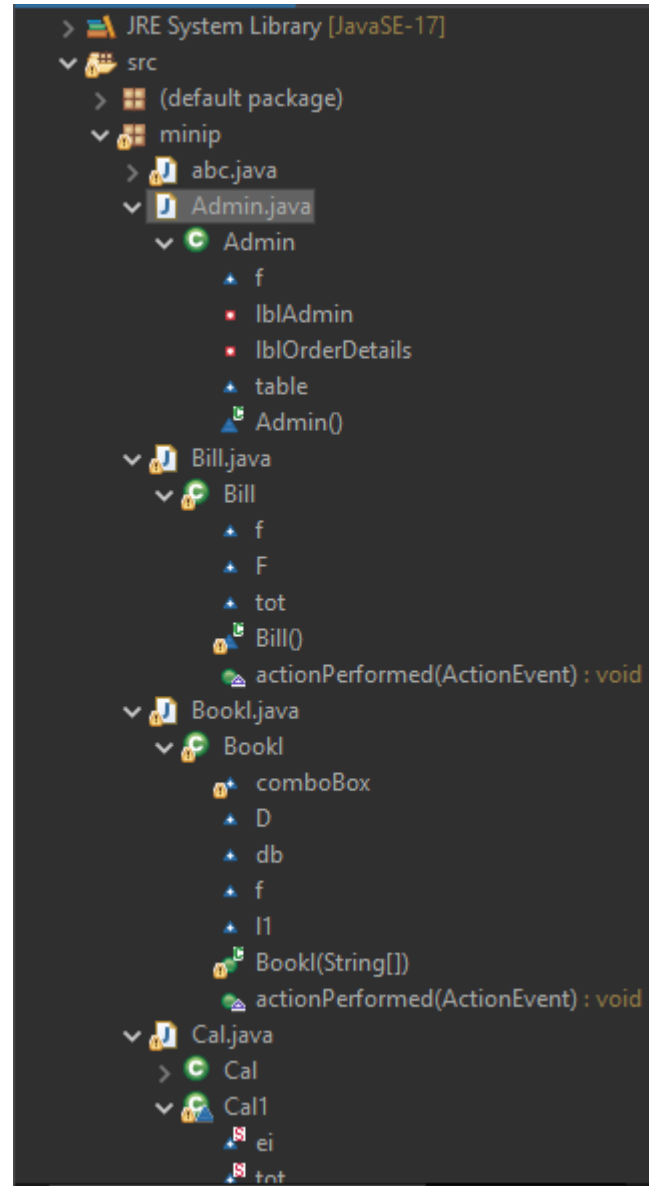


Fig: (code) Classes used



```
abc.java X
1 package minip;
2
3 import java.awt.*;
4
5
6
7
8
9 public class abc implements ActionListener
10 { JFrame f = new JFrame("Welcome");
11     /**
12      * Launch the application.
13      */
14     public static void main(String[] args)
15     {
16         EventQueue.invokeLater(new Runnable()
17         {
18             public void run()
19             {
20                 try { abc a = new abc(); }
21                 catch (Exception e) { e.printStackTrace(); }
22             }
23         });
24     }
25
26     /**
27      * Create the frame.
28      */
29     public abc()
30     {
31         f.getContentPane().setBackground(Color.WHITE);
32         f.setSize(1380,750);
33         f.getContentPane().setLayout(null);
34         f.setVisible(true);
35         f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
36
37         JLabel lblNewLabel = new JLabel("");
38         Image img = new ImageIcon(this.getClass().getResource("images/main.png")).getImage();
```

```
abc.java Admin.java X
1 package minip;
2
3 import java.sql.*;
4
5
6
7
8
9 public class Admin
10 { JFrame f = new JFrame("Admin");
11     JTable table;
12     private JLabel lblAdmin;
13     private JLabel lblOrderDetails;
14
15     Admin()
16     { f.getContentPane().setBackground(new Color(255, 204, 255));
17       f.setSize(1380,750);
18       f.getContentPane().setLayout(null);
19       f.setVisible(true);
20       f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
21
22       DefaultTableModel model = new DefaultTableModel();
23       table = new JTable(model);
24       table.setBounds(100,100,1000,1000);
25       f.getContentPane().add(table);
26
27       JScrollPane scrollPane = new JScrollPane(table);
28       scrollPane.setBounds(192,348,1000,171);
29       f.getContentPane().add(scrollPane);
```



FIGURE 9.1 FIRST PAGE (HOME PAGE)

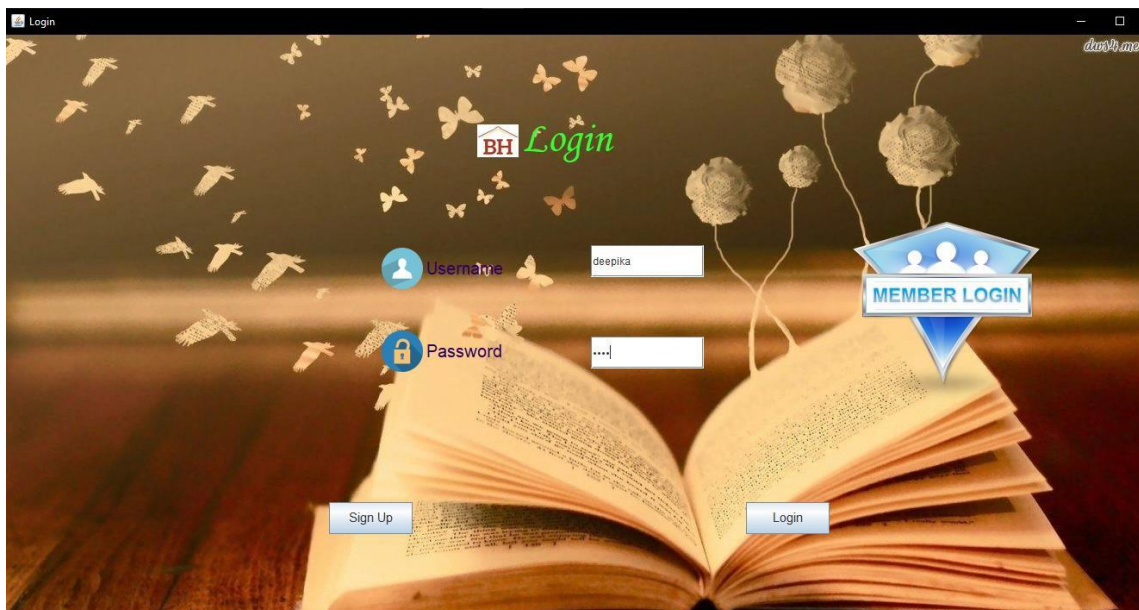
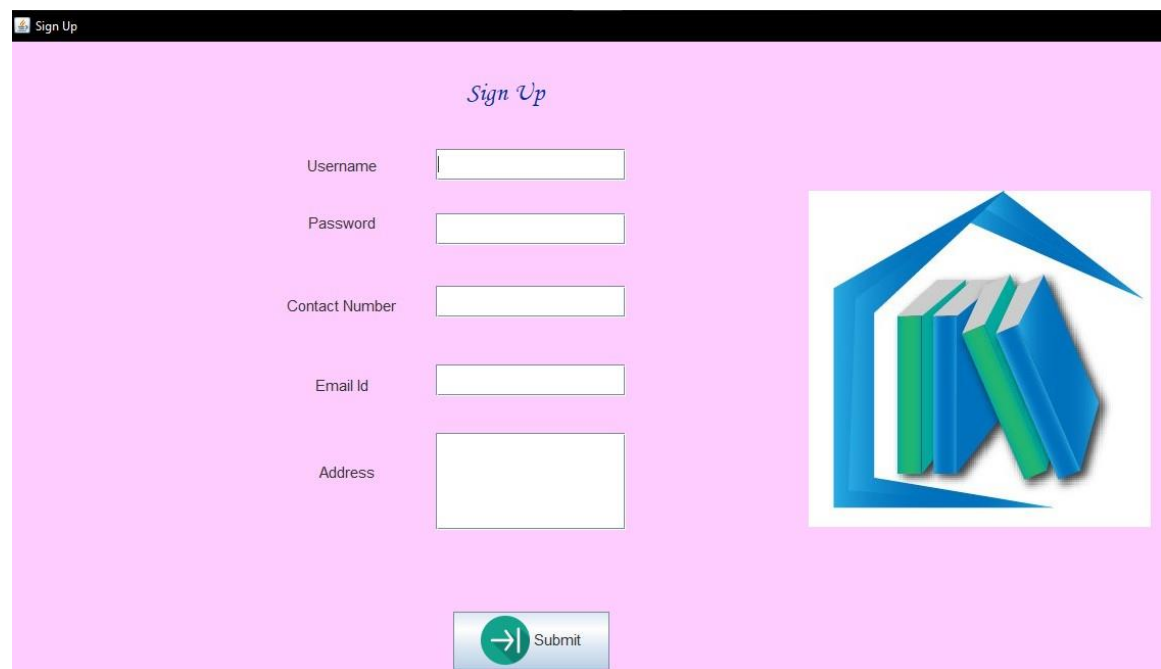


FIGURE 9.2 LOGIN



The image shows a web browser window titled "Sign Up". The page has a light pink background. At the top center, the text "Sign Up" is written in a cursive font. Below this, there are five input fields arranged vertically, each with a label to its left: "Username", "Password", "Contact Number", "Email Id", and "Address". To the right of these fields is a graphic of a blue house outline with several books inside it. At the bottom center, there is a "Submit" button with a green arrow icon.

FIGURE 9.3 SIGNUP PAGE

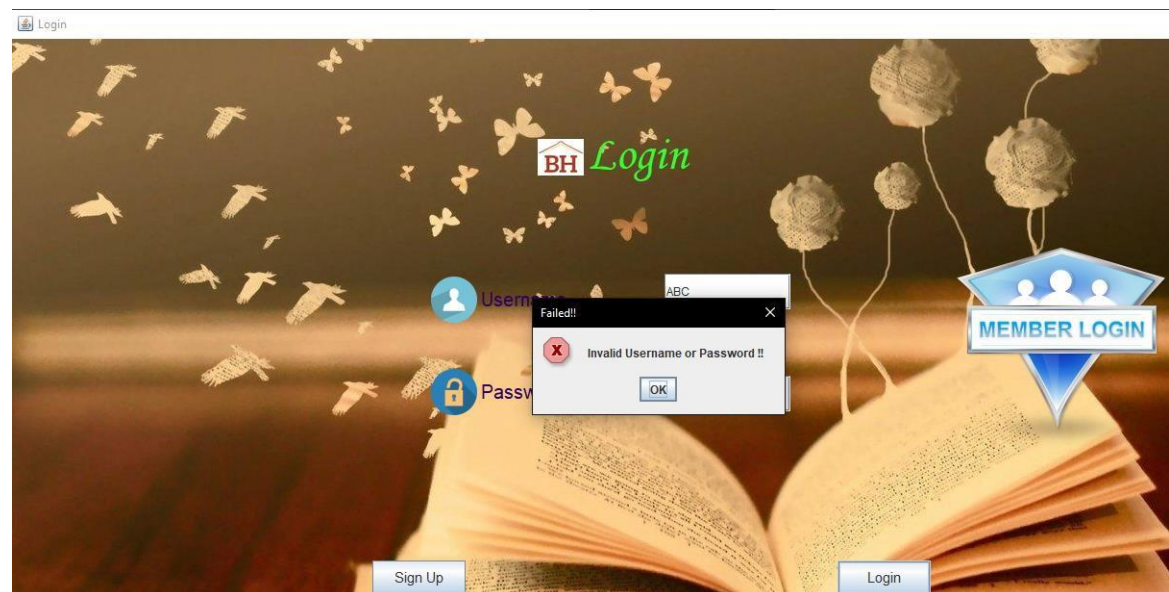


FIGURE 9.4 INCASE OF WRONG LOGIN



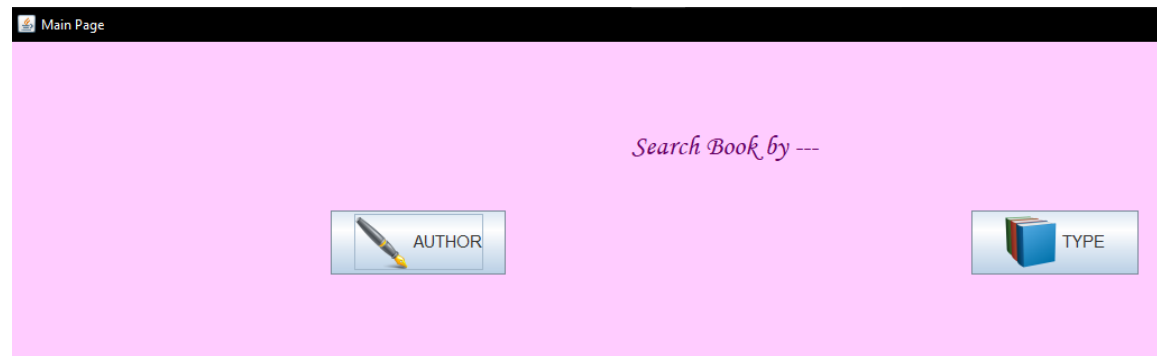


FIGURE 9.5 SEARCH BOOK

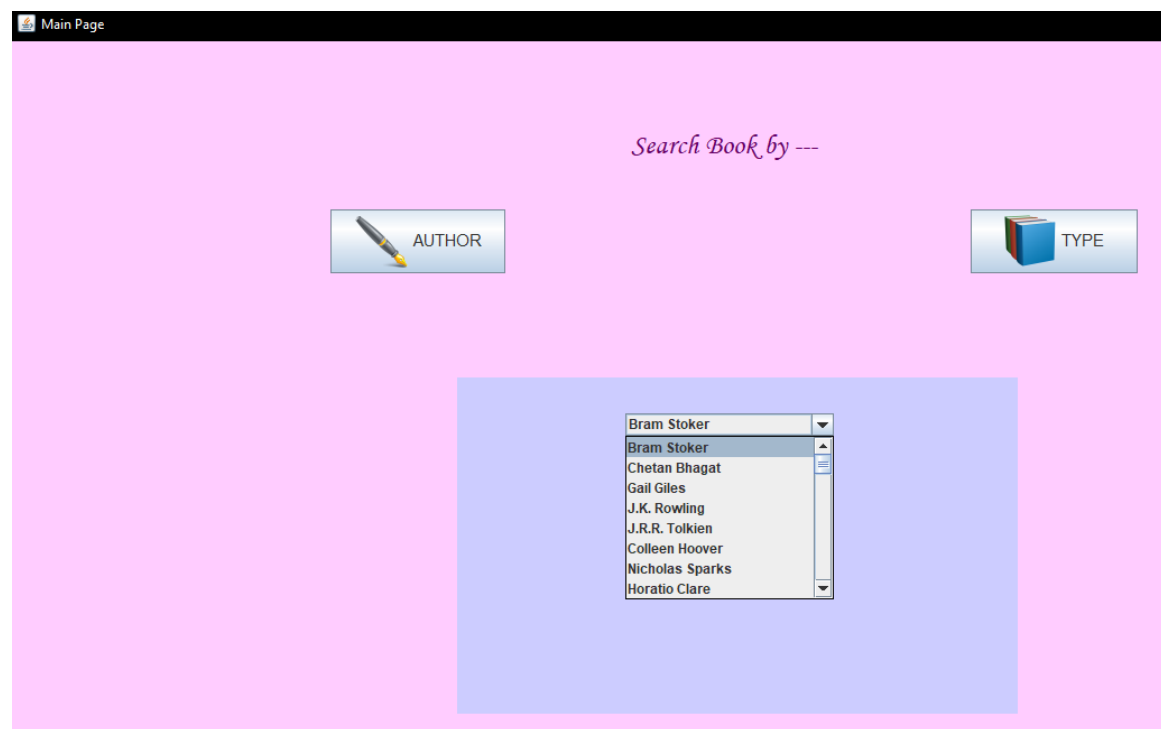


FIGURE 9.6 SEARCH BOOK BY AUTHOR

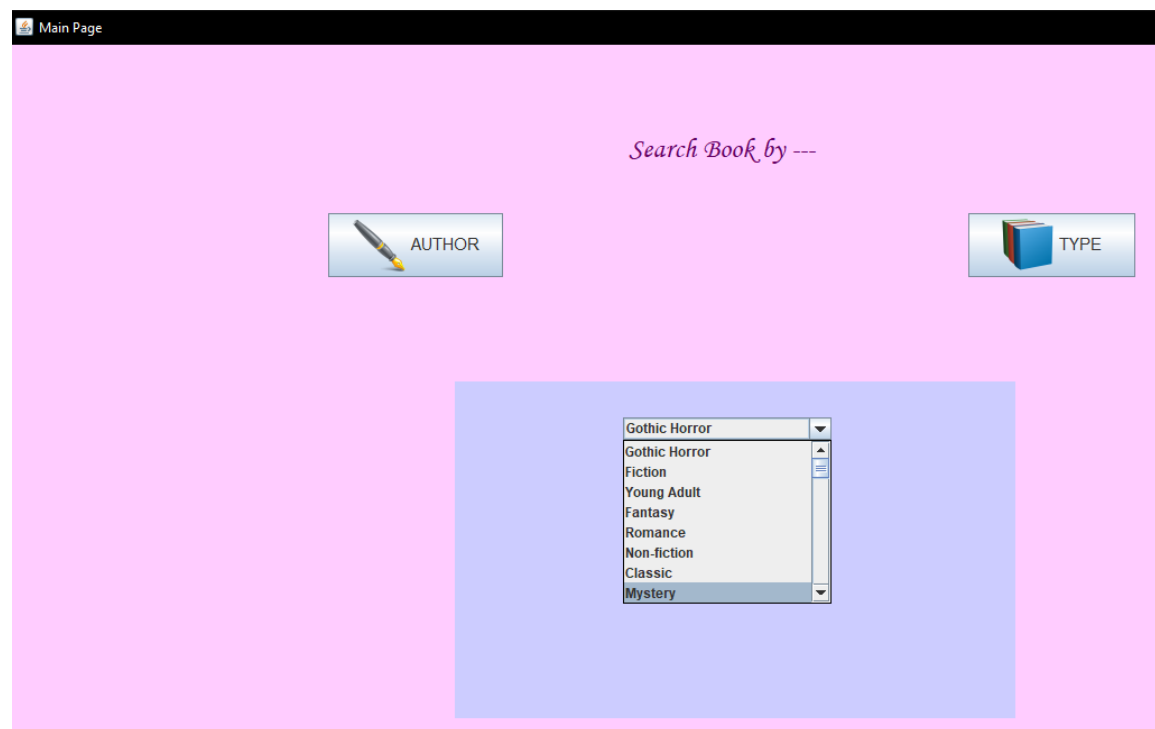


FIGURE 9.7 SEARCH BOOK BY TYPE

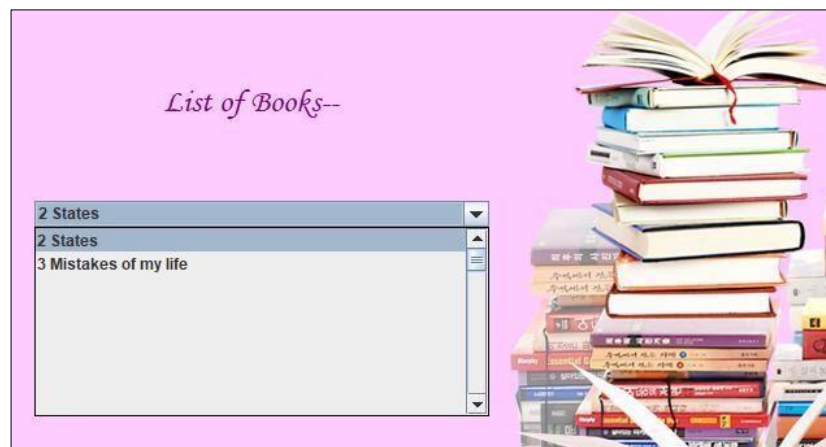


FIGURE 9.8 LIST OF BOOKS

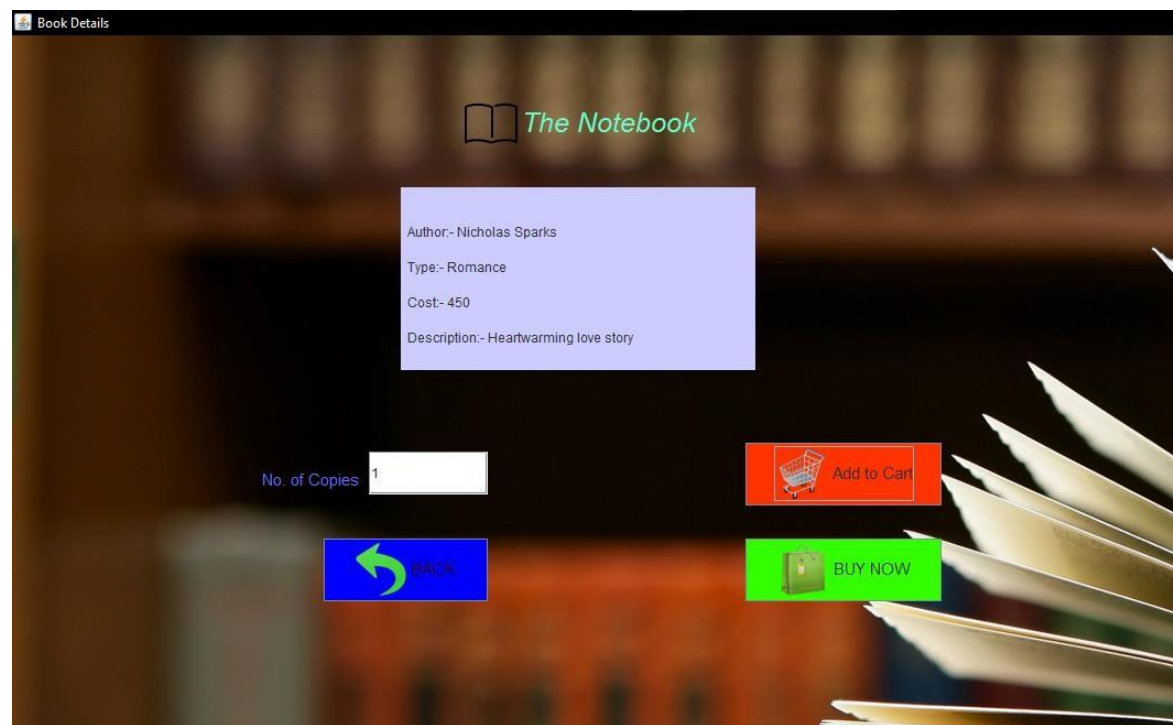


FIGURE 9.9 BOOK DETAILS

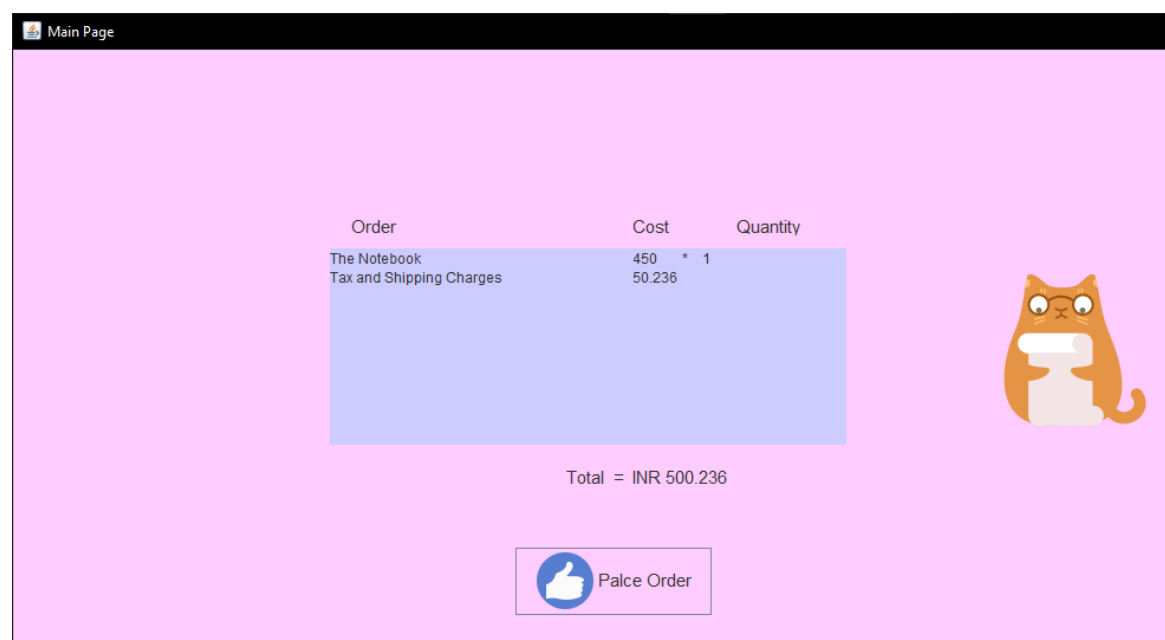


FIGURE 9.10 BILL DETAILS

## 9.1 DATABASE SNAPSHOTS:

```
mysql> use book_hub;
Database changed
mysql> show tables;
+-----+
| Tables_in_book_hub |
+-----+
| book                |
| info                |
| orders              |
+-----+
3 rows in set (0.14 sec)
```

```
mysql> select * from info;
+-----+-----+-----+-----+-----+
| un      | psswd | cont      | ei                                     | addr                |
+-----+-----+-----+-----+-----+
| abc     | 1245  | 123467    | abc@gmail.com                       | asdfgh              |
| chiru   | chi12 | 6874309   | chiranjevispam123@gmail.com         | Bangalore            |
| deepika | 2003  | 91088284  | deepika953singh@gmail.com           | 6th A Cross Bangalore |
| divya   | 1998  | 9845639   | divyasinghn98@gmail.com             | Bangalore            |
| kavya   | 123   | 1234569   | kavyaachuta@gmail.com               | Andhra               |
| skay    | bmsk  | 9141894   | bm.saikrishna2003@gmail.com         | Bangalore            |
+-----+-----+-----+-----+-----+
```

FIGURE 9.11 USER INFORMATION FROM DATABASE

```
mysql> select * from book;
+-----+-----+-----+-----+-----+-----+-----+
| title          | author      | type        | isbn | cost | stock | descp                |
+-----+-----+-----+-----+-----+-----+-----+
| Dracula        | Bram Stoker | Gothic Horror | 111  | 600  | 50    | Classic vampire novel |
| 2 States       | Chetan Bhagat | Fiction      | 222  | 300  | 800   | Romantic comedy about cultural differences |
| My Fault       | Gail Giles   | Young Adult  | 333  | 500  | 700   | Drama about guilt and redemption |
| Harry Potter   | J.K. Rowling | Fantasy      | 444  | 1000 | 120   | Magical adventures at Hogwarts |
| The Hobbit     | J.R.R. Tolkien | Fantasy      | 555  | 450  | 900   | Adventure in Middle-earth |
| It Ends With Us | Colleen Hoover | Romance      | 1010 | 600  | 100   | Emotional contemporary romance |
| The Notebook   | Nicholas Sparks | Romance      | 2020 | 450  | 180   | Heartwarming love story |
| Icebreaker     | Horatio Clare | Non-fiction   | 3030 | 700  | 80    | Journey through the Arctic seas |
| To Kill a Mockingbird | Harper Lee | Classic      | 666  | 550  | 200   | Southern Gothic novel |
| The Great Gatsby | F. Scott Fitzgerald | Classic      | 777  | 700  | 120   | Exploration of the American Dream |
| The Catcher in the Rye | J.D. Salinger | Fiction      | 888  | 400  | 300   | Coming-of-age novel |
| The Da Vinci Code | Dan Brown   | Mystery      | 999  | 800  | 150   | Thriller involving art and cryptography |
+-----+-----+-----+-----+-----+-----+-----+
```

FIGURE 9.12 BOOK INFORMATION FROM DATABASE

```
mysql> select * from orders;
```

un	ei	title	cost	qu	time
deepika	deepika953singh@gmail.com	2 States	15	1	28-01-2024 23:02:16
deepika	deepika953singh@gmail.com	2 States	15	1	28-01-2024 23:37:21
deepika	deepika953singh@gmail.com	2 States	15	1	28-01-2024 23:37:23
deepika	deepika953singh@gmail.com	Harry Potter	30	1	28-01-2024 23:43:36
deepika	deepika953singh@gmail.com	Harry Potter	30	1	28-01-2024 23:43:40
deepika	deepika953singh@gmail.com	The Hobbit	25	1	28-01-2024 23:44:04
deepika	deepika953singh@gmail.com	The Hobbit	25	1	28-01-2024 23:44:05
deepika	deepika953singh@gmail.com	Icebreaker	700	1	29-01-2024 20:24:24
deepika	deepika953singh@gmail.com	It Ends With Us	600	2	29-01-2024 20:24:40
deepika	deepika953singh@gmail.com	It Ends With Us	600	1	29-01-2024 20:31:54
deepika	deepika953singh@gmail.com	Icebreaker	700	1	29-01-2024 21:04:53
deepika	deepika953singh@gmail.com	My Fault	500	1	30-01-2024 23:25:50
deepika	deepika953singh@gmail.com	The Hobbit	450	1	31-01-2024 12:14:56
deepika	deepika953singh@gmail.com	Harry Potter	1000	1	31-01-2024 14:02:20
deepika	deepika953singh@gmail.com	My Fault	500	2	01-02-2024 21:46:39

FIGURE 9.13 ORDER NOTED

## CHAPTER 10

### CONCLUSION

The Book Hub introduces a sophisticated and innovative online bookstore system that sets itself apart through seamless integration with external commercial online bookstores. Developed using Java, JDBC, and MySQL, the system prioritizes user convenience and offers expanded options for customers. The web-based interface allows easy searching for books by author or genre, and the integration with external inventories provides comprehensive information on titles, prices, and availability.

The graphical user interface (GUI) ensures a user-friendly experience, with a Java-based control function managing customer inputs and interacting with the database for efficient information retrieval. The implementation of the SMTP protocol further enhances the system's efficiency by sending confirmation emails to customers upon order placement.

A notable feature of The Book Hub is its software bridge connecting to external online bookstores, broadening the range of available options and providing a richer and more diverse shopping experience. The architecture, leveraging Java Swing for GUI, JDBC for database connectivity, and MySQL for data management, ensures a robust and scalable solution.

Overall, The Book Hub delivers an advanced online bookstore that goes beyond traditional offerings, combining advanced technology with user-friendly features to enhance the overall online book shopping experience for customers.

## REFERENCES

- [1]. [https://www.tutorialspoint.com/java/java\\_swing](https://www.tutorialspoint.com/java/java_swing)
- [2]. <https://www.javatpoint.com/>
- [3]. <https://www.geeksforgeeks.org/java/>
- [4]. <https://www.geeksforgeeks.org/java/stackoverflow.com>
- [5]. Java The Complete Reference - 7th Edition by Herbert Schildt
- [6]. Java A Beginners Guide – 6<sup>th</sup> Edition by Schildt