**1. Introduction**

**1.1. Executive Summary of the project**

Book store management is designed to deal with the customer queries regarding information about different types of books. This is an online System that matches the customer queries regarding the author, publishers, title, price, latest updates, storage information, upgrades etc. against the information stored in the centralized database.

In order to create strong market impact, the issue of any type of books must have enough attractive features so as to attract the professionals. Through outstanding customer service, can gain customer loyalty, increase the number of customer referrals and provide more customers with lucrative preventive maintenance services.

To assist its services staff in retrieving information needed for their work and to meet the heightened service expectations of its customers, the client Web-enabled its internal information systems.so that it could better serve customers, the client also sought to improve the timely exchange of data.

Improved customer satisfaction by reducing the need for customers to fill out forms and provide information about their book to customer service representatives during the check-in process. The representatives can more quickly access the book history and suggest which periodic maintenance services should be performed. They can also determine what parts are in stock for the various publications, which helps customer service representatives keep customers informed about their queries while respecting their time. This improved level of service results in higher customer loyalty.

**2. Profile of Problem Assigned**

The objective and scope of my Project Book Shop Management System is to record the details various activities of user. It will simplifies the task and reduse the paper work. During implementation every user will be given appropriate training to suit their specific needs. Specific support will also be provided at key points within the academic calendar. Training will be provided on a timely basis, and you will be trained as the new is Book Shop Management System rolled out to your area of responsibility.

**3. Objective of the Project**

The objective of this project is to create and implement a software for the Public bookstore. The software will be used primarily by Public students. The software will allow users to create and maintain individual secured accounts, search the Public Bookstore database for textbooks, and make secured online credit card purchases. Users will also be able to contact site administrators. The software makes purchasing textbooks quicker, easier, and more convenient.

* Create different system users and assign different roles with related permissions.
* Manage all the account details such as user name, phone numbers, address.
* Websites, email addresses of the entire customer from one central location.
* Manage all the details regarding features of the books such Company, book dimension, book cost.
* Track all the customers and their contact details.
* Group the contacts together in a single account according to some criteria.
* Capture, View and edit all user transactions.
* Confirmation of end user identity and will verify which users are authorized to receive support.
* Maintain history of each customer and their related information about the book sale, & support related transactions.
* View all the details of all the interactions made with the customer.

**4. System Requirements**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **NAME** | **HARDWARE** |
| 1 | Processer | Platinum Dual Core(upgraded),Intel |
| 2 | Processor Speed | More than 1.2Ghz |
| 3 | RAM | 1GB |
| 4 | Hard Disk Capacity | 40GB |
| 5 | Monitor | SVGA colour monitor |

|  |  |  |
| --- | --- | --- |
| **S. No.** | **NAME** | **SOFTWARE** |
| 1 | Platform | Platform independent |
| 2 | Development Language | JAVA(Front End) |
| 3 | Database Tool | Wamp Server |

**4.2. Function to be provided**

**1. Login:-** This frame provide the security to the software by getting a login ID.

**2. Signup:-**This frame is used to create a login id for you.

**3.** **Book Registration:-** This frame is for the registration the book in the database. This module help to maintain the stokes of the book with the help of storage alert entity. Module contains the isbn, name, publisher, mrp and discount related to the book. It also help to select weather the book is new or old.

**4.** **Distributor:-** This frame for storing the details related to the distributer. This module helps to main the proper record of the distributer. Module contains the name, mobile, email, ifcs code of his account and address related to the distributer.

**5.** **Import:-** This frame used to import in already registered book from the same distributer i.e. it is used import the books in the storage.

**6.** **OutBill:-**This creates the pdf of the bill generated by the shopkeeper for the customer. And that bill automatically stored in the admin hard disk for the future use.

**7. View Books:-** This frame help to view all the registered book in the database of the Book Store.

**8.** **View Distributer:-** This frame help to view all the distributer in the database of the Book Store.

**4.3. Procession Environment H/W and S/W**

Processing is a simple programming environment that was created to make it easier to develop visually oriented applications with an emphasis on animation and providing users with instant feedback through interaction.

**4.3.1. User Interfaces:**

There will be only one user interfaces that will accompany this software: The administrators.

Administrators will be required to login at all times. However, they will have limit access via the web-interface only being able to pull predefined reports. The administrators will have to logon to a host machine inside the Public Bookstore network in order to build reports and ensure backups are running.

**4.3.2. Hardware Interfaces:**

There are no special hardware interface requirements.

**4.4.3. Software Interfaces:**

There are no special software interface requirements.

**4.5.4. Communication Interfaces:**

There are no special communication interfaces requirements.

**5. Feasibility Analysis**

Whatever we think need not be feasible .It is wise to think about the feasibility of any problem we undertake. Feasibility is the study of impact, which happens in the organization by the development of a system. The impact can be either positive or negative. When the positives nominate the negatives, then the system is considered feasible. Here the feasibility study can be performed in two ways such as technical feasibility and Economical Feasibility.

**Technical Feasibility**: We can strongly say that it is technically feasible, since there will not be much difficulty in getting required resources for the development and maintaining the system as well. All the resources needed for the development of the software as well as the maintenance of the same is available in the organization here we are utilizing the resources which are available already.

**Economical Feasibility**:- Development of this application is highly economically feasible .The organization needed not spend much m one for the development of t he system already available. The only thing is to be done is making an environment for the development with an effective supervision. I f we are doing so , we can attain the maximum usability of the corresponding resources .Even after the development , the organization will not be in a condition to invest more in t he organization .There fore , the system is economically feasible.

**6. Project Planning**

**6.1. Team Structure**

Three members are working on this Project.

1. Puneet Goyal(13BCS1193)

2. Priyanka Verma(13BCS1191)

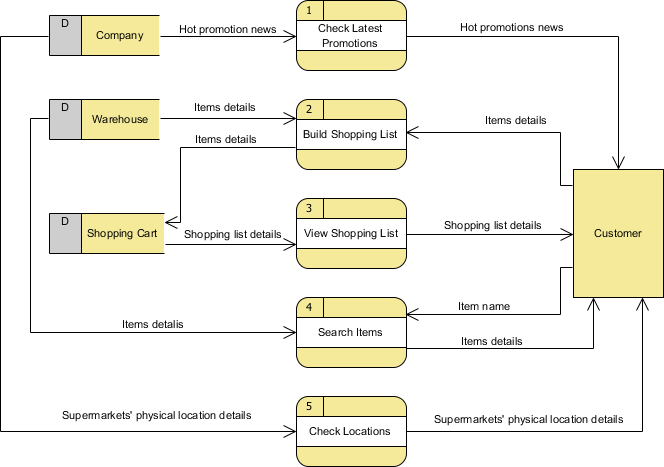
3. Rabal Setia(13BCS1195)

**2.2. Development Schedule**

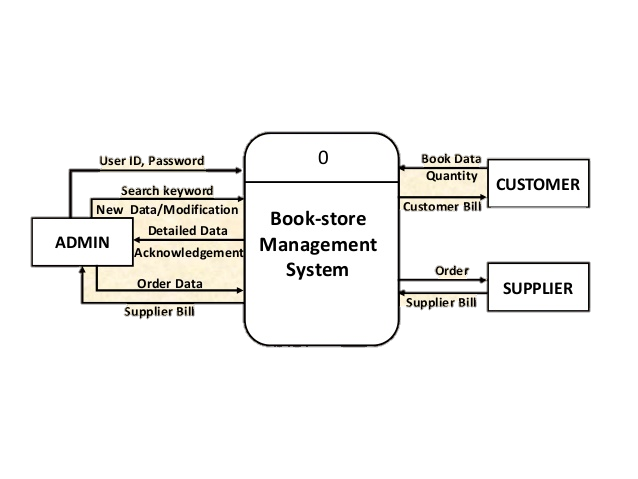
|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.No.** | **Date** | **Day** | **Topic** |
| 1 | 26-8-2015 | Wednesday | Developed Basic idea of Project |
| 2 | 28-8-2015 | Friday | ER Diagram of the Project |
| 3 | 28-8-2015 | Friday | SRS of the Project |
| 4 | 10-9-2015 | Thursday | DFD Related to the Project |
| 5 | 10-9-2015 | Thursday | Start Developing the frames |
| 6 | 17-10-2015 | Thursday | Complete the development of frame |
| 7 | 20-10-2015 | Friday | Start working on the logic |
| 8 | 1-11-2015 | Sunday | Almost complete the Project  (only finishing left) |
| 9 | 10-11-2015 | Tuesday | Project Completed |

**7. System Requirements Specification**

**7.1. External Interface & Flow Diagram**



**7.2. Data Flow Diagrams (Type 0)**



DFD Type 0

**7.3. Data Dictionary**

There are Three tables used in our software which are being created. One for Admin containing admin record one for Distributer and other for Login. containing all record of the book

1. Book Reg.
2. Distributer.
3. Login

Book Reg. Table Descriptions:-

|  |  |  |
| --- | --- | --- |
| **Create Shop** | **Name of Columns** | **Data Types** |
| **Store\_name** | varchar(30) |
| **isbn\_no** | varchar(30) |
| **book\_name** | varchar(30) |
| **auther** | varchar(30) |
| **publisher** | varchar(30) |
| **mrp** | int(30) |
| **discount** | double(30,4) |
| **selling** | double(30,4) |
| **qunatity** | int(30) |
| **version** | varchar(30) |
| **alert** | int(30) |

Distributer Table Decryptions:-

|  |  |  |
| --- | --- | --- |
| **Distributer** | **Name of Columns** | **Data Types** |
| **name** | varchar(30) |
| **mobile** | varchar(10) |
| **email** | varchar(30) |
| **address** | varchar(30) |
| **account** | int(30) |
| **ifcs** | int(30) |
| **bname** | varchar(30) |

Login Table Decryptions:-

|  |  |  |
| --- | --- | --- |
| **Login** | **Name of Columns** | **Data Types** |
| **Name** | varchar(40) |
| **Userid** | varchar(40) |
| **email** | varchar(40) |
| **password** | varchar(40) |

**7.4. Software System Attributes**

**7.4.1. Reliability**

The average time to failure shall be 30 days. In the event that a server does crash, a backup server will be up and running within the hour.

**7.4.2. Availability**

The Public Bookstore shall be available to users 24 hours a day, 7 days a week, with the exception of being down for maintenance no more than one hour a week. If the system crashes, it should be back up within one hour.

**7.4.3. Security**

Users will be able to access only their own personal information and not that of other users. Purchases will be handled through a secure server to ensure the protection of user’s credit card and personal information.

**7.4.4. Maintainability**

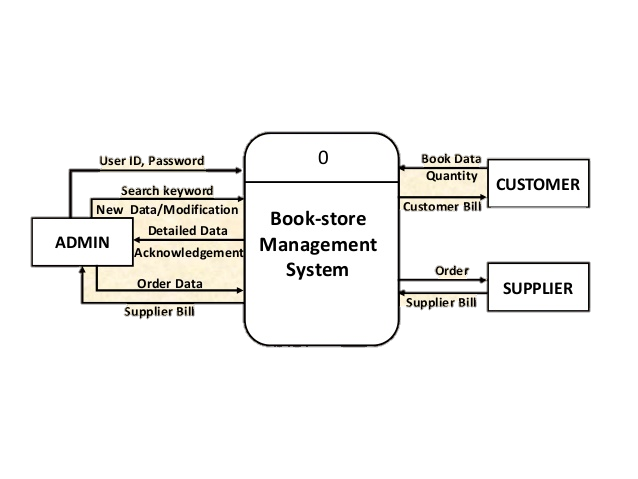
Any updates or defect fixes shall be able to be made on server-side computers only without any patches required by the user.

**7.4.5. Portability**

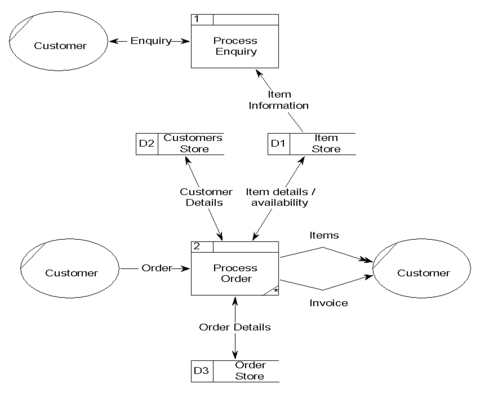
Nothing required

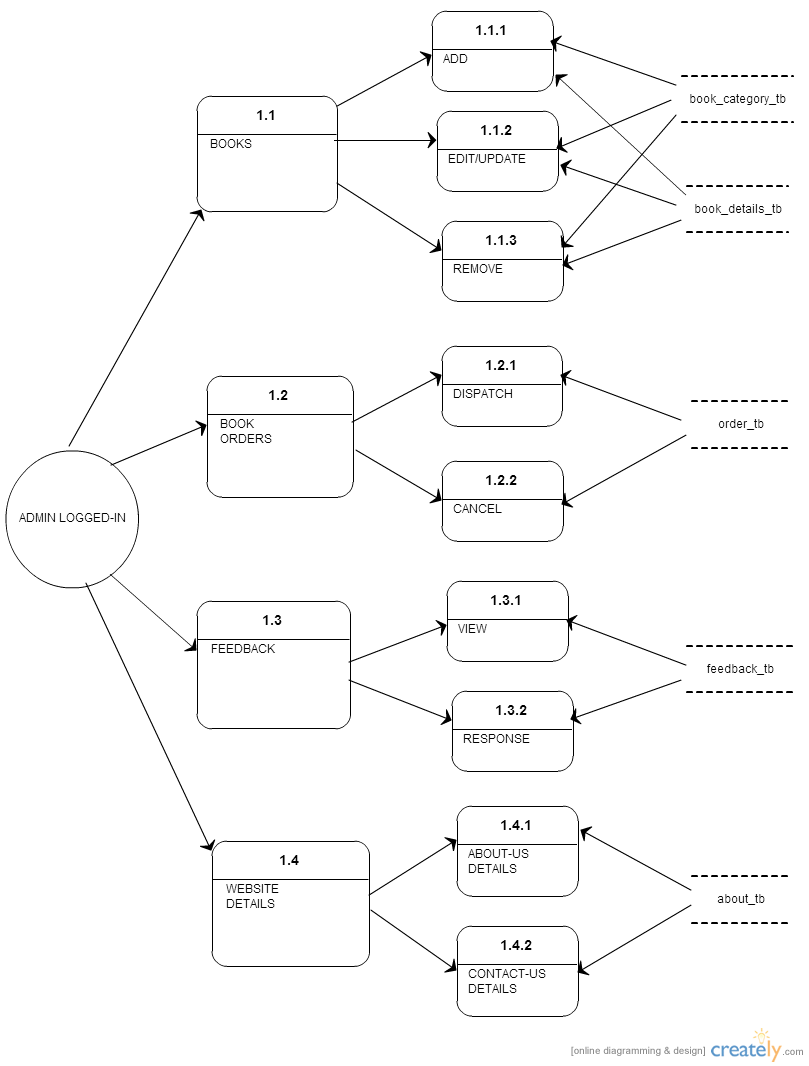
**8. Design**

**8.1. Detail Data Flow Diagrams (Type 0 to Type 2)**



DFD Type 0

  
DFD Type 1



DFD Type 2

**8.2. Pseudo Code**

Pseudocode is an informal [high-level](https://en.wikipedia.org/wiki/High-level_programming_language) description of the operating principle of a [computer program](https://en.wikipedia.org/wiki/Computer_program) or other [algorithm](https://en.wikipedia.org/wiki/Algorithm).

It uses the structural conventions of a [programming language](https://en.wikipedia.org/wiki/Programming_language), but is intended for human reading rather than machine reading. Pseudocode typically omits details that are essential for machine understanding of the algorithm, such as [variable declarations](https://en.wikipedia.org/wiki/Variable_declaration), system-specific code and some [subroutines](https://en.wikipedia.org/wiki/Subroutines). The programming language is augmented with [natural language](https://en.wikipedia.org/wiki/Natural_language) description details, where convenient, or with compact mathematical notation. The purpose of using pseudocode is that it is easier for people to understand than conventional programming language code, and that it is an efficient and environment-independent description of the key principles of an algorithm. It is commonly used in textbooks and scientific publications that are documenting various algorithms, and also in planning of computer program development, for sketching out the structure of the program before the actual coding takes place.

No standard for pseudocode syntax exists, as a program in pseudocode is not an executable program. Pseudocode resembles, but should not be confused with [skeleton programs](https://en.wikipedia.org/wiki/Skeleton_(computer_programming)) which can be [compiled](https://en.wikipedia.org/wiki/Compiler) without errors

Pseudocode of our program is:-

Algorithm:-

1. Start  
   2. Login   
   3. Register the Book

4. Entre the detail about the distributer  
   5. Check the status of the book in the View Book  
   6. Check the status of the Distributer in the View Distributer.  
   7. Import the book.  
   8. Generate the Out Bill for the customer.

9. PDF of generated bill is display.

10. End.

**9. Test Plans**

**9.1. Functional Performance Stress Test**

**Performance testing** is an empirical technical investigation conducted to provide stakeholders with information about the quality of the product or service under test with regard to speed, scalability and/or stability characteristics. It is also the superset of other classes of performance-related testing such as load and stress testing.

A **load test** is a performance test focused on determining or validating performance characteristics of the product under test when subjected to workload models and load volumes anticipated during production operations.

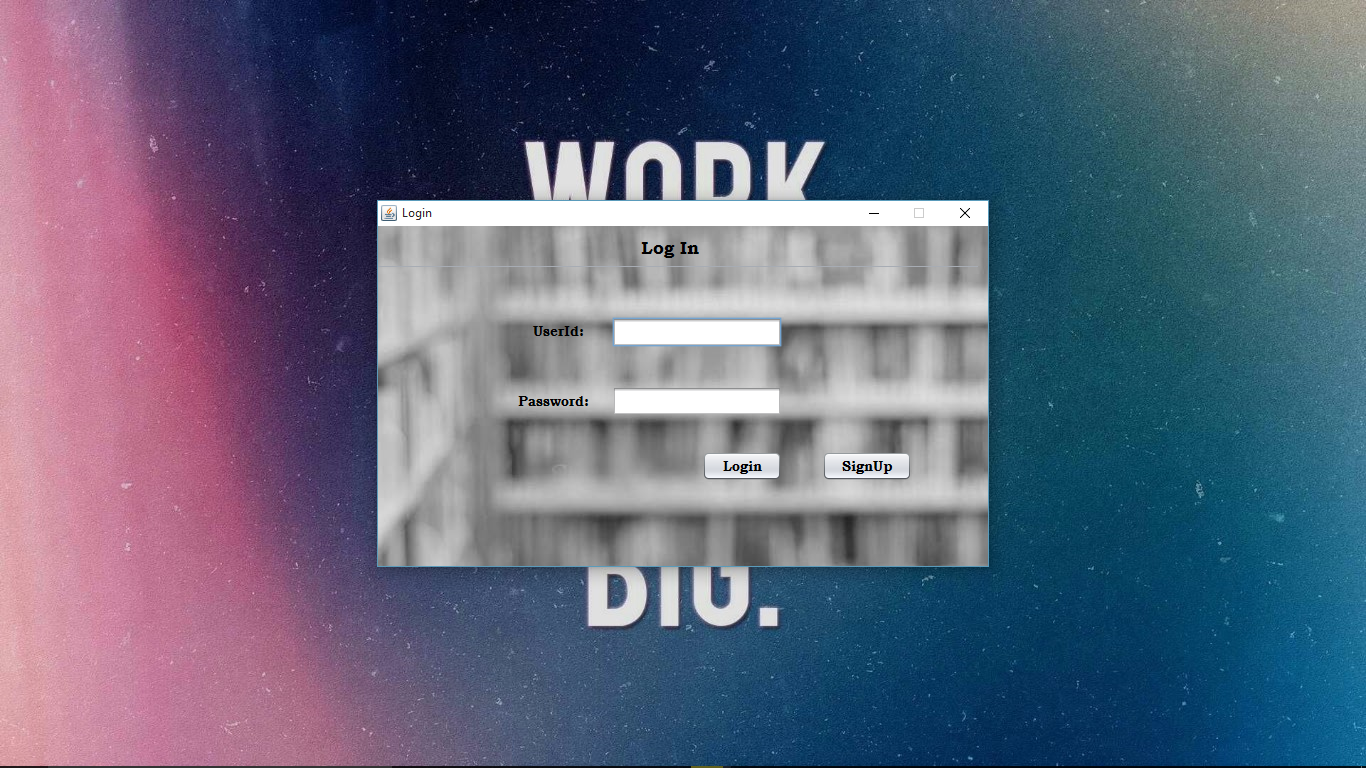
A **stress test** is a performance test focused on determining or validating performance characteristics of the product under test when subjected to workload models and load volumes beyond those anticipated during production operations. Stress tests may also include tests focused on determining or validating performance characteristics of the product under test when subjected to workload models and load volumes *while* the product is subjected to other stressful conditions, such as limited memory, insufficient disk space or server failure.

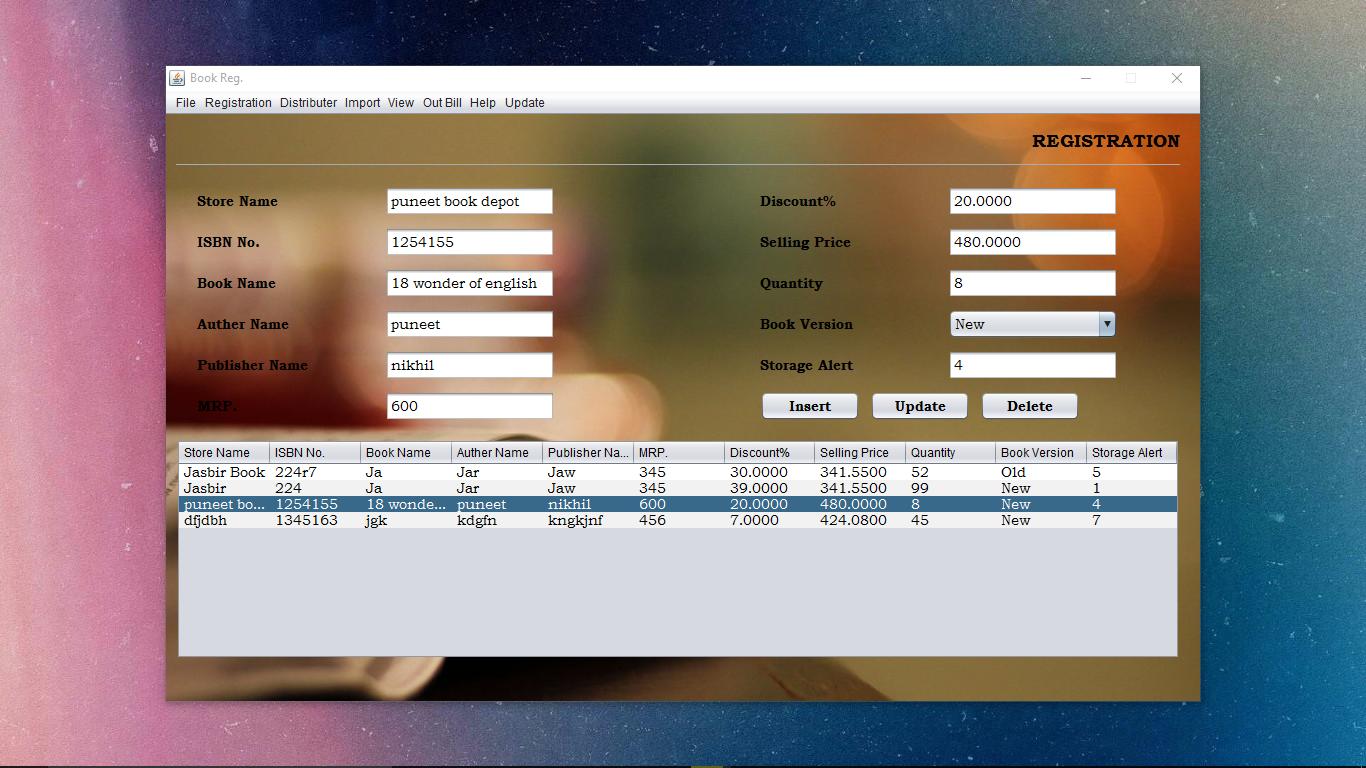
To pinpoint them we use a variety of tools:

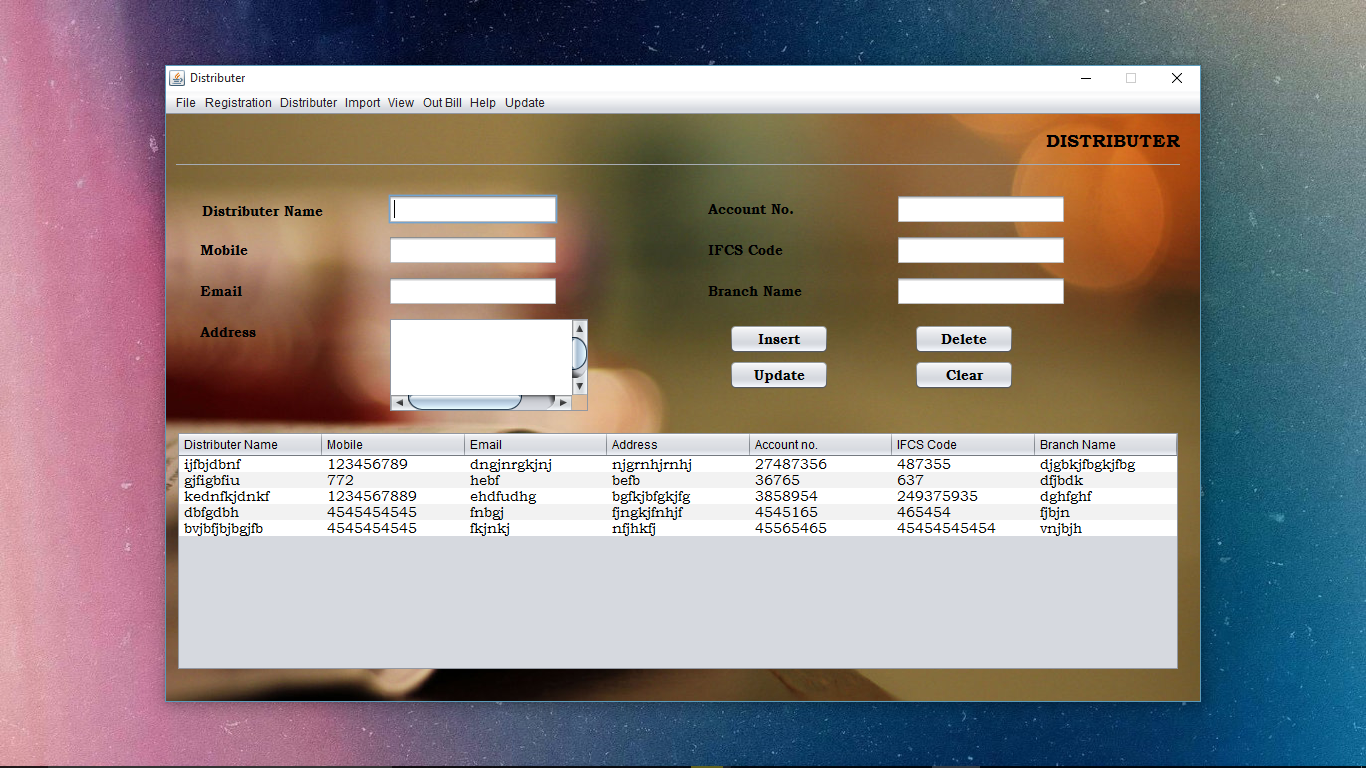
* at the **application level**, developers can use profilers to spot inefficiencies in their code (for example poor search algorithms)
* at the **database level**, developers and DBAs can use database-specific profilers and query optimizers
* at the **operating system level**, system engineers can use utilities such as top, PerfMon (on Windows) to monitor hardware resources such as CPU.

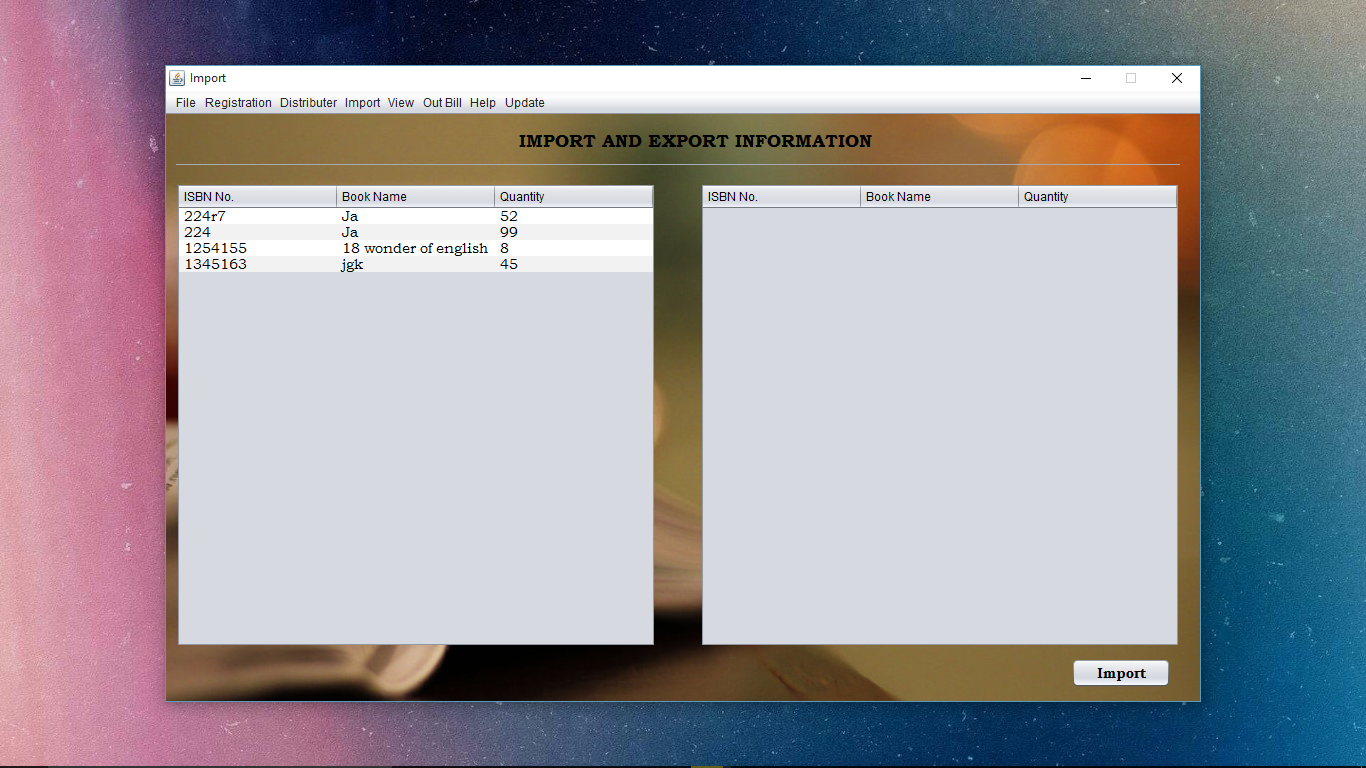
All the above given test are Performed by our team members with a positive result. Our product has successfully completed all tests (**Performance, load, stress**) with high accuracy.

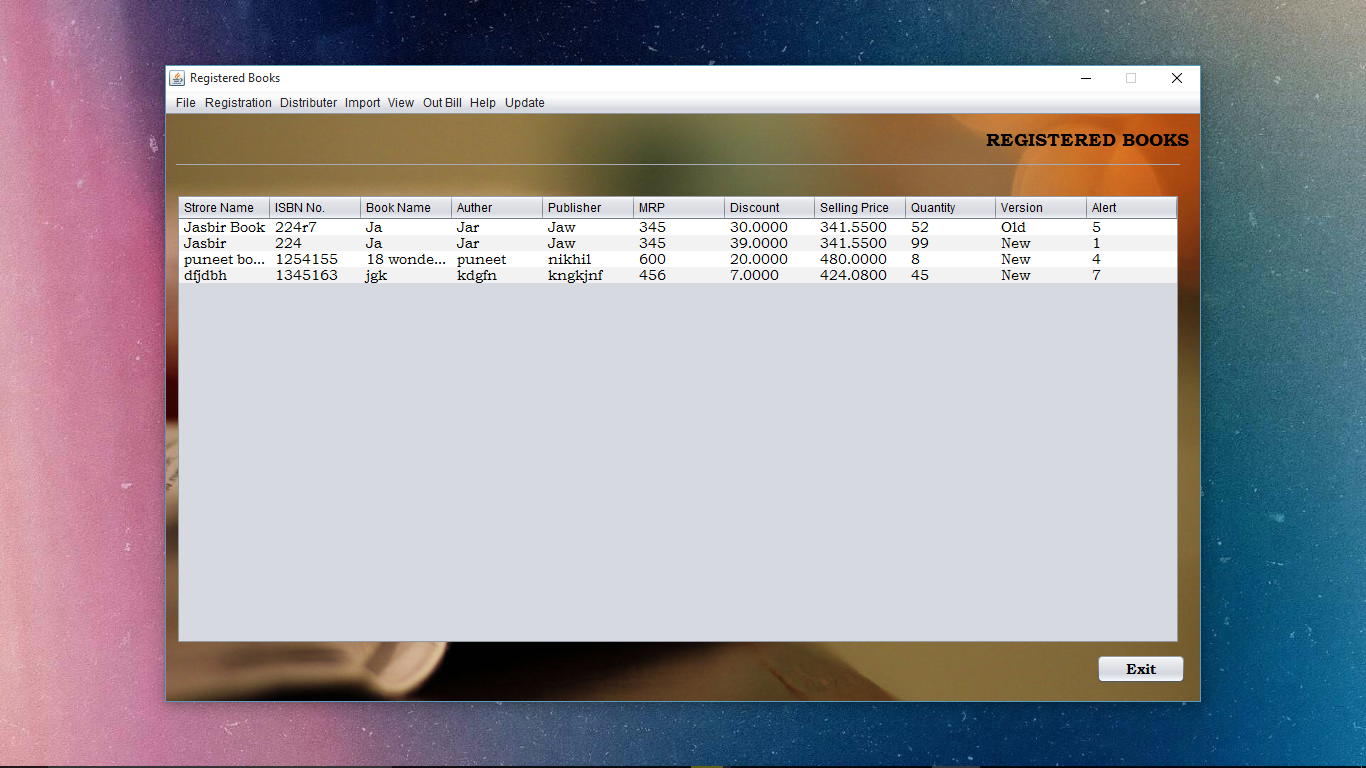
**10. Screen short of Pages of the Project**

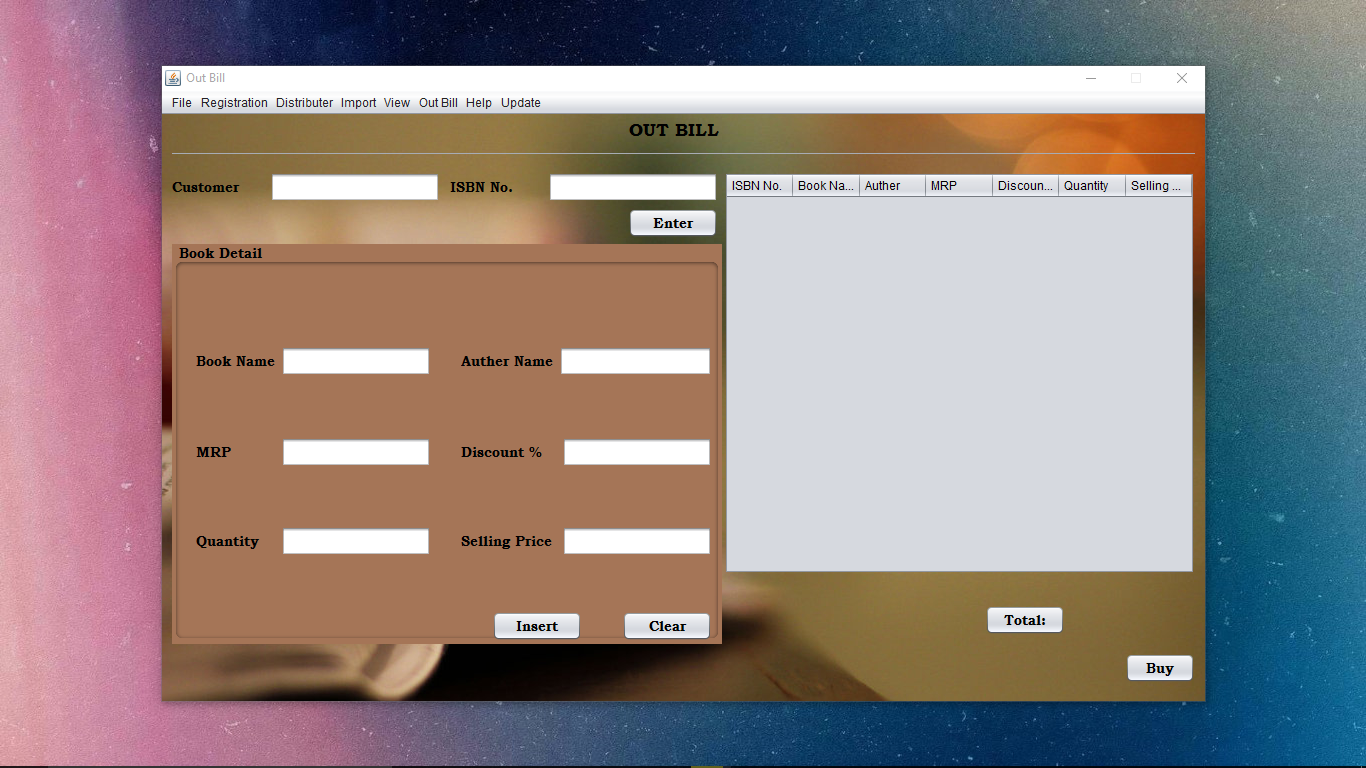












**11. Project Legacy**

**11.1. Current Status of Project**

Completed

**11.2. Remaining Area of Concerns**

None

**11.3. Technical and Managerial Lessons Learnt**

Projects vary in terms of purpose, cost, magnitude and the timelines involved.

Yet, they all have common features and the lessons learned from one project can easily be incorporated in another, circumstances permitting.

Some of the experience thus gleaned is revealed below. This is by no means an extensive list of all the project lessons learned, but a few of the most relevant, are stated herewith:

**Lessons Learned**

* The success of a project is largely dependent on the skills and strengths of the people involved. Therefore, a project needs to have a dedicated, talented set of individuals working towards a common goal.
* Together with leadership skills, the project manager needs to be aware of the strengths and weaknesses of his/her staff, so that the talents are harnessed and the shortfalls downplayed for the benefit of the project.
* A champion team and a team of champions are indeed different. The former would lead to a successful project whilst the latter would yield to a conflict of egos, each chasing an individual goal.
* It pays to know who the decision makers are. Such individuals may not always be readily visible, but they will be calling the shots, so developing a strong line of communication with such individuals will reap benefits in the long run.
* If you have the knowledge and experience to make a decision, then you should go ahead and so, without expecting top managers to spoon feed you at every turn.
* Procrastination does not work. After assimilating the relevant information, decisions need to be made. Wrong decisions can be salvaged, if discovered early; but right decisions cannot be postponed. So, Carpe Diem, (seize the day), as advocated by the popular maxim.
* When things go wrong, as they invariably will; excuses will not work. Find an alternative course of action or remedial propositions instead. Allocating blame only causes dissention and hostility, searching for solutions will bring the team together.
* Be pro-active in your approach. Reactivity is just not good enough.
* Be open to change. Sometimes, you may find that the things you knew along may not be correct at this given time, under these specific conditions.
* Know what resources are available. Not just those under your purview but those which are at the discretion of other teams. Sometimes, others may be happy to help. After all, the *favor bank* concept which is colloquially referred to as the 'you scratch my back and I will scratch yours' philosophy, is apparent in the business world too.
* Paperwork and documentation are necessary for reporting purposes. But when making decisions, placing too much reliance on data which may have changed within a surprisingly short timeframe pays few dividends, especially in an unpredictable environment.
* Know your customer and know the objectives of the project at hand. If any significant changes need to be made, do so, but remember you need to consult the customer first.
* Respect your leader and his/her decisions. Sometimes, you may not agree with these. That is fine. Voice your objections, especially if they are reasonable. But once an action has been decided upon, even if it is contrary to your idea of what should have been done, support it, and try to make it a success.
* Take account of all the known facts. Try to make sense of it, but don't blindly force-fit scenarios into a pre-established mould. Such scenarios may have been right before, and will, in all likelihood, be right once again, but maybe just not in this case.
* Do not be afraid of taking calculated risks. After all, as the adage goes,*a ship is safe in the harbor, but that is not what ships were built for*.
* When things go wrong, know who you can turn to for help.
* Always disclose information to those, who will need it. This is not the time or place for obtaining an edge over another by keeping crucial data close to your chest. People, who know what is expected of them and have the means of doing so, will play a pivotal role in making the project a success.
* Use modern technology and time tested management skills to your advantage.
* Good communication is that which will stop mistakes from becoming failures. Mistakes happen and recovery is always possible. But failure is a dead-end street.
* Do not blindly rush into decisions. Careful thought needs to be given to the circumstances at hand prior to engaging in decision making. This will save time in the long run by minimizing the need to redo work.

**Conclusion**

Repetitive mistakes are the best avoided. Project lessons learned should be documented so that future team leaders can make use of the learning experience of others in order to avoid the same pitfalls themselves.

**12. Bibliography**

1. Java 2 -The Complete Reference 5Th Ed - Herbert Schildt - 2002

2. www.javatpoint.com

3. www,javatutorial,com