**Index**

|  |  |  |
| --- | --- | --- |
| **Chapter Nos.** | **Chapter Name** | **Page Nos.** |
| 01. | Introduction   * Introduction * Existing System * Problem in Existing System * Needs for Computerizations |  |
| 02. | Proposed System   * Proposed System * Hardware and Software Specifications * Feasibility Study * Fact Finding Techniques |  |
| 03. | Analysis   * Nomenclature( ERD, DFD symbols) * Entity Relationship Diagram * Data Flow Diagram |  |
| 04. | System Design   * Data Element Dictionary * Table Design * Program Specification * Menu Design * Input Screen Design |  |
| 05. | Testing Procedures |  |
| 06. | Output Screen |  |
| 07. | Implementation Procedure |  |
| 08. | User Manual |  |
| 09. | Scope for Future Enhancement |  |
| 10. | Conclusions |  |

**Introduction**

**To**

**System**

**Introduction**

The “**Library Management System”** project was designed for Yashwantrao Chavan Science College, Karad, to provide a Desktop based Application developed in JAVA to automate the day to day activities of college library. The main aim of the project is to provide an effective working platform to computerize the whole college library.

This project has been designed in such a way that it suits to all other colleges library. It assists in maintaining Master Entries such as Library Details, Books Category Information, Publisher Information, Books Information, Member Information, as well as it handles issue and returns of books. In addition of that, it maintains the track of books purchased from suppliers and their payments.

It will help to adding, modifying and viewing records. Cost effective and readily available system that provides dynamic and wide Varity of business logic is needed.

It will provide various management information reports. The system mainly proves beneficial to manager as it is analysis friendly. In other words, user can generate various reports and analyze the figure related with Member List, Books List, Issued Books, Available Books, Supplier List, Supplier Payment Register etc.

The main aim of developing this Library Management System based application is to manage the College Library. The main advantage is to provide better solution to the Library who wants to main information quickly and reduces the man power and cost. Here users interact with Front End designed using JAVA and all information can be stored at the back end MySQL Server, This Project was done in three months under the guidance of our college professors.

**Existing**

**System**

**Existing system**

**The Existing System is –**

* The existing system is manual system. Needs to be converted into automated system.
* In the existing system all the documents are not contain any paper work which is handled there is no chance of getting documents corrupted.
* There are many drawbacks of the existing system such as requirement of large number of human resource and papers along with higher chance of losing data and tearing books.
* Less Security.
* It is not reliable.
* Existing system is a time consuming.
* Accuracy not guaranteed.

**Drawbacks –**

* As the system totally manually it is difficult to maintain the track of books availability, issued and returned books, calculate fines, maintains supplier order and its receipts as well as their outstanding etc. that’s why it was consume his lot of time.
* A computerized management system is required.
* All records store in pages while spend out of time and human power.
* It is difficult to get a quick reference in registers
* No Online Transaction Process
* Business working poorly

**The Solution of these Problem -**

The development of the new system contains the following activities, which try to automate the entire process keeping in view of the database integration approach.

* User friendliness is provided in the application with various controls.
* The system makes the overall project management much easier and flexible.
* There is no risk of data mismanagement at any level while the project development is under process.
* It provides high level of security with different level of authentication.
* Users from any part of the world can make use of the system.
* New system will process accurate results.
* New system will be much better in performance as compared to existing one.

**Proposed**

**System**

**Proposed System**

The front-end development tool was JAVA which allows visualization to build the Library Management System. The back-end code was done with fully object-oriented. The JAVA is easy-to-use and efficient.

The back-end database development tool used was MySQL Server. It is able to handle large amounts of data while maintaining data integrity and provides a number of senior management and data distribution functions. These two development tools are powerful, and a good interface for development.

**Features of The System –**

The Features & easy to manage for this project are:**-**

* The System is user friendly.
* Cost effective, Secured Data.
* The coding language is JAVA which is easy to understand which follows the C/C++ language syntax.
* The forms are designed in MDI form Computerized System.

**Objectives of The System *–***

The Objectives for this project are:**-**

* Master Entries such as Library Details, Books Category Information, Publisher Information, Books Information, Member Information etc.
* It handles issue and returns of books.
* It maintains the track of books purchased from suppliers and their payments.
* Keeps the track of books stock.
* Provide all reports which contain Member List, Books List, Issued Books, Available Books, Supplier List, Supplier Payment Register etc.

**Scope of The System –**

* The provided system cannot be used to calculate profit loss report as there are many other things have to be considered in calculating the profit loss report.
* The target user for this system is the only Admin of the Library Management System.
* The system helps to overcome data redundancy and it also saves time.
* Limited Customer reach

**System**

**Analysis**

**Fact Finding Techniques**

A key part of feasibility is gathering information about the present system. The analyst knows what information to gather to make of it.

**Questionnaires:-**

It allows analyst to collect information from a group of individuals who may or may not be using the system thus resulting some times in irrelevant data & data redundancy.

**Interviews:-**

Analysts use interview to collect information from individuals who they consider should be the sources, who are current users of the existing system. The analyst should have a face conversation with the users & administrator of the system & fixed set of questions is prepared.

**Record review *:-***

Consisting of analyzing the previous operations in the company & forecasting the new future schemes. Record include table name, date& time creation, user login etc.

**Observation:-**

If information is not collected from the other fact-finding method, then observation method is used. In this method analyst observes flow of documents, way the process is carried out steps followed person involved etc.

**Feasibility Study**

The main objective of feasibility study is to test the technical, social and economic of developing a system. This is done by investigation the existing system in the area under investigation and generation idea about the new system. Prior to the system developed a thorough study of the system is carried out involves…….

* Identification of user requirement.
* Identification how different tasks are carried out.
* Identification whether proposed system can meet the user requirement.
* Providing technical, economical, operational feasibility of the proposed system.

The details of the feasibility study are given below:-

**TECHNICAL FEASIBILITY :-**

* The system is developed by using NetBeans 8 and JDK 1.8 which has already installed on the computer system. The machine configuration also supports the system.
* The system being user friendly, data entre & reports generation is made easy.
* Easy retrieval and access of data is provided.

**OPERATIONAL FEASIBILITY :-**

* Cost of implementation of the system as well as the installation that to consider in the operational feasibility. After implementation of the system there is need to arrange training program for users of the system.
* Expenditure for this system is also a part of this of his study

**ECONOMIC FEASIBILITY :-**

Economical feasibility is basically a cost benefit analysis. The cost concerned is….

* The cost for system development.
* The cost of hardware, software for the system is presented economical.
* The benefit can be seen in form cost cutting due to fast working.
* The user training is also included in cost of making the software, so it is beneficial deal.

**System Requirements**

For implementing this project, NetBeans IDE, JDK 1.8 and MySQL Server database used as a s/w platform.

**Software Requirement :-**

* **Operating System :** Windows 7/8/10
* **Front End ( IDE & Tool)** : NetBeans IDE and JDK 1.8
* **Back End :** MySQL Server
* **Reporting :** Jasper Reports

**Hardware Requirement :-**

* **Processor :** Intel(R) Core(TM) i5-6300U CPU

2.40GHz 2.50 GHz

* **RAM :**  Min 8GB RAM
* **Hard Disk :** 256 GB (or more)
* **Other Equipment :** Printer, Keyboard, Mouse etc.

**Introduction**

**to**

**JAVA & MySQL Server**

**Introduction to Software**

**Front End :**

* **Introduction to Java**

Java is a small, simple, safe, object oriented, interpreted or dynamically optimized, byte coded, architectural, garbage collected, multithreaded programming language with a strongly typed exception-handling for writing distributed and dynamically extensible programs.

Java is an object oriented programming language. Java is a high-level, third generation language like C, FORTRAN, Small talk, Pearl and many others. You can use java to write computer applications that crunch numbers, process words, play games, store data or do any of the thousands of other things computer software can do.

Special programs called applets that can be downloaded from the internet and played safely within a web browser. Java a supports this application and the follow features make it one of the best programming languages.

* It is simple and object oriented
* It helps to create user friendly interfaces.
* It is very dynamic.
* It supports multithreading.
* It is platform independent
* It is highly secure and robust.
* It supports internet programming

Java is a programming language originally developed by Sun Microsystems and released in 1995 as a core component of Sun's Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code which can run on any Java virtual machine (JVM) regardless of computer architecture.

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun made available most of their Java technologies as free software under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java and GNU Class path.

The Java language was created by James Gosling in June 1991 for use in a set top box project. The language was initially called Oak, after an oak tree that stood outside Gosling's office - and also went by the name Green - and ended up later being renamed to Java, from a list of random words. Gosling's goals were to implement a virtual machine and a language that had a familiar C/C++ style of notation.

**Primary goals**

There were five primary goals in the creation of the Java language:

1. It should use the object-oriented programming methodology.
2. It should allow the same program to be executed on multiple operating systems.
3. It should contain built-in support for using computer networks.
4. It should be designed to execute code from remote sources securely.
5. It should be easy to use by selecting what were considered the good parts of other object-oriented languages.

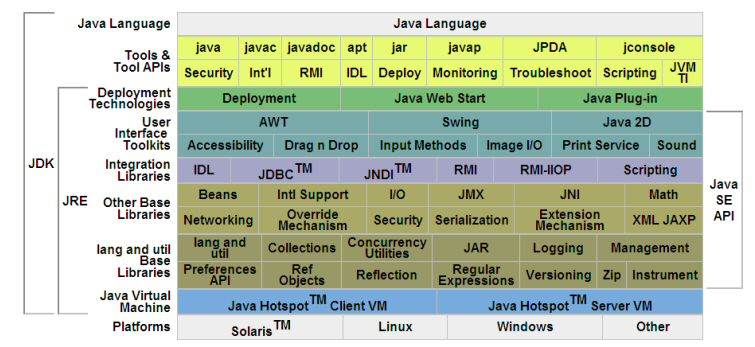
The Java platform is the name for a bundle of related programs, or platform, from Sun which allow for developing and running programs written in the Java programming language. The platform is not specific to any one processor or operating system, but rather an execution engine (called a virtual machine) and a compiler with a set of standard libraries which are implemented for various hardware and operating systems so that Java programs can run identically on all of them.

Different "editions" of the platform are available, including:

* Java ME (Micro Edition): Specifies several different sets of libraries (known as profiles) for devices which are sufficiently limited that supplying the full set of Java libraries would take up unacceptably large amounts of storage.
* Java SE (Standard Edition): For general purpose use on desktop PCs, servers and similar devices.
* Java EE (Enterprise Edition): Java SE plus various APIs useful for multi-tier client-server enterprise applications.

The Java Platform consists of several programs, each of which provides a distinct portion of its overall capabilities. For example, the Java compiler, which converts Java source code into Java bytecode (an intermediate language for the Java Virtual Machine (JVM)), is provided as part of the Java Development Kit (JDK). The sophisticated Java Runtime Environment (JRE), complementing the JVM with a just-in-time (JIT) compiler, converts intermediate bytecode into native machine code on the fly. Also supplied are extensive libraries (pre-compiled into Java bytecode) containing reusable code, as well as numerous ways for Java applications to be deployed, including being embedded in a web page as an applet.There are several other components, some available only in certain editions.

The essential components in the platform are the Java language compiler, the libraries, and the runtime environment in which Java intermediate bytecode "executes" according to the rules laid out in the virtual machine specification.



**Java Virtual Machine**

The heart of the Java Platform is the concept of a "virtual machine" that executes Java bytecode programs. This bytecode is the same no matter what hardware or operating system the program is running under. There is a JIT compiler within the Java Virtual Machine, or JVM. The JIT compiler translates the Java bytecode into native processor instructions at run-time and caches the native code in memory during execution.

The use of bytecode as an intermediate language permits Java programs to run on any platform that has a virtual machine available. The use of a JIT compiler means that Java applications, after a short delay during loading and once they have "warmed up" by being all or mostly JIT-compiled, tend to run about as fast as native programs. Since JRE version 1.2, Sun's JVM implementation has included a just-in-time compiler instead of an interpreter.

Although Java programs are Platform Independent, the code of the Java Virtual Machine (JVM) that execute these programs are not. Every Operating System has its own JVM.

**Class libraries**

In most modern operating systems, a large body of reusable code is provided to simplify the programmer's job. This code is typically provided as a set of dynamically loadable libraries that applications can call at runtime. Because the Java Platform is not dependent on any specific operating system, applications cannot rely on any of the existing libraries. Instead, the Java Platform provides a comprehensive set of standard class libraries, containing much of the same reusable functions commonly found in modern operating systems.

The Java class libraries serve three purposes within the Java Platform. Like other standard code libraries, they provide the programmer a well-known set of functions to perform common tasks, such as maintaining lists of items or performing complex string parsing. In addition, the class libraries provide an abstract interface to tasks that would normally depend heavily on the hardware and operating system. Tasks such as network access and file access are often heavily dependent on the native capabilities of the platform. The Java java.net and java.io libraries implement the required native code internally, then provide a standard interface for the Java applications to perform those tasks. Finally, when some underlying platform does not support all of the features a Java application expects, the class libraries can either emulate those features using whatever is available, or at least provide a consistent way to check for the presence of a specific feature.

**Platform independence**

One characteristic, platform independence, means that programs written in the Java language must run similarly on any supported hardware/operating-system platform. One should be able to write a program once, compile it once, and run it anywhere.

This is achieved by most Java compilers by compiling the Java language code halfway (to Java bytecode) – simplified machine instructions specific to the Java platform. The code is then run on a virtual machine (VM), a program written in native code on the host hardware that interprets and executes generic Java bytecode. (In some JVM versions, bytecode can also be compiled to native code, either before or during program execution, resulting in faster execution.) Further, standardized libraries are provided to allow access to features of the host machines (such as graphics, threading and networking) in unified ways. Note that, although there is an explicit compiling stage, at some point, the Java bytecode is interpreted or converted to native machine code by the JIT compiler.

The first implementations of the language used an interpreted virtual machine to achieve portability. These implementations produced programs that ran more slowly than programs compiled to native executables, for instance written in C or C++, so the language suffered a reputation for poor performance. More recent JVM implementations produce programs that run significantly faster than before, using multiple techniques.

One technique, known as just-in-time compilation (JIT), translates the Java bytecode into native code at the time that the program is run, which results in a program that executes faster than interpreted code but also incurs compilation overhead during execution. More sophisticated VMs use dynamic recompilation, in which the VM can analyze the behavior of the running program and selectively recompile and optimize critical parts of the program. Dynamic recompilation can achieve optimizations superior to static compilation because the dynamic compiler can base optimizations on knowledge about the runtime environment and the set of loaded classes, and can identify the hot spots (parts of the program, often inner loops, that take up the most execution time). JIT compilation and dynamic recompilation allow Java programs to take advantage of the speed of native code without losing portability.

Another technique, commonly known as static compilation, is to compile directly into native code like a more traditional compiler. Static Java compilers, such as GCJ, translate the Java language code to native object code, removing the intermediate bytecode stage. This achieves good performance compared to interpretation, but at the expense of portability; the output of these compilers can only be run on a single architecture. Some see avoiding the VM in this manner as defeating the point of developing in Java; however it can be useful to provide both a generic bytecode version, as well as an optimised native code version of an application.

**Automatic memory management**

One of the ideas behind Java's automatic memory management model is that programmers be spared the burden of having to perform manual memory management. In some languages the programmer allocates memory for the creation of objects stored on the heap and the responsibility of later deallocating that memory also resides with the programmer. If the programmer forgets to deallocate memory or writes code that fails to do so, a memory leak occurs and the program can consume an arbitrarily large amount of memory. Additionally, if the program attempts to deallocate the region of memory more than once, the result is undefined and the program may become unstable and may crash. Finally, in non garbage collected environments, there is a certain degree of overhead and complexity of user-code to track and finalize allocations. Often developers may box themselves into certain designs to provide reasonable assurances that memory leaks will not occur.

In Java, this potential problem is avoided by automatic garbage collection. The programmer determines when objects are created, and the Java runtime is responsible for managing the object's lifecycle. The program or other objects can reference an object by holding a reference to it (which, from a low-level point of view, is its address on the heap). When no references to an object remain, the Java garbage collector automatically deletes the unreachable object, freeing memory and preventing a memory leak. Memory leaks may still occur if a programmer's code holds a reference to an object that is no longer needed—in other words, they can still occur but at higher conceptual levels.

The use of garbage collection in a language can also affect programming paradigms. If, for example, the developer assumes that the cost of memory allocation/recollection is low, they may choose to more freely construct objects instead of pre-initializing, holding and reusing them. With the small cost of potential performance penalties (inner-loop construction of large/complex objects), this facilitates thread-isolation (no need to synchronize as different threads work on different object instances) and data-hiding. The use of transient immutable value-objects minimizes side-effect programming.

Comparing Java and C++, it is possible in C++ to implement similar functionality (for example, a memory management model for specific classes can be designed in C++ to improve speed and lower memory fragmentation considerably), with the possible cost of adding comparable runtime overhead to that of Java's garbage collector, and of added development time and application complexity if one favors manual implementation over using an existing third-party library. In Java, garbage collection is built-in and virtually invisible to the developer. That is, developers may have no notion of when garbage collection will take place as it may not necessarily correlate with any actions being explicitly performed by the code they write. Depending on intended application, this can be beneficial or disadvantageous: the programmer is freed from performing low-level tasks, but at the same time loses the option of writing lower level code. Additionally, the garbage collection capability demands some attention to tuning the JVM, as large heaps will cause apparently random stalls in performance.

Java does not support pointer arithmetic as is supported in, for example, C++. This is because the garbage collector may relocate referenced objects, invalidating such pointers. Another reason that Java forbids this is that type safety and security can no longer be guaranteed if arbitrary manipulation of pointers is allowed.

**Performance**

Java's performance has improved substantially since the early versions, and performance of JIT compilers relative to native compilers has in some tests been shown to be quite similar. The performance of the compilers does not necessarily indicate the performance of the compiled code; only careful testing can reveal the true performance issues in any system.

**Java Runtime Environment**

The Java Runtime Environment, or *JRE*, is the software required to run any application deployed on the Java Platform. End-users commonly use a JRE in software packages and Web browser plugins. Sun also distributes a superset of the JRE called the Java 8 SDK (more commonly known as the JDK), which includes development tools such as the Java compiler, Javadoc, Jar and debugger.

* One of the unique advantages of the concept of a runtime engine is that errors (exceptions) should not 'crash' the system. Moreover, in runtime engine environments such as Java there exist tools that attach to the runtime engine and every time that an exception of interest occurs they record debugging information that existed in memory at the time the exception was thrown (stack and heap values). These Automated Exception Handling tools provide 'root-cause' information for exceptions in Java programs that run in Itemion, testing or development environments.

**Back End :**

* **Introduction to MySQL Server**

MySQL is an open source relational database management system and, since it is open source, starter versions are free to use. MySQL is used in more than 10 million installations and part of its success is down to the fact that it runs on more than 20 platforms. Since MySQL is open source, some versions are free and many host providers’ customers are able to request MySQL databases of their own, thus for small to medium size applications MySQL is a far more compatible, far cheaper alternative to Microsoft Access as well as being known for its reliability, ease of use and performance, traits that are not often apparent with Microsoft’s offering.

MySQL is even a competitor to Microsoft’s SQL Server and Oracle, offering tens of thousands of dollars in savings over the competition. MySQL is one of the many variants of database management systems that are based on the SQL (Structured Query Language) database query language. SQL was developed in the 1970s by IBM and it is an ANSI/ISO standard for creating, modifying and retrieving data. Java and MySQL are a combination used by many because Java has easy to use, powerful MySQL connectivity built into it and can perform queries quickly.

**System Design**

**Data Flow Diagram(DFD)**

**Context level Data Flow Diagram**

Member

Place and get Order

Supplier

Payment to supplier

Return Book

Issue Book

Manage Masters

Generate Reports

Admin

**First level Data Flow Diagram**

Login

Details

**Admin**

Master

Entries

**r**

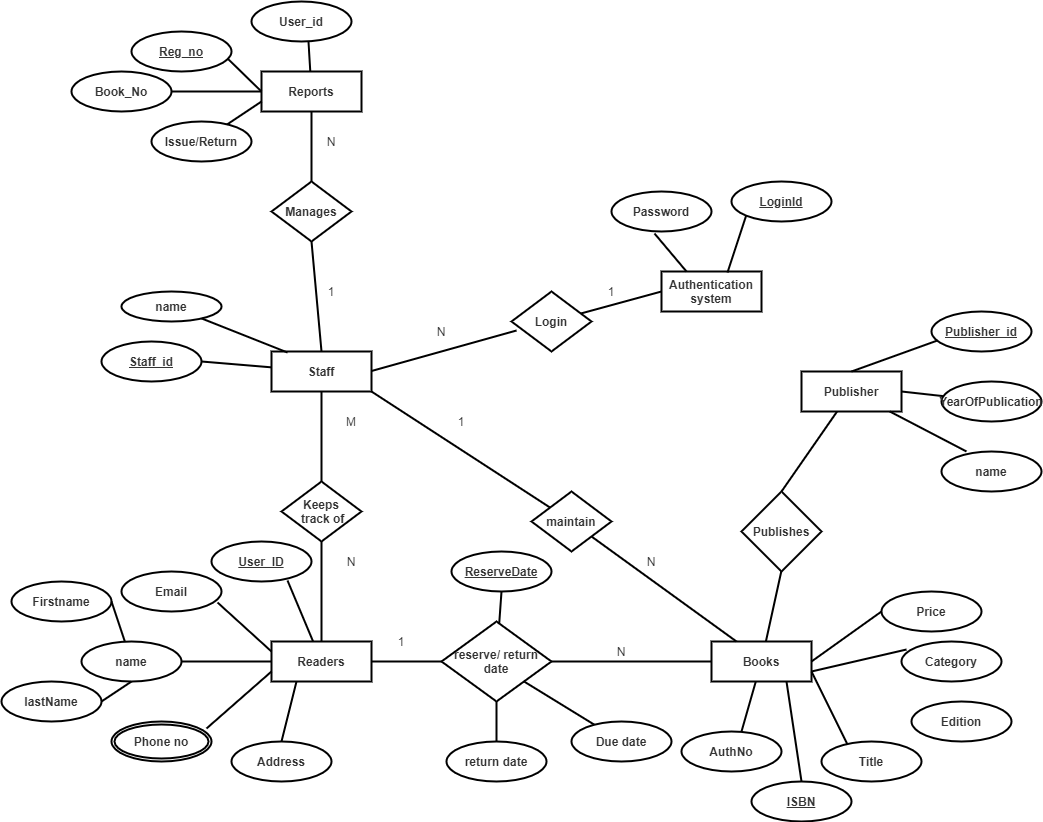
SuppPayment

Report

Order

ReturnBook

**Entity Relationship Diagram (ERD)**



mobileno

address

Supplier

**Data Dictionary**

**Data Dictionary**

**Table Name:-**  Login Table

**Description:-** It Store Login Information.

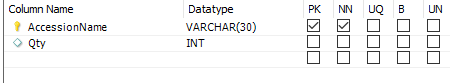
**Primary Key:-** Username



**Table Name:-** Accession

**Description:-** The table contain Accession details.

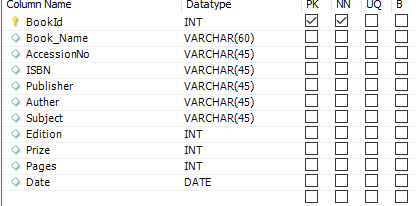
**Primary Key:-** AccessionName



**Table Name:-** Book

**Description:-** The table contain book information.

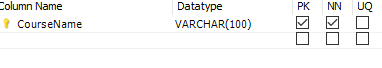
**Primary Key:-** Book Id



**Table Name:-** Course

**Description:-** This table is inform by course Information.

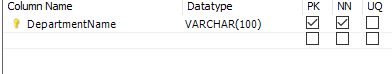
**Primary Key:-** CourseName



**Table Name:-** Department

**Description:-**This Table Contain Department details.

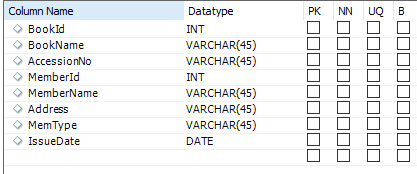
**Primary Key:-** DepartmentName



**Table Name:-** Issue

**Description:-** This Table Contain by Books Issue Information.

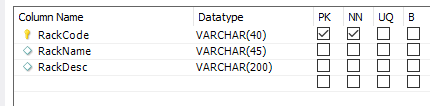
**Primary Key:-** Id



**Table Name:-** Location

**Description:-** This Table Contain By Location details.

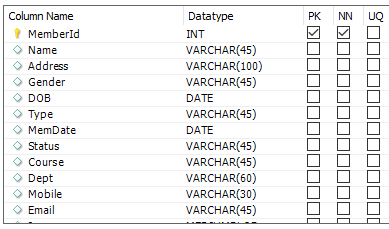
**Primary Key:-** RackCode



**Table Name:-** Member

**Description:-** This Table Contain By Members Id Information.

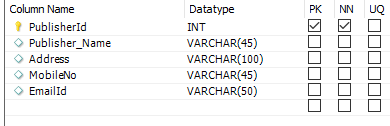
**Primary Key:-** Menber Id



**Table Name:-** Publisher

**Description:-** This Table Contain By PublisherId details.

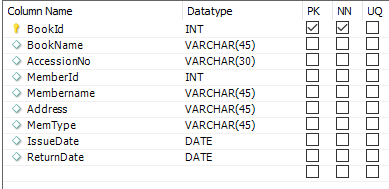
**Primary Key:-** PublisherId



**Table Name:-** Returnbook

**Description:-** This Table Contain By Returnbook details.

**Primary Key:-** Book Id



**Table Name:-** Subject

**Description:-** This table contain by subject names details.

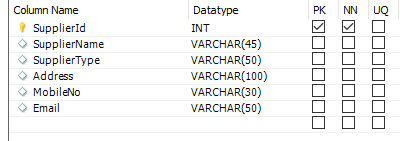
**Primary Key:-** ID



**Table Name:-** Supplier

**Description:-** This table contain by supplier details.

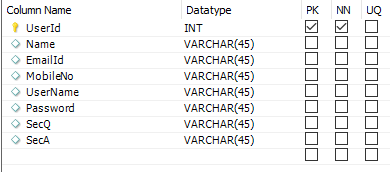
**Primary Key:-** Suplier Id



**Table Name:-** User

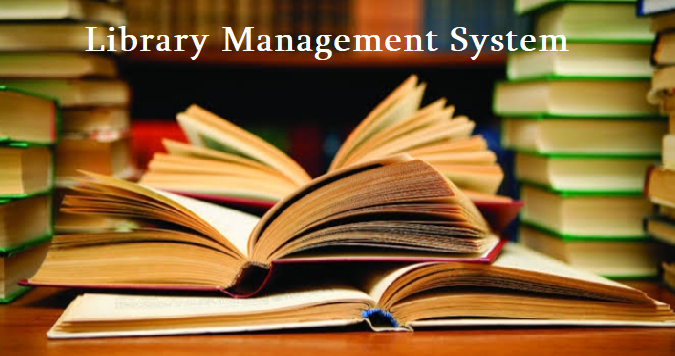
**Description:-**This table contain by user details.

**Primary Key:-** User Id

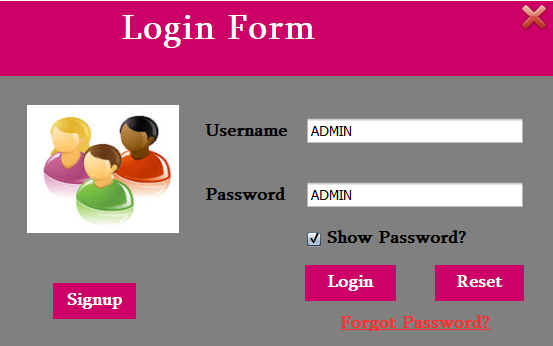


**Input & output screen**

**Splash Screen:-**

****

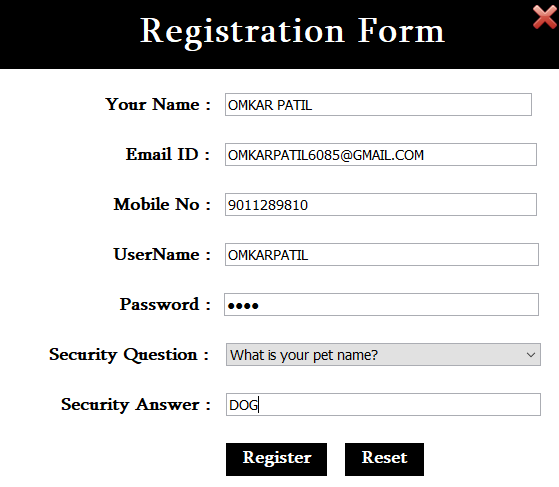
**Login Form:-**



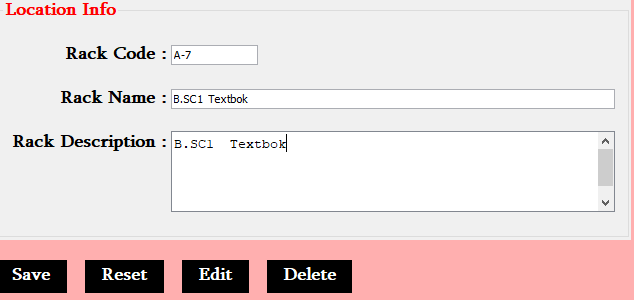
**Main Form:-**



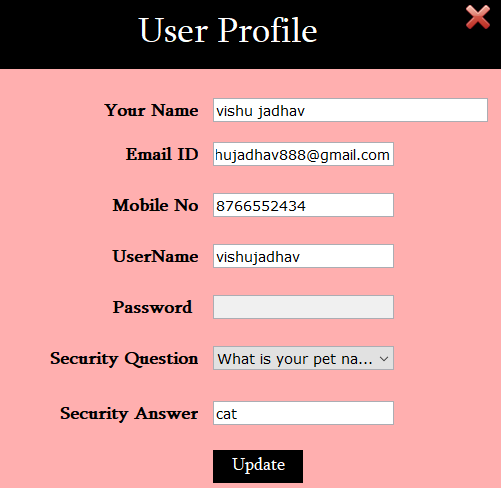
**Registration Form:-**



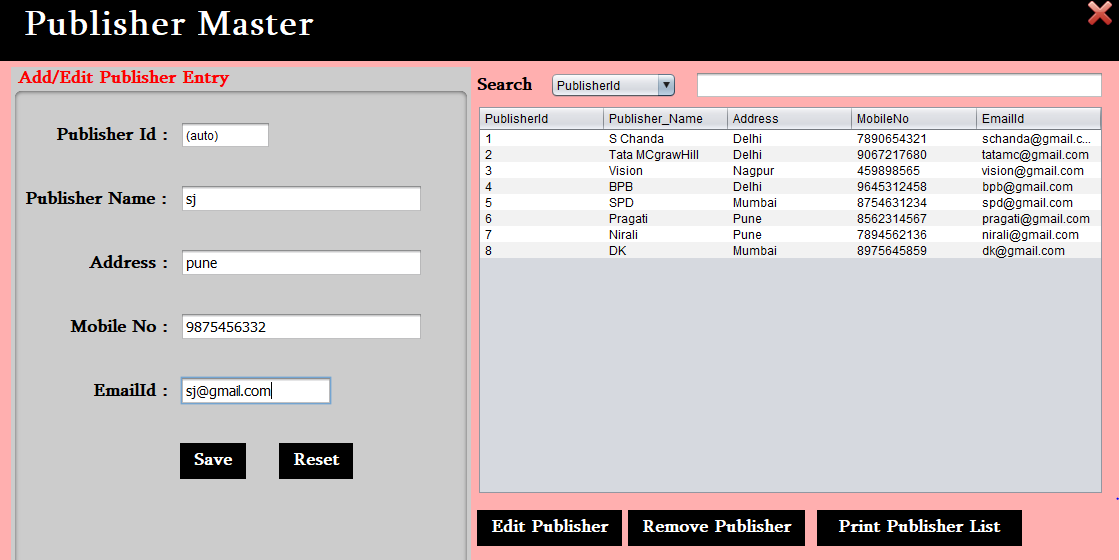
**Location Master:-**



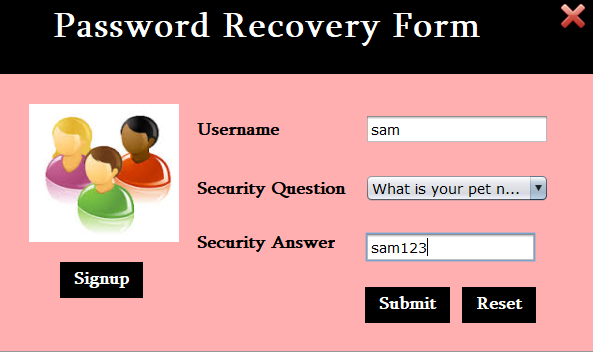
**User Profile:-**



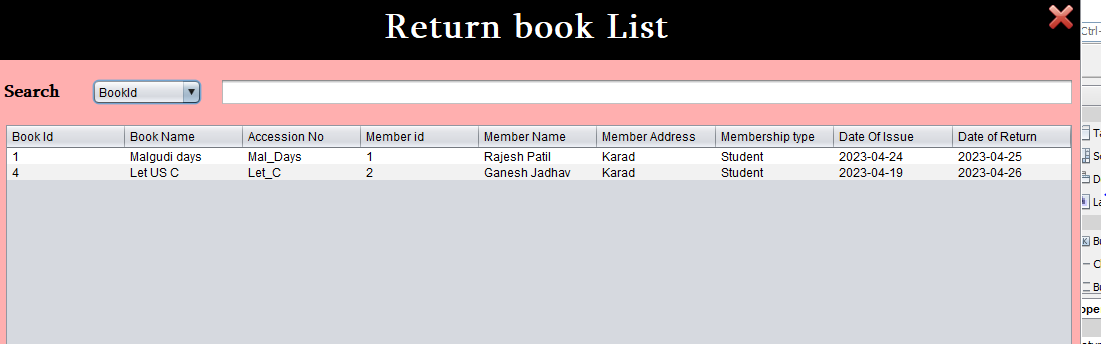
**Publisher Master:-**



**Password Recovery Form:-**

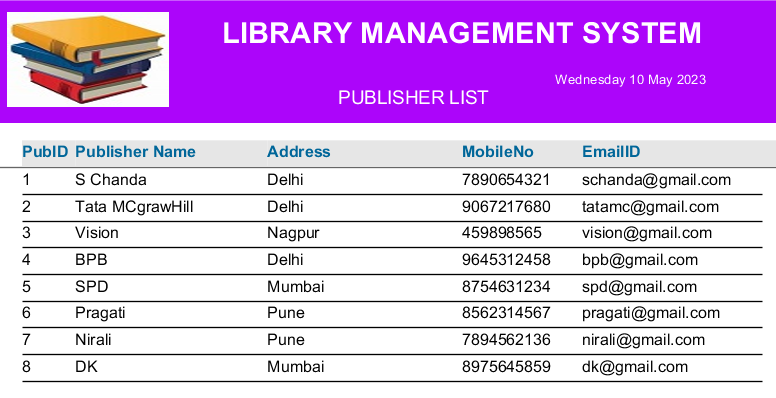


**Return book List:-**

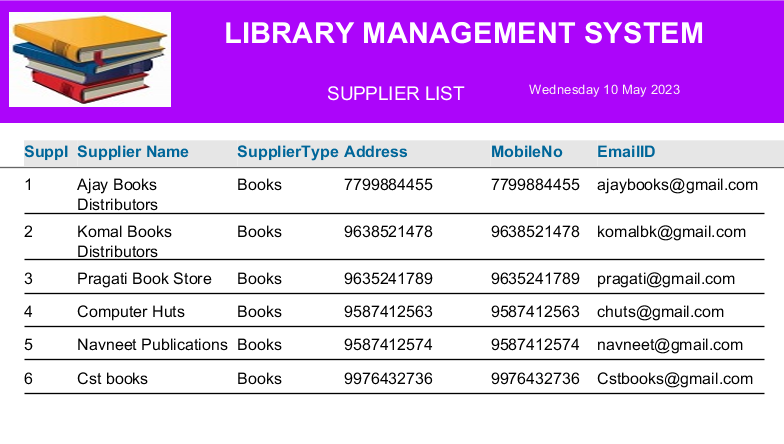


**Report**

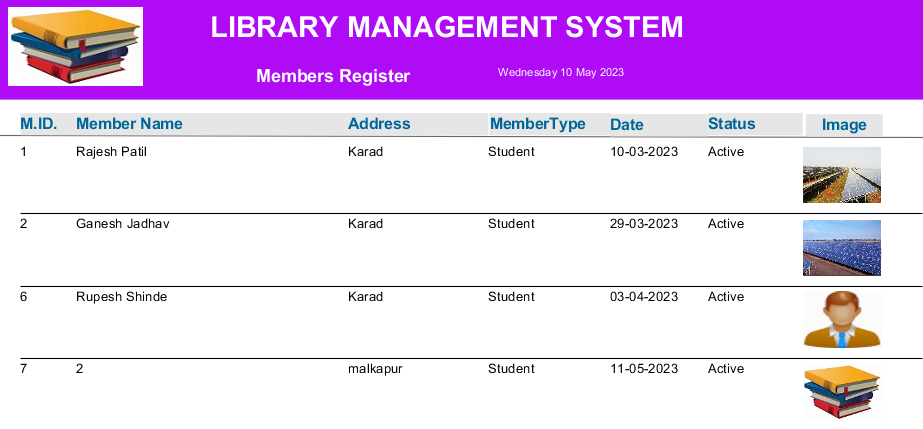
**Publisher List :-**

****

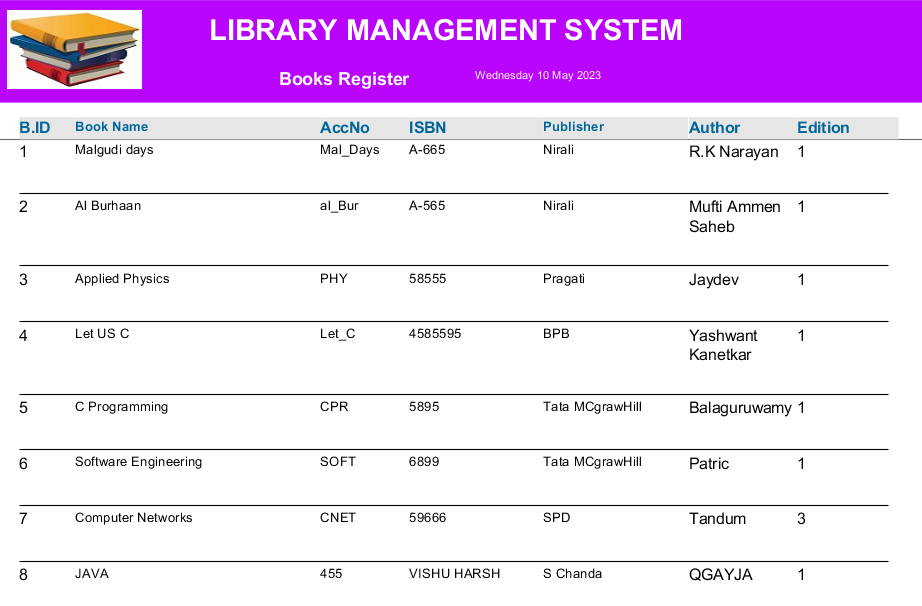
**Supplier List :-**

****

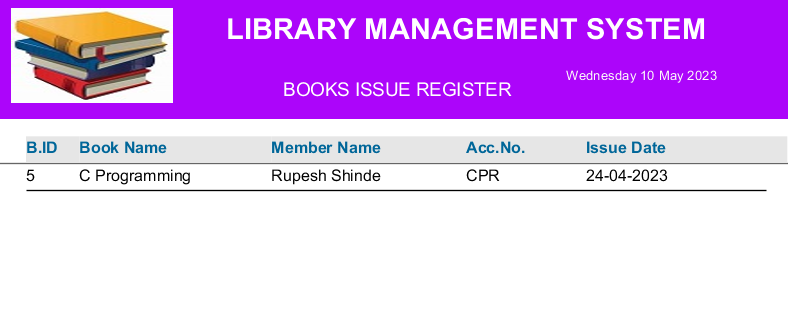
**Members Register :-**

****

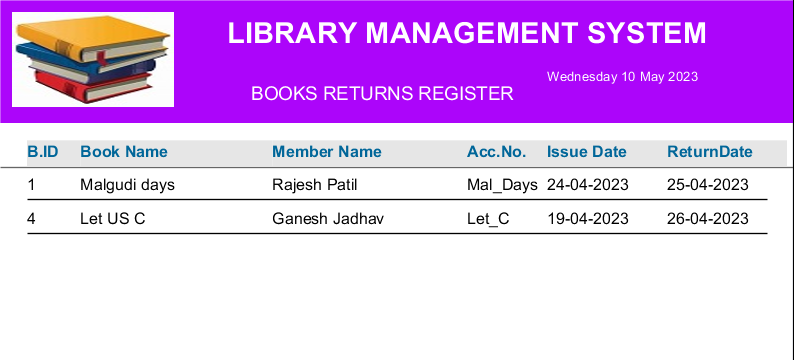
**Books Register :-**

****

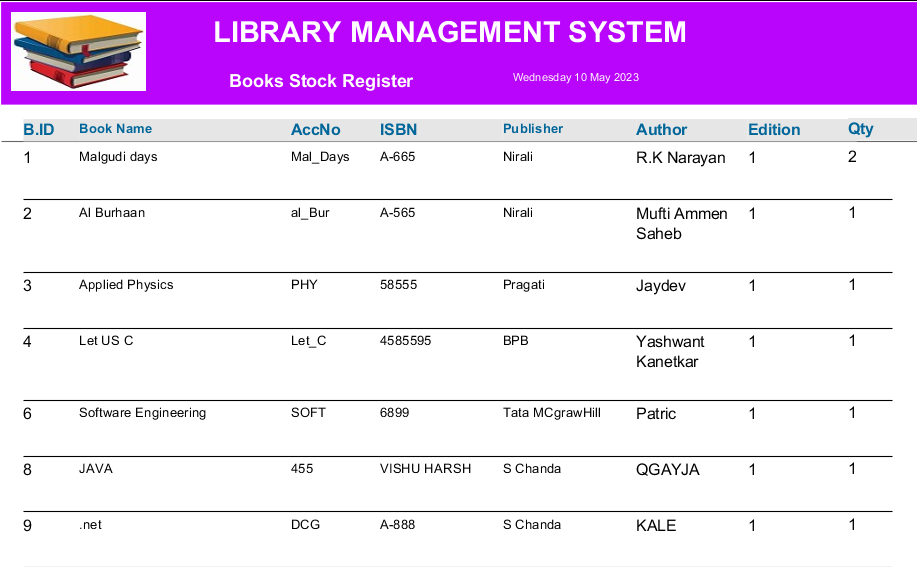
**Books Issue Register :-**

****

**Books Returns Register :-**

****

**Books Stock Register :-**

****

**User manual**

**USER MANUAL**

**User Manual: -**

First when user visit to a Software You will find following menus.

* **Splash Form**

Splash forms are forms XAF WinForms applications show automatically at the application's startup and when users log in.

* **Login**

 a user ID and password combination that allows users to access a website or application.

* **Main Form**

This is the First page of Software

* **Login**

In This Form we Show all Detail about User Login

After Login You will find following menus.

* **Main Form**

the main way to access other forms, load and save files, control trajectory playback, change various global program settings, access help, and to quit the program.

* **Library Information Form**

 a collection of materials, books or media that are accessible for use and not just for display purposes

* **Books Category Form**

  fiction and non-fiction

* **Publisher Form**

Click the Add-ons button in the upper-right corner

* **Books Form**

 the published format of a story

* **Member Form**

a document used by social and recreational organizations to collect information from prospective members.

* **Supplier Form**

 used to gather information about suppliers.

* **Purchase Order to Supplier Form**

Create a purchase order.

* **Material Receipt from**

 refer to data, such as quantities and lot numbers, associated with items that are used for work done on-site by your employees.

* **Purchase returns Form**

when the buyer of merchandise, inventory, fixed assets, or other items sends these goods back to the seller.

* **Issue Book Form**

 a specific change in the book during the printing of an edition.

* **Return Books Form**

a book in accounting where the goods that are returned to the supplier are recorded.

* **Supplier Payment Form**

 the last step in the purchase to pay cycle, when a company pays an outside vendor for purchased goods or services.

* **Books Register**

 usually placed at the entrance to the visitation room and/or funeral/memorial service, and is signed by the guests before entering

* **Change Password**

Under "Signing in to Google," tap Password

**Etc.**

**System Testing**

**System testing**

Testing is the major control measure used during software development. Its basic function is to detect errors in the software. Thus the goal of testing is to uncover the requirements, design and coding errors in the programs.

Main activities performed in this phase are:

* Black box Testing
* White Box Testing
* Unit Testing

**WHITE BOX TESTING :-**

For doing this testing process the person have to access to the source code of the Item to be tested. So it is essential that the person doing this white box testing have some knowledge of the program being tested.

**BLACK BOX TESTING :-**

This is otherwise called as functional testing. In contrary to white box testing here the person who is doing the black box testing need not have the programming knowledge .This is because the person doing the black box testing would access the output and would perform thorough functionality testing to check whether the developed module or Items

**UNIT TESTING :-**

In this testing procedure, individual program will be tested with respect output from the system. After compilation of program some test data as per specification will be used for testing of the program to see if it works as specified.

**FUTURE ENHANCEMENT**

**Future Enhancement**

* The Library Management System could reduce the manual work and physical entities of the system.
* The system can be further enhanced by proposing an advance Management system.
* The system can be further enhanced by proposing an advance Facilities.
* In future due to increase in records of database file, data redundancy occurs.
* More modules can be included to keep track of occurrence.
* Generation of financial reports financial hierarchical detailed information about profit.
* System can be implemented as Online Web Application as well as Android App.
* We have not included various offers options in current system
* We have not maintained the process of mobile repairing

**Conclusion**

**Conclusion**

Finally, we can conclude that the system we had developed will eliminate the existing system’s drawbacks and limitations to maximum extent. And provide the user with a Item of high quality, standard sand excellence. Hence, it will be very profitable to the and satisfaction will also be achieved because the delay and in convenience caused due to existing system will be eliminated.

During the project, we followed all principles of System Analysis and Design. We want through all the stages of Software Development Life Cycle i.e. starting with Requirement Analysis till Final Implementation of the Software.

**BIBLOGRAPHY**

* **Book:**
* Murach’s publication Java Programming (3rd Edition)
* The Java Complete Reference – Herbert Schildt – Oracle Press
* The Java for Programmers – Deitel Publications
* The Introduction to Java Programming - Y. Daniel Liang - Armstrong Atlantic State University
* **Reference:**
* [www.stackoverflow.com](http://www.stackoverflow.com)
* <http://www.javatpoint>.com
* <https://www.tutorialspoint.com/>