Library Management System

Description	Page 2
DB Schema	Page 6
Class Diagram	Page 7

The project (Library Management System) is a Library management software for tracking and managing the daily transactions in a Library. The software (Library Management System) was built using java, which focus on basic operations in a library such as registering new members, adding new books and a user-friendly interface to facilitate borrowing, and returning books.

"Library Management System" is a windows application designed to help users maintain and organize daily activities in a library. Our software is easy to use for both beginners and advanced users. It features a familiar and user-friendly interface, combined with searching, insertion, and reporting capabilities. The report generation for a library system helps to get a good idea on which are the books borrowed by the members.

The software (Library Management System) has three main modules.

- Insertion to Database Module – User-friendly input screen
- Extraction from Database module – Attractive Output Screen
- Report Generation module – Borrowed book list & Available book list

PROPSED SYSTEM

Proposed system is an automated Library Management System. Through our software use can add members, add books, issue, and return books in quick time. Our proposed system has the following advantages.

- User friendly interface
- Fast access to database
- Less error
- Look and feel Environment
- Quick transaction

All the manual difficulties in managing the Library have been rectified by implementing computerization.

FEASIBILTY ANALYSIS

Whatever we think need not be feasible. It is wise to think about 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 could be either positive or negative. When the positives outweigh the negatives, then the system is considered feasible. Here the feasibility study can be performed in two ways such as technical feasibility and Economic Feasibility.

Technical Feasibility:

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

Economic Feasibility

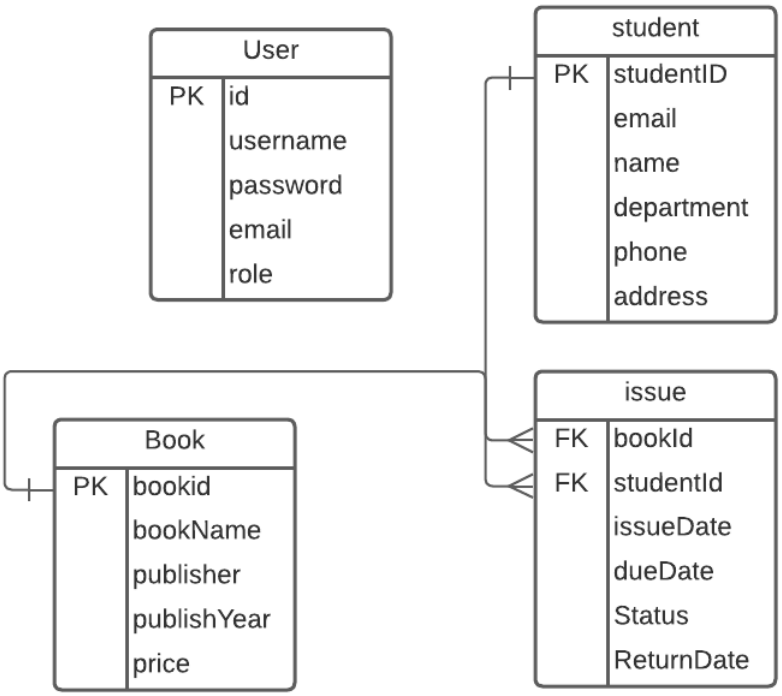
Development of this application is highly economically feasible. It is cost effective for the organization since the development cost and maintenance is low. The only thing is to be done is making an environment for the development with an effective supervision. Even

after development, the organization will not be in a condition to invest more money. Therefor, the system in economically feasible.

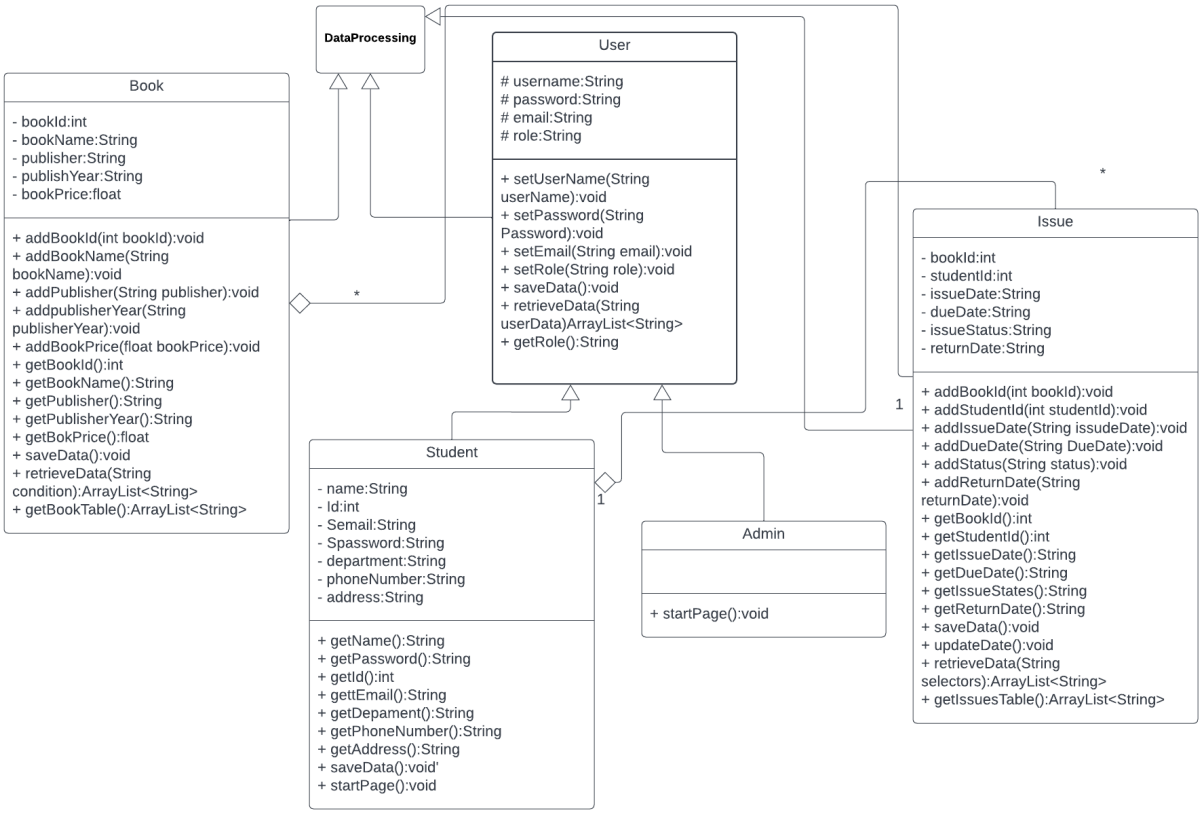
SOFTWARE CONFIGURATION

Operating System	: Windows 10, Windows 11.
Language	: Java
Data Base	: Microsoft SQL server

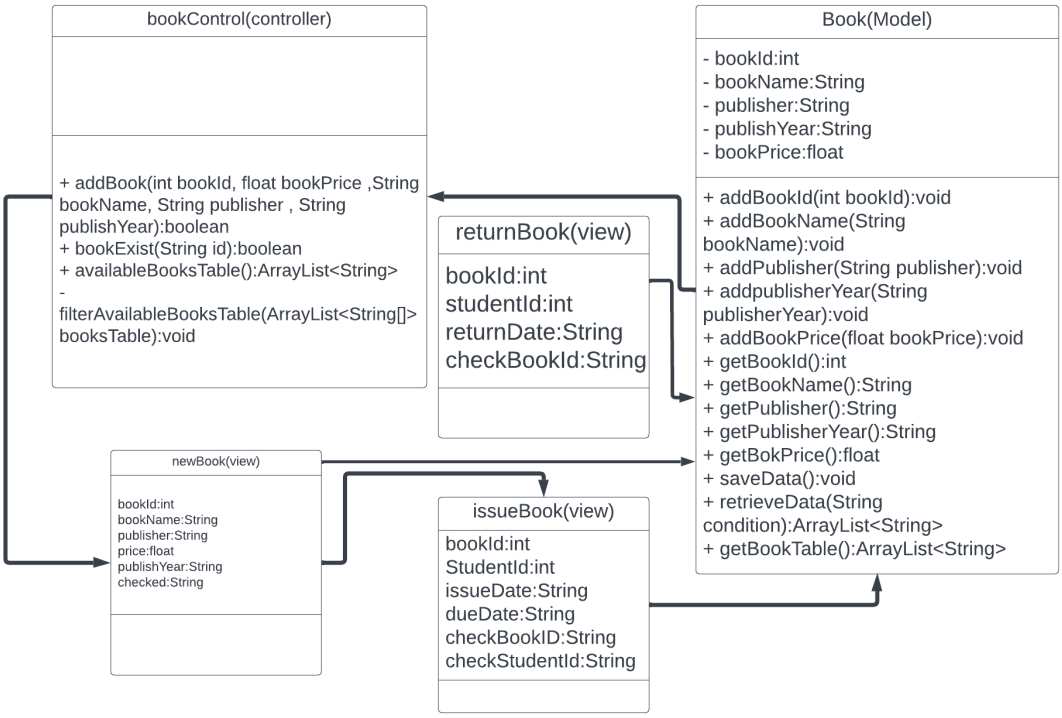
DataBase Schema:

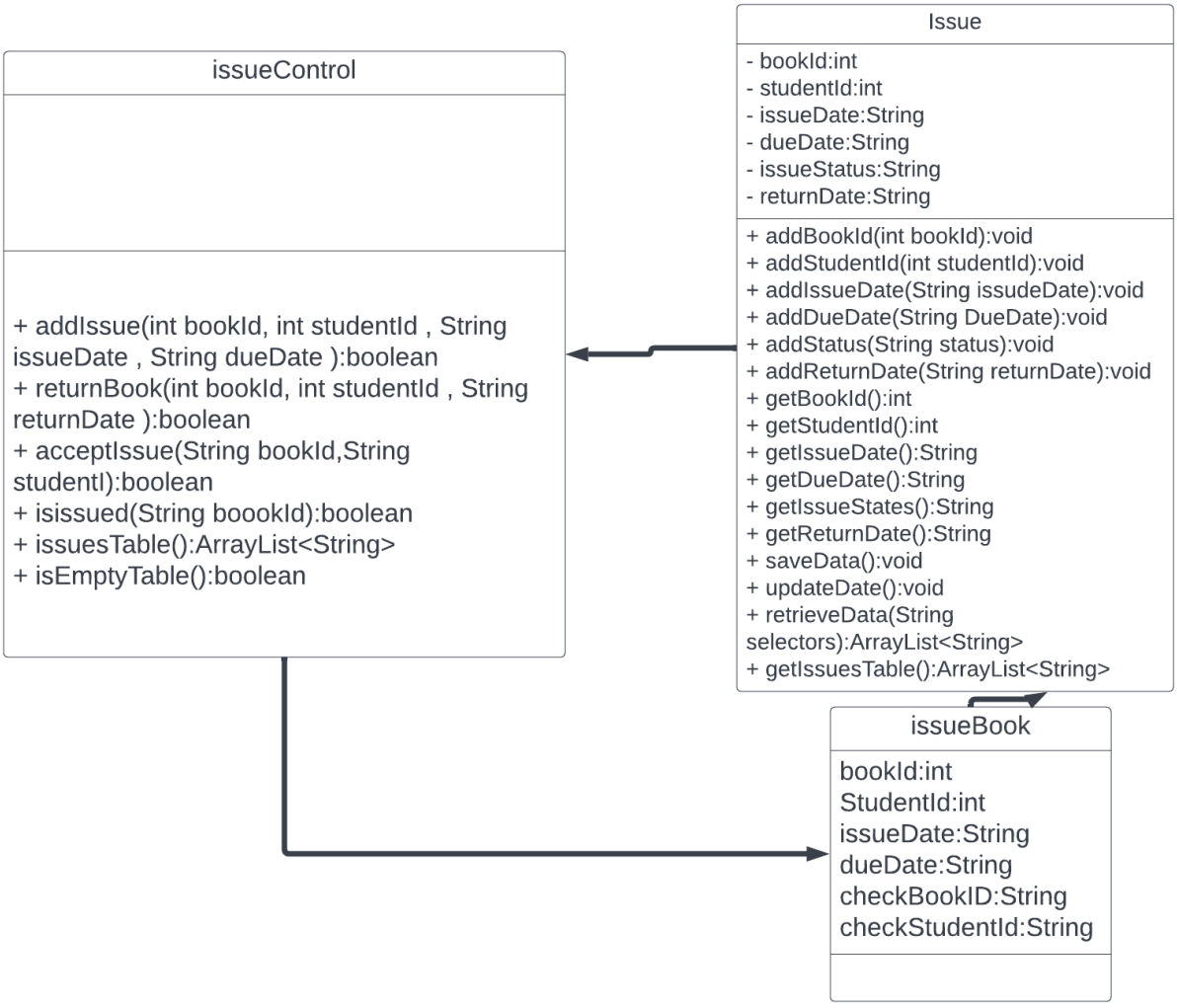


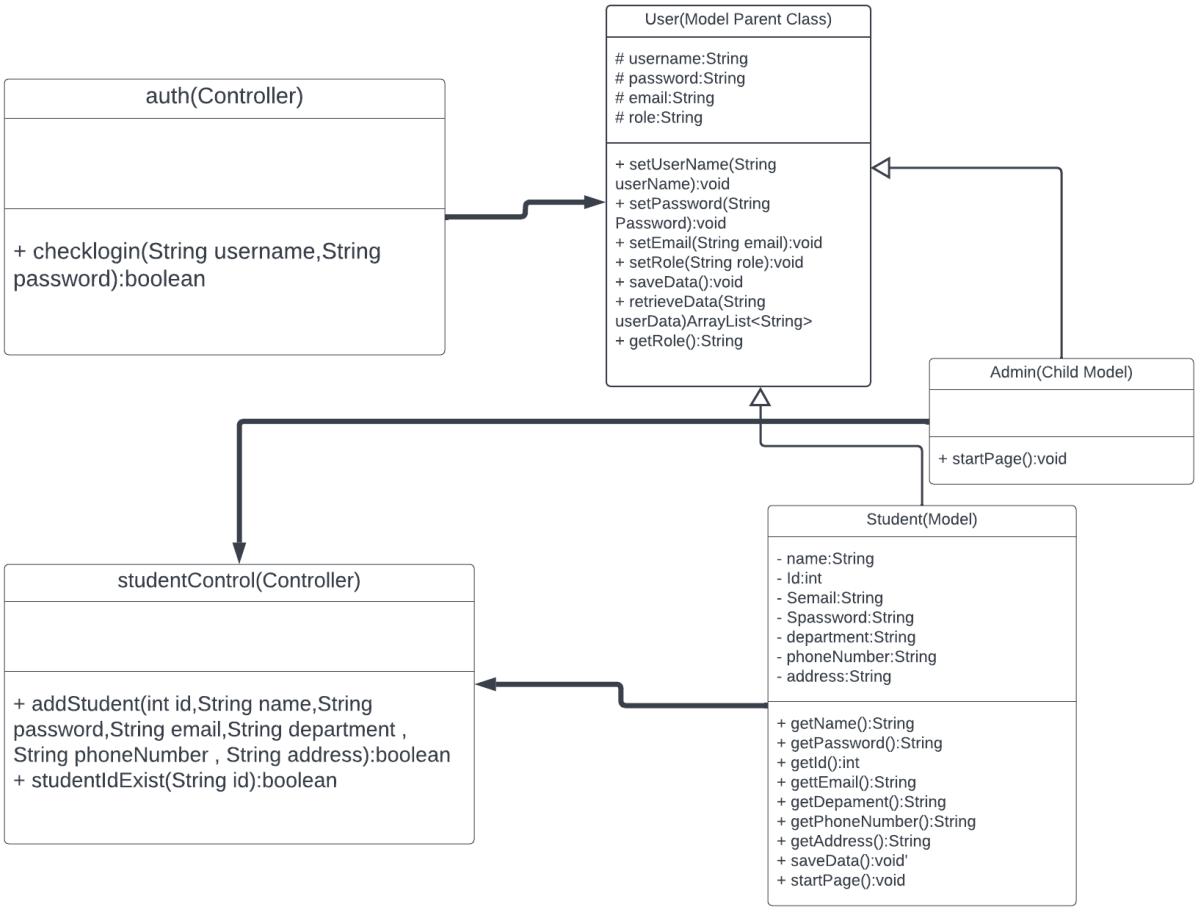
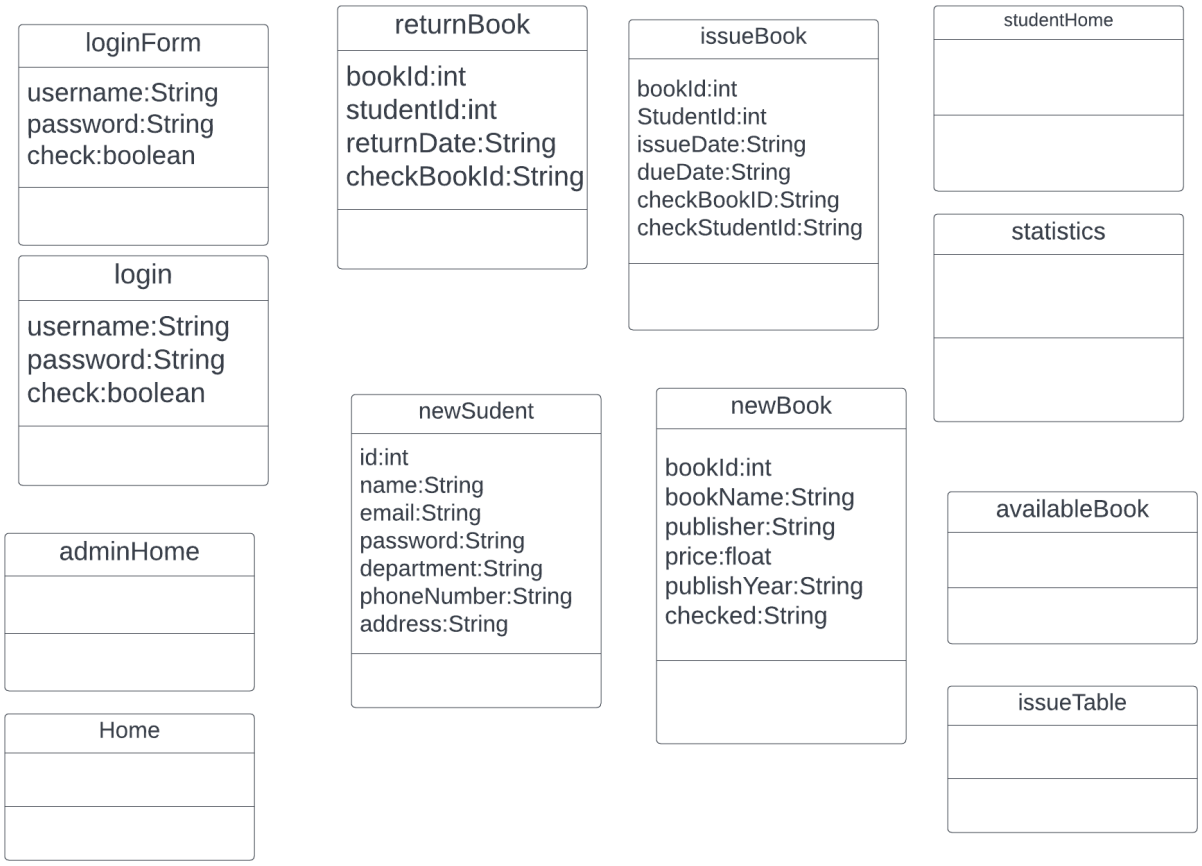
Model class diagram:



MVC Class Diagrams:




issueControl
+ addIssue(int bookId, int studentId , String issueDate , String dueDate):boolean
+ returnBook(int bookId, int studentId , String returnDate):boolean
+ acceptIssue(String bookId,String studentI):boolean
+ isissued(String boookId):boolean
+ issuesTable():ArrayList<String>
+ isEmptyTable():boolean
Issue
- bookId:int
- studentId:int
- issueDate:String
- dueDate:String
- issueStatus:String
- returnDate:String
+ addBookId(int bookId):void
+ addStudentId(int studentId):void
+ addIssueDate(String issudeDate):void
+ addDueDate(String DueDate):void
+ addStatus(String status):void
+ addReturnDate(String returnDate):void
+ getBookId():int
+ getStudentId():int
+ getIssueDate():String
+ getDueDate():String
+ getIssueStates():String
+ getReturnDate():String
+ saveData():void
+ updateDate():void
+ retrieveData(String selectors):ArrayList<String>
+ getIssuesTable():ArrayList<String>
issueBook
bookId:int
StudentId:int
issueDate:String
dueDate:String
checkBookID:String
checkStudentId:String


User(Model Parent Class)
# username:String
# password:String
# email:String
# role:String
+ setUserName(String userName):void
+ setPassword(String Password):void
+ setEmail(String email):void
+ setRole(String role):void
+ saveData():void
+ retrieveData(String userData)ArrayList<String>
+ getRole():String
auth(Controller)
+ checklogin(String username,String password):boolean
Admin(Child Model)
+ startPage():void
Student(Model)
- name:String
- Id:int
- Semail:String
- Spassword:String
- department:String
- phoneNumber:String
- address:String
+ getName():String
+ getPassword():String
+ getId():int
+ gettEmail():String
+ getDepament():String
+ getPhoneNumber():String
+ getAddress():String
+ saveData():void'
+ startPage():void
studentControl(Controller)
+ addStudent(int id,String name,String password,String email,String department , String phoneNumber , String address):boolean
+ studentIdExist(String id):boolean

Views class diagram: