**Design Specification document**

**High School Book Tracking Application**

Authors: Umashankar Rajaram (A20391260)

Aishwarya Anand (A20331867)

Adnan Haider (A20324647)

Jagruti Vichare (A20378092)

**1 Introduction**

The requirement specification document is designed to specify the user requirements and can be referred in next stages of the development of the software product. It gives the specification of the software product to be developed which are agreed upon by the customer.

**1.1 Goals and Objectives**

The primary goal of this document is to illustrate the High School Book Tracking Application's uses and functionalities to be covered as specified by the customer.

**1.2 Statement of Scope**

The High School Book Tracking Application is intended for high school students, their parents, staffs and teachers. It is basically designed to be able to keep track of books checked out to students, their due dates etc.

A staff when logged into application, should be able to add/delete books, checkout books to students for a particular period of time and release checked out books. A teacher when logged into application should be able to checkout/release only those books for which he/she takes classes. A student/parent can only see his/her checked out books information and should be able to modify his/her personal details.

**1.3 Software Context**

The High School Book Tracking application should be secured as only authorized users can access this application and the access should be granted based on access levels. The system should not take more than 5 seconds to respond to a request. Only a staff/Admin has a right to add books to this application from a pool of books available online. Once the books are available, they can be checked out to students for a period of time (6 months). Once a book is checked out to a student, he can log in to the application to check the details such as due date, due amount etc. for the checked out books.

This application makes it easy for staffs/teachers to track all the books and the amount due for each and every book checked out to a particular student. If a student fails to return the book on time, late fees of $1/month will be added to due amount.

**1.4 Major Constraints**

The application may not work properly if used with browsers other than google chrome or mozilla.

There won’t be any password recovery/forgot password feature nor any lockout phase after numerous entries of invalid password.

**2 Data Design**

**2.1 Internal software data structures**

The data for the front-end are basically extracted out of the database using the query mechanism. Post retrieval, the resident data will be stored in the form of *list* and *hash* data structures depending upon the kind of data, which are then passed among the software components as per the operations.

**2.2 Global data structures**

The software architecture work with the following data structures, which are of class type, each having their own set of attributes, their data type, maximum length and default value.

1. Student object

|  |  |  |  |
| --- | --- | --- | --- |
| **Student** | | | |
| ***Member*** | ***Type*** | ***Max Length (in Bytes)*** | ***Default Value*** |
| UserID | int | 4 | 0 |
| Name | string | 40 | NULL |
| Access Level | int | 4 | 0 |
| Contact No | int | 4 | 0 |
| Parent Name | string | 40 | NULL |
| Parent Contact No | int | 4 | 0 |
| Amount Due | float | 4 | 0 |

1. Faculty object

|  |  |  |  |
| --- | --- | --- | --- |
| **Faculty** | | | |
| ***Member*** | ***Type*** | ***Max Length***  ***(in Bytes)*** | ***Default Value*** |
| UserID | int | 4 | 0 |
| Name | string | 40 | NULL |
| Access Level | int | 4 | 0 |

1. Staff/Admin object

|  |  |  |  |
| --- | --- | --- | --- |
| **Staff/Admin** | | | |
| ***Member*** | ***Type*** | ***Max Length***  ***(in Bytes)*** | ***Default Value*** |
| UserID | int | 4 | 0 |
| Name | string | 40 | NULL |
| Access Level | int | 4 | 0 |

1. Book object

|  |  |  |  |
| --- | --- | --- | --- |
| **Book** | | | |
| ***Member*** | ***Type*** | ***Max Length*** | ***Default Value*** |
| ***(in Bytes)*** |
| ISBN | string | 40 | NULL |
| BookID | int | 4 | 0 |
| Book Name | string | 4 | 0 |
| Author | string | 4 | 0 |
| Price | float | 40 | NULL |
| Total Available | int | 4 | 0 |
| Total Checked out | int | 4 | 0 |

1. Class object

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | | | |
| ***Member*** | ***Type*** | ***Max Length*** | ***Default Value*** |
| ***(in Bytes)*** |
| ClassID | int | 4 | 0 |
| ClassName | string | 40 | NULL |
| DeptName | string | 40 | NULL |

1. Transaction object

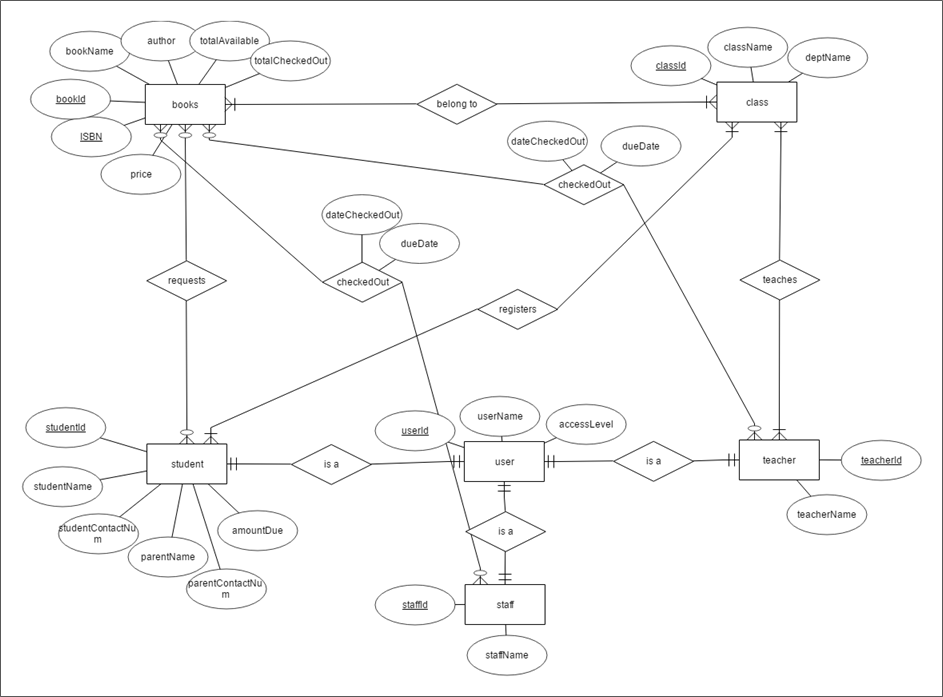
|  |  |  |  |
| --- | --- | --- | --- |
| **Transaction** | | | |
| ***Member*** | ***Type*** | ***Max Length*** | ***Default Value*** |
| ***(in Bytes)*** |
| TransID | int | 4 | 0 |
| BookID | int | 4 | 0 |
| NoOfBooks | int | 4 | 0 |
| TransDate | string | 20 | NULL |
| DueDate | string | 20 | NULL |

1. Fee object

|  |  |  |  |
| --- | --- | --- | --- |
| **Fee** | | | |
| ***Member*** | ***Type*** | ***Max Length*** | ***Default Value*** |
| ***(in Bytes)*** |
| UserID | int | 4 | 0 |
| AmountDue | float | 4 | 0 |

**2.4 Database design**

The features of High School Book Tracking Application with respect to the user accounts are shown below in the form of an ER-Diagram:



**The schema for above ER-Diagram:**

* **books** (varchar bookId, varchar bookName, varchar author, varchar ISBN, int price, int totalAvailable, int totalCheckedOut)
* **student** (varchar studentId, varchar userId, varchar studentName, int studentContactNum, varchar parentName, int parentContactNum, int amountDue)
* **requests** (varchar bookId, varchar studentId)
* **class** (varchar classId, varchar className, varchar deptName)
* **belongTo**(varchar bookId, varchar classId)
* **registers**(varchar studentId, varchar classId)
* **teacher**(varchar teacherId, varchar userId, varchar teacherName)
* **teaches**(varchar teacherId, varchar classId)
* **staff**(varchar staffId, varchar userId, varchar staffName)
* **checkedOut**(varchar userId, varchar bookId, varchar studentId, date dateCheckedOut, date dueDate)
* **user**(varchar userId, varchar userName, varchar accessLevel, varchar password)

Based on the ER-Diagram above, we have the following set of relationships among the objects:

* **Belongs to:**

This relationship stores details of all the books with their corresponding class details i.e which book is referred by which class.

* **Requests:**

This relationship stores details of all the books requested by students.

* **CheckedOut:**

This relationship stores details of all the books checked out by staffs/teachers along with date it is checked out and its due date.

*Attributes:*

**1**. **dateCheckedOut -** This attribute specifies the date at which book has been checked out for a student.

**2. dueDate -** This attribute specifies the due date for every checked out book.

* **Registers:**

This relationship stores information of all the students along with the classes they are registered for.

* **Teaches:**

This relationship stores details of all the teachers along with the classes they take.

* **Is a:**

This relationship is an inheritance relationship. We have multiple data objects which inherit from a user data object.

**3 Architectural and component-level design**

**3.1 System Structure**

Architecture diagram for High School Book Tracking application:



**High School Book tracking application:** Application is written using Spring Boot framework running on apache tomcat application server, which serves as a service accessible to user at anywhere anyplace through internet. It also provides an API which can be integrated with other system for flexibility and extensibility.

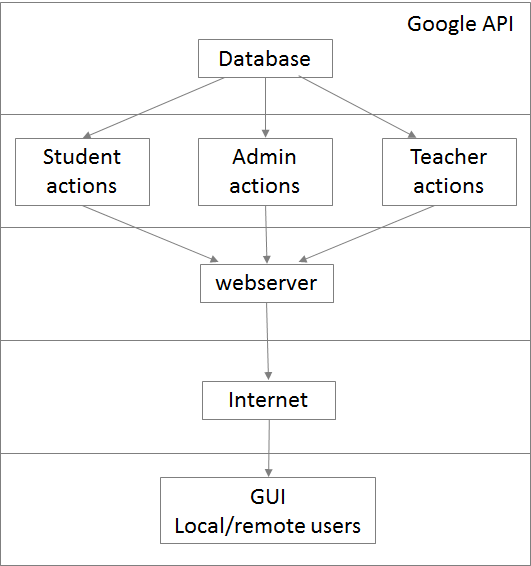
Architecture is composed of following component/layers:

**Controller**: Act at front to serve any valid request to application as well it also handles the security of application. Controller consist of handlers that is able to redirect request and response form server by using view layer for sending HTML or JSP response to user.

**View Layer**: View layer is responsible to rendering and redirecting pages based on request made by handler of controller.

**Service Layer**: This layer is responsible for handling all the business application functionality processing as well some implementation logic of application.

**DAO layer**: DAO layer shares the responsibility of performing CURD (Create, update, read and write) database transactional operation for service layer.



**3.2 Description for Component**

**Component 1 - Registration**

Input user information and save the details in the database.

Only teachers and staffs can do the registration of students.

**Component 2 - Add/Delete books**

Only staffs and teachers can add/delete books.

Teachers can only add/delete books of classes they teach for.

Input bookId and update local database accordingly.

**Component 3 - View/search books**

Input bookId or author or name of the book and details of the corresponding book will be shown

**Component 4 - Request books**

Input studentId and bookId and request will be gone to staff/teacher for the approval.

Students are not allowed to checkout books. Only staff/teacher can checkout books.

**Component 5 - Checkout books**

Input studentId, bookId and userId and update the database with checkedout date and due date.

**Component 6 - Return books**

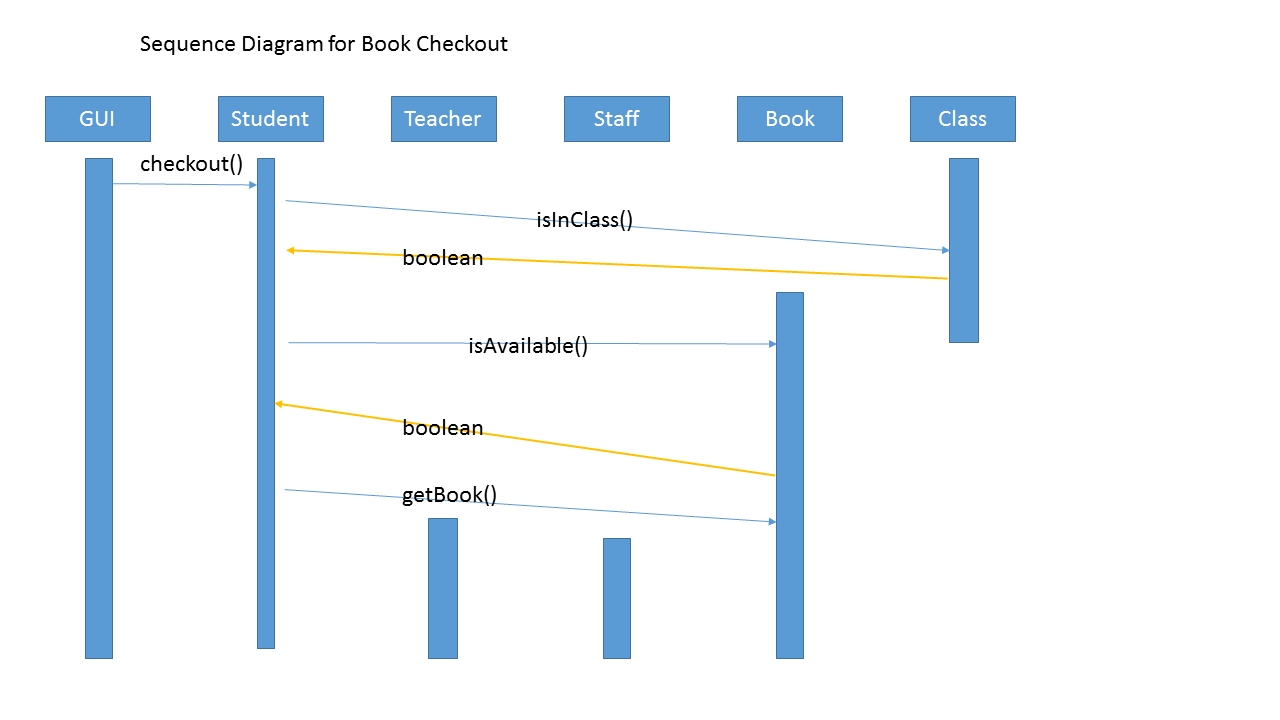
Input bookId and return the book.

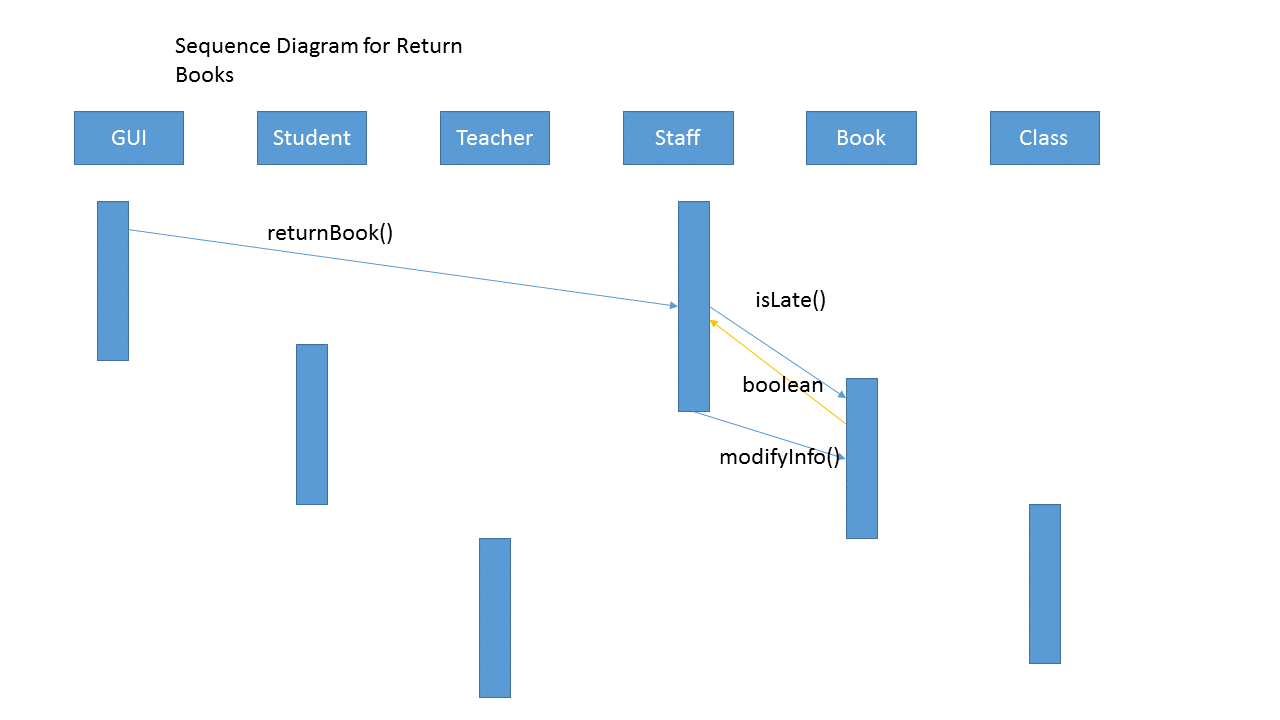
**Component 7 - Payment**

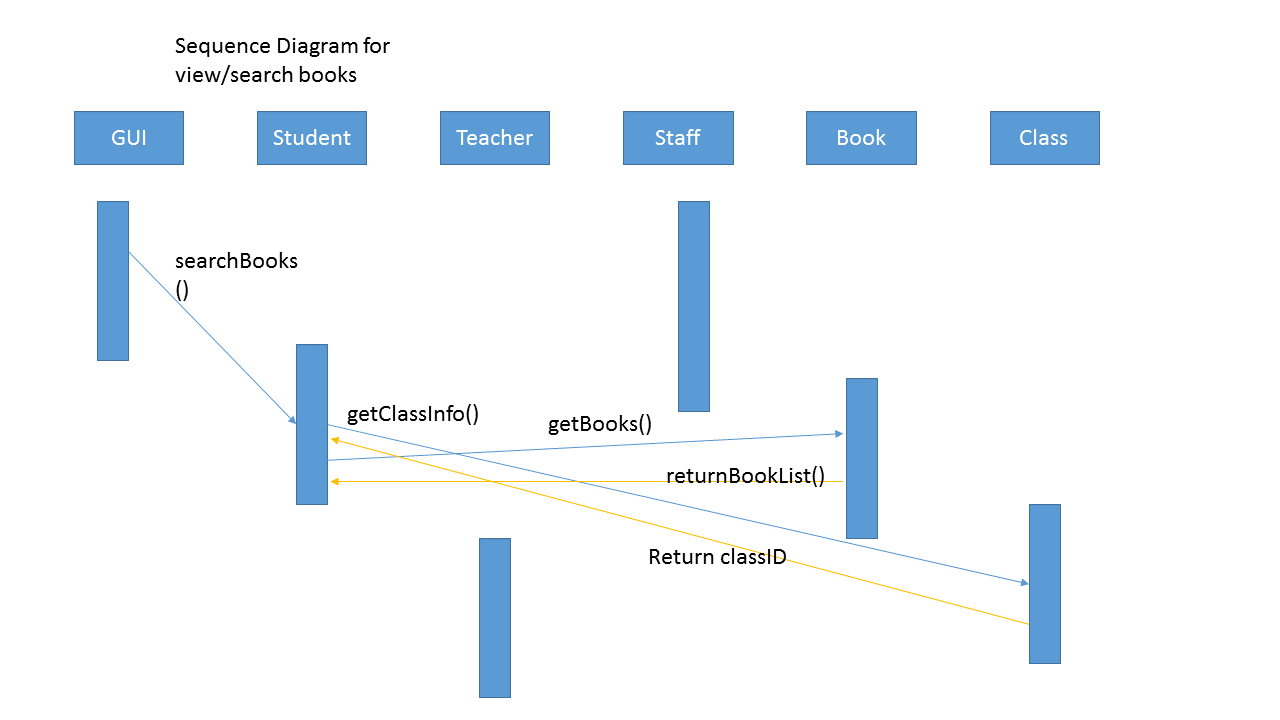
Input bookId, check the account credits and make the payement.

**3.3 Dynamic Behavior for Component**

Sequence Diagrams for different components:

****

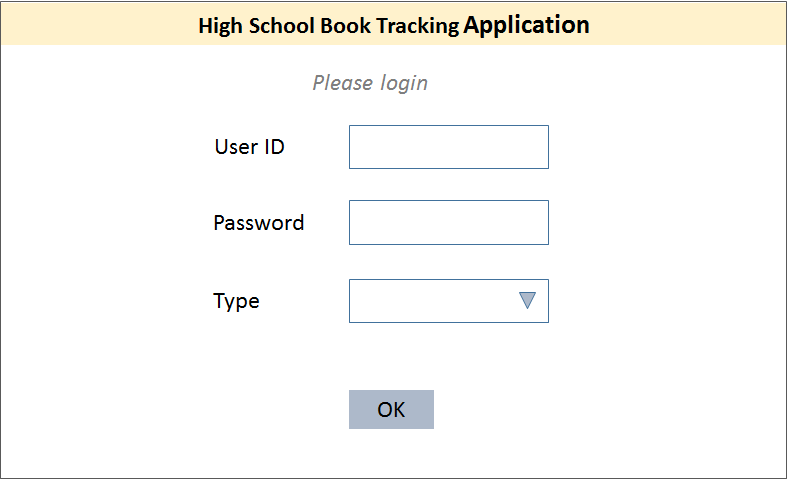
****

****

**4 User interface design**

In this section provides UI of the High School Book tracking application and details of how the application will actually look like when launched from a browser.

1.) Following screen will be dispalyed when launch the application. User can be of three types, namely 'teacher', 'student' and 'staff/admin'. Based on the type of the user, access level is found out which defines what information is to be shown or hidden in the application when user log in.



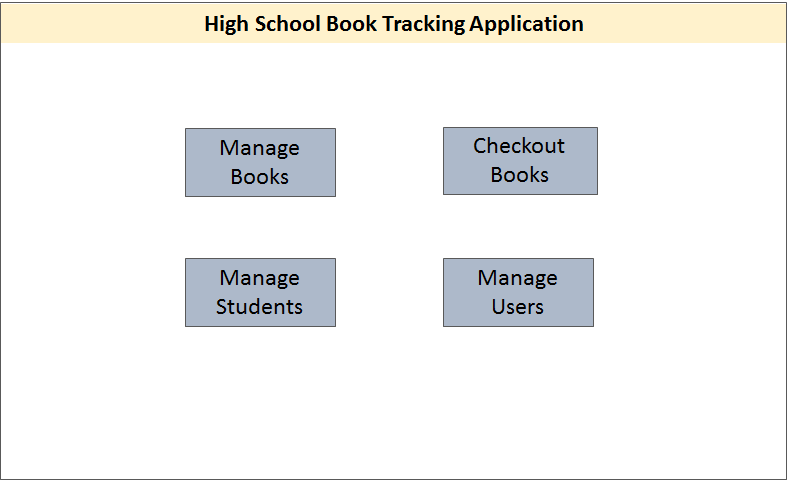
2.) Staff/Admin and teachers will see following screen when logged into the application. They will have three options.

Manage books option will allow them to view books, search books, add/delete books.

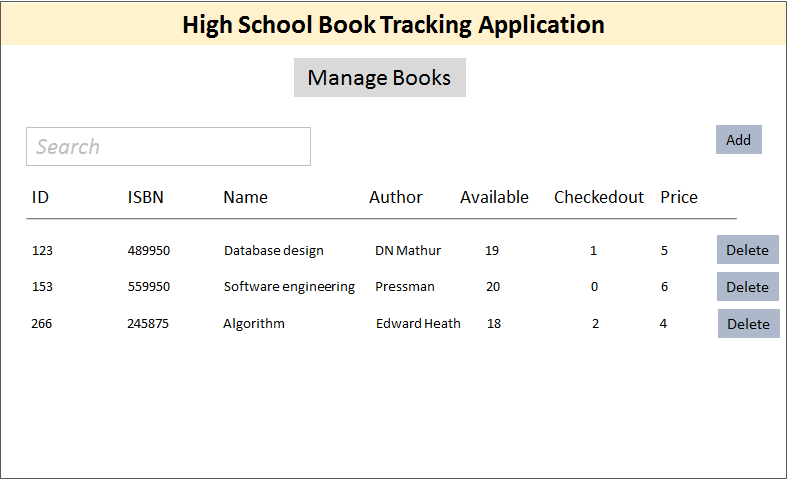
Checkout books will list all the books along with studentId of students who requested the books and a checkout option. It will also have the details of all the books already checkedout.

Manage students will provide an option to view/search all the student details and their due amount etc.

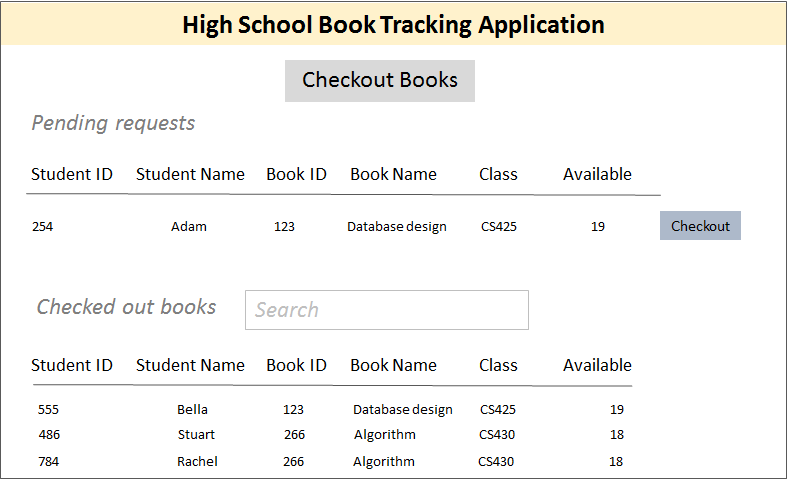
Manage users will display all the users' details with their access levels information(this option is only available for staff/admin users).



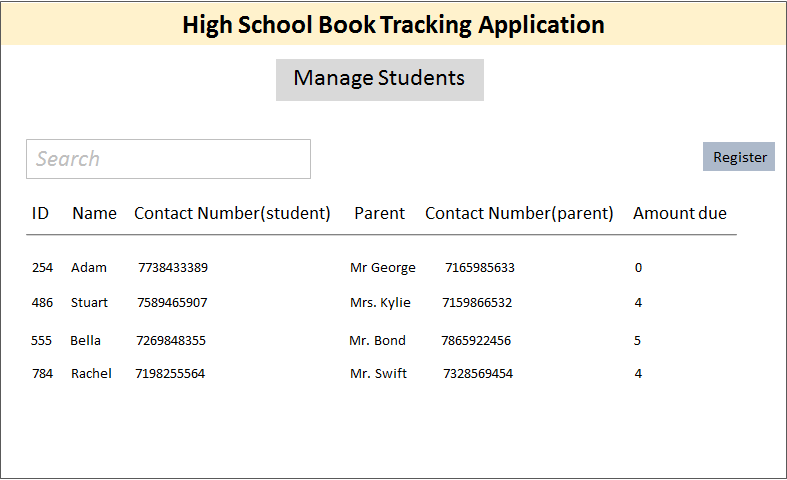
3.) When user clicks on Manage books, following screen will appear. Staff or teachers can view or search any book by its name, author, ISBN etc. They can also add or delete a particular book.



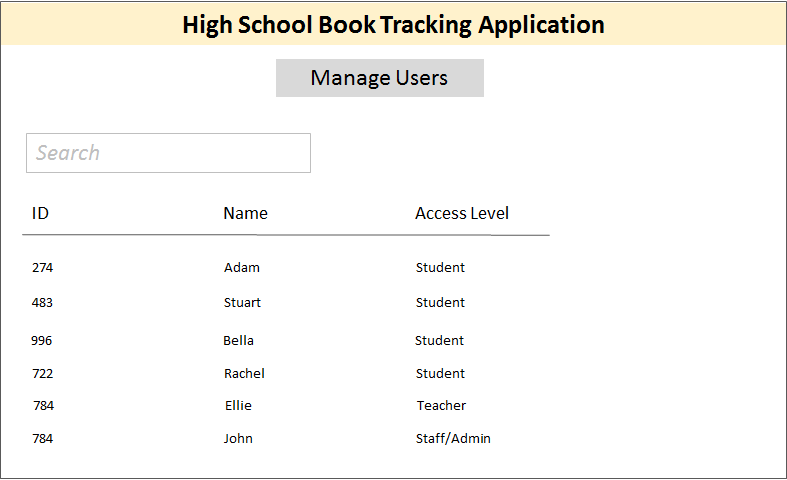
4.) If user clicks on Checkout books option on the home page, following screen will appear. Staff or teacher will be able to checkout books to students from this page. Also they can view details of all the checked out books.



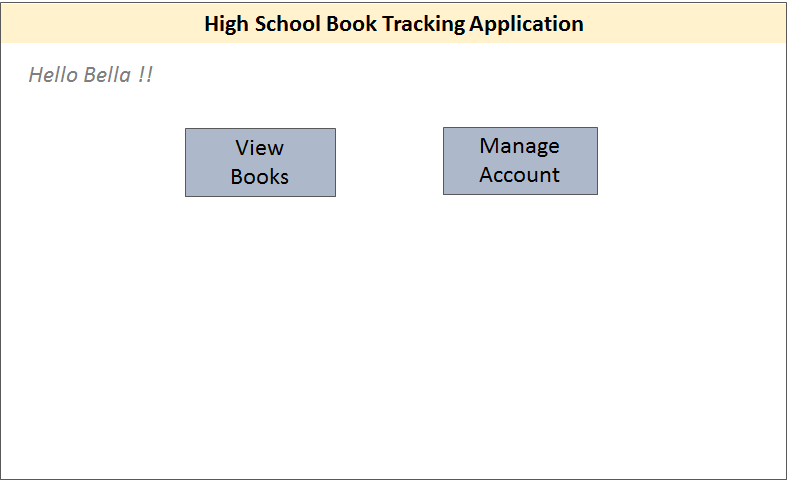
5.) If user clicks on Manage students option on the home page, following screen will appear. Staff or teachers can see details of all the students and their parent information on this page. They will also have an option to register a new student on this page.



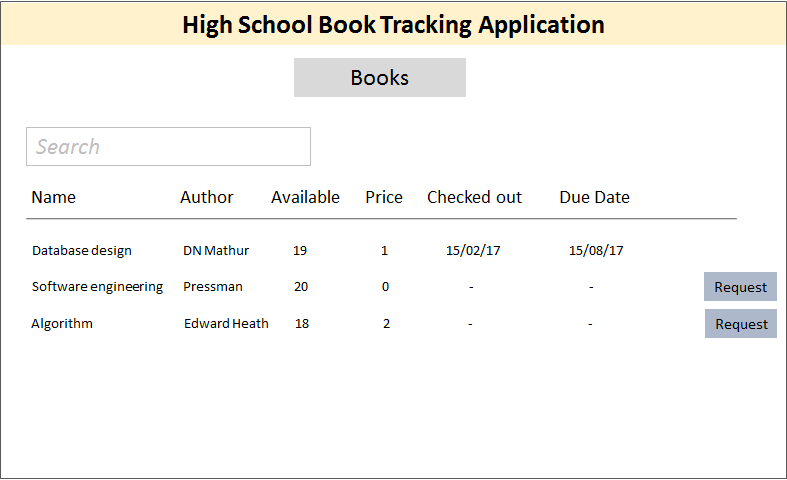
6.) If user clicks on Manage users option on the home page, following screen will appear. This option is only available for staff/admin users. Admin can see all the user details with their access levels on this page.



7.) If a student logs in to the application, the following home page will appear. Students can view books available in the library, request checkout books, view/update their account details.

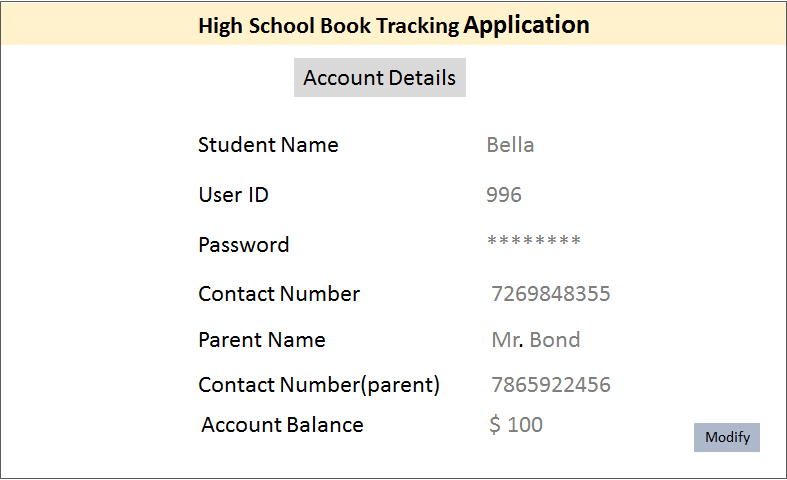


8.) If student clicks on View books option on the home page, following screen will appear. Student can see details of all the books available in the library as well as books that he/she has already checkedout.



9.) If student clicks on Manage account option on the home page, following screen will appear.

Here student can see his/her account details and modify if needed.



**5 Restrictions, limitations, and constraints**

This application has below mentioned restrictions :

* Student can checkout any number of books, also books of a particular class for which he is not registered.
* Staff/Admin can not modify book information.
* Teacher can not Add/Delete any books.
* Only a single book can be added or checked out at a time.
* Staff/Admin can not checkout a book for himself/herself.

**6 Testing Issues**

To excel in meeting needs our client we want our software to be well tested during all phases of software development lifecycle. We will cover most of the critical, major and minor scenarios while performing Unit testing. We will perform functional testing though automation testing using scripts as well as we will also perform manual testing for some core functionality of software. To maintain a good quality standard of product will also go through the integration and perform regular regression testing of our software. All these testing will happen in different phases to provide a good quality product being consistent, complete and highly available. We will conduct tests, where the test do not need to know the internal of the application (black-box testing).

**6.1 Classes of tests**

To develop a good software we want all our classes to be covering at least 99% unit test coverage. We will perform different types of testing to ensure good quality software as follows:

**Use case testing:** We will test every use case shown in previous sections. This will be done by having providing fake data into the database and then running a test suite for use case testing.

*Example:* 1) Test whether user can check out more books than registered classes. 2) Book search capability by book name or book number 3) List students with books checked *out*

**Security testing:** This testing will allow us to ensure our application abides by security standards.

*Example*: 1) While logging we need to determine the roles of user for providing authorization of a functionality. 2) Only admins can see information about every other user.

**Validation Testing:** We will perform upper and minimum bound validation testing on input parameter of our software.

*Example:* we will perform validation on username like it should have min of 7 and maximum of 25 characters.

**Book delivery testing :** Since the main goal of the application is to track books, we need to ensure books are being correctly tracked.

*Example*: 1) Before issuing a book a book needs to be available or else we will give an error message on screen. 2) We need to check the availability of book before issuing book to any user. 3) Adding a new book should reflect while performing search operation.

**Error Testing:** This is one of the most common testing performed while software testing, as the name itself suggest that while testing we will perform some error testing scenarios and test the software.

**State transition:** As in our software there are state involved, so we need to do the testing for transition of state of objects.

*Example*: As we will be handling payment functionality that requires our software to monitor the state of payment. 1) Check that when a fee has been payed.

Software should be able to determine late submission, which require its book state to move to late state.

**6.2 Expected software response**

As we want our software to be highly available and to perform high though put we will be using cutting – edge tools like jmeter, soapUI to measure the performance of software. Also we are taking extra step to meet the needs to our clients by adding additional measures like:

**Health Monitoring -** we will add a hook to our application server apache tomcat to monitor the health of server that will provide us the information about availability of our software.

While adding new feature to software it’s a normal trend that older features takes more than expected time than before. To solve this problem we will maintain Test suits that will keeps up updated on different feature response time.

**6.3 Performance bounds**

Few feature of our software are quite core to system and so it’s a good idea to keep a performance bound on theses functionality. As we are dealing with tier architecture system it’s hard to say the exact response time on each and every cases, however our core functionality performance bound are as follows.

|  |  |  |
| --- | --- | --- |
| **Functionality** | **Min Time** | **Max Time** |
| **Home Page** | Less than 0.5 sec | Not more than 1.5 sec |
| **Login Functionality** | Less than 1 sec | Not more than 2 sec |
| **Book Search (on index table data like book name)** | Less than 1.2 sec | Not more than 2 sec |
| **Book Search (on non-index table data like publication date )** | Less than 1.5 sec | Not more than 2.5 sec |
| **Fee Payment** | Less than 2 sec | Less than 2.8 sec |
| **Add Book** | Less than 1.5 sec | Not more than 2.3 sec |
| **Delete Book** | Less than 1.5 sec | Not More than 2.3 sec |
| **Return Book** | Less than 1.5 sec | Not More than 2.3 sec |

**6.4 Identification of critical components**

We identify the following as critical components of our library tracking system.

1. **Security:** We will test every use case where users’ login credentials and their access roles are properly defined before giving them their respective privileges. We will also try encrypting the data containing the login credentials.
2. **Transactions:** We will test the Fee-payment system thoroughly to ensure that the transactions involving the due amounts of the users are properly tracked and their accounts are duly updated after each transaction.

**References:**

1. <https://suspiciosum.files.wordpress.com/2012/12/design-specification.pdf>
2. <http://www.mhhe.com/engcs/compsci/pressman/graphics/Pressman5sepa/common/cs2/design.pdf>
3. https://cow.ceng.metu.edu.tr/Courses/download\_courseFile.php?id=3634