**Requirement 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 Usage Scenario**

**2.1 User Profiles**

All the users of this application should be granted access to the system before they can use it. Based on the access level, there are three types of users :

Staff/Admin

Staff/Admin should be able to do following functions :

* Add/Delete a book .
* Checkout/Release book to a student.
* Search books by its department, class, book name or ISBN number.
* List all authorized users and their access levels.
* List all students and parents information with amount due.

Teacher :

Teacher should be able to do following functions :

* Checkout/Release only those books for which he/she teaches classes for.
* List all the students of his class and their parent information with amount due.
* List all the books of which he/she teaches classes.

Student/Parent :

Student/Parent should be able to do following functions :

* List all the books that he/she has checked out.
* Add/Update personal information (Parents information, contact information).
* Search for books in the system.
* Check out books with a limit on the number of books equal to the number of registered classes for the student.

**2.2 Use-Cases**

Add/Delete a book

* Actor (online catalog) automatically adds books.
* User (admin) can explicitly add/delete books as well.

Checkout/Release book

* Users(staff/teachers) can checkout or release books and when they do so our applications modifies state.

Search books by its department, class, book name or ISBN number.

* Users can use search functionality of application to display this information.

List all authorized users and their access levels.

* Admins can use search functionality for this info.
* Our application also interacts with separate, external person database for this info.

List all students and parents information with amount due

* Users can collect this info only allowed to admin.

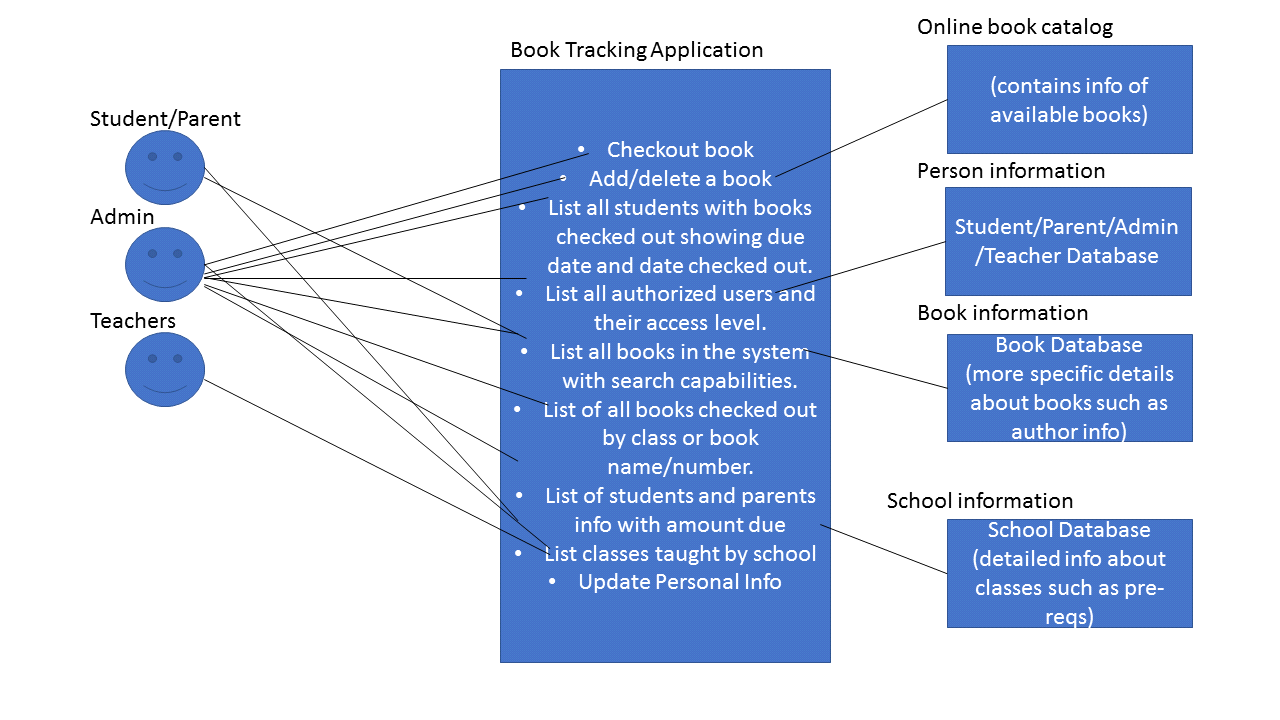
List all books in the system

* Any user can do this.
* Our application information interacts with an external book data base.

List students with books showing due data and data checked out

* Admin can gather this info.
* Teachers can for only their classes.

**2.3 Use-Case Diagram**



**2.4 Use-Case Descriptions**

* Description of Add/Delete a book : Book tracking application interacts with an external online catalog to collect data. However, since some books that are available online may not be available at a school, a admin can explicitly modify the set of books available at a school
* Description of Checkout book: Staffs/teachers with valid login credentials can check out/Release a book from the list of books that are available at the school.
* Description: Search books by its department, class, book name or ISBN number: The search bar should be able to filter by department, class, book name, and ISBN number.
* Description of List all authorized users and their access levels: The admin should be able to get this information.
* Description of List all students and parents information with amount due: The admin will be able to search and collect this information. The information will include name, id, student or parent, and the amount due in dollars.
* Description of List all books in the system: This includes all books in the school’s system. This will be a subset of the books available in the *books database and online catalog*.
* Description of List of students with books showing due date and date checked out: The information will include a book id which is an key into the book data base to gather more information such as author and quantity.

**2.5 Special Usage Considerations**

* Parents and students should be able to pay the fees and amount due information should be updated accordingly.
* As books are checked out or released, the total available and total checked out quantities should change for a book
* Security should be used when adding/updating/deleting information. For e.g. students cannot modify other student’s info.
* Admin should be able to grant access to students/parents for using the book tracking app.
* Books must be grouped by the class they are in, also when searching for a book.
* Books can belong to multiple classes and a class can contain multiple books.

**3 Data Model and Description**

Note: that the following attributes does not include information exclusive to database design (such as primary keys).

**3.1 Data objects**

A High School Book Tracking Application stores following information :

* Books :

It stores book information along with total available books and total checked out books. Every book is uniquely identified by an ISBN number.

Attributes :

**1. ISBN** - It is a unique ISBN number given to every book

**2. bookId** - It is a unique school generated ID

**3. bookName** - This attribute specifies name of the book

**4. author** - This attribute specifies author of the book

**5. price** - This attribute specifies price of the book

**6. totalAvailable** - This attribute gives count of total available books

**7. totalCheckedOut** - This attribute gives count of total checked out books

* Classes :

It stores details of all the classes conducted in high school. This information can be used to check which student is registered for which class or which teacher teaches which class.

Attributes :

**1. classId** - This attribute is the unique ID assigned to each class

**2. className** - This attribute specifies name of each class

**3. deptName** - This attribute specifies department name of each class

* Users :

It stores details of all the users who are going to use this application. All the users are uniquely identified by their user ID. Users can be of three types 'Staff/Admin', 'Teacher' and 'Student/Parent' and based on their type access level is granted to them.

Attributes :

**1. userId** - This attribute is the unique ID given to every user

**2. userName** - This attribute specifies name of every user

**3. accessLevel** - This attribute specifies the access level of every user

a.Staff/Admin  
 Attributes:  
 1. **staffId** - This attribute is the unique ID given to every staff  
 2. **staffName** - This attribute specifies name of every staff/admin

b.Teacher  
 Attributes:  
 1. **teacherId** - This attribute is the unique ID given to every teacher  
 2. **teacherName** - This attribute specifies name of every teacher

c.Student/Parent

Attributes:  
 1. **studentId** - This attribute is the unique ID given to every student  
 2. **studentName** - This attribute specifies name of every student

**3. studentContactNum -** This attribute specifies student's contact details

**4. parentName -** This attribute specifies name of every student's parent

**5. parentContactNum -** This attribute specifies student's parent's contact details

**6. amountDue** - This attribute specifies the due amount for every student

**3.2 Relationships**

* Belongs to

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

* CheckedOut

This relationship stores details of all the books checked out by students 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.

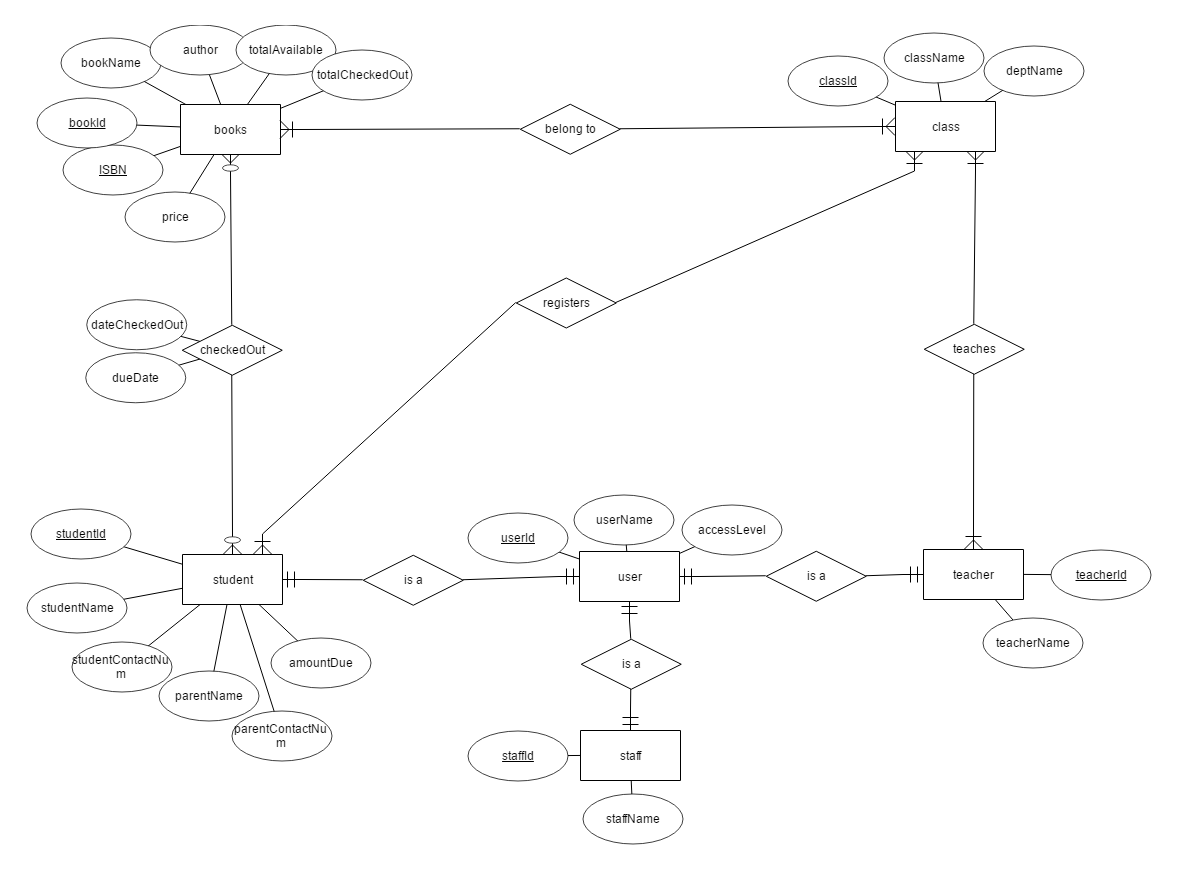
* Teaches

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

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

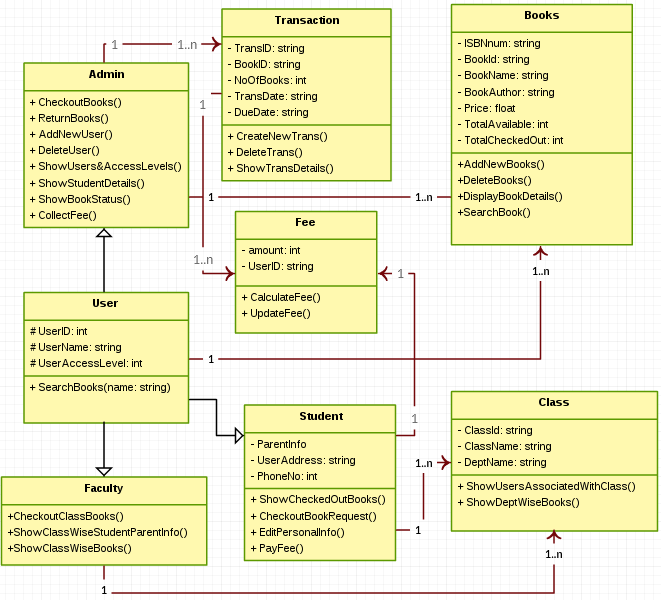
**3.3 Complete data model**

The features of High School Book Tracking Application is shown below in the form of ER-Diagram :



**4 Functional Model and Description**

**4.1 Class diagram**

****

**4.2 Software Interface Description**

**4.2.1 External Machine Interfaces**

* Front-end Software : NodeJs
* Back-end Software : Java
* Application Server : Apache Tomcat

**4.2.2 External System Interfaces**

* Platform : x86, amd64
* Memory: 512 MB or more,
* Hard Disk: 40 GB or more
* Operating Systems : Ubuntu, Windows, Mac
* Browser Support : Chrome and Mozilla

**4.2.3 Human Interfaces**

The software renders a good GUI for users. The system administrator can operate on the library system, performing the required task such as create, update, viewing the details of the book.

* It allows user to view quick reports like Book CheckedOut/Returned in between particular time.
* It provides a search facility based on different criteria.
* All the modules provided with the software must fit into this graphical user interface and accomplish to the standard defined.
* The design should be simple and all the different interfaces should follow a standard template.
* The user interface should be able to interact with the user management module and a part of the interface must be dedicated to the login/logout module.

**User Login Interface:**

New users can register with their personal information and create an account. The login window will prompt the user to type his credentials. If the user entered either his username or password incorrectly then an error message appears.

**Admin’s Control Panel:**

This control panel will allow an admin to add/remove users, edit book information and manage lending options.

**Search:**

A user can search and find book information. Also s/he can do advanced search with fields like book title, domain, author, etc. Admins can search for student and transactions.

**5 Behavioral Model and Description**

**5.1 Software Interface Description**

The book tracking application will mutate state when actors (both users and external data stores) are updated. Information related to books, classes, people (student/staff/teacher) can be updated. Since the goal of the application is to track books, the events that will trigger such updates are related to actions that would happen on books, such as check out, check in, pay fee, increment fee, add book, delete book, list books.

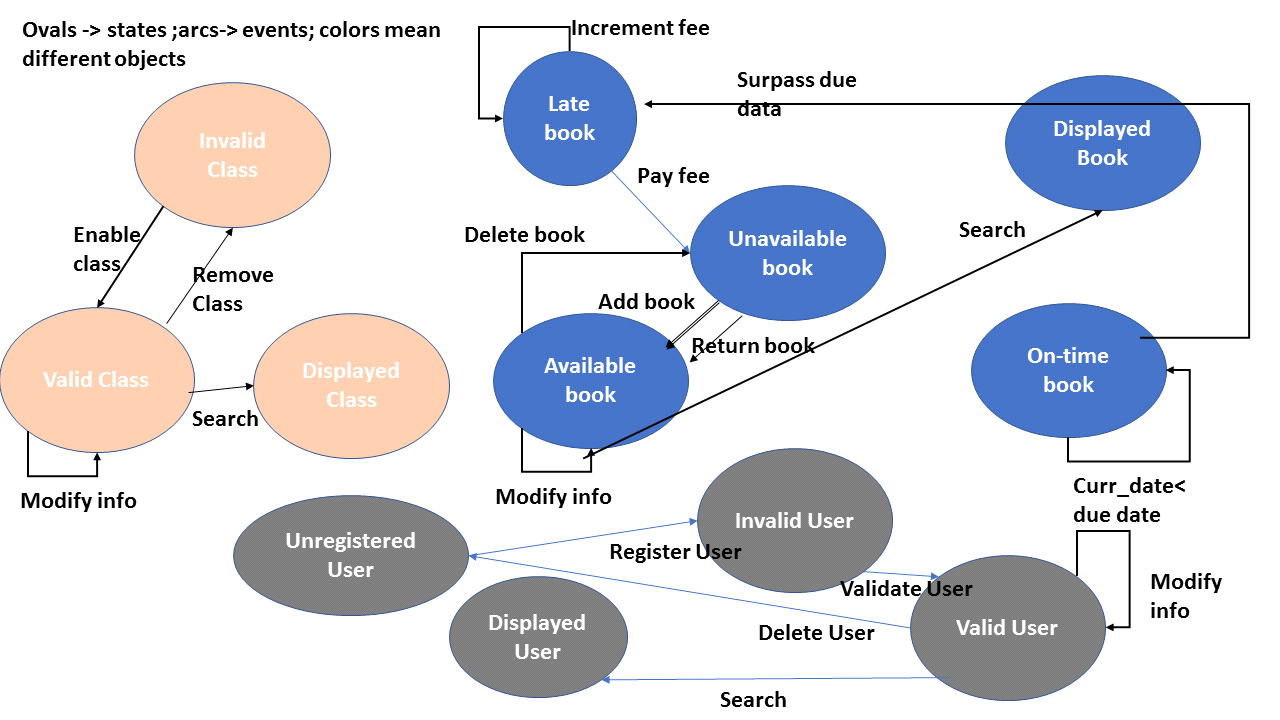
**5.1.2 Events**

* Register User
* Validate User
* Add book
* Pay fee
* Surpass due date
* Delete book
* Enable class
* Remove class
* Modify info
* Search
* Return book
* Update/Increment fee

**5.2.2 States**

* Late book
* Unavailable book
* Available book
* On-time book
* Invalid user
* Valid user
* Displayed user
* Displayed class
* Valid Class
* Invalid class
* Displayed book

**5.2 Statechart Diagram**



**6 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.

**7 Validation Criteria**

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).

**7.1 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.

**7.2 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.

**7.3 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 |

**References:**

1. <https://www.slideshare.net/Chetan2608/software-requirements-specification-of-library-management-system>
2. <http://www.cs.iit.edu/~oaldawud/CS487/project/requirement_specification_document_template.htm>
3. <http://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database>
4. <https://web.cs.dal.ca/~hawkey/3130/SEBackground4.pdf>