DESIGN SPECIFICATION DOCUMENT

For a Book tracking Application

*CS487 – Software Engineering*

The document provides information about the characteristics of the Book Tracking application project to set criteria the developers will need to meet.

TEAM:

- Praveen Kumar Selvan

- Vijay Bharathi

- Dayalini Nagaraj

- Parinitha Hirehal

1.INTRODUCTION:

The document describes the software design specifications for a book tracking application called Book-e-Tracker. This tool/application mainly maintains a small repository of school books and tracks the books, classes, staff and students who check out the books.

# 1.1 Goals and Objectives

Book-e-tracker is a web application built using node-js and MySQL to aid its student users and staff users to manage their book checkouts and returns by simplifying the whole process.

1.2 Statement of Scope:

The Book-e-Tracker is an application which intends to automate the process of tracking books. Basically, it is a web application built using node js to provide an interface for users to perform various operations. The system’s scope is mainly to provide the users/actors the ability to perform the below functionalities:

The student will be able to:

* Register for the application
* Login to the application
* Register for courses
* Search for books
* Checkout the books -Return the books
* Extend the due date
* Logout

The staff will be able to:

* Register for the application
* Login to the application
* Search for the books
* Checkout the books - Return the books
* Extend the due date
* Authorize the book request
* Validate the book requests

The Admin will be able to:

* Provide different level of access to different users
* Maintain the book stack
* Add books
* Update the Books
* Delete books
* Add Users Information
* Delete Users Information
* Update Users Information
* Place order for the books
* Cancel orders
* Pay for the books

REQUIREMENTS:

The below table mentions the requirements to implement in our application Book-e-Tracker at a very high level.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| Requirement No | Description | Priority | Tested |
| 1 | The Repository of School books shall be maintained | High | No |
| 2 | The student shall check out Books (By Students-Limited by number of courses taken) | High | No |
| 3 | The staff shall check out Books (By Teacher- Limited to his/her department) | High | No |
| 4 | The Student/Staff shall return Books | High | No |
| 5 | The Admin Shall Add/Update/Delete the Book Information | High | No |
| 6 | The Admin shall Add/Update/Delete the Student Information | High | No |
| 7 | The Admin shall Add/Update/Delete the Course Information | High | No |
| 8 | ‘Search Book’ option for the application shall be enabled | Desired | No |
| 9 | The application shall track the Books | High | No |
| 10 | The application shall track the classes | High | No |
| 11 | The application shall track the students | High | No |
| 12 | The application shall track the total Number of books | High | No |
| 13 | The application shall track the total Number of books checked out | High | No |
| 14 | Database: MYSQL Server shall be implemented in the application | Desired | No |
| 15 | The application shall request for the ID and password | High | No |
| 16 | Security (Access based on users) aspects shall be implemented in the application | High | No |
| 17 | Response Time shall be implemented for 5 Seconds for the application | High | No |
| 18 | Reports (List of all students who checked out books with due date and checked out date) shall be generated. | High | No |
| 19 | Reports(List of all authorized Users with access level) shall be generated. | High | No |
| 20 | Reports (List of all books in system with search feature) shall be generated. | High | No |
| 21 | Reports(List of all orders by order type) shall be generated. | High | No |
| 22 | Reports (List of all students and parents with due amount) shall be generated. | High | No |
| 23 | Reports (List of books checked out by Class/Book name/Book number) shall be generated. | High | No |
| 24 | Notifications shall be enabled | Desired | No |
| 25 | Autogenerate Fields: Book ID | High | No |
| 26 | Autogenerate Fields: Student ID | High | No |
| 27 | Autogenerate Fields: Staff ID | High | No |
| 28 | Student log details shall be recorded. | Desired | No |
| 29 | Time Stamp(For Check out and return) shall be recorded. | High | No |
| 30 | Pickup confirmation Number | High | No |
| 31 | Online Payment for due amount for admin shall be recorded. | Desired | No |
| 32 | Order Types shall be uploaded | High | No |
| 33 | Order Confirmation Number shall be enabled for the application | High | No |

Requirements Table: (Functional and Non-functional Requirements)

# 1.3 Software Context

As mentioned earlier, the Book-e-Tracker is a web application providing the interface for book transactions.

The Application contains a LOGIN SCREEN prompting for a Student ID/ Staff ID and the password. The Login screen also contains the ‘register’ option for the first-time users.

Based on the Kind of a login, the options appear in the next screen. Suppose if the user is a student, then a STUDENT HOME SCREEN appears where s/he will be allowed to search, checkout and return books. If the user is a staff, STAFF HOME SCREEN appears, where s/he can authorize for the requests of books and validate the requests from students. And they can also search for books, checkout the books and return the books.

The Checkout Screen will be used for displaying the item's title, description and selected quantity.

The admin has all the privileges open for him/her. The ADMIN HOME SCREEN will have an option to add/update/delete the book information and the user information.

A PAYMENT SCREEN will be displayed for the admin to pay for the orders of books.

(Note: We will implement many more screen as we progress).

# 1.4Constraints

* The application should be compatible on all systems and browsers
* The application should be able to handle multiple connections simultaneously

# 2.0 Data Design

At a high level, we have three major layers in the application

1. Front End: This is developed using HTML, CSS and JavaScript. Front End serves as the input / output layer. It accesses Form elements and sends to mid-tier and displays the reports or responses from server.
2. Backend: Our backend is a database in MySQL that contains tables for representing books, users, courses, orders, and relationships among them.
3. Mid–Tier Layer: This is the server which runs in node js. Mid-Tier layer gets the request from frontend and queries the database to create and return results. This layer is made up of multiple modules that performs various functionalities like checkout book, search book, create reports, etc.

# 2.1 Internal Software Data Structure

The input from user (Front End) is sent to the mid-tier layer as a query string parameter or as a POST variable. The Mid-Tier layer stores these input parameters as a local variable for creating queries to database. All the results from running queries on database are stored in local variables while creating the HTML output and then discarded after use.

# 2.2 Global Software Data Structure

We have a global connection pool object which establishes connections to the database using node.js driver for MySQL. This is done by creating a DB wrapper which creates connection pool to the database. Every module in the mid-tier layer which requires to query the database will require a connection from the DB wrapper and release it after use.

# 2.3 Temporary Data Structure

To maintain the login state of the user – we use a temporary session object to verify the authentication of the user after logging in. The session object will create a session cookie that authenticates the user and will be sent on each request. Before routing to the page – application will validate if the cookie is present else redirect to the login page. The cookie will be deleted from browser when user logs out or when he closes the browser.

# 2.4 Data Base

The following tables will be created in the data base to store information about books, users, orders and courses and the relationships between them. The tables are created in MySQL database.

1. **Books Table:**

Table that contains information about the books

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Datatype** | **Size** | **Nullable** | **Constraint** |
| School\_Book\_ID | Integer | 15 | No | Primary Key |
| Book\_Name | Varchar | 30 | No |  |
| ISBN | Integer | 16 | No |  |

1. **Users Table:**

Table that contains details of persons with a valid username and password who can login to the system

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FieldName | Datatype | Size | Nullable | Constraint |
| User\_Name | Varchar | 15 | No | PrimaryKey |
| Password | Varchar | 15 | No |  |
| Name | Varchar | 15 | No |  |
| Access\_level | Varchar | 10 | No |  |

1. **Student Table:**

Table that contains all student information

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FieldName | Datatype | Size | Nullable | Constraint |
| Student\_Id | Integer | 10 | No | Primary Key |
| Parent\_Name | Varchar | 20 |  |  |

1. **Teachers Table:**

Table that contains all teacher information

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FieldName | Datatype | Size | Nullable | Constraint |
| Staff\_Id | Integer | 10 | No | Primary Key |

1. **Admin table**

Table that contains information about admin

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FieldName | Datatype | Size | Nullable | Constraint |
| Admin\_Id | Integer | 10 | No | Primary Key |

1. **Courses Table:**

Table containing all course information

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FieldName | Datatype | Size | Nullable | Constraint |
| Course\_Id | Integer | 10 | No | Primary Key |
| Course\_Name | Varchar | 20 |  |  |

1. **Order Table:**

Table containing order information

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FieldName | Datatype | Size | Nullable | Constraint |
| Order\_Id | Integer | 10 | No | Primary Key |
| Order\_Type | Varchar | 20 |  |  |
| Order\_Description | Varchar | 20 |  |  |

**Association Tables:**

1. **Placed by Table:**

Table describing a relationship between admin and order

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FieldName | Datatype | Size | Nullable | Constraint |
| Order\_Id | Integer | 10 | No | Foreign key on orders table  Part of Primary Key |
| Admin\_Id | Integer | 10 | No | Foreign key on admin table  Part of primary key |
| Placed\_Date | Date |  |  |  |
| Order\_Status | Varchar | 20 |  |  |

1. **Borrowed By Table:**

Table describing relationship between user and book

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FieldName | Datatype | Size | Nullable | Constraint |
| User\_Name | varchar | 15 | No | Foreign Key on User Table  Part of Primary Key |
| School\_Book\_ID | Integer | 15 | No | Foreign Key on Books Table  Part of Primary Key |
| Amount\_Due | Integer |  |  |  |
| Due\_Date | Date |  |  |  |
| Return\_Date | Date |  |  |  |
| Checkout\_Date | Date |  |  |  |

1. **Enrolls In Table**

Table describing relationship between student and course

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FieldName | Datatype | Size | Nullable | Constraint |
| Course\_Id | Integer | 10 | No | Foreign Key on Courses Table  Part of Primary Key |
| Student\_Id | Integer | 10 | No | Foreign Key on Students Table  Part of Primary Key |

1. **Teaches Table**

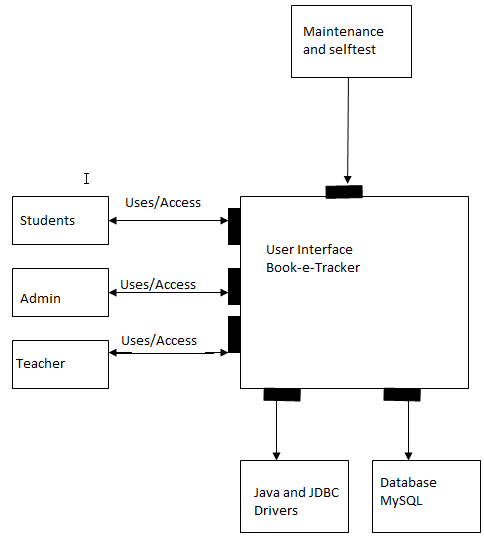
Table describing relationship between teacher and course

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FieldName | Datatype | Size | Nullable | Constraint |
| Course\_Id | Integer | 10 | No | Foreign Key on Courses Table  Part of Primary Key |
| Staff\_Id | Integer | 10 | No | Foreign Key on Staff Table  Part of Primary Key |

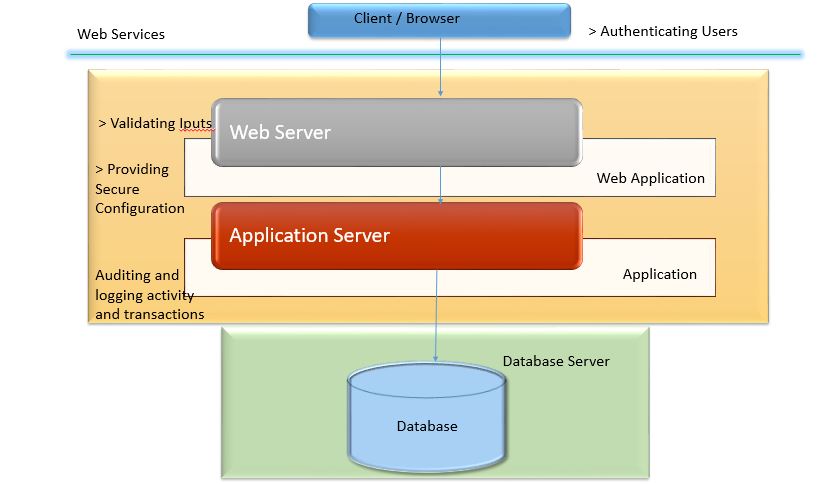
3.0 ARCHITECTURAL AND COMPONENT LEVEL DESIGN

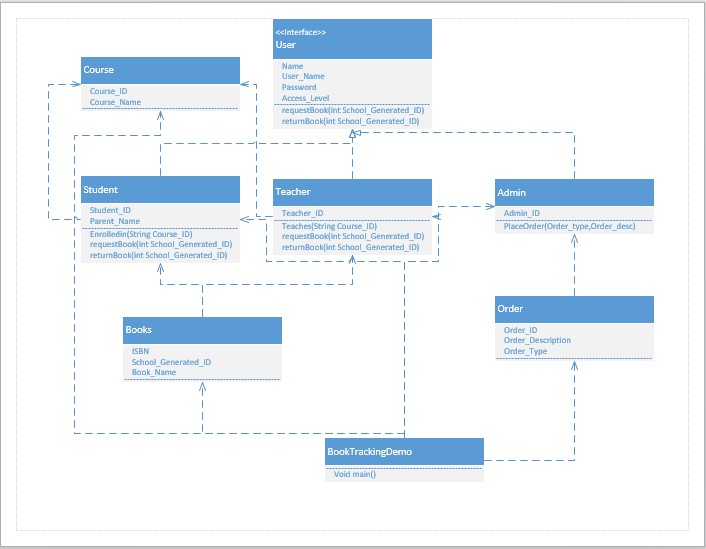
3.1 Architectural Diagram

The Architectural Context diagram for the software is shown below. As it is clear from the diagram, various actors are admin, students and teacher, which use the system. The subordinate systems that will be used by the software are the database (MySQL) and Java. Also JDBC, ODBC driver is required to connect java to database. Maintenance system is super ordinate, which needs the software for testing and maintenance purposes.

  
  
**Architectural Context Diagram**

There are various types of architectures to represent the context. Basically the architectures can be divided as system architecture and application architectures. The system architecture is as follows:

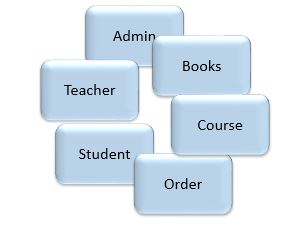


There are many ways to represent application architectures such as call and return, pipes and filters, Object oriented architecture etc. In our case call and return architecture and Object oriented architecture are shown. However, the main emphasis is on object oriented architecture. Firstly, Call and Return architecture is drawn .In this, there are various functions which further call the application functions. These functions then return the required data. The call and return architecture for the Book-e-Tracker Application system is shown below.  
  
The Call and return architecture has a main program called the “BookTrackingDemo”, which controls all others. Basically, it selects the functions described in components below it. The user program calls External communication management, which further calls GUI implemented in project. Now depending on the access levels, it calls the various functions. And these functions return some values that are data.

Now from the above figure, each class becomes a component and it interacts (interfaces) with each other. The interfaces are described in more detail in the next section.

3.2 Description of Components

We totally have 6 below mentioned components. A detailed description of each software component contained within the architecture is presented.



***Component ADMIN and Interface description of the component Admin***

* **Processing narrative of component ADMIN**

The component Admin contains the class Admin. It contains the functions, which sets all the related data into and out of the database. The responsibilities of this component is to declaring variables of the basic data of the Admin and implementing functions which mainly is to place order and interacts with the database.

**Algorithmic description of component Admin**  
  
Component Admin:  
Admin will have a Admin ID and functions to interact with the database.Admin will be directed to the Management Screen where they will be able to add a new user, accept/deny Orders (Book request), add/edit book details.

The below algorithm shows the interaction of Admin component with the Order class.

Start  
Declare variables of the basic data  
Set the basic data of the Admin component in the database

Get the data from the database  
Execute the function Place Order  
End

* **Design class hierarchy**

OrderAdmin

***Component TEACHER and Interface description of the component Teacher***

* **Processing narrative of component Teacher**

The component Teacher contains the class Teacher. It contains the functions, which retrieves all the related data into and out of the database via admin. Teacher will be directed to the default Home screen which has the following options View available/search Books, My Orders, My Books Tab.

* **Algorithmic description of component:**  
    
  *Component Teacher:*  
  Teacher will have a Teacher ID and functions to interact with the Admin. The Teacher will be able to request the books, return the books The below algorithm shows the interaction of Teacher component with the Admin class.

Start  
Declare variables of the basic data  
Execute the request book and return book calls to admin.  
Get the data from the database via admin class  
End

* **Design class hierarchy**

Admin

***Component STUDENT and Interface description of the component STUDENT***

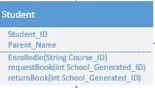
* **Processing narrative of component:**

The component STUDENT contains the class STUDENT. It contains the functions, which retrieves all the related data into and out of the database via admin . STUDENT will be directed to the default Home screen which has the following options View available/search Books, My Orders, My Books Tab.

* **Algorithmic description of component:**  
    
  Component STUDENT:

STUDENT will have a STUDENT ID and functions to interact with the Admin. The STUDENT will be able to request the books, return the books. The below algorithm shows the interaction of Teacher component with the Admin class.

Start  
Declare variables of the basic data  
Execute the request book and return book calls to admin.  
Get the data from the database via admin class  
End

* **Design class hierarchy**  
  Admin

***Component BOOKS and Interface description of the component BOOKS***

* **Processing narrative of component:**

The component BOOKS contains the class BOOKST. It contains the request related data into and out of the database via admin . Books displays all the books the user has checked out with due date and an option to return book.

* **Algorithmic description of component:**

*Component Books:*

* Displays all the books the user has checked out with due date and an option to return book
* Start
* Display all the books the user has checked out with due date and option to return.
* End
* **Design class hierarchy**

Order

***Component ORDER and Interface description of the component ORDER***

* **Processing narrative of component:**

The component ORDER contains the class ORDER. It contains the requests related data into and out of the database via admin. On request, book is added to ‘My Orders’ Tab where user can track the status of his request.

* **Algorithmic description of component:**  
    
  *Component Order*:  
  Stores and processes all the orders by Users.
* Start
* Process the Requests
* End
* **Design class hierarchy**

AdminOrder

***Component COURSE and Interface description of the component COURSE***

* **Processing narrative of component COURSE**

The component COURSE contains the class COURSE. Ot contains all the data related to te course and stores the list of books related to the course.

* **Algorithmic description of component Course**  
    
  *Component Course:*  
  Stores all the course information, student information and books information
* Start
* Store the student information
* Store the books information
* Display the information when requested
* End
* **Design class hierarchy**

Course

3.3 Dynamic Behavior For Component N

**3.3.1 Interaction Diagrams**

The scenario mentions the sequence of interaction among various actors for Requesting a book by Teacher:



The scenario mentions the sequence of interaction among various actors for Requesting a book by the Student:



The scenario mentions the sequence of interaction among various actors for Returning a book by the Student:



4.0 User interface design

This section presents the graphical user interface of the Book-e-Tracker web application that will have the form of a lightweight web application, where all functionalities are grouped visually and logically into thematic units, according to the types of entities they are associated with. The design provides for a responsive site that will work on both desktop and mobile tablet devices. The sections that follow contain mock views of a selection of the most important aspects of the GUI, along with textual descriptions of their purpose and contents.

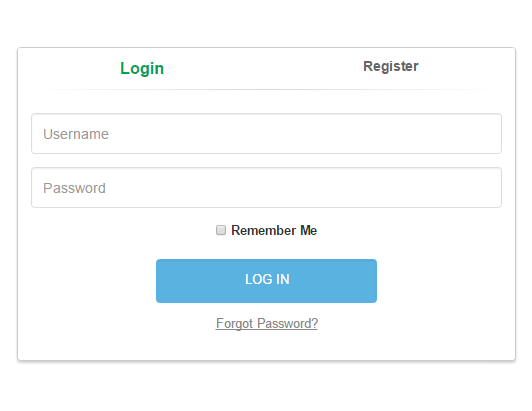
# 4.1 Description of the user interface

From “Index” page user can reach “Login Page” page. On successful login, user will have access to the following pages “Home”, “Search Books”, “Borrowed Books”,” Wish List”, “Holds”, “Setting”, “Administration”, “Faculty and Department”, “Process Requests”, “Report” “Inventory”, “Contact”, “Help”. All these pages cover necessary functionality of system. It is easy to navigate between these pages. Note: The users will not have access to all pages, pages available to the user depends on the user’s access level.

# 4.1.1 Screen images

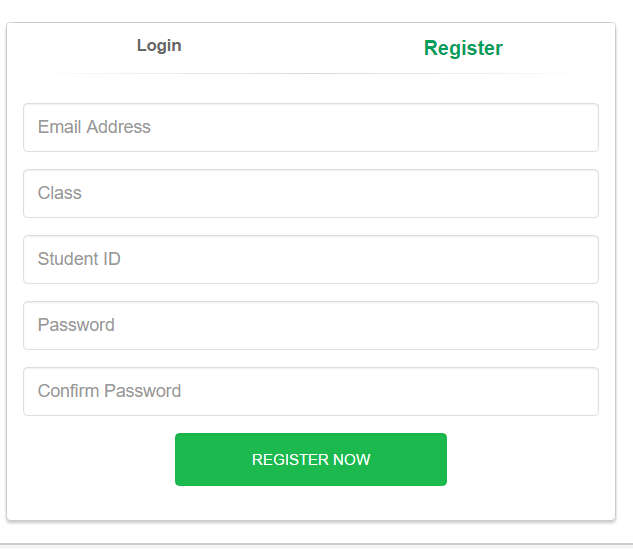
The GUI for the Book-e-Tracker system has been kept quite simple and follows a conventional style. Its very user friendly and do not require any training what so ever to work on.  
Below is a detailed images and description of the UI. The object and actions performed when you click them are also identified.

**Login Screen**:

Each user has and username and password. After entering username and password you click on login and the program verifies the information entered. Upon verification, the program either allows you to log in or a message will be displayed that the login was not verified.  
  
The program also checks whether the user is the Admin, Professor or Student. Depending upon this information the program gives the respective interfaces there by restricting user to perform any other task except that can be performed by him/her.

**Registration Screen:**

User (Students) also have the option to register, wherein the student is required to enter his Email Address, Class and Student ID. Upon Validation of the credentials student account is created.



**Index Page:**

Index page has descriptive characters; it contains a list of main system’s functionality and contact information. After login “Index” page changes to Home Page and guides user on how to work with system.



**Home Page:**

Home Page for each user(Student, Professor, Admin) varies and displays the respective feature/functions available to them.



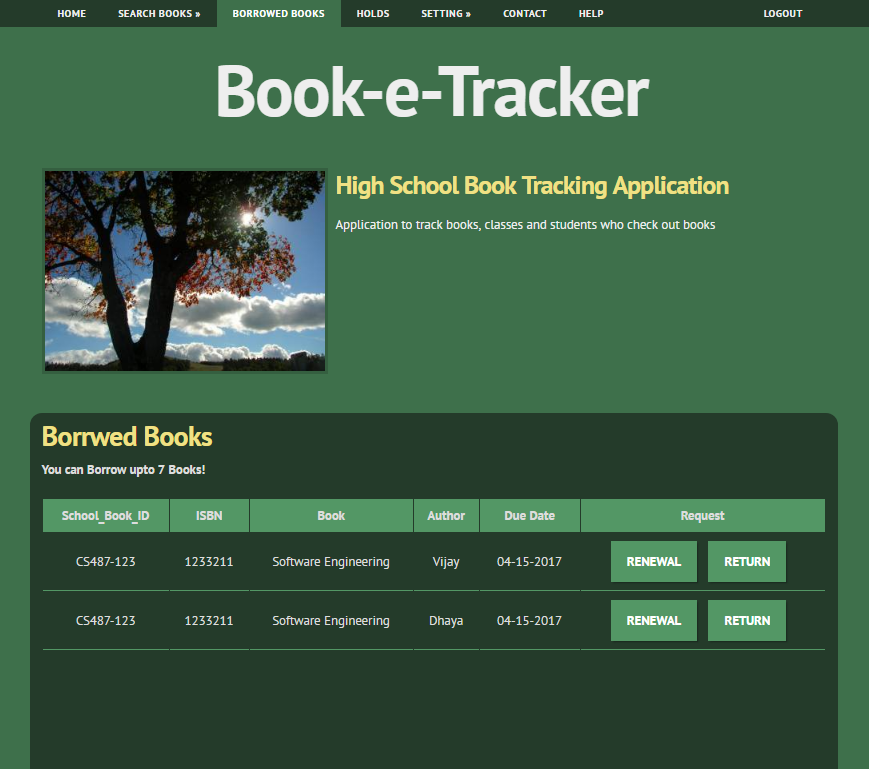
**Request Books:**

Users can add books to wish list (like a cart), and then initiate a request for a book or remove books from the wish list. Once approved by the Teacher or Admin the book is available to the user

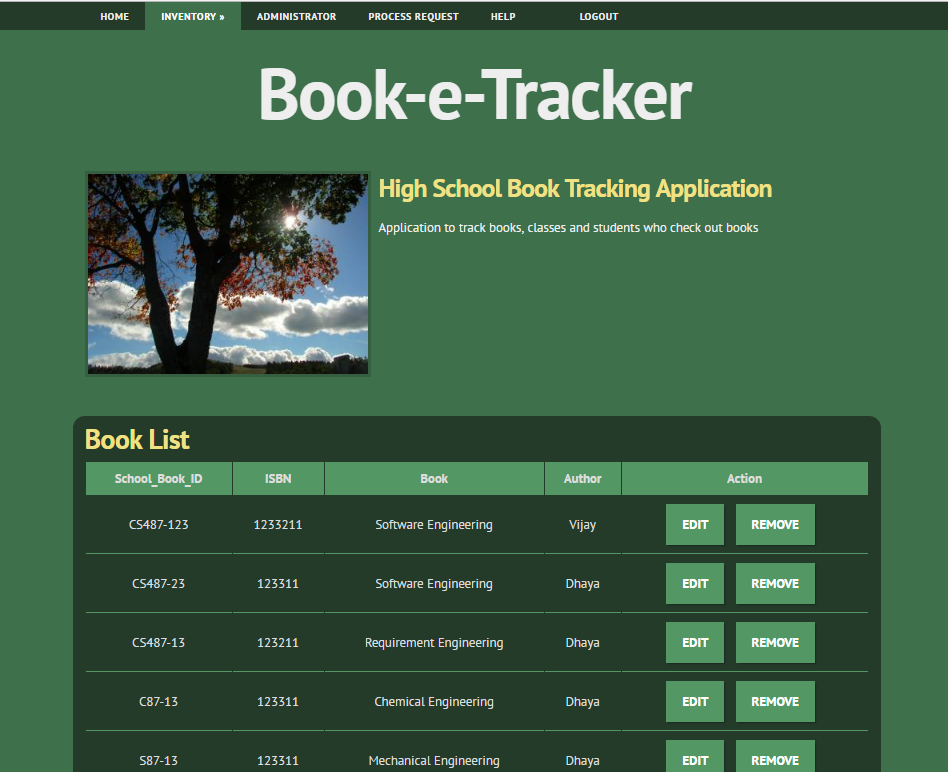
# 

**Borrowed Books:**

User can view the books they have borrowed under the Borrowed Books Tab. Each user can borrow upto 7 books. Details of the books borrowed and the due date is listed. Users have the option to either renew the book or return the book back.



# Admin:

 Inventory page is available only to the Admin, admin has the ability to add, edit or remove books from the Inventory.

# 4.1.2 Objects and actions

The above shown screenshots of the web application consist of the following components that holds the website together. **Front End Elements** (HMTL, CSS, JS)-Navigation Structure, Page layout, Logo, Images, Contents, Graphic Design. **Back End Elements**(Node-JS): Cart, Site Search, Contact forms, Security, Multimedia. All the Components mentioned above are not implemented are in Stages of Development

# 4.2 Interface design rules

To improve the usability of an application it is important to have a well-designed interface. These “Eight Golden Rules of Interface Design" are a guide to good interaction design.   
  
**1) Strive for consistency.**Consistent sequences of actions should be required in similar situations; identical terminology should be used in prompts, menus, and help screens; and consistent commands should be employed throughout.   
  
**2) Enable frequent users to use shortcuts.**As the frequency of use increases, so do the user's desires to reduce the number of interactions and to increase the pace of interaction. Abbreviations function keys, hidden commands, and macro facilities are very helpful to an expert user.   
  
**3) Offer informative feedback.**For every operator action, there should be some system feedback. For frequent and minor actions, the response can be modest, while for infrequent and major actions, the response should be more substantial.   
  
**4) Design dialog to yield closure.**Sequences of actions should be organized into groups with a beginning, middle, and end. The informative feedback at the completion of a group of actions gives the operators the satisfaction of accomplishment, a sense of relief, the signal to drop contingency plans and options from their minds, and an indication that the way is clear to prepare for the next group of actions.   
  
**5) Offer simple error handling.**As much as possible, design the system so the user cannot make a serious error. If an error is made, the system should be able to detect the error and offer simple, comprehensible mechanisms for handling the error.   
  
**6) Permit easy reversal of actions.**This feature relieves anxiety, since the user knows that errors can be undone; it thus encourages exploration of unfamiliar options. The units of reversibility may be a single action, a data entry, or a complete group of actions.   
  
**7) Support internal locus of control**.  
Experienced operators strongly desire the sense that they are in charge of the system and that the system responds to their actions. Design the system to make users the initiators of actions rather than the responders. 

**8) Reduce short-term memory load.**  
The limitation of human information processing in short-term memory requires that displays be kept simple, multiple page displays be consolidated, window-motion frequency be reduced, and sufficient training time be allotted for codes, mnemonics, and sequences of actions.

# 4.3 Components available

Book-e-Tracker is a web application it has the following three primary components

1. **Application** – Web Application Component, Enterprise Application Component
2. **Database** – MySQL Database
3. **OtherComponents**- HTTP Listener, Web browser

# 4.4 UIDS description

No user interface development system has been used in the development of the Book-E-Tracker system.

5.0 Restrictions, limitations, and constraints

The below mentioned items explains the cases which impact the specification, design, or implementation of the software system. These items may also contribute to restrict the scalability and performance of the system as well.

Constraints:

* The users must have their correct usernames and passwords to enter the Application.
* Only the Admin will be able to approve the order of books
* The admin will be able to update and modify the database based on the availability of books

Assumptions and dependencies

* The users have sufficient knowledge of computers.
* The computer should have the product installed correctly.
* The users know the English language, as the user interface will be provided in English
* The software uses SQL as database. The system should be connected with the server to access the database.
* The processing speed of the operations should be high.

6.0 Testing Issues

Software testing is an important element of software utility assurance and represents the ultimate review of specification, design and coding. The increasing visibility of the software as a system element and the costs associated with project failure are motivating forces for well planned, through testing.

Though the test phases is often thought of as separate and distinct from the development effort first develop and then test, Testing is a concurrent process that provides valuable information for the development team.

In general, these properties indicate the extent to which the component or system under test:

* Meets the client requirements and design
* Responds correctly to all types of input
* Function performance and time taken to execute it.
* User friendly
* Can be installed and run in its intended [environments](https://en.wikipedia.org/wiki/Operating_environment), and
* Achieves the general result its stake holders desire.

6.1 Classes of tests

Types of tests to be conducted:

* Unit Testing
* Integration Testing
* Functional Testing
* System Testing
* White Box Testing
* Black Box Testing
* Performance Testing

**Unit Testing:**

Unit testing is a software development process in which the small parts called units of an application, are individually tested by the developer. The update, delete, view , UI elements are tested by the developers.

**Functional Testing:**

The functionality and the features of the application are tested in this phase. All the features mentioned in the requirement has to be implemented and validated. According to our application, add, delete, update, search, view, order, payment, login and logout are the main functionalities that has to be tested for validation.

**White Box Testing:**

Test the software based on the internal structures and working of the application, as opposed to its functionality.

**Black Box Testing:**

The whole application is tested based on GUI, Design, Features and flow of the application and validated based on that.

6.2 Expected software response

Based on each scenarios and modules, test cases will be generated and analysis will be done. If the actual results are different from the expected then we need to go through the software code and fix the error.

It also includes UI specifications, Data to be displayed and the performance of the software based on the response time.

**Function and Expected Results:**

* + **Log In and Logout**

**Case1**: If the user is already a member then the user should be able to login using the username and password.

**Case 2** : If the user is not a member, then he/she has to create a new account by clicking “New User” button.

**Case 3:** On click on ‘New User’ button, the user should be taken to a page with a form.

**Case 4:** The new user should be able to fill all the required information and able to create a new account.

**Case 5:** Based on the user information the users access level must be decided and the access are provided based on that.

**Case 6:** If the user is logged in then the user must be able to log out.

* + **Welcome Screen**

**Case1:** The details of the user has to be displayed in the screen and other information related to the user.

**Case 2:** Search bar must be available.

**Case 3:** Other UI elements like necessary buttons and texts should be displayed and align according to the design.

* + **Add/ Update/Delete**

**Case1:** Students must be able to add their details, update their information and delete any unnecessary information.

**Case2**: Student must be able to fill all the required information like name, student id and courses registered. If the required information is not filled then “error message” should be displayed.

**Case 3**: Teachers must be able to fill their information and courses handled by them. If the required details are not filled, then error message should be displayed.

**Case 4:** Admin must be able to add the books to the database, availability along with course id, author and book name.

**Case5**: The users should be able to modify the information or delete the information.

* + **Search**

**Case 1**: The users should be able to search the books based on the title, keywords, author, course name.

**Case2**: The results of the search must be displayed to the user.

**Case3**: The user must be able to clear the search contents and search for other books.

**Case4:** The user must be able to select the book from the search contents.

* + **Order**

**Case1:** The UI should align with the design

**Case2:** The user must be able to select the book and proceed to order it by clicking order button.

**Case3:** The user must be able to order the book based on the availability and the course registered for. If the user does not satisfy the conditions, then the appropriate error message should be displayed.

**Case4:** The user should be able to order the book and notified once the order has been confirmed.

* + **Payment**

**Case1:** The user should be able to pay the amount due for the book.

**Case2:** If the payment is successful, the user should be notified with success message else appropriate error message should be displayed.

**Case3:** The balance amount due should be updated in the user profile.

* + **Order Approval**

**Case1:** The Admin should be able to view all the pending approvals.

**Case2:** The Admin should be able to approve the order based on certain conditions.

**Case3:** The Admin should be able to add book availability, the checkout date and the return date based on the user profile.

* + **Return book.**

**Case1**: The Admin should be able to update the status of the book and the user’s status based on the return date.

**Case2**: The user should be able to view the full details of the order along with borrow, payment and return date and the status of the order once the admin has updated it.

6.3 Performance bounds

**1. Response Time**

The time taken to create account, add, delete and update information should not take more time. The amount of time required to retrieve the details from the database should be less. The response message to the user or to the admin should be delivered immediately and there should not be any delay. When large number of users are using the system at same time, then the process can take more time and the user must be notified with proper message.

**2. Capacity**

The system should be capable of handling large number of users at the same time. The database should be accessed by multiple users at the same time and the database should be updated accordingly. Internal errors should not occur and it should not be crashed.

**3. Resource Availability and Utilization**

The books available should be updated by the admin frequently according to the orders. The users will be able to access it and order only based on the availability.

**4. Browser Compatibility**

This software will run only on web browsers based on their versions and the Internet connection is mandatory.

6.4 Identification of critical components.

The users and the admin should be logged in to the system. The system is connected to the database server.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Testing | Description | Expected Result | Performance bound | Critical Components |
| Log in | Validate whether the student, teacher and admin are able to login | The student, teacher and the admin should be able to login. | Should not be logged in if user and password does not match. | Users table. |
| Register Courses | Validate whether the users are able to register for the courses | The users should be able to register for the courses. |  | Users, Courses Information. |
| Add/Update/Delete books | Validate whether the admin is able to add, update and delete books. | The admin should be able to add, update and delete books |  | Books information. |
| Search | Validate whether the student and teacher are able to search the books. | The search results has to be displayed |  | Books information. |
| Book request | Validate that the users are able to request and validate book request | The users should be able to request the book |  | Books Information.. |
| Authorise request | Validate whether the admin is able to confirm the book request. | The admin should be able to authorise the request |  | Users, Books and orders information. |
| Payment | Validate whether the admin is able to process the payment. | The admin should be able to process the payment |  | Books, Users, Orders Information. |
| Extend due date | Verify that the users are able to extend the due date. | The users should be able to extend the due date. |  | Books and Orders Information |
| Cancel Order | Validate that the admin is able to cancel the orders. | The admin should be able to cancel the orders. |  | Orders and Books Information. |
| Return books | Validate whether the users are able to return the books. | The users should be able to return the books and the information is updated. |  | Users, Books and Orders. |