**Library**

**Introduction:**

In this project LIBRARY, It will clearly explained about what are all the types of books and is categories with its Authors. Here BooksID contains with PRIMARY KEY and CategoriesID , AutherID contains FOREIGN KEY. Each table Book, Author, Categories contains set of **entities**. A **primary key** is used to ensure data in the specific column is unique. A **Foreign Key** is a database **key** that is used to link two tables together.

**Database:**

A **database** is an organized collection of structured information, or data, typically stored electronically in a computer system.

* Entities : BookID, BookName, BookPrice, AuthorID, AuthorName, AuthorAge, CategoriesID, CategoriesName.
* Table Design:

|  |  |
| --- | --- |
| **Books Detail** |  |
| BookID | INT (primary key) |
| BookName | VARCHER |
| BookPrice | INT |
| CategoryID | INT (foreign key) |
| AuthorID | INT (foreign key) |
| **Authors Detail** |  |
| AuthorID | INT (primary key) |
| AuthorName | VARCHAR |
| Author Age | INT |
| **Category Detail** |  |
| CategoryID | INT (primary key) |
| CategoryType | VARCHAR |

* SQL for Creating Tables:

1. Create table Books(

BookID INT,

BookName VARCHAR(20),

BookPrice INT,

CategoryID INT,

AuthorID INT

)

1. Create table Authors(

AuthorID INT,

AuthorName VARCHAR(20),

AuthorAge INT

)

ALTER TABLE Books ADD CONSTRAINT FK\_Books\_Authors FOREIGN KEY (AuthorsID) REFERENCES Authors(AuthorsID)

1. Create table Category(

CategoryID INT,

CategoryType VARCHAR(20)

)

ALTER TABLE Books ADD CONSTRAINT FK\_Books\_Category FOREIGN KEY (CategoryId) REFERENCES Category(CategoryId)

**LibraryAPI**

**Login Test and Login Fail Test:**

**Method :** Logintest

**Description :** logins with username and password

**Parameter :** username and password as parameter is passed

**Returns :** login success

**Method : Login fail test**

**Description : login fails due to wrong username and password**

**Parameter : username and password parameter is passed**

**Returns : Login Success**

**CRUD API**

* REST Contains the following methods
* POST – INSERT (CREATE) – Adds a Resource
* PUT – Update – Updates a Resource
* GET – Read – Reads an Resource

|  |  |  |
| --- | --- | --- |
| **CRUD** | **REST Method** | **SQL Statement** |
| Create | POST | INSERT |
| Read | GET | SELECT |
| Update | PUT | UPDATE |
| Delete | DELETE | DELETE |

**Method Name :** GetBooks

**Description :** Getting the book values from Table Books

**Parameters :** BookID

**Returns :** s

**DB Table :** Books

**Method Name :** GetAllBooks

**Description :** Getting all the book values from the Table Books

**Parameters :** Library

**Returns :** books

**DB Table :**Books

**Method Name :** InsertBooks

**Description :** Inserting Books values in Table Books from DB

Library

**Parameters :** None

**Returns** : books

**DB Table :** Books

**Method Name :** updateBooks

**Description :** Updating Books values in Table Books

**Parameters :** updBooks

**Returns :** numOfRows

**DB Table :** Books

**Method Name :** DeleteBooks

**Description :** Delete Books values from Tale Books

**Parameters :** bookID

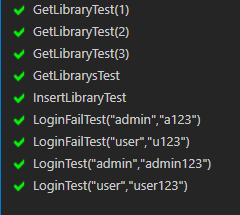
**Returns :** numOfRows

**DB Table :** Books

**TEST PLAN**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case Number | Step Number | Description | Return Value | Expected | Actual |
|  |  |  |  |  |  |
| ST- 001 | 1 | Insert Books | Book Id | >0 | >0 |
|  | 2 | Get Books for inserted BookID | Inserted Books | NOT NULL Name = Inserted name | NOT NULL Name = Inserted name |
|  | 3 | Update Books | Number of Rows | = 1 | = 1 |
|  | 4 | Get Books for updated Books | Updated Books | NOT NULL BookID = Updated bookId | NOT NULL BookID = Updated bookId |
|  | 5 | Delete Books | Has Been Deleted | True | True |
|  | 6 | Get Deleted Books | Books | NULL | True |

**Unit Testing of DB**



**Method Name : GetLibrary(1)**

**Description : Get Library values for testing(Unit Test)**

**Parameter : NULL**

**Returns : Book Details**

**Method Name : GetLibraryTest(2)**

**Description : Get Library values for testing (unit Test)**

**Parameter : BookID**

**Returns : Book Details**

**Method Name : GetLibraryTest(3)**

**Description : Get Library values for testing(Unit Test)**

**Parameter : BookID**

**Returns : BookID**

**Method Name : GetLibraryTest(3)**

**Description : Get Library values for testing(Unit Test)**

**Parameter : NULL**

**Returns :**

**Method Name : InsertLibraryTest**

**Description : Get Library values for testing(Unit Test)**

**Parameter : BookID**

**Returns : NULL**

**Method Name : InsertLibraryTest**

**Description : Get Library values for testing(Unit Test)**

**Parameter : BookID**

**Returns : NULL**

**Method Name : InsertLibraryTest**

**Description : Get Library values for testing(Unit Test)**

**Parameter : BookID**

**Returns : NULL**

**Method Name : LoginFailTest()**

**Description : If user name and password are incorrect then Login Failed**

**Parameter : NULL**

**Returns : LoginFails**

**Method Name : LoginTest**

**Description : “User name= user” “ Password =user123”**

**Parameter : NULL**

**Returns : Login success**

**Method Name : LoginTest**

**Description : “User name= admin” “ Password =admin123”**

**Parameter : NULL**

**Returns : Login success**

**API TESTING**

**CreateBooksAsync:**

**Method Name: CreateBooksAsync**

**Description: Insert given details of a Books into the database with API**

**using http request**

**Parameter: NULL**

**Returns: Inserted details of the Books in json format**

**DB Table: Books**

**GetBooksAsync :**

**Method Name: GetBooksAsync**

**Description: Get details of a Books based on Books entities API using http request**

**Parameter: BookID**

**Returns: Required details of the Books in json format**

**DB Table: Books**

**GetBooksAsync :**

**Method Name: GetBooksAsync**

**Description: Get Books entities with API using http request**

**Parameter: None**

**Returns: Required details of the Booksin json format**

**DB Table: Books**

**UpdateBooksAsync :**

**Method Name: UpdateBooksAsync**

**Description : Update the details of Books based on BooksID**

**with API using http request**

**Parameter: BookID**

**Returns: Details of the updated Books in json format**

**DB Table: Books**

**DeleteBooksAsync :**

**Method Name : DeleteBooksAsync**

**Description : Delete details of a Books based on BookID with API using http request**

**Parameter : BookID**

**Returns : Number of rows deleted**

**DB Table : Books**