|  |  |
| --- | --- |
| Logo_FPT_University_doc | **MINISTRY OF EDUCATION AND TRAINING** |

**FPT UNIVERSITY**

**CAPSTONE PROJECT DOCUMENT**

BSMS

**Report #3 – Software Design Specification**

|  |  |
| --- | --- |
|  | |
| Group Member | SE03975 - Olasehinde Ezekiel Olaoluwa |
| SE\*\*\*\*\* - Kanda Joshua David |
| SE\*\*\*\*\* - Shunom Peter Bala |
| Supervisor | Bui Đinh Chien |
| Ext. Supervisor |  |
| Project Code | BSMS |

- Hanoi, 05/2012 -

Table of Contents

[1 PURPOSE 4](#_Toc483851876)

[2 ARCHITECTURE DESIGN 4](#_Toc483851877)

[2.1 Choice of Architecture Design 4](#_Toc483851878)

[2.1.1 Our System Architecture 4](#_Toc483851879)

[2.1.2 ASP.NET MVC ARCHITECTURE 5](#_Toc483851880)

[2.2 Architectural Presentation 6](#_Toc483851881)

[2.3 Component/Package Design 7](#_Toc483851882)

[3 DETAILED DESIGN 7](#_Toc483851883)

[3.1 Common Design 7](#_Toc483851884)

[3.2 Authentication Management 7](#_Toc483851885)

[3.2.1 Class Diagram 7](#_Toc483851886)

[3.2.2 Class Description 7](#_Toc483851887)

[3.2.3 Register Customer/Staff 9](#_Toc483851888)

[3.2.4 Login 11](#_Toc483851889)

[3.2.5 Logout 13](#_Toc483851890)

[3.2.6 Forget Password 14](#_Toc483851891)

[3.3 Category Management 14](#_Toc483851892)

[4 DATABASE DESIGN 15](#_Toc483851893)

[4.1 Entity Relationship Diagram 15](#_Toc483851894)

[4.2 Database Diagram 16](#_Toc483851895)

[4.3 Table Descriptions 16](#_Toc483851896)

[4.3.1 User 16](#_Toc483851897)

[4.3.2 Role 17](#_Toc483851898)

[4.3.3 Author 17](#_Toc483851899)

[4.3.4 Category 18](#_Toc483851900)

[4.3.5 Genre 18](#_Toc483851901)

[4.3.6 Book 19](#_Toc483851902)

[4.3.7 Book\_Category 20](#_Toc483851903)

[4.3.8 Producer 20](#_Toc483851904)

[4.3.9 Book\_SoftCopy 20](#_Toc483851905)

[4.3.10 Book\_Author 21](#_Toc483851906)

[4.3.11 WatchList 21](#_Toc483851907)

# PURPOSE

This document is proposed with intent of describing the technical design of our bookstore Management system, mainly to give development team an overall overview of the system’s architecture, thus bringing them to a level of enlightenment of how the system should work and how to implement it. This section contains the following

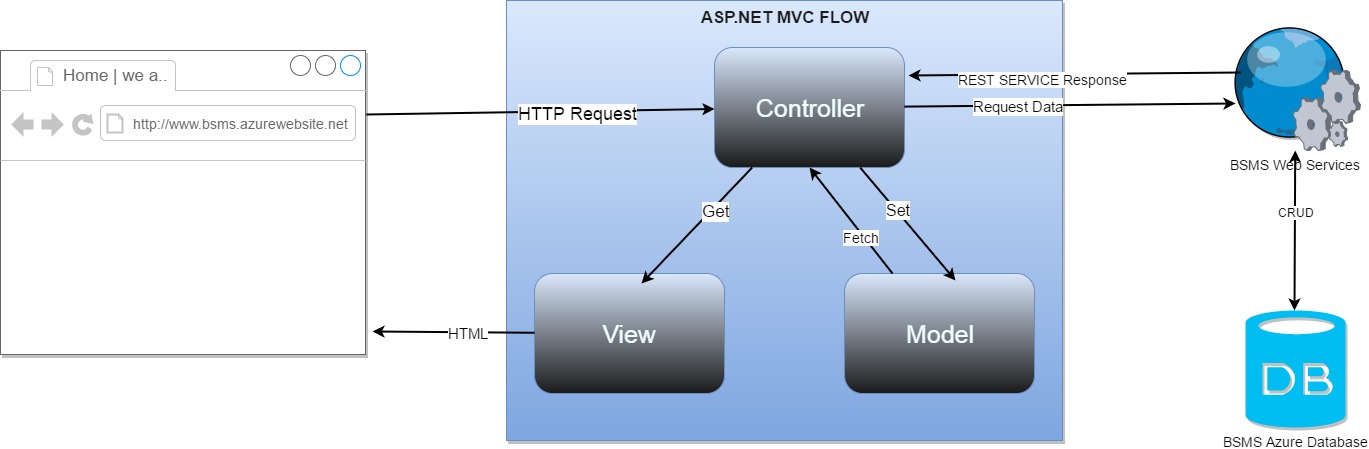
* System Architecture Design
* Database Design
* Detail Design

# ARCHITECTURE DESIGN

## Choice of Architecture Design

### Our System Architecture

BSMS web application uses the MVC (Model-View and Controller) architecture together with ASP.NET Web Services (XML,JSON), which has been one of the best software design pattern this days, This will allow our code to be more readable, more understandable and editable than the native coding pattern.



* **Models :** Model objects are the parts of the application that implement the logic for the application's data domain. Often, model objects retrieve and store model state in a database. For example, a Product object might retrieve information from a database, operate on it, and then write updated information back to a Products table in a SQL Server database.

In small applications, the model is often a conceptual separation instead of a physical one. For example, if the application only reads a dataset and sends it to the view, the application does not have a physical model layer and associated classes. In that case, the dataset takes on the role of a model object.

In Our system the model will be referencing to the BSMS web service to complete database related tasks.

* **Views :** Views are the components that display the application's user interface (UI). Typically, this UI is created from the model data. An example would be an edit view of a Products table that displays text boxes, drop-down lists, and check boxes based on the current state of a Product object..
* **Controllers :** This serves as the system handler, it handles user input and what the system do with them. When request is send from the client side the asp.net send a request to the controller, which after the controller will find the equivalent get method and them render the called view.

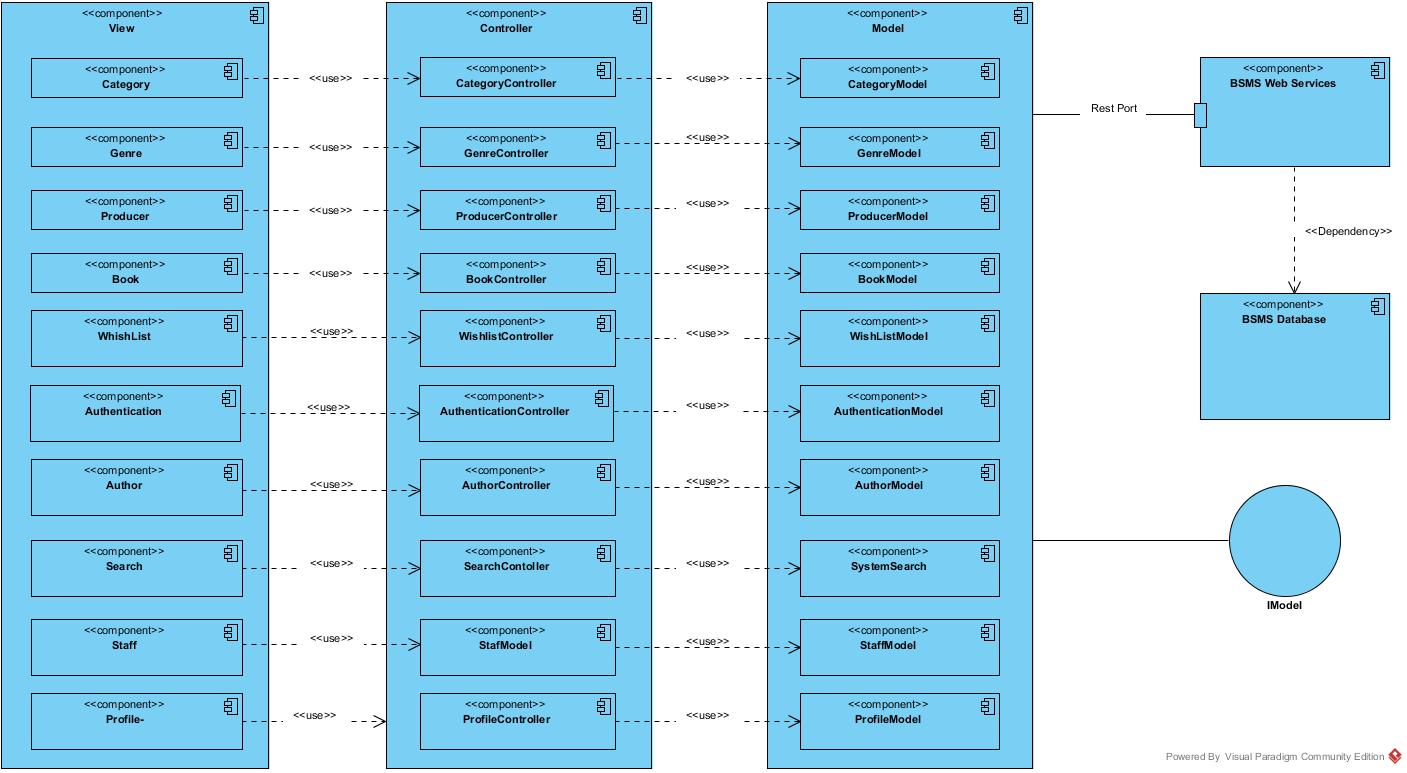
 Controllers are the components that handle user interaction, work with the model, and ultimately select a view to render that displays UI. In an MVC application, the view only displays information; the controller handles and responds to user input and interaction. For example, the controller handles query-string values, and passes these values to the model, which in turn might use these values to query the database**.**

* **Web Service**: This is the service that connects to the database and fetches data for the controller to make use of; CRUD methods will be implemented using this web service.

### FEATURES OF THE ASP.NET MVC FRAMEWORK

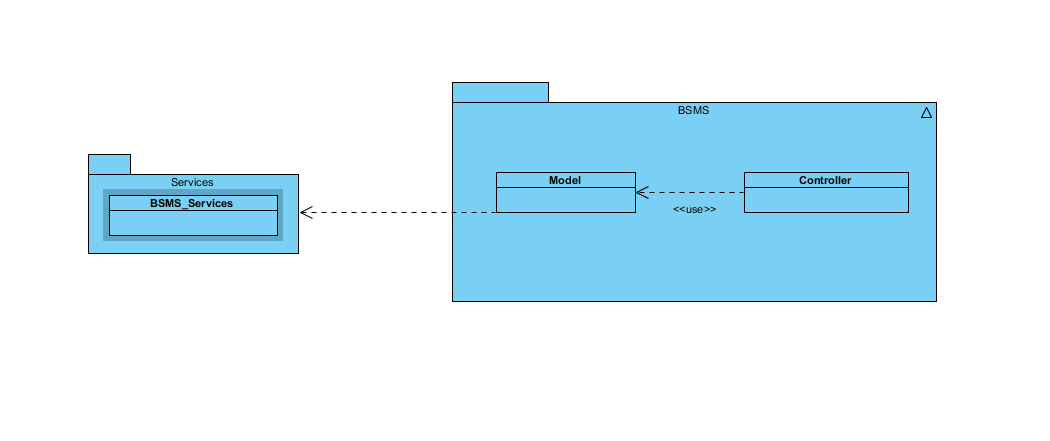
* Separation of application tasks (input logic, business logic, and UI logic), testability, and test-driven development (TDD). All core contracts in the MVC framework are interface-based and can be tested by using mock objects, which are simulated objects that imitate the behavior of actual objects in the application. You can unit-test the application without having to run the controllers in an ASP.NET process, which makes unit testing fast and flexible. You can use any unit-testing framework that is compatible with the .NET Framework.
* An extensible and pluggable framework. The components of the ASP.NET MVC framework are designed so that they can be easily replaced or customized. You can plug in your own view engine, URL routing policy, action-method parameter serialization, and other components. The ASP.NET MVC framework also supports the use of Dependency Injection (DI) and Inversion of Control (IOC) container models. DI enables you to inject objects into a class, instead of relying on the class to create the object itself. IOC specifies that if an object requires another object, the first objects should get the second object from an outside source such as a configuration file. This makes testing easier.
* Extensive support for ASP.NET routing, which is a powerful URL-mapping component that lets you build applications that have comprehensible and searchable URLs. URLs do not have to include file-name extensions, and are designed to support URL naming patterns that work well for search engine optimization (SEO) and representational state transfer (REST) addressing.
* Support for using the markup in existing ASP.NET page (.aspx files), user control (.ascx files), and master page (.master files) markup files as view templates. You can use existing ASP.NET features with the ASP.NET MVC framework, such as nested master pages, in-line expressions (<%= %>), declarative server controls, templates, data-binding, localization, and so on.
* Support for existing ASP.NET features. ASP.NET MVC lets you use features such as forms authentication and Windows authentication, URL authorization, membership and roles, output and data caching, session and profile state management, health monitoring, the configuration system, and the provider architecture.

## Component Diagram



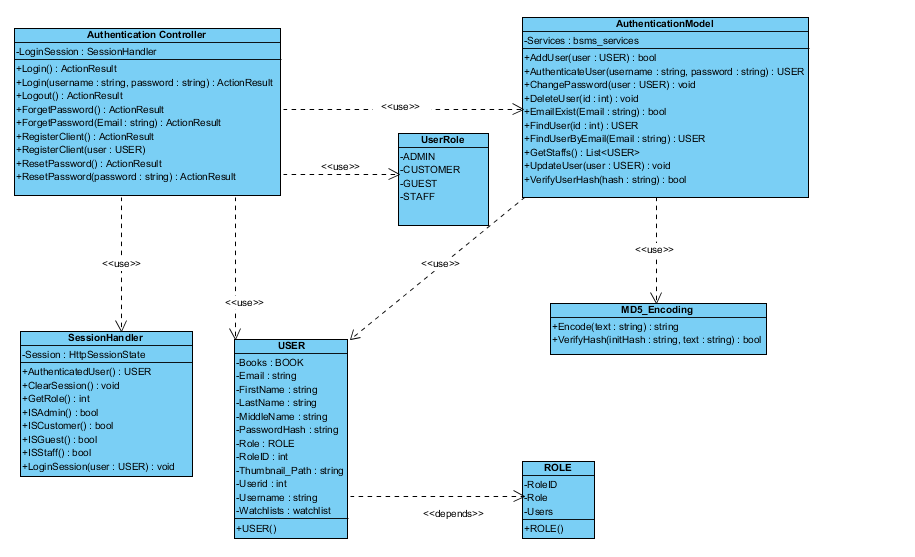
# DETAILED DESIGN

## Common Design



## Authentication Management

### Class Diagram



### Class Description

|  |  |  |
| --- | --- | --- |
| Class | Dependency | Description |
| User | Role | The user class holds all user data and information including all the usernames and passwords in the system, this is known as the BSMS user recognition methodology. |
| Role | N/A | This class is intended to hold unique set of roles for users, since the user uses this class as dependency, user has a role, which is how the system recognise the type of user coming into the system |
| AuthenticationController | N/A |  |
| SessionHandler | N/A |  |
| AuthenticationModel | USER |  |
| MD5\_Encoding | N/A |  |
| UserRole | N/A |  |

#### User

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | User | | | |
| **Description** | User data, this class holds all the user information including the password and username | | | |
| **Base Class** | Authentication | | | |
| **Constructor** | Default | | | |
| **Prototype** |  | | | |
| **Source File** | User.cs | | | |
| **Namespace** | BSMS.Model | | | |
| **Attributes** | Name | Type | Description | |
| Name | String | N/A | |
| Username | String | Users username is contained in this attribute | |
| PasswordHash | String | This contains 64 digit MD5 encrypted password | |
| Email | String |  | |
| RoleID | int | This hold the role of user in number format ex. (1:”Staff” , 2:”Admin”) | |
|  | Thumbnail | String | This hold the thumbnail path in the system | |
| **Methods** | Name | Input | Output | Description |
| Authentication | Username  Password | User | This function check if user is valid or not, returns null if not valid and user information if valid |
| LoggedOn | N/A | Boolean | This function is intended to check if any user is logged on to the system |

#### Role

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | Role | | | |
| **Description** | This contain all the user roles that exists in the system | | | |
| **Base Class** | N/A | | | |
| **Constructor** | Default | | | |
| **Prototype** |  | | | |
| **Source File** | Role.cs | | | |
| **Namespace** | BSMS.Model | | | |
| **Attributes** | Name | Type | Description | |
| Roleid | int | This is a unique integer value that holds a unique id for each role that exist in the system | |
| Role | String | Role of this role e.g. (“Admin, Staff” .... ) | |
| **Methods** | Name | Input | Output | Description |

#### Authentication Controller

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | AuthenticationController | | | |
| **Description** | This class controls everything they deals with User and Roles | | | |
| **Base Class** | System.Web.Mvc.Controller | | | |
| **Constructor** | N/A | | | |
| **Prototype** |  | | | |
| **Source File** | AuthenticationController.cs | | | |
| **Namespace** | BSMS.Controller | | | |
| **Methods** | Name | Input | Output | Description |
| Login | N/A | ActionResult | This method make a request to the Http Get in order to request for a page |
| Login | String, String | ActionResult | This method call makes request to Http Post and calls the Authentication Model class to verify user detail |
| Logout | N/A | ActionResult | On this method call E-U will be logged out of the system and redirect to the login page |
| ForgetPassword | N/A | ActionResult | Http Method call to request forget password page |
| ForgetPassword | String | ActionResult | Http Post method call to act on user input |
| RegisterUser | N/A | ActionResult | Request made to register user or customer page |
| RegisterUser | User | ActionResult | Process new user Http Post method |

#### Session Handler

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | SessionHandler | | | |
| **Description** | This class acts as session assistant, in other words this class handles all session related activities | | | |
| **Base Class** | N/A | | | |
| **Constructor** | Default | | | |
| **Prototype** |  | | | |
| **Source File** | SessionHandler.cs | | | |
| **Namespace** | BSMS.Session | | | |
| **Attributes** | Name | Type | Description | |
| Session | HttpSessionState | The Session is an instance of the HttpSessionState, which contains the application context Session | |
| **Methods** | Name | Input | Output | Description |
|  | AuthenticatedUser | N/A | USER | This method return the currently logged in user, and returns it’s object |
| ClearSession | N/A | N/A | This method is used to clear application context session |
| GetRole | N/A | Integer | This method will be used to get role of currently loggeduser |
| ISAdmin | N/A | Boolean | This method is mean’t to check if currently logged in user is Admin |
| ISCustomer | N/A | Boolean | This method is mean’t to check if currently logged in user is Customer |
| ISGuest | N/A | Boolean | This method is mean’t to check if there is no logged user in the system |
| ISStaff | N/A | Boolean | This method is mean’t to check if currently logged in user is Staff |
| LoginSession | USER | N/A | This method is mainly used to create a login session, which mean this will be used only when users needs to login. |

#### Authentication Model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | AuthenticationModel | | | |
| **Description** | This is a class that handles all operations that involves Authentication including CRUD Operations, thus this class have direct connection to BSMS SOAP Webservices. | | | |
| **Base Class** | N/A | | | |
| **Constructor** | Default | | | |
| **Prototype** |  | | | |
| **Source File** | Authentication.cs | | | |
| **Namespace** | BSMS.Model | | | |
| **Attributes** | Name | Type | Description | |
| Services | bsms\_services | This is a private instance of “BSMS.SERVICES” | |
| **Methods** | Name | Input | Output | Description |
| AddUser | USER | Boolean | This method is use to call the webservices method inorder to add a new user into the system |
| AuthenticateUser | String, String | USER | This method is proposed to check if login credential is correct or not, this return null if username or password is incorrect and User Object otherwise |
| ChangePassword | USER | N/A | This method is used to change user passwords on reset password request |
| DeleteUser | Int | N/A | This static method is used to delete user from BSMS |
| FindUser | Int | USER | This static method return User Object which matches the id of user inpt |
| FindUserByEmail | Int | USER | This method is used to find user by email, since the email address of each user is unique |
| GetStaffs | N/A | List<USER> | This method is used to get all avaliable staff in the system |
| UpdateStaff | USER | N/A | This is used to update staff informations |
| VerifyUserHash | String | Boolean | This method is mainly used to verify hash code of users |

#### MD5\_Encoding

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | MD5\_Encoding | | | |
| **Description** | This class handles encoding, like encode, decode | | | |
| **Base Class** | MD5\_Encoding | | | |
| **Constructor** | Default | | | |
| **Prototype** |  | | | |
| **Source File** | MD5\_Encoding.cs | | | |
| **Namespace** | BSMS.Model | | | |
| **Methods** | Name | Input | Output | Description |
| Encode | String | String | This function mainly encode input string to MD5 encryption |
| VerifyHash | String,String | Boolean | This function is intended to check if user password match its encypted format |

#### UserRole

|  |  |  |  |
| --- | --- | --- | --- |
| Class | UserRole | | |
| **Description** | This class acts like enum class, as this keeps static variables of each types of users that exist in the system | | |
| **Base Class** | UserRole | | |
| **Constructor** | Default | | |
| **Prototype** |  | | |
| **Source File** | UserRole.cs | | |
| **Namespace** | BSMS.Model | | |
| **Attribute** | Name | Type | Description |
| ADMIN | int | This contains role id of Admin users |
| CUSTOMER | int |
|  | GUEST | int | This contains role id of Guest users |
|  | STAFF | int | This contains role id of Staff users |

### Register Customer/Staff

BSMS guest who might have intention of becoming a customer, or Admin who wish to register a new staff in the system, both the guest and the admin will make use of the same page, but what happens behind every actor will be different.

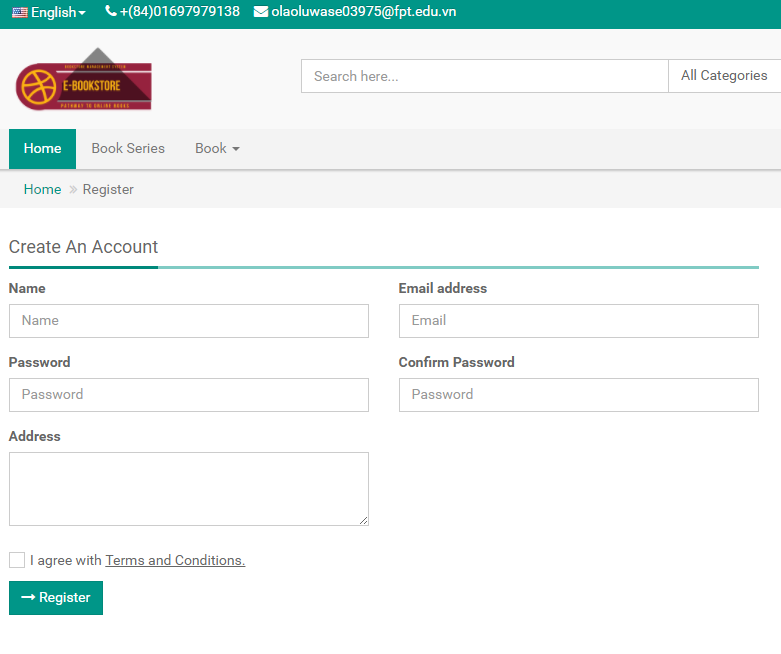
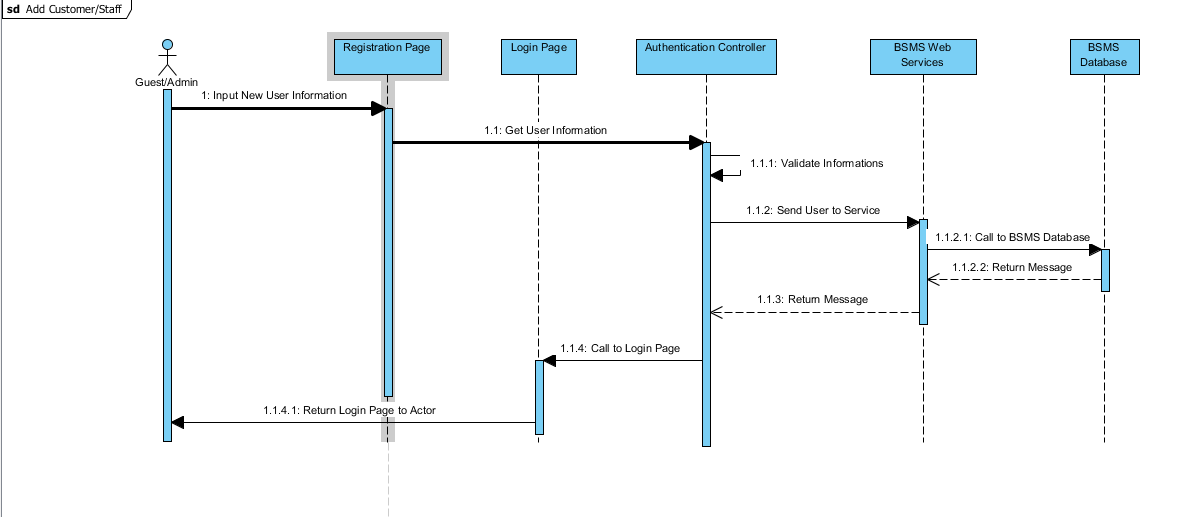


Figure 3.2.3

**Table 4-1:** Register Customer/Staff

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Object/Control Name** | **Type** | **Required** | **Length** | **Description** |
| 1 | Name | Text-Field | Yes | 3-20 | Customer Name |
| 2 | Email | Email-Field | Yes | 7-30 | Customer Email Address |
| 3 | Password | Password-Field | Yes | > 6 | User password |
| 4 | Confirm Password | Password-Field | Yes | > 6 | Password verification |
| 5 | Address | MultiLine-Field | No | Max 50 | Customers Address |
| 6 | cbAccept | CheckBox | N/A | N/A | Check i agree with terms and condition to activate Register button |
| 7 | Register | Button | N/A | N/A | Registration button |

#### Sequence Diagram



### Login

#### Description

Account holder of BSMS who might want to take action on the system, which might be required of him/her to login into the system, this applies to all stockholders of the system apart from the guest.

#### Screen Design

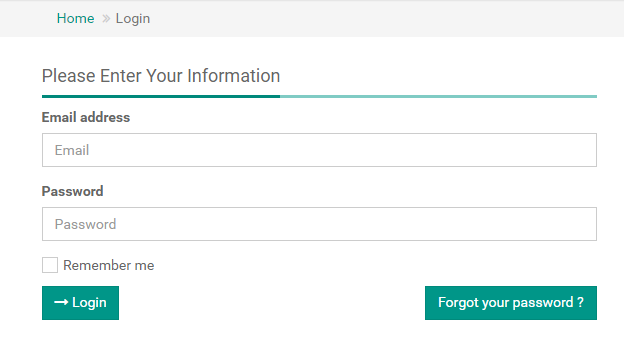
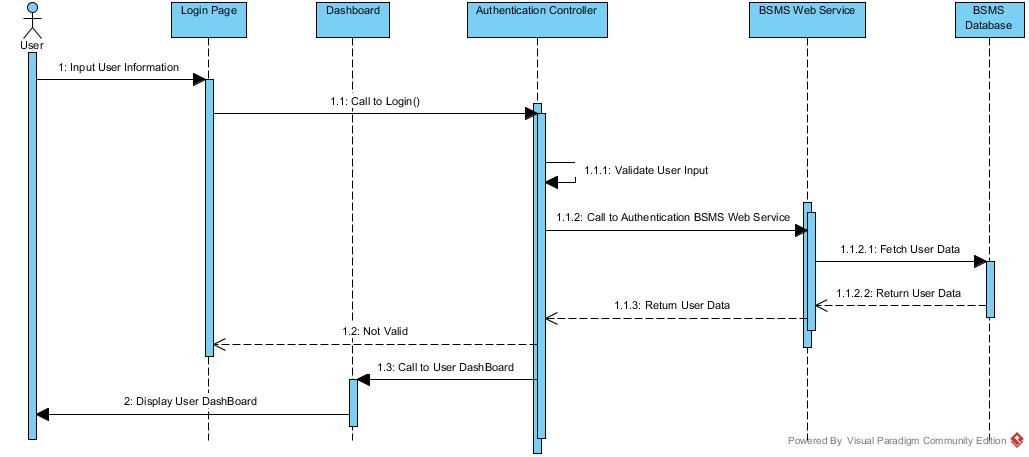


Figure 3.2.3

**Table 4-2:** Login

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Object/Control Name** | **Type** | **Required** | **Length** | **Description** |
| 2 | Email | Email-Field | Yes | 7-30 | Customer Email Address |
| 3 | Password | Password-Field | Yes | > 6 | User password |
| 6 | cbRememberMe | CheckBox | N/A | N/A | Remember me, to keep user detail incase of next login |
| 7 | Login | Button | N/A | N/A | Login button |

#### Sequence Diagram

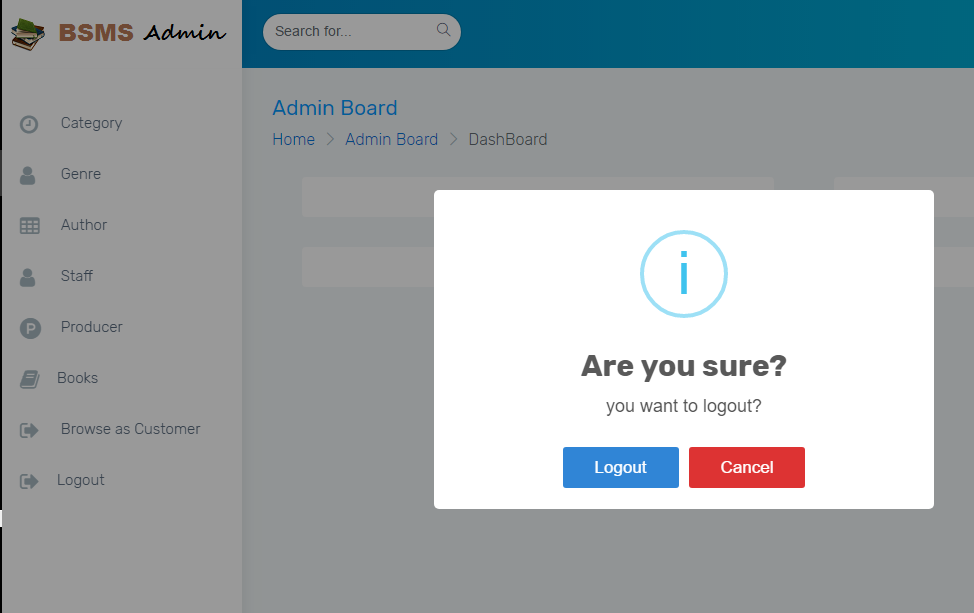


### Logout

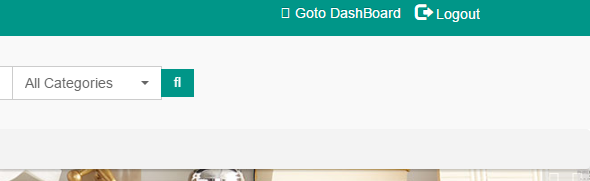
#### Description

Logged user sometime might like to make use of the logout functionality, in order for guest users not to take advantage of their logged session.

#### Screen Design

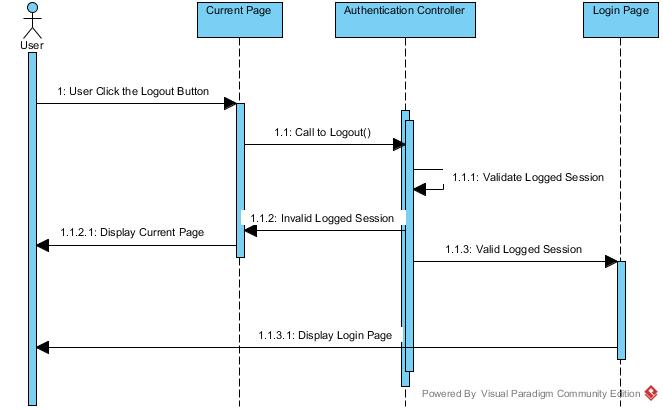


For Admin and Staff users



For customers

#### Sequence Diagram

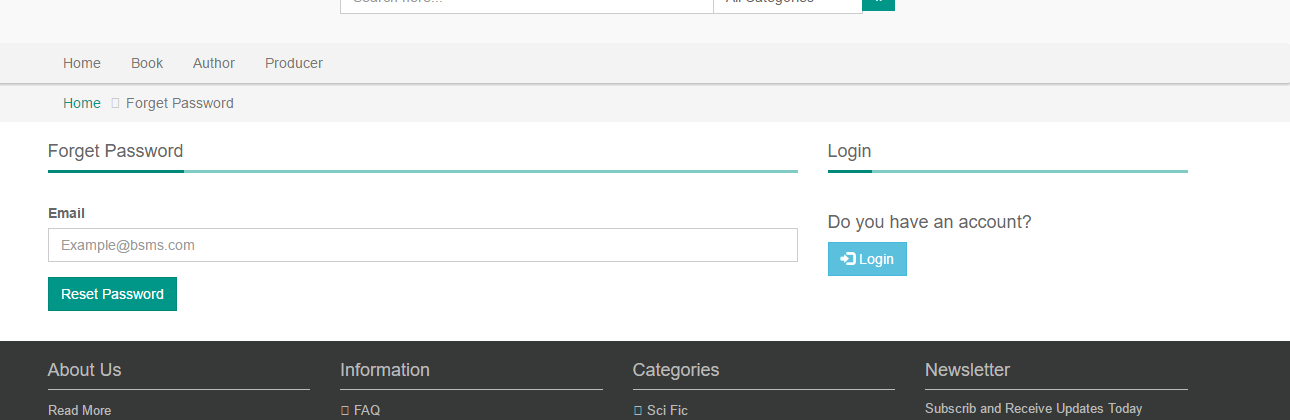


### Forget Password

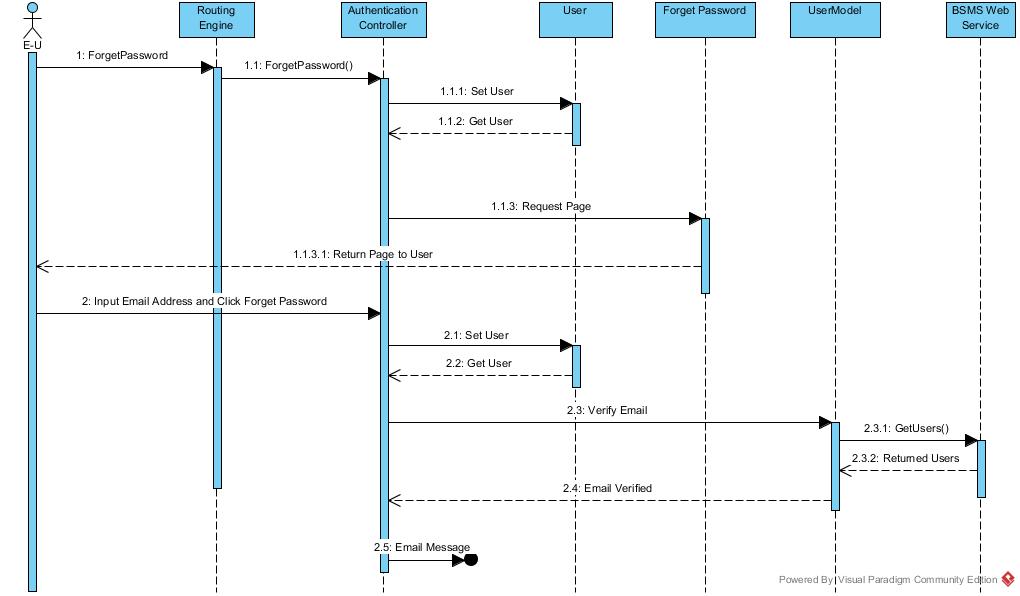
#### Description

BSMS doesn’t expect her E-U to have the brain of computer, in case if any of her users forget their password, BSMS has provide a simple way to recover their password using the same email used while registering.

#### Screen Design

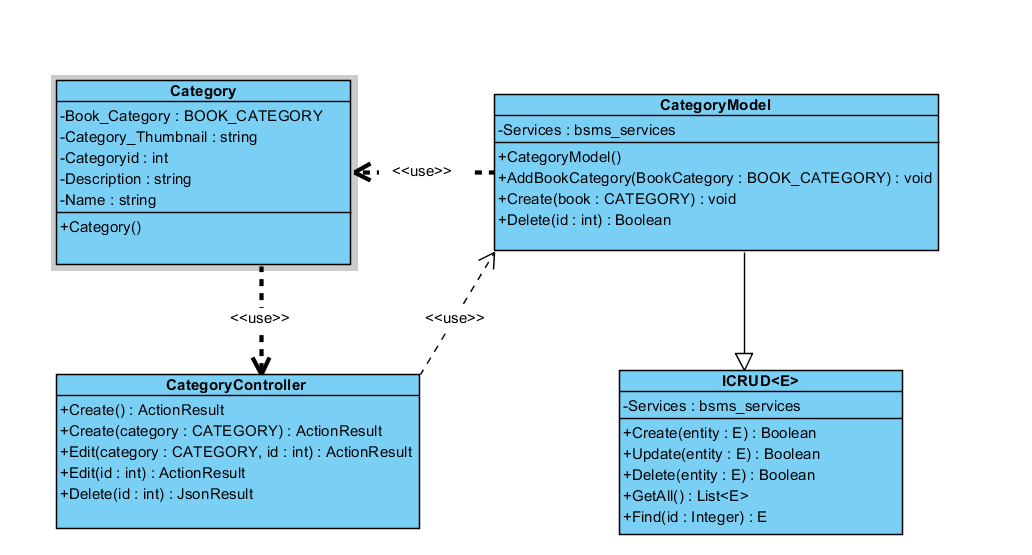


#### Sequence Diagram



## Category Management

### Class Diagram



### Class Description

|  |  |  |
| --- | --- | --- |
| Class | Dependency | Description |
| Category | Role | The category class is a linq mapped entity class that holds all informations regarding category in BSMS. |
| ICRUD | N/A | This Interface is intended to create rules for all model class and sometimes the controller can also make use of this |
| CategoryModel | N/A | This class provide connection to BSMS SOAP services, CRUD operations will be handled by this model class |
| CategoryController | N/A | This controller class extends from the base class “Controller” which allow request to be mapped to it. |

#### Category

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Category | | |
| **Description** | this class holds all category information including name, and description | | |
| **Base Class** | Object | | |
| **Constructor** | Default | | |
| **Prototype** |  | | |
| **Source File** | Category.cs | | |
| **Namespace** | BSMS.bsms.locahost | | |
| **Attributes** | Name | Type | Description |
| Name | String | N/A |
| Book\_Category | BOOK\_CATEGORY | N/A |
| Category\_Thumbnail | String | This field contains the image of a category |
| CategoryID | Int | This is a unique identifyer for each categories, and this is what serves as key BSMS uses in recorgnising a category |
| Description | String | This field contains description for category |

#### Category Model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | CategoryModel | | | |
| **Description** | This class provide connection to BSMS SOAP services, CRUD operations will be handled by this model class | | | |
| **Base Class** | Object | | | |
| **Constructor** | Default | | | |
| **Prototype** |  | | | |
| **Source File** | Category.cs | | | |
| **Namespace** | BSMS.Model | | | |
| **Attributes** | Name | Type | Description | |
| Services | Bsms\_services | A private instance of bsms soap web services | |
| **Methods** | Name | Input | Output | Description |
| AddBookCategory | BOOK\_CATEGORY | N/A | This method is use to add new book category into BSMS |
| Create | Category | void | This helper method is used to insert new category into the system |
|  | Delete | Int | Boolean | this is a static internal type of method, whcih allows existing category to be deleted as the name Delete implies |

#### CategoryController

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | CategoryController | | | |
| **Description** | This class provide connection to BSMS SOAP services, CRUD operations will be handled by this model class | | | |
| **Base Class** | System.web.controller | | | |
| **Constructor** | N/A | | | |
| **Prototype** |  | | | |
| **Source File** | CategoryController.cs | | | |
| **Namespace** | BSMS.Controller | | | |
| **Methods** | Name | Input | Output | Description |
| Create | N/A | ActionResult | This is serves as http get handler |
| Create | Category | ActionResult | This is a web method that handles user input for create |
|  | Edit | Int | ActionResult | This is a controller web method that handles all http get for Edit functionality |
| Edit | Category, int | ActionResult | This is a controller http post method that handles edit functionalities for categories |
| Index | N/A | ActionResult | http get method that handles list page |
| Delete | Int | JsonResult | http delete method that handles delete request from bsms |

### Add Category

This feature allows admin of BSMS to add new category into the system

#### Screen Design

#### Sequence Diagram

### Edit Category

This feature allows admin of BSMS to add new category into the system

#### Screen Design

#### Sequence Diagram

### List Category

This feature allows admin of BSMS to add new category into the system

#### Screen Design



#### Sequence Diagram

### Delete Category

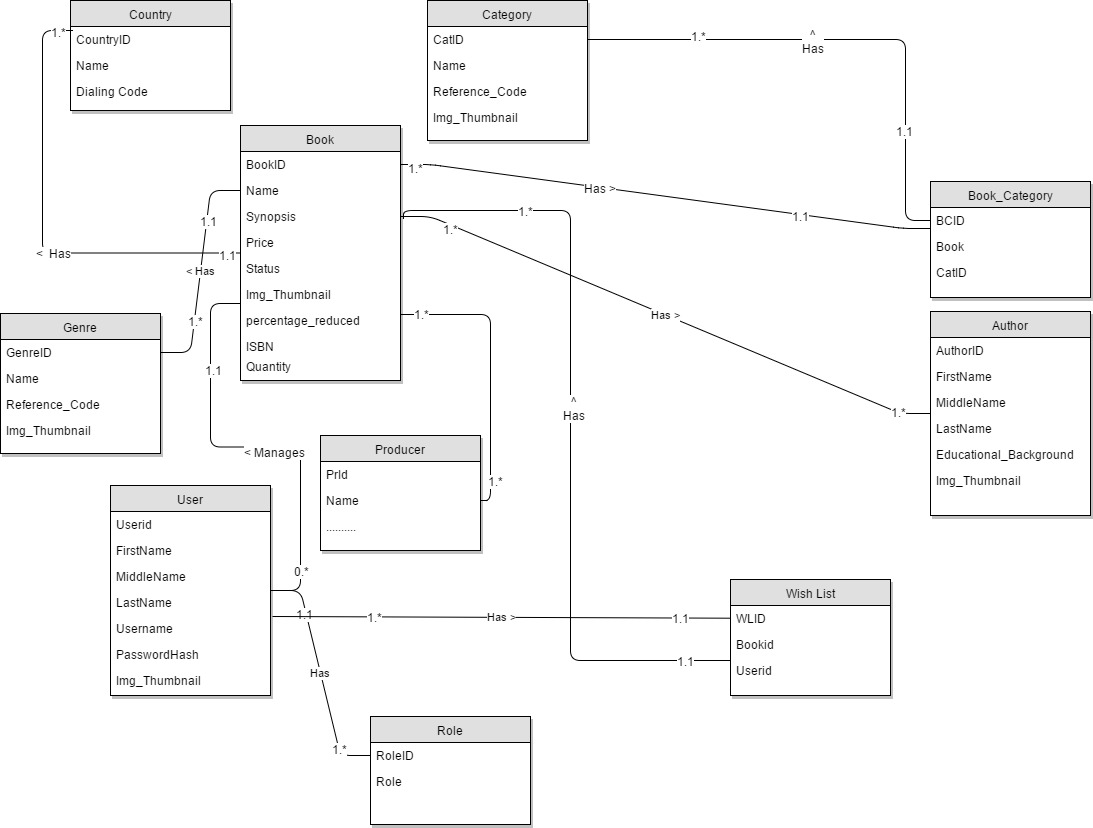
This feature allows admin of BSMS to add new category into the system

#### Screen Design

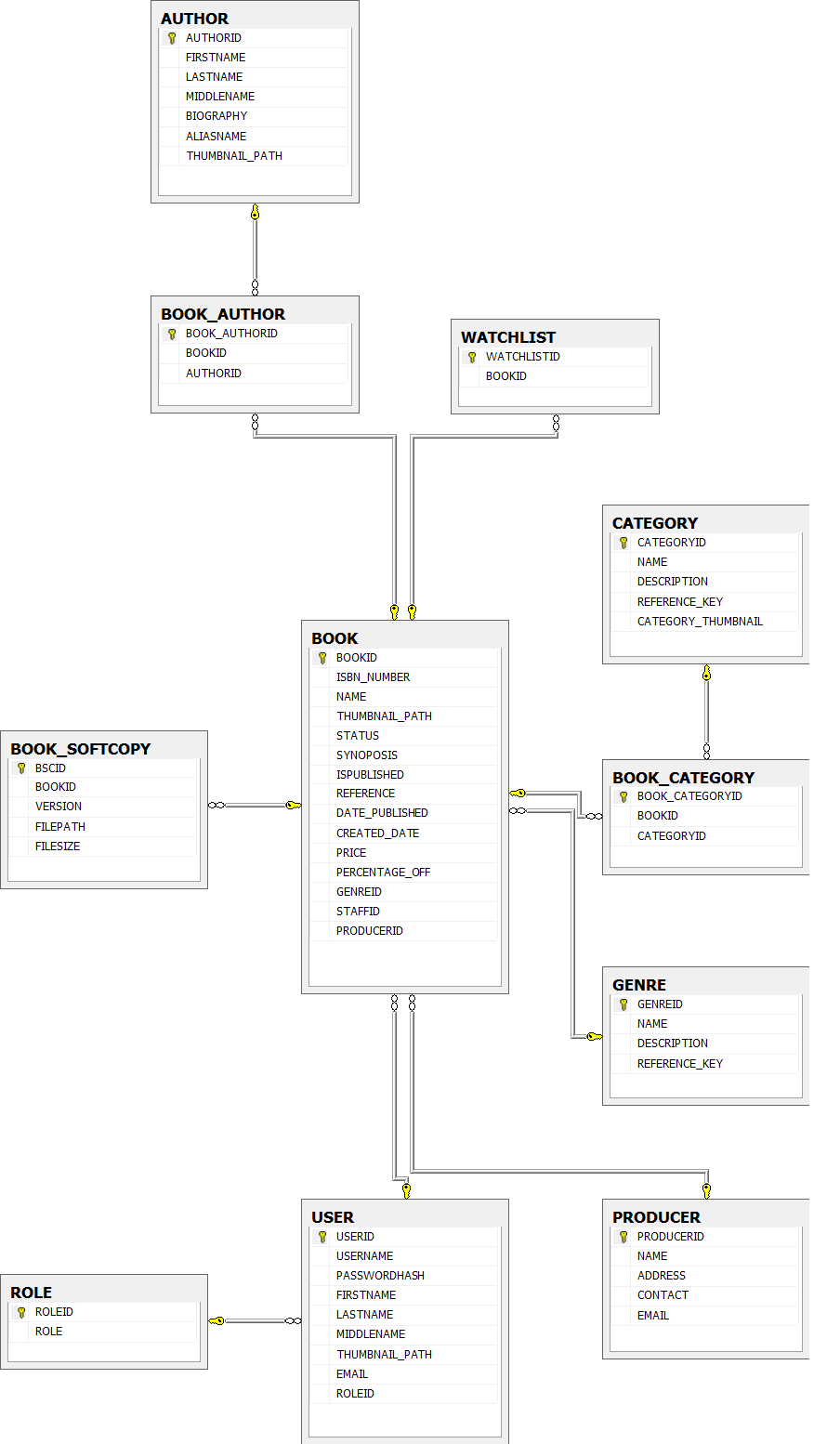
#### Sequence Diagram

# DATABASE DESIGN

## Entity Relationship Diagram



## Database Diagram



## Table Descriptions

### User

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Attribute | Type | Constraints | Description |
| 1 | USERID | INT | Primary key | User’s unique identifier |
| 2 | USERNAME | VARCHAR | Not null |  |
| 3 | PASSWORDHASH | VARCHAR | Not Null | This contains the password hash of users in MD5 format |
| 4 | FIRSTNAME | VARCHAR | Not null | User’s name |
| 5 | LASTNAME | VARCHAR | N/A | User’s name |
| 6 | MIDDLENAME | VARCHAR | N/A | User’s name |
| 7 | THUMBNAIL\_PATH | VARCHAR |  | This attribute contains user’s image path |
| `8 | EMAIL | VARCHAR | Not null | User’s unique email address |
| 9 | ROLEID | INT | Foreign Key | Role foreign key, this is how the system recognise different type of users |

### Role

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Attribute | Type | Constraints | Description |
| 1 | ROLEID | INT | Primary Key | Role unique identifier, auto increment field |
| 2 | ROLE | VARCHAR | Not Null | Role name |

### Author

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Attribute | Type | Constraints | Description |
| 1 | AUTHORID | INT | Primary Key | Author unique identifier, auto increment field |
| 2 | FIRSTNAME | VARCHAR | Not null | Author’s First name |
| 3 | LASTNAME | VARCHAR | Not Null | Author’s Last name |
| 4 | MIDDLENAME | VARCHAR | N/A | Author’s Middle name |
| 5 | BIOGRAPHY | VARCHAR | Not null | Full detail about author |
| 6 | ALIASNAME | VARCHAR | N/A | Author’s nick, prefix or any name the is popularly use to refer to such author |
| 7 | THUMBNAIL\_PATH | VARCHAR | Not Null | Image path of author is store in this attribute |

### Category

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Attribute | Type | Constraints | Description |
| 1 | CATEGORYID | INT | Primary Key | Category unique identifier, auto increment field |
| 2 | NAME | VARCHAR | Not null | Category name |
| 3 | DESCRIPTION | VARCHAR | Not null | Category description |
| 4 | REFERENCE\_KEY | VARCHAR | N/A | The reference key set the link to each category |
| 5 | CATEGORY\_THUMBNAIL | VARCHAR | Not Null | Image path of category is store in this attribute |

### Genre

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Attribute | Type | Constraints | Description |
| 1 | GENREID | INT | Primary Key | Genre unique identifier, auto increment field |
| 2 | NAME | VARCHAR | Not null | Genre name |
| 3 | DESCRIPTION | VARCHAR | Not null | Genre description |
| 4 | REFERENCE\_KEY | VARCHAR | N/A | The reference key set the link to each Genre |

### Book

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Attribute | Type | Constraints | Description |
| 1 | BOOKID | INT | Primary Key | Book unique identifier, auto increment field |
| 2 | ISBN\_NUMBER | BIGINT | Not null | Unique book identifier |
| 3 | NAME | VARCHAR | Not null | Book name |
| 4 | THUMBNAIL\_PATH | VARCHAR | N/A | Book image path address |
| 5 | STATUS | BIT | Null | Book status Null = “Not attended”, 0 = “Rejected” and 1=”Approved” |
| 6 | ISPUBLISHED | BIT | Not Null |  |
| 7 | SYNOPOSIS | VARCHAR | Not null | this holds information of book overview. |
| 8 | REFERENCE | VARCHAR | N/A | This contains information all available references |
| 9 | DATE\_PUBLISHED | DATETIME | N/A | Date this book was published, normally the date admin approves the book |
| 10 | CREATED\_DATE | DATETIME | N/A | Date this book was initially created |
| 11 | PRICE | FLOAT | Not null | Price of this book |
| 12 | PERCENTAGE\_OFF | INT | null | Percentage discount |
| 13 | GENREID | INT | Foreign Key | Book genre |
| 14 | STAFFID | INT | Foreign Key | Staff who controls this book |
| 15 | PRODUCERID | INT | Foreign Key | Producer of this book |

### Book\_Category

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Attribute | Type | Constraints | Description |
| 1 | BOOK\_CATEGORYID | INT | Primary Key |  |
| 2 | BOOKID | INT | Foreign Key |  |
| 3 | CATEGORYID | INT | Foreign Key |  |

### Producer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Attribute | Type | Constraints | Description |
| 1 | PRODUCTID | INT | Primary Key |  |
| 2 | NAME | INT | Not Null |  |
| 3 | ADDRESS | INT | Not Null |  |
| 4 | CONTACT | VARCHAR | Null |  |
| 5 | ADDRESS | VARCHAR | Null |  |

### Book\_SoftCopy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Attribute | Type | Constraints | Description |
| 1 | BSCID | INT | Primary Key |  |
| 2 | BOOKID | INT | Foreign Key |  |
| 3 | VERSION | FLOAT | Not Null |  |
| 4 | FILEPATH | VARCHAR | Null |  |
| 5 | FILESIZE | INT | Null |  |

### Book\_Author

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Attribute | Type | Constraints | Description |
| 1 | BOOK\_AUTHORID | INT | Primary Key |  |
| 2 | BOOKID | INT | Foreign Key |  |
| 3 | AUTHORID | INT | Foreign Key |  |

### WatchList

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Attribute | Type | Constraints | Description |
| 1 | WATCHLISTID | INT | Primary Key |  |
| 2 | BOOKID | INT | Foreign Key |  |
| 3 | USERID | INT | Foreign Key |  |