**ABOUT AUTHOR**

Copyright © 2018 by Korea Software HRD center

Sixth Generation — WEB Group

Address all inquiries to:

KSHRD Center

Telephone: (855) 23 99 13 14

Email: info.kshrd@gmail.com

Website: www.kshrd.com.kh

**ACKNOWLEDGEMENT**

We would like to express our deepest appreciation to **Dr. Kim Tae Kyung**, the director of Korea Software HRD center, who has handed out our group for the final project of **Seav Phov** website for basic course project. Furthermore, he is an admirable mentor and facilitator who always find solutions to assist us to complete this website. We would like to thank our instructor **Mr. Voy Rathana** and **Mr. Ven Kimsoer** whose contribution in stimulating suggestions and encouragement, helped us to coordinate our project especially in implementing this project.

**PREFACE**

This document is a technical report of **Seav Phov** project for our WEB group, named **Seav Phov**, which is developed and implemented by the 6th generation students of Korea Software HRD center during basic course. The vital thing about this application is that students have studied the techniques and concept of implementing web apps by using spring (Java Programming Language).

1. **INTRODUCTION**

Nowadays, technology is very important for Cambodian like we know Khmer Academy is a website that provides E-Learning service (videos) and **DEH TLAI** is an E-Commerce website. So we want to develop a website that provide E-Reading service called **All Khmer Docs**.

**All Khmer Docs** is a website that provide conveniences for people in finding, sharing their useful documents to others while they can also easily access documents anywhere, anytime whenever they want. **All Khmer Docs** is created by 4th generation students of Korea Software HRD center in 2016 during basic course.

However, the UI (user interface) of **All Khmer Docs** website is still not interesting and some features are not working, so we decide to fix and update website‘s UI in order to attract users to use our applications. We also changed website’s name from **All khmer Docs** to **Seav Phov**.

* 1. **Goals**

**Seav Phov** is created to provide many useful documents in Khmer for Cambodian people. The goals of this website is to collect many useful documents, provide E-Reading Service and make people use documents comprehensively. The website enables people to share their documents to other. When user upload documents to our website, it will be displayed on our website as public content. Our website provides a lot of knowledge for people who like reading, especially students.

* 1. **Motivation**

To provide E-Reading service to Cambodian, our team has developed this **Seav Phov** website with useful features and interesting UI that make user feel comfortable when they use it. We make this document to help you to understand about implementation process. This website users can read, upload, share, comment, download, and manage on their documents.

* 1. **Contribution**

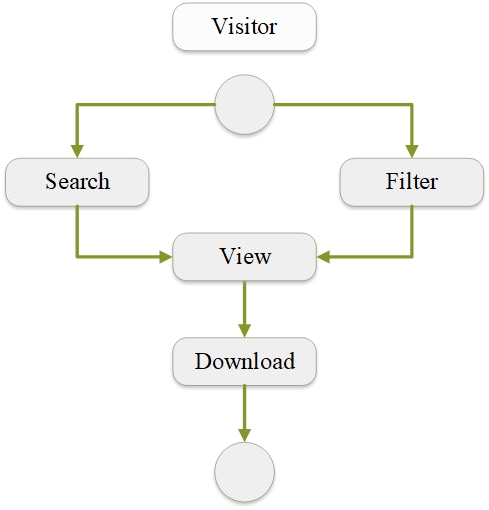
**Seav Phov** is a website that provides a lot of documents in both Khmer and English. In this website we improve UI, features, and add more documents such as document from universities, student’s assignment etc. Users can find documents in this website for read and share to their friends. If users have any idea about documents they also can comment on that documents.

1. **REQUIREMENT**

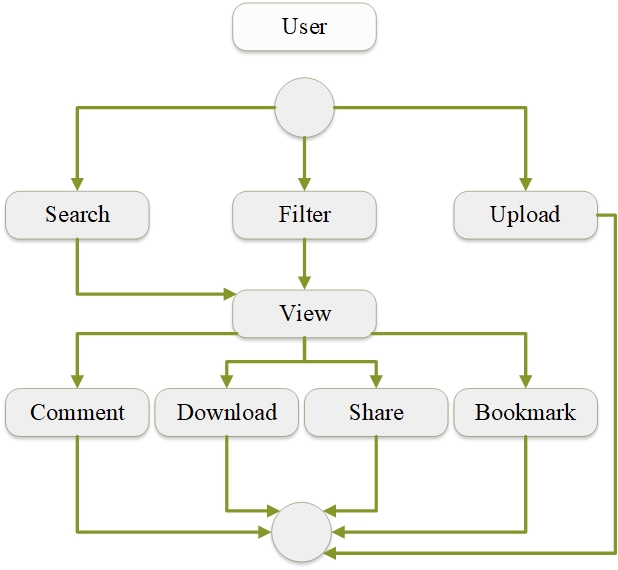
In order to achieve our goal, we need to make some plans. This section will detail what our software need include the Business Modeling and System Requirement Modeling.

* 1. **Business Modeling**

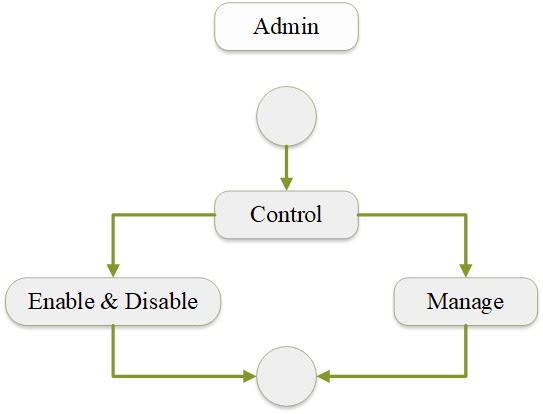
A business model is the way in which a project generates revenue and makes a profit from project operations. Analysts use the metric gross profit as a way to compare the efficiency and effectiveness of a firm's business model. Gross profit is calculated by subtracting the cost of goods sold from revenues. Our web-based application called **Seav Phov** that created in order to provide reading service. The business modeling is told about how users get the resources and share documents to other users. Our web-based application focus on three type of contributors such as: Visitor, User and Admin.



**Figure 1**. Diagram of what visitor can do: visitor can search and filter for documents then can view or read and also can download.



**Figure 2.** Diagram of what user can do: user can search, filter, view, comment, download, share, and bookmark for documents. They can also upload documents if they have.



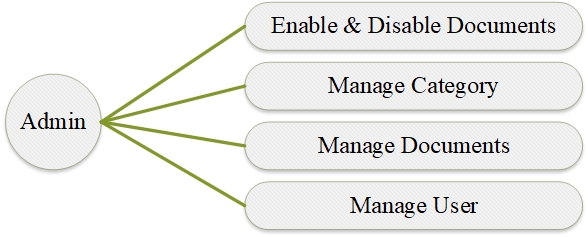
**Figure 3.** Activity Diagram of how admin manage documents and users: the administrative control two point of documents such as enable and disable documents and user’s profile.

* 1. **System Requirement Modeling**
     1. **Problem of Statement**

As the requirement above, all users are possible to search for documents and view. However, there are difficulties such as:

**For visitor:** Cannot comment, bookmark, share, and upload.

**For user:** Cannot restrict on their documents as view-only or downloadable.

* + 1. ** System Requirement**

**Figure 4.** Use Case Diagram: In this Use Case diagram, there are three kinds of contributors: Admin, User, and Visitor. Each of them has their own permission to use our system as follow:

**Admin:** can manage users and documents. They also can enable and disable documents or user’s profile.

**User:** when they login in system to read, comment, bookmark, search, share.

**Visitor:** when they login, they can only search, filter, view and download.

1. **ANALYSIS**

Analysis is the process of studying a procedure or business in order to identify its goals and purposes and create systems and procedures that will achieve them in an efficient way. Another view sees system analysis as a problem-solving technique that breaks down a system into its component pieces for the purpose of the studying how well those component parts work and interact to accomplish their purpose.

* 1. **Purpose**

The Core purpose of developing **Seav Phov** is:

**Admin:**

* Upload and manage documents
* Manage categories
* Create and manage user account
* Enable and disable comment

**User:**

* Login with Facebook
* Search documents
* Filter documents
* View documents
* Download documents
* Comment on Documents
* Upload documents
* Bookmark
* Share to Social Media
* Feedback

**Visitor:**

* Search Documents
* Filter Documents
* View Documents
* Download Documents
* Feedback
  1. **Scope**

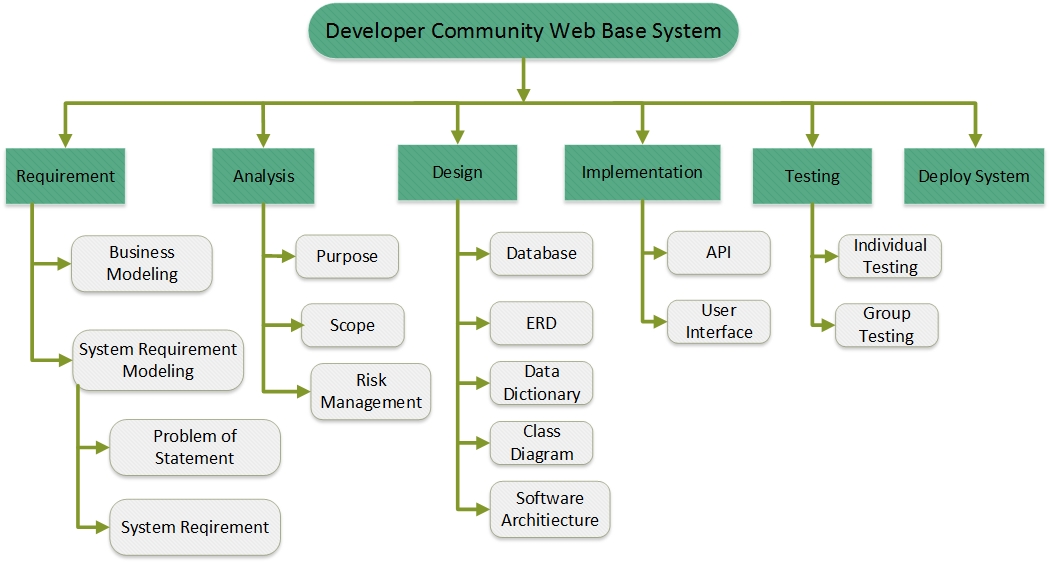
There is some scope’s plan that we have to develop this web application such as:

* Login with Facebook
* Online Reading
* Upload and Download Document
* Search and Filter
* Share to Facebook
* Bookmark
* Comment and Feedback
  1. **Risk Management**

Risk management is the identification, evaluation, and prioritization of risks followed by coordinated and economical application of resources to minimize, monitor, and control the probability or impact of unfortunate events or to maximize the realization of opportunities.

Normally, every project always has risk. So, we have to be identification and management the Risk of project because it is essential part of building project.

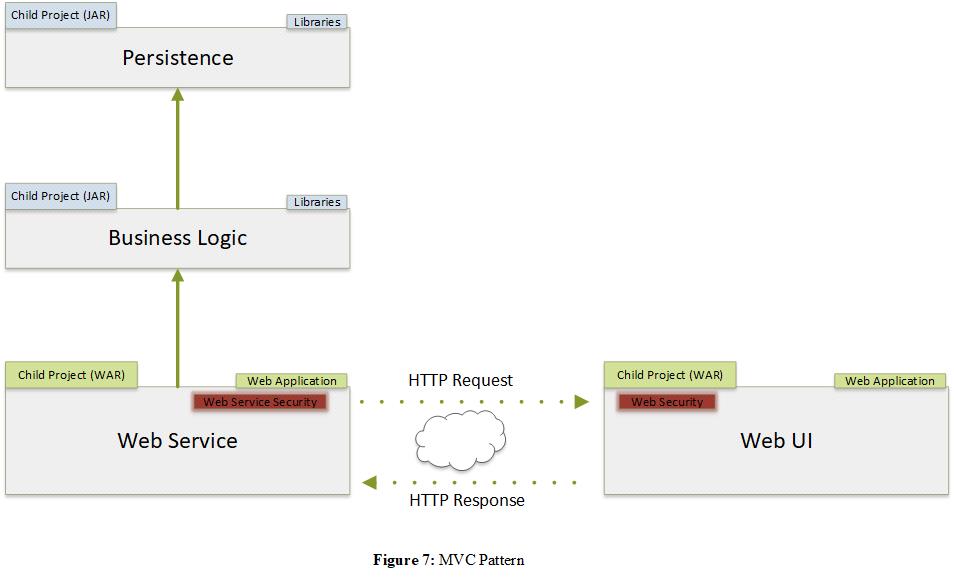
|  |  |
| --- | --- |
| **Risk** | **Description** |
| Crash Project. | Difficult to manage any changing in project requirement or scope. |
| Technical factors | User don’t have experience with system. |
| Server Down | When many users access at the same time. |

1. **PLANNING** 
   1. **Tree Structure View**

**Figure 5.** Tree Structure View, in this picture show about our scope planning.

* 1. **Work Breakdown Structure (WBS)**

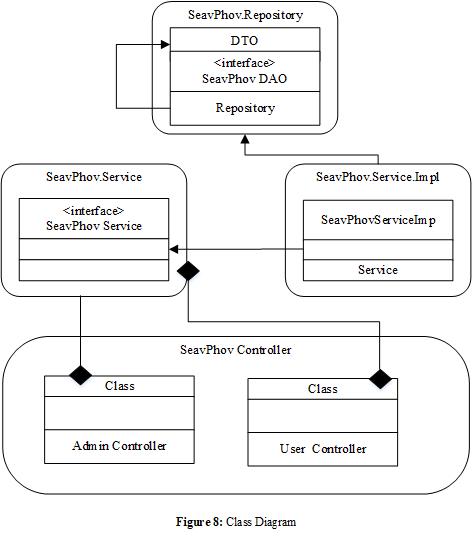
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task Name** | **Start Date** | **End Date** | **Assigned To** | **Duration** |
| 1. Requirement | | | | |
| 1.1 Business Modeling | 12-June-2018 | 14-June-2018 | Chumeng | 3 days |
| 1.2 System Requirement Modeling | 15-June-2018 | 16-June-2018 | Chumeng, Siyeak | 2 days |
| 1.2.1 Problem of Statement | 16-June-2018 | 17-June-2018 | Siyeak | 2 days |
| 1.2.2 System Requirement | 17-June-2018 | 19-June-2018 | Siyeak, Rady | 3 days |
| 2. Analysis | | | | |
| 2.1 Purpose | 19-June-2018 | 20-June-2018 | Rady, Hongly | 2 days |
| 2.2 Scope | 20-June-2018 | 21-June-2018 | Rady | 2 days |
| 2.3 Risk Management | 22-June-2018 | 23-June-2018 | Rady, Hongly | 2 days |
| 3. Planing | | | | |
| 3.1 Tree Structure View | 26-June-2018 | 27-June-2018 | Chumeng | 2 days |
| 3.2 Work Breakdown Structure (WBS) | 27-July-2018 | 29-July-2018 | Hongly | 3 days |
| 4. Design | | | | |
| 4.1 ERD | 2-July-2018 | 3-July-2018 | Chumeng | 2 days |
| 4.2 Data Dictionary | 2-July-2018 | 3-July-2018 | Chumeng | 1 day |
| 4.3 Database Design | 4-July-2018 | 4-July-2018 | Chumeng, Hongly |  |
| 4.4 Spring Boot MVC | 4-July-2018 | 4-July-2018 | Bunhak | 1 day |
| 4.5 Class Diagram | 5-July-2018 | 5-July-2018 | Hongly ,Seiha | 1 day |
| 4.6 Software Architecture | 5-July-2018 | 5-July-2018 | Siyeak, Rady | 1 day |
| 4.8 Interface Design | 6-July-2018 | 15-July-2017 | All Member | 10 days |
| 4.9 Source Code | 23-July-2017 | 10-August-2017 | All Member | 19 days |
| 5. Implement | | | | |
| 5.1 User Front-End | | | | |
| 5.1.2 Home Page | 16-July-2018 | 20-July-2018 | Seiha, Chumeng | 5 days |
| 5.1.3 List Documents by Category Page | 16-July-2018 | 20-July-2018 | Rady, Hongly |
| 5.1.4 Read Details Page | 16-July-2018 | 20-July-2018 | Chumeng, Rady |
| 5.1.5 User Upload | 16-July-2018 | 20-July-2018 | Hongly, Bunhak |
| 5.1.6 User Bookmarked | 19-July-2018 | 20-July-2018 | Seiha, Hongly |
| 5.2 Admin Front-End | | | | |
| 5.2.1 Dashboard | 21-July-2018 | 22-July-2018 | Bunhak | 2 days |
| 5.2.2 Manage Documents | 21-July-2018 | 22-July-2018 | Hongly |
| 5.2.3 Manage Categories | 21-July-2018 | 22-July-2018 | Chumeng |
| 5.2.4 Manage User’s Accounts | 21-July-2018 | 22-July-2018 | Seiha |
| 5.2.5 Manage Comments | 21-July-2018 | 22-July-2018 | Rady |
| 5.3 User Back-End | | | | |
| 5.3.1 Login | 23-July-2018 | 27-July-2018 | Hongly | 5 days |
| 5.3.2 Home Page | 23-July-2018 | 27-July-2018 | Rady | 5 days |
| 5.3.3 Search | 23-July-2018 | 27-July-2018 | Chumeng | 4 days |
| 5.3.4 List Document by Category Page | 27-July-2018 | 02- August -2018 | Hongly ,Chumeng | 7 days |
| 5.3.5 Read Details Document Page | 27- July -2018 | 02- August -2018 | Siyeak, Hongly | 7 days |
| 5.3.6 User report on Document (if not logged in redirect to login) | 27- July -2018 | 31-July-2018 | Rady | 5 days |
| 5.3.7 Bookmark (if not logged in redirect to login) | 27- July -2018 | 31- August -2018 | Hongly | 5 days |
| 5.3.7 Bookmark (if not logged in redirect to login) | 27- July -2018 | 31- August -2018 | Hongly | 5 days |
| 5.3.8 Feedback | 27- July -2018 | 31- July -2018 | Seiha | 5 days |
| 5.3.9 Comment (if not logged in redirect to login) | 27- July -2018 | 02- August -2018 | Chumeng | 7 days |
| 5.3.3 User Upload & User Page | 1-August-2018 | 9-August-2018 | Hongly, Chumeng | 10 days |
| 5.4 Admin Back-End | | | | |
| 5.4.1 Dashboard View | 1-August-2018 | 6-August-2018 | Seiha | 6 days |
| 5.4.2 Feedback | 1-August-2018 | 7-August-2018 | Siyeak | 7 days |
| 5.4.3 Manage Category | 1-August-2018 | 6-August-2018 | Hongly | 6 days |
| 5.4.4 Manage Document | 7-August-2018 | 12-August-2018 | Chumeng, Hongly | 6 days |
| 5.4.5 Manage User’s Account, Role | 7-August-2018 | 12-August-2018 | Rady,Bunhak | 6 days |
| 5.4.6 Report Dashboard | 7-August-2018 | 12-August-2018 | Chumeng | 6 days |
| 7. Submit Assignment | 21-August-2018 | 21-August-2018 | Hongly | 1 days |
| 8. Deploy System | 21-August-2018 | 23-August-2018 | All Member | 3 days |

1. **DESIGN**
   1. **MVC Pattern**

**Figure 7.** MVC Pattern

* 1. **Class Diagram**

There are 4 modules such as:

* business-logic
* persistence
* web-service
* web-ui

**Figure 8.** Class Diagram

* 1. **API Document**

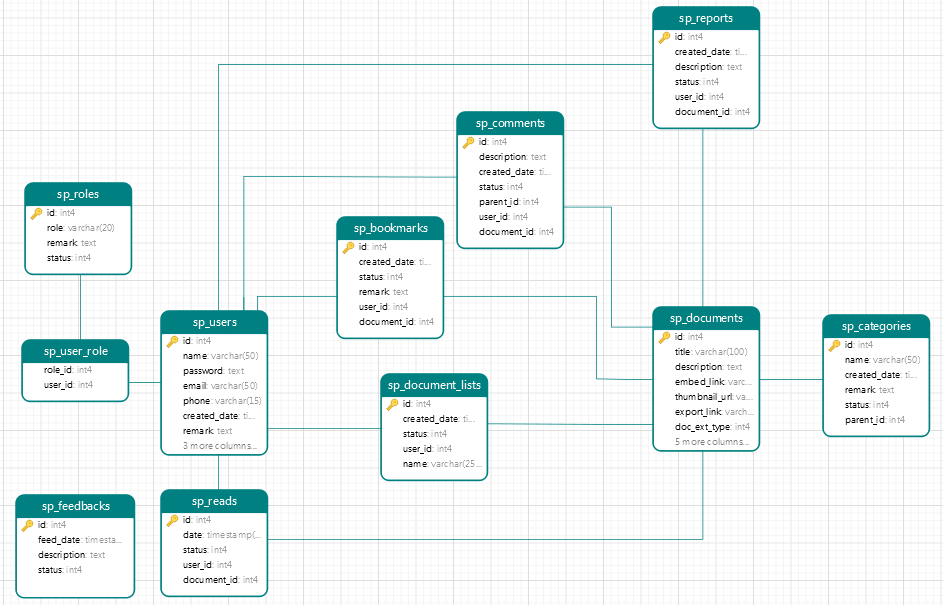
API (Application Programming Interface) documentation, also known as Programmers documentation, is a deliverable of technical writing in which a technical writer develops instructions about how to effectively use a [software API](http://en.wikipedia.org/wiki/Application_programming_interface), hardware, or web-API.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Screen** | **URL** | **Method** | **Request**  **(JSON Format)** | **Response**  **(JSON Format)** | **File** |
| 1 | **BOOKMARK**  addBookmark | /api/v1/bookmark | POST | {  “status”:“string”,  “remark”:”string”,  “userId”:0,  “documentId”:0  } | {  “message”: “Insert Successfully”,  “status”: true  } | BookmarkRestController.java |
| 2 | **CATEGORY**  findCategory | /api/v1/category | GET |  | {  "message": "Category Found",  "status": true  } | CategoryRestController.java |
| 3 | **CATEGORY**  findParentCategory | /api/v1/parent-categories | GET |  | {  "message": "category Found",  "status": true  } | CategoryRestController.java |
| 4 | **CATEGORY**  findChildCategory | /api/v1/child-categories | GET |  | {  "message": "category Found",  "status": true  } | CategoryRestController.java |
| 5 | **CATEGORY**  findOne | /api/v1/category/{id} | GET | {  “id”: 0  } | {  "message": "category Found",  "status": true  } | CategoryRestController.java |
| 6 | **CATEGORY**  insert | /api/v1/category | POST | {  "categoryName": string,  "parentId": 0,  "remark": string, "childCategories": null,  "status": string  } | {  “message”: “Insert Successfully”,  “status”: true  } | CategoryRestController.java |
| 7 | **CATEGORY**  deleteDocument | /api/v1/category/{id} | DELETE | {  “id”: 0  } | {  “message”:”Deleted”,  “status”:true  } | CategoryRestController.java |
| 8 | **CATEGORY**  update | /api/v1/category | PUT | {  “id”: 0  } | {  “message”: “Category has been update successfully”,  “status”:true  } | CategoryRestController.java |
| 9 | **CATEGORY**  updateStatus | /api/v1/categorystatus | PUT | {  “id”: 0  } | {  “message”: “Category has been update successfully”,  “status”:true  } | CategoryRestController.java |
| 10 | **CATEGORY**  findSubLevelTwoCategory | /api/v1/categories-by-parent/{id} | GET | {  “id”: 0  } | {  "message": "category Found",  "status": true  } | CategoryRestController.java |
| 11 | **COMMENT**  updateCommentStatus | /api/v1/admin/comment | PUT | {  “id”: 0  } | {  “message”: “Comment’s status updated”,  “status”:true  } | CommentRestController.java |
| 12 | **COMMENT**  getAllComment | /api/v1/comment/{documentId} | GET | {  “documentId”: 0  } | {  "message": "Retrieve comments successfully!",  "status": true  } | CommentRestController.java |
| 13 | **COMMENT**  addComment | /api/v1/comment | POST | {  “description”: string,  “status”:0,  “parentId”:0,  “reply”:list<Comment>,  “userId”:0,  “documentId”:0  } | {  “message”:”Insert Successfully”,  “status”: true  } | CommentRestController.java |
| 14 | **COMMENT**  getOneNewComment | /api/v1/lastcomment | GET |  | {  “message”:”Retrieve comment successfully”,  “status”:true  } | CommentRestController.java |
| 15 | **COMMENT**  getNewReply | /api/v1/new-reply/{parentId} | GET | {  “parentId”:0  } | {  “message”:”Retrieve comment successfully”,  “status”:true  } | CommentRestController.java |
| 16 | **DOCUMENT**  deleteDocument | /api/v1/documents/{id} | DELETE | {  “id”:0  } | {  “message”:”Document has been delete successfully”,  “status”:true  } | DocumentRestController.java |
| 17 | **DOCUMENT**  findNewUploadedDocument | /api/v1/documents/latest | GET |  | {  “message”:”Retrieved documents successfully”,  “status”:true  } | DocumentRestController.java |
| 18 | **DOCUMENT**  findMostViewedDocuments | /api/v1/documents/most-view | GET |  | {  “message”:”Retrieved documents successfully”,  “status”:true  } | DocumentRestController.java |
| 19 | **DOCUMENT**  findPublicSchoolDocuments | /api/v1/documents/public-document | GET |  | {  “message”:”Retrieved documents successfully”,  “status”:true  } | DocumentRestController.java |
| 20 | **DOCUMENT**  updateDocumentStatus | /api/v1/document | PUT | {  “id”: 0  } | {  “message”:”Document’s status has been updated successfully”,  “status”:true  } | DocumentRestController.java |
| 21 | **FEEDBACK**  feedback | /api/v1/feedback | POST | {  “description”:string,  “status”: true  } | {  “message”:”Insert Successfully”,  “status”:true  } | FeedbackRestController.java |
| 22 | **REPORT**  addReport | /api/v1/report | POST | {  “description”:string,”userId”:0,  “documentId”:0  } | {  “message”:”Insert Successfully”,  “status”:true  } | ReportRestController.java |
| 23 | **REPORT**  findReport | /api/v1/report | GET |  | {  “message”:”Retrieved reports successfully”,  “status”: true  } | ReportRestController.java |
| 24 | **ROLE**  addRole | /api/v1/role | POST | {  “role”:string,  “remark”:string,  “status”:0  } | {  “message”:”Insert Successfully”,  “status”:true  } | RoleRestController.java |
| 25 | **USER ROLE**  updateRole | /api/v1/user-role | PUT | {  “userId”:0  } | {  “message”:”Update User Role Successfully”,  “status”:true  } | UserRestController.java |

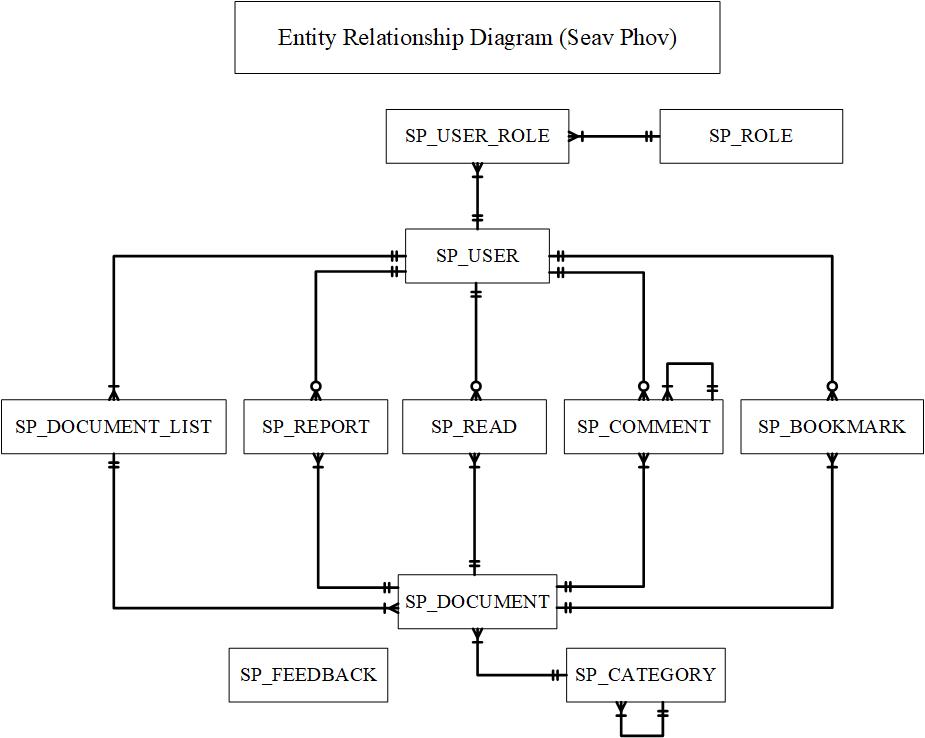
* 1. **Data Dictionary**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Table Name** | **Field Name** | **Datatype** | **Required** | **PK/FK** | **Default** | **Reference** | **Description** |
| sp\_users | id | INTEGER | Y | PK |  |  |  |
| name | VARCHAR(50) | Y |  |  |  |  |
| password | VARCHAR(20) | N |  |  |  |  |
| email | VARCHAR(50) | N |  |  |  |  |
| phone | VARCHAR(15) | N |  |  |  |  |
| created\_date | TIMESTAMP | Y |  | now() |  |  |
| remark | TEXT | N |  |  |  |  |
| status | INTEGER | Y |  | 1 |  | 1: Active  0: Deactivate |
| role\_id | INTEGER | Y | FK |  | sp\_roles |  |
| profile | VARCHAR(100) | N |  |  |  |  |
| social\_id | VARCHAR(100) | N |  |  |  |  |
| sp\_ documents | id | INTEGER | Y | PK |  |  |  |
| title | VARCHAR(50) | Y |  |  |  |  |
| description | TEXT | N |  |  |  |  |
| embed\_link | VARCHAR(255) | Y |  |  |  |  |
| thumbnail\_url | VARCHAR(255) | Y |  |  |  |  |
| export\_link | VARCHAR(255) | Y |  |  |  |  |
| doc\_ext\_type | INTEGER | Y |  |  |  | 1:PDF, 2:PPTX, 3:WORD |
| view | INTEGER | N |  |  |  |  |
| category\_id | INTEGER | Y | FK |  | sp\_categories |  |
| document\_list\_id | INTEGER | Y | FK |  | sp\_document\_lists |  |
| status | INTEGER | Y |  |  |  |  |
| created\_date | TIMESTAMP | Y |  | now() |  |  |
| drive\_file\_thumbnail\_id | VARCHAR(255) | Y |  |  |  |  |
| drive\_file\_document\_id | VARCHAR(255) | Y |  |  |  |  |
| sp\_document\_lists | id | INTEGER | Y | PK |  |  |  |
| name | VARCHAR(50) | Y |  |  |  |  |
| created\_date | TIMESTAMP | Y |  | now() |  |  |
| status | INTEGER | Y |  | 1 |  | 1: Active  0: Deactivate |
| user\_id | INTEGER | Y | FK |  | sp\_users |  |
| drive\_folder\_id | VARCHAR(255) | Y |  |  |  |  |
| sp\_categories | id | INTEGER | Y | PK |  |  |  |
| name | VARCHAR(50) | Y |  |  |  |  |
| created\_date | TIMESTAMP | Y |  | now() |  |  |
| remark | TEXT | N |  |  |  |  |
| status | INTEGER | Y |  | 1 |  | 1: Active  0: Deactivate |
| parent\_id | INTEGER | N |  |  |  |  |
| sp\_reads | id | INTEGER | Y | PK |  |  |  |
| date | TIMESTAMP | Y |  | now() |  |  |
| status | INTEGER | Y |  | 1 |  | 1: Active  0: Deactivate |
| user\_id | INTEGER | Y | FK |  | sp\_users |  |
| document\_id | INTEGER | N | FK |  | sp\_documents |  |
| sp\_comments | id | INTEGER | Y | PK |  |  |  |
| description | TEXT | Y |  |  |  |  |
| created\_date | TIMESTAMP | Y |  | now() |  |  |
| parent\_id | INTEGER | N |  |  |  |  |
| status | INTEGER | Y |  | 1 |  | 1: Active  0: Deactivate |
| document\_id | INTEGER | N | FK |  | sp\_documents |  |
| user\_id | INTEGER | Y | FK |  | sp\_users |  |
| sp\_feedbacks | id | INTEGER | Y | PK |  |  |  |
| feed\_date | TIMESTAMP | N |  | now() |  |  |
| feed\_des | TEXT | N |  |  |  |  |
| status | INTEGER | Y |  | 1 |  | 1: Active  0: Deactivate |
| sp\_bookmarks | id | INTEGER | Y | PK |  |  |  |
| created\_date | TIMESTAMP | Y |  | now() |  |  |
| remark | TEXT | N |  |  |  |  |
| status | INTEGER | Y |  | 1 |  | 1: Active  0: Deactivate |
| document\_id | INTEGER | N | FK |  | sp\_documents |  |
| user\_id | INTEGER | Y | FK |  | sp\_users |  |
| sp\_reports | id | INTEGER | Y | PK |  |  |  |
| created\_date | TIMESTAMP | Y |  | now() |  |  |
| description | TEXT | Y |  |  |  |  |
| document\_id | INTEGER | N | FK |  | sp\_documents |  |
| user\_id | INTEGER | Y | FK |  | sp\_users |  |
| status | INTEGER | Y |  |  |  |  |
| sp\_roles | id | INTEGER | Y | PK |  |  |  |
| name | VARCHAR(20) | Y |  |  |  |  |
| created\_date | TIMESTAMP | Y |  | now() |  |  |
| remark | TEXT | N |  |  |  |  |
| status | INTEGER | Y |  | 1 |  | 1: Active  0: Deactivate |
| sp\_user\_role | user\_id | INTEGER | Y | FK | PK |  | sp\_users |
|  | role\_id | INTEGER | Y | FK |  |  | sp\_roles |

* 1. **Database Diagram**

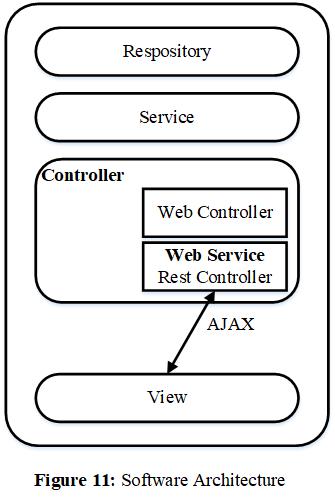
**Figure 9.** Relational Diagram

* 1. **ERD Diagram**

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases.

**Figure 10.** Entity Relational Diagram

* 1. **Software Architecture**

Spring Web Service + AJAX to view data in page, use Thymeleaf for view only

**Figure 11.** Software Architecture

As in the above figure, describe the flow of MVC

* Web Browser sends request for data through AJAX
* AJAX access to Java classes to invoke the Web Service Rest Controller
* Web Service Rest Controller classes invoke to Service classes model architecture
* Service classes interacts with data source(database) and retrieve or manipulation data
* The Web Service Rest Controller response back to the browser

**SELECTED TECHNOLOGY:**

**Server:**

* Spring: use in MVC pattern as Controller which to communicate with Model and View. It is used to request and response from server.

**Client:**

* Spring Boot: is the framework from Spring that designed to simplify develop Spring Application.
* Spring MyBatis: is a persistence framework that automatic mapping among SQL databases and objects.
* HTML: HyperText Markup Language is a set of markup tag used for create web pages.
* CSS: Cascading Style Sheet is the style sheet language used for apply style to the HTML document.

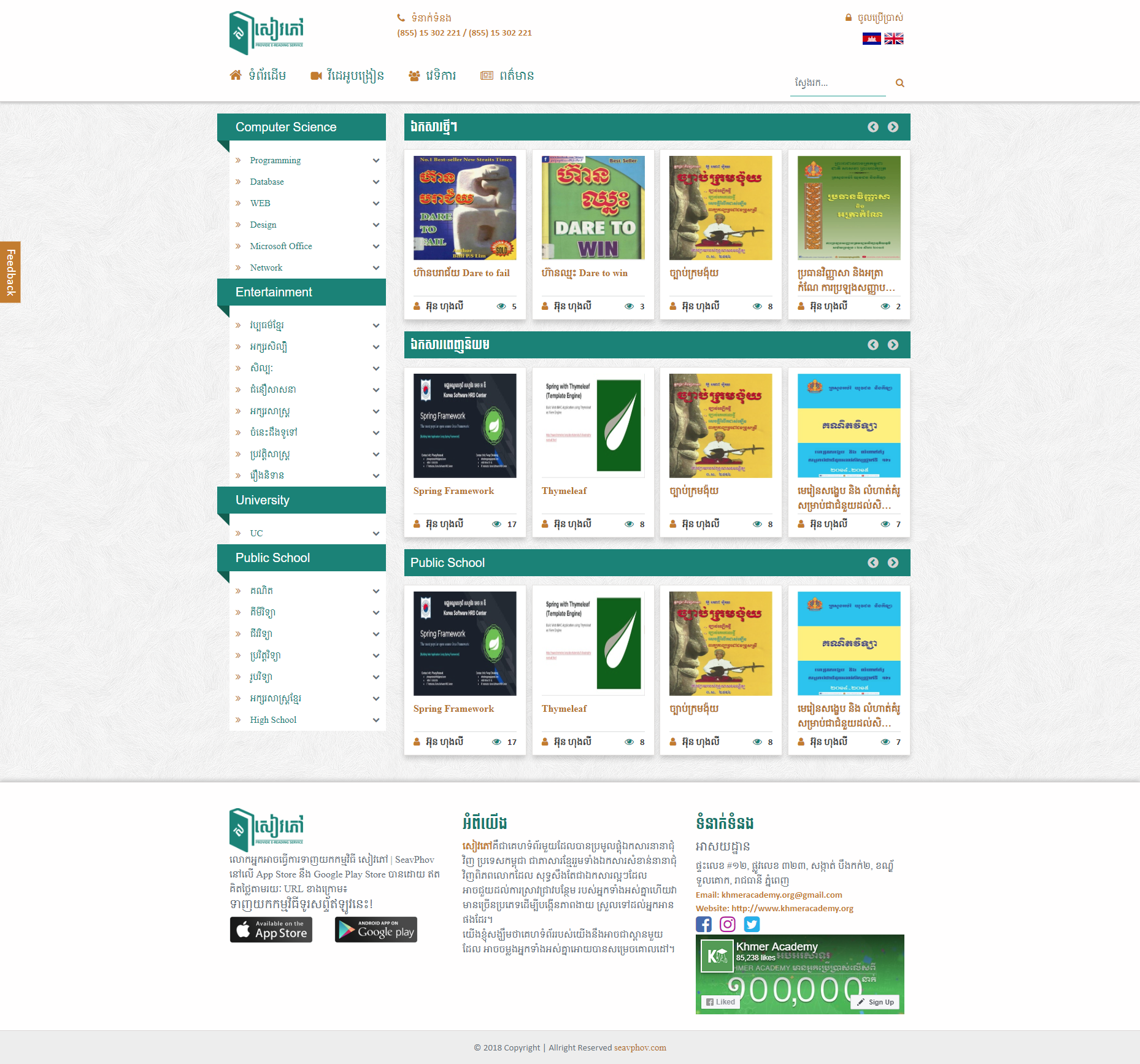
Our application is based on MVC pattern that applied with Spring Web Service and AJAX. The Spring Web Service and AJAX has 4 main components to use in software development:

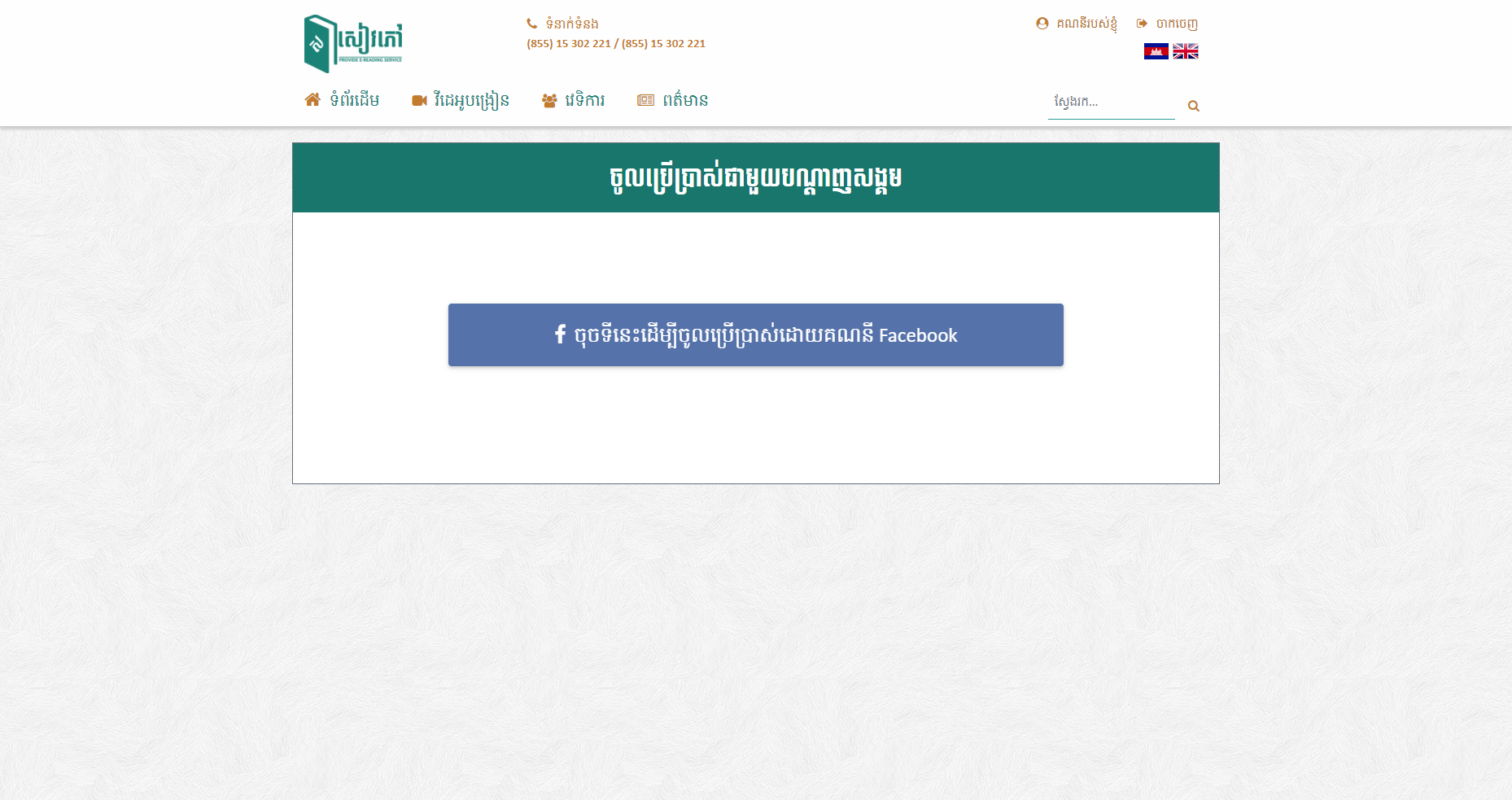
A Repository: repository class serves in the persistence layer of the application as a Data Access Layer (DAO) that contains all your database access logic.

A Service: all your business logics are contains in service class.

A View: is a collection of classes representing the elements in the user interface (all of the things the user can see and respond to the screen, such as buttons, display, boxes, etc)

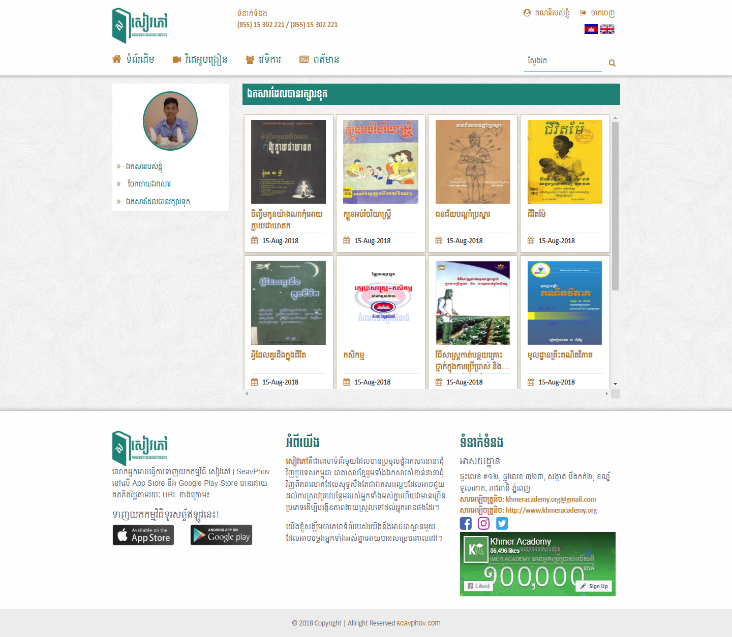
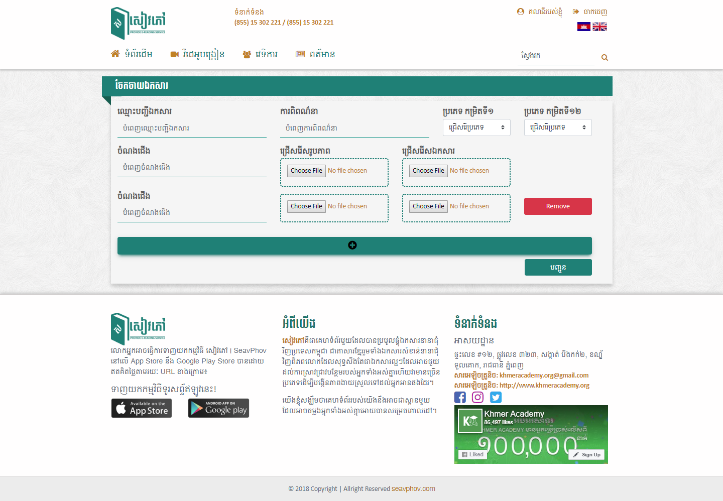
A Controller: represents the classes connecting the service and the view, and is used to communicate between classes in the service and view.

* 1. **Interface Design**

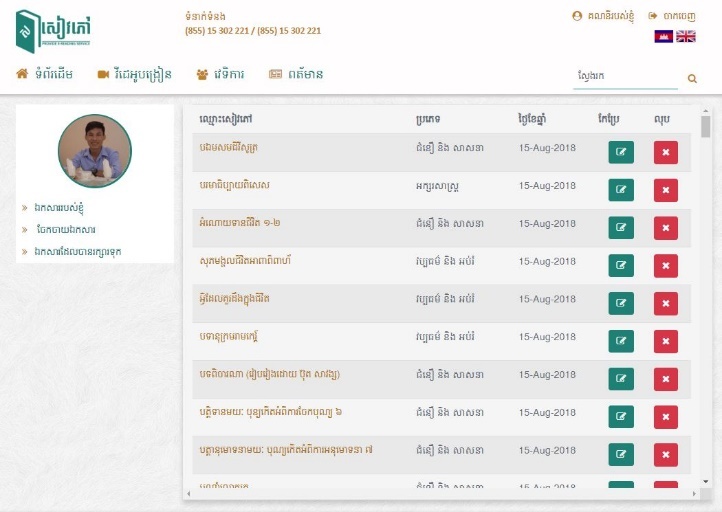
**Figure 12.** Home Page  
**Figure 13.** Login Page



**Figure 14.** Read Page

**Figure 15:** User’s Bookmarked Documents Page

**Figure 16.** Upload Documents Page

  
**Figure 17.** Manage User’s Document Page

* 1. **Source Code**
* **CategoryRepository**.**java**

|  |
| --- |
| package org.khmeracademy.seavphov.repository;  import org.apache.ibatis.annotations.\*;  import org.khmeracademy.seavphov.model.Category;  import org.springframework.stereotype.Repository;  import java.util.List;  @Repository  public interface CategoryRepository {  @Select("SELECT pr.id,pr.name,pr.parent\_id,pr.status FROM sp\_categories pr LEFT JOIN sp\_categories ch ON pr.id = ch.parent\_id " +  "WHERE pr.parent\_id=0 AND pr.status=1 GROUP BY pr.id ORDER BY pr.id ASC")  @Results({  @Result(property = "id",column = "id"),  @Result(property = "categoryName",column = "name"),  @Result(property = "parentId",column = "parent\_id"),  @Result(property = "childCategories", column = "id", many = @Many(select = "findLevelOneChildCategoriesByParentId"))  })  public List<Category> findAllCategories();  @Select("SELECT pr.id,pr.name,pr.parent\_id,pr.status FROM sp\_categories pr LEFT JOIN sp\_categories ch ON pr.id = ch.parent\_id " +  "WHERE pr.parent\_id=#{id} AND pr.status=1 GROUP BY pr.id")  @Results({  @Result(property = "id",column = "id"),  @Result(property = "categoryName",column = "name"),  @Result(property = "parentId",column = "parent\_id"),  @Result(property = "childCategories", column = "id", many = @Many(select = "findLevelTwoChildCategoriesByParentId"))  })  public List<Category> findLevelOneChildCategoriesByParentId(@Param("id") int id);  @Select("SELECT pr.id,pr.name,pr.parent\_id,pr.status FROM sp\_categories pr LEFT JOIN sp\_categories ch ON pr.id = ch.parent\_id " +  "WHERE pr.parent\_id=#{id} AND pr.status=1 GROUP BY pr.id")  @Results({  @Result(property = "id",column = "id"),  @Result(property = "categoryName",column = "name"),  @Result(property = "parentId",column = "parent\_id")  })  public List<Category> findLevelTwoChildCategoriesByParentId(@Param("id") int id);  @Select("SELECT id, name,created\_date, remark, parent\_id,status FROM sp\_categories WHERE id= #{id}")  @Results({  @Result(property = "id",column = "id"),  @Result(property = "categoryName",column = "name"),  @Result(property = "parentId",column = "parent\_id"),  @Result(property = "createdDate",column = "created\_date")  })  public Category findOneCategory(Integer id);  //Find All Categories For Admin  @Select("SELECT id, name,created\_date, remark, parent\_id, status FROM sp\_categories ORDER BY id DESC")  @Results({  @Result(property = "id",column = "id"),  @Result(property = "categoryName",column = "name"),  @Result(property = "parentId",column = "parent\_id"),  @Result(property = "createdDate",column = "created\_date")  })  List<Category> findAllCategoriesForAdmin();  //Insert Category  @Insert("INSERT INTO sp\_categories (name, created\_date, remark, status, parent\_id) VALUES (#{categoryName}, #{createdDate}, #{remark}, 1, #{parentId})")  @Options(useGeneratedKeys = true,keyProperty = "id")  boolean insertCategory(Category category);  //Delete Category  @Delete("DELETE FROM sp\_categories WHERE id = #{id}")  boolean deleteCategory(int id);  @Select("SELECT id,name,parent\_id,status FROM sp\_categories WHERE parent\_id!=0 AND status=1 GROUP BY id ORDER BY id ASC")  @Results({  @Result(property = "id",column = "id"),  @Result(property = "categoryName",column = "name"),  @Result(property = "parentId",column = "parent\_id"),  })  public List<Category> findAllSubCategories();  //Update Category  @Update("UPDATE sp\_categories SET name=#{categoryName},remark=#{remark} WHERE id=#{id}")  boolean updateCategory(Category category);  //Update status Category  @Update("UPDATE sp\_categories SET status=#{status} WHERE id=#{id}")  boolean updateStatusCategory(Category category);  @Select("SELECT id,name,created\_date,remark,parent\_id,status FROM sp\_categories " +  "WHERE parent\_id IN (SELECT id FROM sp\_categories WHERE parent\_id=0)")  @Results({  @Result(property = "id",column = "id"),  @Result(property = "categoryName",column = "name"),  @Result(property = "parentId",column = "parent\_id"),  })  public List<Category> findAllSubLevelOneCategories();  @Select("SELECT id,name,created\_date,remark,parent\_id,status FROM sp\_categories WHERE parent\_id= #{parentId}")  @Results({  @Result(property = "id",column = "id"),  @Result(property = "categoryName",column = "name"),  @Result(property = "parentId",column = "parent\_id"),  })  public List<Category> findAllCategoriesByParentId(int parentId);  //find Parent Category  @Results({  @Result(property = "id",column = "id"),  @Result(property = "categoryName",column = "name")  })  @Select("SELECT id,name FROM sp\_categories WHERE parent\_id=0")  List<Category> findParentCategory();  //find Child Category  @Results({  @Result(property = "id",column = "id"),  @Result(property = "categoryName",column = "name"),  @Result(property = "parentId",column = "parent\_id")  })  @Select("SELECT id, name, created\_date, remark, status, parent\_id FROM sp\_categories WHERE parent\_id IN (SELECT id FROM sp\_categories WHERE parent\_id=0)")  List<Category> findChildCategory();  @Select("SELECT COUNT(id) FROM sp\_categories")  int categoryCount();  } |

* **BookmarkRepository**.**java**

|  |
| --- |
| package org.khmeracademy.seavphov.repository;    import org.apache.ibatis.annotations.Insert;  import org.khmeracademy.seavphov.model.form.Bookmark;  import org.springframework.stereotype.Repository;    @Repository  public interface BookmarkRepository {  @Insert("INSERT INTO sp\_bookmarks (status,user\_id,document\_id)" +  "VALUES(1,#{userId},#{documentId})")  Boolean insert(Bookmark bookmark);  } |

* **CommentRepository.java**

|  |
| --- |
| package org.khmeracademy.seavphov.repository;    import org.apache.ibatis.annotations.\*;  import org.khmeracademy.seavphov.model.Comment;  import org.springframework.stereotype.Repository;  import java.util.List;    @Repository  public interface CommentRepository {    @Select("SELECT com.id, com.description, com.created\_date, com.status, com.parent\_id, com.user\_id, usr.name, usr.profile, com.document\_id, doc.title " +  "FROM sp\_users usr " +  "INNER JOIN sp\_comments com ON usr.id = com.user\_id " +  "INNER JOIN sp\_documents doc ON com.document\_id = doc.id")  @Results({  @Result(property = "id",column = "id"),  @Result(property = "createdDate",column = "created\_date"),  @Result(property = "parentId",column = "parent\_id"),  @Result(property = "user.id",column = "user\_id"),  @Result(property = "user.name",column = "name"),  @Result(property = "user.profile",column = "profile"),  @Result(property = "document.id",column = "document\_id"),  @Result(property = "document.title",column = "title")  })  List<Comment> findAllComments();    @Select("SELECT com.id, com.description, com.created\_date, com.status, com.parent\_id, com.user\_id, usr.name, usr.profile, com.document\_id, doc.title " +  "FROM sp\_users usr " +  "INNER JOIN sp\_comments com ON usr.id = com.user\_id " +  "INNER JOIN sp\_documents doc ON com.document\_id = doc.id " +  "WHERE com.id = #{id}")  @Results({  @Result(property = "id",column = "id"),  @Result(property = "createdDate",column = "created\_date"),  @Result(property = "parentId",column = "parent\_id"),  @Result(property = "user.id",column = "id"),  @Result(property = "user.name",column = "name"),  @Result(property = "user.profile",column = "profile"),  @Result(property = "document.id",column = "id"),  @Result(property = "document.title",column = "title")  })  Comment findOneComment(int id);    @Update("UPDATE sp\_comments SET status=#{status} WHERE id=#{id}")  boolean updateCommentStatus(org.khmeracademy.seavphov.model.form.Comment comment);    @Insert("INSERT INTO sp\_comments" +  "(description, status, parent\_id, user\_id, document\_id) " +  "VALUES(#{description}, #{status}, #{parentId}, #{user.id}, #{document.id})")  @Options(useGeneratedKeys = true,keyProperty = "id")  public boolean insert(Comment comment);    // User Comments  @Select("SELECT cmt.id, cmt.description, cmt.created\_date, cmt.status, cmt.parent\_id, " +  "cmt.user\_id, u.name, u.profile, cmt.document\_id, d.title FROM sp\_users u " +  "INNER JOIN sp\_comments cmt ON cmt.user\_id = u.id " +  "INNER JOIN sp\_documents d ON cmt.document\_id = d.id " +  "WHERE cmt.parent\_id = 0 AND cmt.document\_id = #{documentId} AND cmt.status=1 GROUP BY cmt.id, u.name, u.profile, d.title")  @Results({  @Result(property = "id", column = "id"),  @Result(property = "description", column = "description"),  @Result(property = "createdDate", column = "created\_date"),  @Result(property = "status", column = "status"),  @Result(property = "parentId", column = "parent\_id"),  @Result(property = "reply", column = "id", many = @Many(select = "findUserReplyComment")),  @Result(property = "user.id", column = "user\_id"),  @Result(property = "user.name", column = "name"),  @Result(property = "user.profile", column = "profile"),  @Result(property = "document.id", column = "document\_id"),  @Result(property = "document.title", column = "title")  })  public List<Comment> findAllUserComments(int documentId);    @Select("SELECT cmt.id, cmt.description, cmt.created\_date, cmt.status, cmt.parent\_id, " +  "cmt.user\_id, u.name, u.profile, cmt.document\_id, d.title FROM sp\_users u " +  "INNER JOIN sp\_comments cmt ON cmt.user\_id = u.id " +  "INNER JOIN sp\_documents d ON cmt.document\_id = d.id " +  "WHERE cmt.parent\_id = #{id} AND cmt.status=1 GROUP BY cmt.id, u.name, u.profile, d.title")  @Results({  @Result(property = "id", column = "id"),  @Result(property = "description", column = "description"),  @Result(property = "parentId", column = "parent\_id"),  @Result(property = "user.id", column = "user\_id"),  @Result(property = "user.name", column = "name"),  @Result(property = "user.profile", column = "profile"),  @Result(property = "document.id", column = "document\_id"),  @Result(property = "document.title", column = "title")  })  public List<Comment> findUserReplyComment(@Param("id") int id);    @Select("SELECT cmt.id, cmt.description, cmt.created\_date, cmt.status, cmt.parent\_id, " +  "cmt.user\_id, cmt.document\_id FROM sp\_comments cmt WHERE cmt.id=#{id} AND cmt.status=1")  @Results({  @Result(property = "id", column = "id"),  @Result(property = "description", column = "description"),  @Result(property = "createdDate", column = "created\_date"),  @Result(property = "parentId", column = "parent\_id"),  @Result(property = "user.id", column = "user\_id"),  @Result(property = "document.id", column = "document\_id")  })  public Comment findUserOneComment(int id);    @Select("SELECT cmt.id, cmt.description, cmt.created\_date, cmt.status, cmt.parent\_id, cmt.user\_id, u.name, u.profile, " +  "cmt.document\_id, d.title FROM sp\_users u " +  "INNER JOIN sp\_comments cmt ON cmt.user\_id = u.id " +  "INNER JOIN sp\_documents d ON cmt.document\_id = d.id " +  "WHERE cmt.status=1" +  "ORDER BY cmt.id DESC LIMIT 1")  @Results({  @Result(property = "id", column = "id"),  @Result(property = "description", column = "description"),  @Result(property = "createdDate", column = "created\_date"),  @Result(property = "parentId", column = "parent\_id"),  @Result(property = "user.id", column = "user\_id"),  @Result(property = "user.name", column = "name"),  @Result(property = "user.profile", column = "profile"),  @Result(property = "document.id", column = "document\_id"),  @Result(property = "document.title", column = "title")  })  public Comment findOneLastComment();    @Select("SELECT cmt.id, cmt.description, cmt.created\_date, cmt.status, cmt.parent\_id, cmt.user\_id, u.name, u.profile, " +  "cmt.document\_id, d.title FROM sp\_users u " +  "INNER JOIN sp\_comments cmt ON cmt.user\_id = u.id " +  "INNER JOIN sp\_documents d ON cmt.document\_id = d.id " +  "WHERE cmt.parent\_id = #{parentId} AND cmt.status=1 ORDER BY cmt.id DESC LIMIT 1")  @Results({  @Result(property = "id", column = "id"),  @Result(property = "description", column = "description"),  @Result(property = "createdDate", column = "created\_date"),  @Result(property = "parentId", column = "parent\_id"),  @Result(property = "user.id", column = "user\_id"),  @Result(property = "user.name", column = "name"),  @Result(property = "user.profile", column = "profile"),  @Result(property = "document.id", column = "document\_id"),  @Result(property = "document.title", column = "title")  })  public Comment findNewReply(int parentId);    } |

* **DocumentListRepository.java**

|  |
| --- |
| package org.khmeracademy.seavphov.repository;    import org.apache.ibatis.annotations.\*;  import org.khmeracademy.seavphov.model.DocumentList;  import org.springframework.stereotype.Repository;    @Repository  public interface DocumentListRepository {    @Insert("INSERT INTO sp\_document\_lists(name,drive\_folder\_id,status,user\_id) VALUES(#{name},#{driveFolder},#{status},#{user.id})")  @Options(useGeneratedKeys = true,keyProperty = "id")  public boolean insert(DocumentList documentList);    @Select("SELECT id,name,drive\_folder\_id,created\_date,status,user\_id FROM sp\_document\_lists WHERE id=#{id}")  @Results({  @Result(property = "user.id",column = "user\_id"),  @Result(property = "createdDate",column = "created\_date"),  @Result(property = "driveFolder",column = "drive\_folder\_id")  })  public DocumentList findOneDocumentList(int id);    } |

* **DocumentRepository.java**

|  |
| --- |
| package org.khmeracademy.seavphov.repository;  import org.apache.ibatis.annotations.\*;  import org.khmeracademy.seavphov.model.Comment;  import org.khmeracademy.seavphov.model.Document;  import org.khmeracademy.seavphov.model.DocumentList;  import org.khmeracademy.seavphov.utility.DocumentFilter;  import org.khmeracademy.seavphov.provider.DocumentProvider;  import org.khmeracademy.seavphov.utility.Paging;  import org.springframework.stereotype.Repository;  import java.util.List;  @Repository  public interface DocumentRepository {  @Select("SELECT doc.id, doc.title, doc.description, doc.embed\_link, doc.export\_link, doc.thumbnail\_url, doc.view, doc.status, doc.created\_date, doc.category\_id, cat.name as category\_name,doc.document\_list\_id, dlst.user\_id, usr.name, usr.profile " +  "FROM sp\_users usr " +  "INNER JOIN sp\_document\_lists dlst ON usr.id = dlst.user\_id " +  "INNER JOIN sp\_documents doc ON dlst.id = doc.document\_list\_id " +  "INNER JOIN sp\_categories cat ON doc.category\_id = cat.id")  @Results({  @Result(property = "id", column = "id"),  @Result(property = "title", column = "title"),  @Result(property = "description", column = "description"),  @Result(property = "embedLink", column = "embed\_link"),  @Result(property = "exportLink", column = "export\_link"),  @Result(property = "thumbnail", column = "thumbnail\_url"),  @Result(property = "view", column = "view"),  @Result(property = "status", column = "status"),  @Result(property = "createdDate", column = "created\_date"),  @Result(property = "category.id", column = "category\_id"),  @Result(property = "category.categoryName", column = "category\_name"),  @Result(property = "documentList.id", column = "document\_list\_id"),  @Result(property = "user.id", column = "user\_id"),  @Result(property = "user.name", column = "name"),  @Result(property = "user.profile", column = "profile")  })  List<Document> findAllDocuments();    //================ Find Document by Category or Search =================//  @SelectProvider( type=DocumentProvider.class,method="findAllDocumentFilter")  @Results({  @Result(property = "embedLink", column = "embed\_link"),  @Result(property = "exportLink", column = "export\_link"),  @Result(property = "thumbnail", column = "thumbnail\_url"),  @Result(property = "category.id", column = "category\_id"),  @Result(property = "category.categoryName", column = "category\_name"),  @Result(property = "documentList.id", column = "document\_list\_id"),  @Result(property = "createdDate", column = "created\_date"),  @Result(property = "user.id", column = "user\_id"),  @Result(property = "user.name", column = "name")  })  List<Document> findAllDocumentByFilter(@Param("filter")DocumentFilter filter, @Param("paging")Paging paging);    //================ Find New Uploaded Document=================//  @SelectProvider(method = "findNewUploadedDocuments",type = DocumentProvider.class)  @Results({  @Result(property = "embedLink", column = "embed\_link"),  @Result(property = "exportLink", column = "export\_link"),  @Result(property = "thumbnail", column = "thumbnail\_url"),  @Result(property = "category.id", column = "category\_id"),  @Result(property = "category.categoryName", column = "category\_name"),  @Result(property = "documentList.id", column = "document\_list\_id"),  @Result(property = "createdDate", column = "created\_date"),  @Result(property = "user.id", column = "user\_id"),  @Result(property = "user.name", column = "name")  })  List<Document> findNewUploadedDocuments(@Param("paging") Paging paging);    //================ Find New Uploaded Document=================//  @SelectProvider(method = "findPublicSchoolDocuments",type = DocumentProvider.class)  @Results({  @Result(property = "embedLink", column = "embed\_link"),  @Result(property = "exportLink", column = "export\_link"),  @Result(property = "thumbnail", column = "thumbnail\_url"),  @Result(property = "category.id", column = "category\_id"),  @Result(property = "category.categoryName", column = "category\_name"),  @Result(property = "documentList.id", column = "document\_list\_id"),  @Result(property = "createdDate", column = "created\_date"),  @Result(property = "user.id", column = "user\_id"),  @Result(property = "user.name", column = "name")  })  List<Document> findPublicSchoolDocuments(@Param("paging") Paging paging);    //================ Count Document with filter=================//  @SelectProvider(method="countDocumentFilter", type=DocumentProvider.class)  public Integer countDocumentFilter(DocumentFilter filter);    //================ Count Document with filter=================//  @SelectProvider(method="countPublicSchoolDocuments", type=DocumentProvider.class)  public Integer countPublicSchoolDocuments();      //================ Count All Document=================//  @Select("SELECT COUNT(id) FROM sp\_documents WHERE status=1")  public Integer countAllDocument();    //================ Find Top Viewed Document=================//  @SelectProvider(method = "findMostViewedDocuments",type = DocumentProvider.class)  @Results({  @Result(property = "embedLink", column = "embed\_link"),  @Result(property = "exportLink", column = "export\_link"),  @Result(property = "thumbnail", column = "thumbnail\_url"),  @Result(property = "category.id", column = "category\_id"),  @Result(property = "category.categoryName", column = "category\_name"),  @Result(property = "documentList.id", column = "document\_list\_id"),  @Result(property = "createdDate", column = "created\_date"),  @Result(property = "user.id", column = "user\_id"),  @Result(property = "user.name", column = "name")  })  List<Document> findMostViewedDocuments(@Param("paging") Paging paging);    //================ Find Document by Document's ID=================//  @Select("SELECT doc.id, doc.title, doc.description, doc.embed\_link, doc.export\_link, doc.thumbnail\_url, doc.view, doc.created\_date, doc.category\_id, cat.name as category\_name,doc.document\_list\_id, dlst.user\_id, usr.name, usr.profile " +  "FROM sp\_users usr INNER JOIN sp\_document\_lists dlst ON usr.id = dlst.user\_id " +  "INNER JOIN sp\_documents doc ON dlst.id = doc.document\_list\_id " +  "INNER JOIN sp\_categories cat ON doc.category\_id = cat.id WHERE doc.id=#{id}")  @Results({  @Result(property = "embedLink", column = "embed\_link"),  @Result(property = "exportLink", column = "export\_link"),  @Result(property = "thumbnail", column = "thumbnail\_url"),  @Result(property = "category.id", column = "category\_id"),  @Result(property = "category.categoryName", column = "category\_name"),  @Result(property = "documentList.id", column = "document\_list\_id"),  @Result(property = "createdDate", column = "created\_date"),  @Result(property = "user.id", column = "user\_id"),  @Result(property = "user.name", column = "name")  })  Document findOneDocument(int id);    //================ Find Document by Document's ID=================//  @Select("SELECT doc.id, doc.title, doc.description, doc.embed\_link, doc.export\_link, doc.thumbnail\_url, doc.view, doc.created\_date, doc.category\_id, cat.name as category\_name,doc.document\_list\_id, dlst.user\_id, usr.name, usr.profile " +  "FROM sp\_users usr INNER JOIN sp\_document\_lists dlst ON usr.id = dlst.user\_id " +  "INNER JOIN sp\_documents doc ON dlst.id = doc.document\_list\_id " +  "INNER JOIN sp\_categories cat ON doc.category\_id = cat.id WHERE doc.document\_list\_id=#{documentListId} AND doc.status=1")  @Results({  @Result(property = "embedLink", column = "embed\_link"),  @Result(property = "exportLink", column = "export\_link"),  @Result(property = "thumbnail", column = "thumbnail\_url"),  @Result(property = "category.id", column = "category\_id"),  @Result(property = "category.categoryName", column = "category\_name"),  @Result(property = "documentList.id", column = "document\_list\_id"),  @Result(property = "createdDate", column = "created\_date"),  @Result(property = "user.id", column = "user\_id"),  @Result(property = "user.name", column = "name")  })  List<Document> findDocumentsByDocumentList(Integer documentListId);    //Delete Document  @Delete("DELETE FROM sp\_documents WHERE id=#{id}")  boolean deleteDocument(int id);    //Update Status Document  @Update("UPDATE sp\_documents SET status=#{status} WHERE id=#{id}")  boolean updateDocumentStatus(Document document);    //================ Find Document by Document's ID=================//    @Select("SELECT doc.id, doc.title, doc.description, doc.embed\_link, doc.export\_link, doc.thumbnail\_url, doc.view, doc.created\_date, doc.category\_id, cat.name as category\_name,doc.document\_list\_id, dlst.user\_id, usr.name, usr.profile " +  "FROM sp\_users usr INNER JOIN sp\_document\_lists dlst ON usr.id = dlst.user\_id " +  "INNER JOIN sp\_documents doc ON dlst.id = doc.document\_list\_id " +  "INNER JOIN sp\_categories cat ON doc.category\_id = cat.id WHERE doc.category\_id=#{categoryId} AND doc.status=1 LIMIT 7")  @Results({  @Result(property = "embedLink", column = "embed\_link"),  @Result(property = "exportLink", column = "export\_link"),  @Result(property = "thumbnail", column = "thumbnail\_url"),  @Result(property = "category.id", column = "category\_id"),  @Result(property = "category.categoryName", column = "category\_name"),  @Result(property = "documentList.id", column = "document\_list\_id"),  @Result(property = "createdDate", column = "created\_date"),  @Result(property = "user.id", column = "user\_id"),  @Result(property = "user.name", column = "name")  })  List<Document> findDocumentsByCategoryId(Integer categoryId);    //================ Update Document by Document's ID=================//    @Update("UPDATE sp\_documents SET view= view+1 WHERE id=#{id}")  boolean updateDocumentView(int id);      //================ Retrieve Documents for User bookmark page =================//    @Select("SELECT doc.id, doc.title, doc.description,doc.view, doc.embed\_link, doc.export\_link, doc.thumbnail\_url, doc.document\_list\_id, doc.category\_id, dlst.user\_id ,bmk.created\_date " +  "FROM sp\_users usr " +  "INNER JOIN sp\_bookmarks bmk ON usr.id = bmk.user\_id " +  "INNER JOIN sp\_documents doc ON doc.id = bmk.document\_id " +  "INNER JOIN sp\_document\_lists dlst ON doc.document\_list\_id = dlst.id " +  "WHERE bmk.user\_id=#{userId} AND doc.status=1")  @Results({  @Result(property = "embedLink", column = "embed\_link"),  @Result(property = "exportLink", column = "export\_link"),  @Result(property = "thumbnail", column = "thumbnail\_url"),  @Result(property = "category.id", column = "category\_id"),  @Result(property = "category.categoryName", column = "category\_name"),  @Result(property = "documentList.id", column = "document\_list\_id"),  @Result(property = "createdDate", column = "created\_date"),  @Result(property = "user.id", column = "user\_id"),  @Result(property = "user.name", column = "name"),  @Result(property = "bookmark.createdDate", column = "created\_date")  })  List<Document> findUserBookmarkedDocument(int userId);    //================ Insert Document =================//  @Insert("INSERT INTO sp\_documents(title,description,thumbnail\_url,embed\_link,export\_link,doc\_ext\_type,view,status,category\_id,document\_list\_id) " +  "VALUES(#{title},#{description},#{thumbnail},#{embedLink},#{exportLink},1,#{view},#{status},#{category.id},#{documentList.id})")  boolean insertDocument(Document document);  //============== Select count Document=================//  @Select("SELECT COUNT(id) FROM sp\_documents")  int documentCount();    //================ Find Documents by User's ID =================//  @Select("SELECT doc.id, doc.title, doc.description, doc.embed\_link, doc.export\_link, doc.thumbnail\_url, doc.view, doc.created\_date, doc.category\_id, cat.name as category\_name,doc.document\_list\_id, dlst.user\_id, usr.name, usr.profile " +  "FROM sp\_users usr INNER JOIN sp\_document\_lists dlst ON usr.id = dlst.user\_id " +  "INNER JOIN sp\_documents doc ON dlst.id = doc.document\_list\_id " +  "INNER JOIN sp\_categories cat ON doc.category\_id = cat.id WHERE usr.id=#{userId} ORDER BY doc.id DESC")  @Results({  @Result(property = "embedLink", column = "embed\_link"),  @Result(property = "exportLink", column = "export\_link"),  @Result(property = "thumbnail", column = "thumbnail\_url"),  @Result(property = "category.id", column = "category\_id"),  @Result(property = "category.categoryName", column = "category\_name"),  @Result(property = "documentList.id", column = "document\_list\_id"),  @Result(property = "createdDate", column = "created\_date"),  @Result(property = "user.id", column = "user\_id"),  @Result(property = "user.name", column = "name"),  @Result(property = "bookmark.createdDate", column = "created\_date")  })  public List<Document> findDocumentByUserId(Integer userId);      //================ Insert Document =================//  @Update("UPDATE sp\_documents SET title=#{title},description=#{description},embed\_link=#{embedLink},thumbnail\_url=#{thumbnail},export\_link=#{exportLink},category\_id=#{category.id} WHERE id=#{id}")  boolean updateDocument(Document document);  } |

* **FeedbackRepository.java**

|  |
| --- |
| package org.khmeracademy.seavphov.repository;  import org.apache.ibatis.annotations.Insert;  import org.apache.ibatis.annotations.Result;  import org.apache.ibatis.annotations.Results;  import org.apache.ibatis.annotations.Select;  import org.khmeracademy.seavphov.model.Feedback;  import org.springframework.stereotype.Repository;  import java.util.List;    @Repository  public interface FeedbackRepository {    @Select("SELECT id,feed\_date,description,status FROM sp\_feedbacks")  @Results({  @Result(property = "feedDate",column = "feed\_date")  })  public List<Feedback> findAllFeedback();    @Insert("INSERT INTO sp\_feedbacks(description,status) VALUES(#{description}, 1)")  public boolean insertFeedback(Feedback feedback);    } |

* **ReportRepository.java**

|  |
| --- |
| package org.khmeracademy.seavphov.repository;    import org.apache.ibatis.annotations.Insert;  import org.apache.ibatis.annotations.Result;  import org.apache.ibatis.annotations.Results;  import org.apache.ibatis.annotations.Select;  import org.khmeracademy.seavphov.model.Report;  import org.springframework.stereotype.Repository;  import java.util.List;    @Repository  public interface ReportRepository {    //========Find All Report =========//  @Select("SELECT rpt.id,rpt.description, rpt.user\_id, usr.name, rpt.document\_id, doc.title, doc.thumbnail\_url, rpt.created\_date " +  "FROM sp\_users usr INNER JOIN sp\_reports rpt ON rpt.user\_id = usr.id INNER JOIN sp\_documents doc ON rpt.document\_id = doc.id")  @Results({  @Result(property = "id",column = "id"),  @Result(property = "createdDate",column = "created\_date"),  @Result(property = "user.id",column = "user\_id"),  @Result(property = "user.name",column = "name"),  @Result(property = "document.id",column = "document\_id"),  @Result(property = "document.title",column = "title"),  @Result(property = "document.thumbnail",column = "thumbnail\_url")  })  public List<Report> findAllReports();    @Insert("INSERT INTO sp\_reports(description,status,user\_id,document\_id)" +  "VALUES (#{description},1,#{userId},#{documentId})")  public boolean insert(org.khmeracademy.seavphov.model.form.Report report);    } |

* **RoleRepository.java**

|  |
| --- |
| package org.khmeracademy.seavphov.repository;  import org.apache.ibatis.annotations.\*;  import org.khmeracademy.seavphov.model.Role;  import org.khmeracademy.seavphov.model.form.UserRole;  import org.springframework.stereotype.Repository;    import java.util.List;  @Repository  public interface RoleRepository {    @Select("SELECT id,role,status,remark from sp\_roles")  @Results({  @Result(property = "id",column = "id"),  @Result(property = "status",column = "status"),  @Result(property = "remark",column = "remark"),  @Result(property = "role",column = "role")  })    public List<Role> findAll();    @Insert("INSERT INTO sp\_roles(role,remark,status) VALUES (#{role},#{remark},1)")  @Options(useGeneratedKeys = true,keyProperty = "id")  public boolean addRole(Role role);    @Select("SELECT id,role,status,remark from sp\_roles WHERE id=#{id}")  @Results({  @Result(property = "id", column = "id"),  @Result(property = "role", column = "role"),  @Result(property = "remark", column = "remark"),  @Result(property = "status", column = "status")  })  public Role findOneRole(Integer id);    @Insert("INSERT INTO sp\_user\_role(role\_id,user\_id) VALUES (#{roleId},#{userId})")  public boolean addUserRole(UserRole roleUser);  } |

* **UserRepository.java**

|  |
| --- |
| package org.khmeracademy.seavphov.repository;  import org.apache.ibatis.annotations.\*;  import org.khmeracademy.seavphov.model.Role;  import org.khmeracademy.seavphov.model.User;  import org.springframework.stereotype.Repository;    import java.util.List;    @Repository  public interface UserRepository {  @Select("SELECT id, name, password, email, phone, created\_date, remark, profile, social\_id, status FROM sp\_users")  @Results({  @Result(property = "id",column = "id"),  @Result(property = "name",column = "name"),  @Result(property = "password",column = "password"),  @Result(property = "email",column = "email"),  @Result(property = "phone",column = "phone"),  @Result(property = "createdDate",column = "created\_date"),  @Result(property = "remark",column = "remark"),  @Result(property = "socialId",column = "social\_id"),  @Result(property = "status",column = "status"),  @Result(property = "roles",column = "id", many = @Many(select = "findRoleByUserId")),  })  public List<User> findAllUsers();    @Select("SELECT id, name, password, email, phone, created\_date, remark, profile, social\_id, status FROM sp\_users " +  "WHERE social\_id LIKE #{socialId} AND status=1")  @Results({  @Result(property = "id",column = "id"),  @Result(property = "name",column = "name"),  @Result(property = "password",column = "password"),  @Result(property = "email",column = "email"),  @Result(property = "phone",column = "phone"),  @Result(property = "createdDate",column = "created\_date"),  @Result(property = "remark",column = "remark"),  @Result(property = "socialId",column = "social\_id"),  @Result(property = "status",column = "status"),  @Result(property = "roles",column = "id", many = @Many(select = "findRoleByUserId")),  })  User findOneUserBySocialId(@Param("socialId") String socialId);    @Select("SELECT urol.role\_id,rol.role, rol.remark, rol.status FROM sp\_roles rol INNER JOIN sp\_user\_role urol ON rol.id = urol.role\_id WHERE urol.user\_id=#{id}")  @Results({  @Result(property = "id",column = "role\_id"),  @Result(property = "role",column = "role"),  @Result(property = "remark",column = "remark")  })  List<Role> findRoleByUserId(int id);    @Insert("INSERT INTO sp\_users( name, email, password, phone, remark, status, profile, social\_id) " +  "VALUES (#{name}, #{email}, #{password}, #{phone}, #{remark}, #{status}, #{profile}, #{socialId})")  boolean insert(User user);    @Insert("INSERT INTO sp\_user\_role( user\_id,role\_id ) " +  "VALUES(#{user.id},#{roleId})")  boolean insertUserRole(@Param("user")User user,@Param("roleId") int roleId);    @Select("SELECT COUNT(id) FROM sp\_users")  int userCount();    } |

* **UserController.java**

|  |
| --- |
| package org.khmeracademy.seavphov.controller.user;  import org.khmeracademy.seavphov.model.\*;  import org.khmeracademy.seavphov.service.\*;  import org.khmeracademy.seavphov.service.implement.GoogleDriveServiceImplement;  import org.khmeracademy.seavphov.utility.DocumentFilter;  import org.khmeracademy.seavphov.utility.Paging;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.beans.factory.annotation.Value;  import org.springframework.security.core.context.SecurityContextHolder;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.validation.annotation.Validated;  import org.springframework.web.bind.annotation.\*;  import org.springframework.web.multipart.MultipartFile;  import javax.validation.Valid;  import javax.validation.constraints.NotNull;  import java.io.IOException;  import java.security.GeneralSecurityException;  import java.security.Principal;  import java.util.ArrayList;  import java.util.List;  @Controller  public class UserController {  String parentFolderID="1XDTUNyN3mL--t0ns0R6Vuwm4GKMqiKWb";  CategoryService categoryService;  DocumentService documentService;  CommentService commentService;  ReportService reportService;  DocumentListService documentListService;  GoogleDriveService googleDriveService;  UploadFileToSeverService uploadFileToSeverService;  @Autowired  public void setGoogleDriveService(GoogleDriveService googleDriveService) {  this.googleDriveService = googleDriveService;  }  @Autowired  public void setUploadFileToSeverService(UploadFileToSeverService uploadFileToSeverService) {  this.uploadFileToSeverService = uploadFileToSeverService;  }  @Autowired  public void setDocumentListService(DocumentListService documentListService) {  this.documentListService = documentListService;  }  @Autowired  public void setReportService(ReportService reportService) {  this.reportService = reportService;  }  @Autowired  public void setCategoryService(CategoryService categoryService) {  this.categoryService = categoryService;  }  @Autowired  public void setDocumentService(DocumentService documentService){  this.documentService = documentService;  }  @Autowired  public void setCommentService(CommentService commentService) {  this.commentService = commentService;  }  @GetMapping(value={"/home","/index"})  public String redirectToHomePage(){  return "redirect:/";  }  @GetMapping("/")  public String homePage(Model model,DocumentFilter filter){  List<Category> categories=categoryService.findAllCategories();  model.addAttribute("categories",categories);  model.addAttribute("filter",filter);  return "user/index";  }  @GetMapping("/document")  public String documentList(DocumentFilter filter, Paging paging, Model model){  System.out.println("TITLE="+filter.getTitle());  System.out.println("CAT="+filter.getCategoryId());  if(filter.getTitle()!=null){  filter.setTitle(filter.getTitle().trim());  }  paging.setLimit(12);  List<Category> categories=categoryService.findAllCategories();  List<Document> documents= documentService.findAllDocumentByFilter(filter,paging);  model.addAttribute("categories",categories);  model.addAttribute("documents",documents);  Category cat= new Category();  if(filter.getCategoryId()!=null || filter.getTitle()==null){  cat=categoryService.findOneCategory(filter.getCategoryId());  }else{  cat= new Category();  cat.setCategoryName("Search Result");  }  if(documents.size()>0){  model.addAttribute("slectedCategory",cat);  }else {  Category cat2= new Category();  cat2.setCategoryName("No document found");  model.addAttribute("slectedCategory",cat2);  }  model.addAttribute("filter",filter);  model.addAttribute("paging",paging);  return "user/document";  } @GetMapping("/document/view/{id}/{documentListId}/{categoryId}")  public String viewDocument(Principal principal , @PathVariable("id") Integer id, @PathVariable("documentListId") Integer documentListId,@PathVariable("categoryId") Integer categoryId, DocumentFilter filter, Model model){  documentService.updateDocumentView(id);  model.addAttribute("filter",filter);  model.addAttribute("document",documentService.findOneDocument(id));  model.addAttribute("documentList",documentService.findDocumentsByDocumentList(documentListId));  model.addAttribute("documentsInCategory",documentService.findDocumentsByCategoryId(categoryId));  return "user/view";  }  @GetMapping("/user/bookmark")  public String userBookmark(Model model){  // Authentication authentication = SecurityContextHolder.getContext().getAuthentication();  // String currentPrincipalName = authentication.getPrincipal().getClass());  // System.out.println(currentPrincipalName);  User user = (User)SecurityContextHolder.getContext().getAuthentication().getPrincipal();  int userId = user.getId();  model.addAttribute("user",user);  model.addAttribute("filter",new DocumentFilter());  model.addAttribute("documents",documentService.findUserBookmarkedDocuments(userId));  return "user/user-bookmark";  }  @GetMapping("/user/upload")  public String userUploadPage(Model model){  model.addAttribute("filter",new DocumentFilter());  User user = (User)SecurityContextHolder.getContext().getAuthentication().getPrincipal();  model.addAttribute("user",user);  model.addAttribute("categories",categoryService.findAllSubLevelOneCategories());  return "user/user-upload";  }  @PostMapping ("/user/upload")  public String userUploadFile(@RequestParam("documentListName") String documentListName, @RequestParam("title") List<String> titles, String description, @RequestParam("documentFile") List<MultipartFile> documentFiles, @RequestParam("thumbnail") List<MultipartFile> thumbnails,Integer categoryLevelOne, Integer categoryLevelTwo){  //======= To prevent White Space====//  description=description.trim();  documentListName=documentListName.trim();  int categoryId;  if(categoryLevelTwo!=null){  categoryId = categoryLevelTwo;  }else{  categoryId = categoryLevelOne;  }  //TODO: ==== Store all files information to an ArrayList ==========//  List<FileInfo> files= new ArrayList<>();  for(int i=0;i<documentFiles.size();i++) {  String title = titles.get(i);  String documentPath = uploadFileToSeverService.uploadDocumentFile(documentFiles.get(i), title, documentListName.trim());  String thumbnailPath = uploadFileToSeverService.uploadThumbnailFile(thumbnails.get(i), title, documentListName.trim());  files.add(new FileInfo(title, description, thumbnailPath, documentPath, categoryId, documentListName.trim()));  }  //TODO: ===== Create Folder In Google Drive for storing Documents ==========//  String driveFolder="";  try {  // driveFolder = googleDriveService.createGoogleDriveFolder(documentListName,"1XDTUNyN3mL--t0ns0R6Vuwm4GKMqiKWb");  driveFolder = googleDriveService.createGoogleDriveFolder(documentListName,parentFolderID);  } catch (GeneralSecurityException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  }  //TODO: ===== Create Document List for User's Document in Database and then get it back ==========//  DocumentList documentList= new DocumentList();  documentList.setName(documentListName);  User user = (User)SecurityContextHolder.getContext().getAuthentication().getPrincipal();  documentList.setUser(user);  documentList.setDriveFolder(driveFolder);  documentList.setStatus(1);  if(documentListService.insert(documentList)){  documentList =documentListService.findOneDocumentList(documentList.getId());  }  //TODO: ===== Upload each file to Google Drive and then return Document Object from the upload method for insert to database ==========//  List<Document> documents = new ArrayList<>();  Category category = categoryService.findOneCategory(categoryId);  for(FileInfo file : files){  try {  documents.add(googleDriveService.uploadDocumentsToGoogleDrive(file.getTitle(),file.getDescription(),file.getThumbnailFilePath(),file.getDocumentFilePath(),category,documentList,driveFolder));  } catch (GeneralSecurityException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  }  }  //TODO: ===== Insert each Document to database ==========//  for(Document document : documents){  documentService.insertDocument(document);  }  String folder = "/opt/SEAVPHOV";  java.io.File path = new java.io.File(folder);  if(path.exists()){  path.delete();  }  return "redirect:/";  }  @GetMapping("/user/document")  public String userDocument(Model model){  User user = (User)SecurityContextHolder.getContext().getAuthentication().getPrincipal();  int userId = user.getId();  model.addAttribute("user",user);  model.addAttribute("documents",documentService.findDocumentsByUserId(userId));  model.addAttribute("filter",new DocumentFilter());  return "user/user-document";  }  @GetMapping("/user/document/{id}")  public String userDocument(Model model,@PathVariable("id") int id){  Document document=documentService.findOneDocument(id);  model.addAttribute("document",document);  model.addAttribute("categories",categoryService.findAllSubLevelOneCategories());  model.addAttribute("documentList",documentListService.findOneDocumentList(document.getDocumentList().getId()));  model.addAttribute("filter",new DocumentFilter());  return "user/user-document-operation";  }  @PostMapping ("/user/document/update")  public String userUpdateDocument( @ModelAttribute Document document, @RequestParam("documentFile") MultipartFile documentFile, @RequestParam("thumbnailFile") MultipartFile thumbnailFile,@RequestParam("categoryLevelOne") Integer categoryLevelOne, Integer categoryLevelTwo){  int categoryId;  if(categoryLevelTwo!=null){  categoryId = categoryLevelTwo;  }else{  categoryId = categoryLevelOne;  }  //TODO: ===== find DocumentList from database in order to get some existing information included Document's google drive folder ==========//  DocumentList documentList = documentListService.findOneDocumentList(document.getDocumentList().getId());  //TODO: ===== Upload the new file to server(temporary file) in order get path of file (we can not get full path of file from multipart file so we need to use this trick) ==========//  String documentPath;  String thumbnailPath;  if(!documentFile.isEmpty()) {  documentPath = uploadFileToSeverService.uploadDocumentFile(documentFile, document.getTitle(), documentList.getName());  }else {  documentPath=null;  }  if(!thumbnailFile.isEmpty()) {  thumbnailPath = uploadFileToSeverService.uploadThumbnailFile(thumbnailFile, document.getTitle(), documentList.getName());  }else {  thumbnailPath=null;  }  //TODO: ===== Get the existing google drive folder of document ==========//  String driveFolder = documentList.getDriveFolder();  //TODO: ===== Upload each file to Google Drive and then return Document Object from the upload method for insert to database ==========//  Category category = categoryService.findOneCategory(categoryId);  Document documentFromDB=documentService.findOneDocument(document.getId());  // String thumbnailIdInGDrive=documentFromDB.  Document document1=new Document();  try {  if(documentPath!=null && thumbnailPath==null){  document1=googleDriveService.uploadDocumentsToGoogleDrive(document.getTitle(),document.getDescription(),null,documentPath,category,documentList,driveFolder);  documentFromDB.setEmbedLink(document1.getEmbedLink());  documentFromDB.setExportLink(document1.getExportLink());  documentFromDB.setTitle(document.getTitle());  documentFromDB.setDescription(document.getDescription());  googleDriveService.delete(documentFromDB.getDocumentDriveFile());  documentFromDB.setDocumentDriveFile(document1.getDocumentDriveFile());  documentFromDB.setCategory(category);  //TODO DELETE OLD DOCUMENT FILE IN DRIVE  }else if(thumbnailPath != null && documentPath == null){  document1=googleDriveService.uploadDocumentsToGoogleDrive(document.getTitle(),document.getDescription(),thumbnailPath,null,category,documentList,driveFolder);  documentFromDB.setThumbnail(document1.getThumbnail());  documentFromDB.setTitle(document.getTitle());  documentFromDB.setDescription(document.getDescription());  googleDriveService.delete(documentFromDB.getThumbnailDriveFile());  documentFromDB.setThumbnailDriveFile(document1.getThumbnailDriveFile());  documentFromDB.setCategory(category);  //TODO DELETE OLD THUMBNAIL FILE IN DRIVE  }else if(documentPath==null && thumbnailPath==null){  documentFromDB.setTitle(document.getTitle());  documentFromDB.setDescription(document.getDescription());  documentFromDB.setCategory(category);  }  else {  document1=googleDriveService.uploadDocumentsToGoogleDrive(document.getTitle(),document.getDescription(),thumbnailPath,documentPath,category,documentList,driveFolder);  documentFromDB.setEmbedLink(document1.getEmbedLink());  documentFromDB.setExportLink(document1.getExportLink());  documentFromDB.setThumbnail(document1.getThumbnail());  documentFromDB.setTitle(document.getTitle());  googleDriveService.delete(documentFromDB.getDocumentDriveFile());  googleDriveService.delete(documentFromDB.getThumbnailDriveFile());  documentFromDB.setDocumentDriveFile(document1.getDocumentDriveFile());  documentFromDB.setThumbnailDriveFile(document1.getThumbnailDriveFile());  documentFromDB.setDescription(document.getDescription());  documentFromDB.setCategory(category);  }  } catch (GeneralSecurityException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  }  String folder = "/opt/SEAVPHOV";  java.io.File path = new java.io.File(folder);  if(path.exists()){  path.delete();  }  //TODO: ===== Update Document to database ====//  documentService.updateDocument(documentFromDB);  return "redirect:/";  }  } |

* **AdminController.java**

|  |
| --- |
| package org.khmeracademy.seavphov.controller.admin;  import org.khmeracademy.seavphov.model.User;  import org.khmeracademy.seavphov.service.\*;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.security.core.context.SecurityContextHolder;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.ui.ModelMap;  import org.springframework.web.bind.annotation.\*;  @Controller  public class AdminController {  RoleService roleService;  @Autowired  public void setRoleService(RoleService roleService) {  this.roleService = roleService;  }  //Inject Feedback's Service  FeedbackService feedbackService;  @Autowired  public void setFeedbackService(FeedbackService feedbackService) {  this.feedbackService = feedbackService;  }  //Inject User's Service  UserService userService;  @Autowired  public void setUserService(UserService userService) {  this.userService = userService;  }  //Inject Report's Service  private ReportService reportService;  @Autowired  public void setReportService(ReportService reportService) {  this.reportService = reportService;  }  //Inject Document's Service  private DocumentService documentService;  @Autowired  public void setDocumentService(DocumentService documentService) {  this.documentService = documentService;  }  //Inject Comment's Service  private CommentService commentService;  @Autowired  public void setCommentService(CommentService commentService) {  this.commentService = commentService;  }  //Inject Comment's Service  private CategoryService categoryService;  @Autowired  public void setCategoryService(CategoryService categoryService) {  this.categoryService = categoryService;  }  // Controller of feedback  @GetMapping("/admin/feedback")  public String feedback(Model model){  User user = (User) SecurityContextHolder.getContext().getAuthentication().getPrincipal();  model.addAttribute("user",user);  model.addAttribute("feedbacks",feedbackService.findAllFeedback());  return "admin/feedback";  }  //Controller of Role Managemnet  @GetMapping("/admin/role")  public String role(Model model){  User user = (User) SecurityContextHolder.getContext().getAuthentication().getPrincipal();  model.addAttribute("user",user);  model.addAttribute("allroles",roleService.findAll());  return "admin/role";  }  // Controller of Report  @GetMapping("/admin/report")  public String report(ModelMap model) {  User user = (User) SecurityContextHolder.getContext().getAuthentication().getPrincipal();  model.addAttribute("user",user);  model.addAttribute("reports",reportService.findAllReports());  return "admin/report";  }  // Controller of document  @GetMapping("/admin/document")  public String document(ModelMap model) {  User user = (User) SecurityContextHolder.getContext().getAuthentication().getPrincipal();  model.addAttribute("user",user);  model.addAttribute("documents",documentService.findAllDocuments());  return "admin/document";  }  // Controller of deleteDocument  @GetMapping("/admin/document/{id}")  public String deleteDocument(@PathVariable("id") int id) {  documentService.deleteDocument(id);  return "admin/document";  }  // Controller of category  @GetMapping("/admin/category")  public String category(ModelMap model) {  User user = (User) SecurityContextHolder.getContext().getAuthentication().getPrincipal();  model.addAttribute("user",user);  model.addAttribute("categories",categoryService.findAllCategoriesForAdmin());  return "admin/category";  }  // Controller of user  @GetMapping("/admin/user")  public String user(Model model) {  User user = (User) SecurityContextHolder.getContext().getAuthentication().getPrincipal();  model.addAttribute("user",user);  model.addAttribute("allroles",roleService.findAll());  model.addAttribute("users",userService.findAllUsers());  return "admin/user";  }  // Controller of dashboard  @GetMapping(value ={"/admin/dashboard", "/admin"})  public String dashboard(Model model) {  User user = (User) SecurityContextHolder.getContext().getAuthentication().getPrincipal();  model.addAttribute("user",user);  model.addAttribute("documents",documentService.findAllDocuments());  model.addAttribute("usercount",userService.userCount());  model.addAttribute("documentcount",documentService.documentCount());  model.addAttribute("categorycount",categoryService.categoryCount());  return "admin/dashboard";  }  // Controller of comment  @GetMapping("/admin/comment")  public String comment(ModelMap model) {  User user = (User) SecurityContextHolder.getContext().getAuthentication().getPrincipal();  model.addAttribute("user",user);  model.addAttribute("comments",commentService.findAllComments());  return "admin/comment";  }  // find one comment  @GetMapping("/admin/comment/{id}")  public String comment(@PathVariable("id") int id) {  return "comment";  }  } |

* **FacebookController.java**

|  |
| --- |
| package org.khmeracademy.seavphov.controller.facebook;  import com.github.scribejava.apis.FacebookApi;  import com.github.scribejava.core.builder.ServiceBuilder;  import com.github.scribejava.core.model.OAuth2AccessToken;  import com.github.scribejava.core.model.OAuthRequest;  import com.github.scribejava.core.model.Response;  import com.github.scribejava.core.model.Verb;  import com.github.scribejava.core.oauth.OAuth20Service;  import org.json.JSONObject;  import org.khmeracademy.seavphov.model.Role;  import org.khmeracademy.seavphov.model.User;  import org.khmeracademy.seavphov.service.UserService;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.beans.factory.annotation.Value;  import org.springframework.context.annotation.PropertySource;  import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  import org.springframework.security.core.Authentication;  import org.springframework.security.core.context.SecurityContextHolder;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestParam;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import java.io.IOException;  import java.util.ArrayList;  import java.util.List;  import java.util.Random;  import java.util.UUID;  @Controller  @RequestMapping("/facebook")  @PropertySource("classpath:fblogin.properties")  public class FacebookController {  @Value("${fb.login.appid}")  private String FB\_APP\_ID;  @Value("${fb.login.secret}")  private String FB\_APP\_SECRET;  @Value("${fb.login.domain}")  private String DOMAIN;  UserService userService;  @Autowired  public void setUserService(UserService userService) {  this.userService = userService;  }  private static final String CALLBACK\_URL = "/facebook/callback";  private static final List<String> SCOPES = new ArrayList<String>() {  private static final long serialVersionUID = 1L;  {  add("public\_profile");  add("email");  add("user\_gender");  }  };  // Facebook API get user information  private static final String USER\_PROFILE\_API\_URL = "https://graph.facebook.com/v2.8/me"  + "?fields=id,name,first\_name,last\_name,gender,email,picture";  @GetMapping("/signin")  public void signin(HttpServletRequest request, HttpServletResponse response) throws IOException {  String secretState = "secret" + new Random().nextInt(999\_999);  request.getSession().setAttribute("SECRET\_STATE", secretState);  @SuppressWarnings("deprecation")  OAuth20Service service = new ServiceBuilder()  .apiKey(FB\_APP\_ID)  .apiSecret(FB\_APP\_SECRET)  .callback(DOMAIN + CALLBACK\_URL)  .scope(String.join(",", SCOPES))  .state(secretState)  .build(FacebookApi.instance());  String authorizeUrl = service.getAuthorizationUrl();  // System.out.println("Authorize URL: " + authorizeUrl);  response.sendRedirect(authorizeUrl);  }  @GetMapping(value = "/callback")  public String callback(@RequestParam(value = "code", required = false) String code,  @RequestParam(value = "state", required = false) String state,  HttpServletRequest request,  HttpServletResponse response) {  try {  request.getSession().getAttribute("SECRET\_STATE");  @SuppressWarnings("deprecation")  OAuth20Service service = new ServiceBuilder()  .apiKey(FB\_APP\_ID)  .apiSecret(FB\_APP\_SECRET)  .callback(DOMAIN + CALLBACK\_URL)  .build(FacebookApi.instance();  final String requestUrl = USER\_PROFILE\_API\_URL;  final OAuth2AccessToken accessToken = service.getAccessToken(code);  final OAuthRequest oauthRequest = new OAuthRequest(Verb.GET, requestUrl);  service.signRequest(accessToken, oauthRequest);  final Response resourceResponse = service.execute(oauthRequest);  // System.out.println("RESPONSE CODE: " + resourceResponse.getCode() + ", MESSAGE: " + resourceResponse.getMessage());  // System.out.println("BODY: " + resourceResponse.getBody());  final JSONObject obj = new JSONObject(resourceResponse.getBody());  // System.out.println("JSON BODY: " + obj.toString()); request.getSession().setAttribute("FACEBOOK\_ACCESS\_TOKEN", accessToken);  User existingUser=userService.findOneUserBySocialId(obj.getString("id"));  if(existingUser==null){  User newUser= new User();  newUser.setName(obj.getString("name"));  try {  newUser.setEmail(obj.getString("email"));  }  catch(Exception e) { newUser.setEmail(null); }  newUser.setPassword(UUID.randomUUID().toString());  newUser.setSocialId(obj.getString("id"));  newUser.setRemark("FACEBOOK USER");  newUser.setStatus(1);  // TODO: 7/24/2018  // JSONObject picture=obj.getJSONObject("picture").getJSONObject("data");  //==== User profile url ===/  String profileUrl = "http://graph.facebook.com/"+obj.getString("id")+"/picture?width=300&height=300";  newUser.setProfile(profileUrl);  //=====Register User to System=======//  userService.insert(newUser);  //=====GET registered USER back=======//  User user = userService.findOneUserBySocialId(obj.getString("id"));  // System.out.println(user);  //=====SET USER ROLE=======/  userService.insertUserRole(user,2);  }  User authUser = userService.findOneUserBySocialId(obj.getString("id"));  //=======Manual Authentication to System========//  Authentication auth = new UsernamePasswordAuthenticationToken(authUser, null, authUser.getRoles());  SecurityContextHolder.getContext().setAuthentication(auth);  System.out.println(">>User login successfully<<");  return "redirect:/";  } catch (Exception e) {  e.printStackTrace();  return "/login";  }  }  } |

1. **EVALUATION**

**Seav Phov** has been updated from **All Khmer Docs** website. In updated website, we have been improved some features and update user interface to be more useful for users. However, every project always have strength and weakness.

* 1. **Project Strength**

Strengths in **Seav Phov** have:

* Sign up using social account
* User can upload documents
* User can manage their uploaded documents
* Download document.
  1. **Project Weakness**

Even we have a lot of strengths but we also have weakness:

* Document Owner cannot restrict on their documents (view only or downloadable).

1. **CONCLUSION**

In addition, **Seav Phov** website is updated from “All Khmer Docs” project. It comes with a new UI which makes users interact better and feel more comfortable in using it. Fixed features make them fully use features. This project **Seav Phov** will help Cambodian in finding good documents for reading. Since the number of documents is increased and most of documents are in Khmer language, people can do research much more efficiently. We hope **Seav Phov** will contribute to enhancement of education system and development of human resources in Cambodia.

**REFERENCES**

1. Interface Design: http://enovel.sabay.com.kh/book, http://docs.khmeracademy.org/
2. Features: http://docs.khmeracademy.org/