**ABOUT AUTHOR**


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 **Mr. Kim Han-Soo**, the country director of Korea Software HRD center, who has handed out our group for the final project of **Khmer Anguli** website for basic course project. Furthermore, he is an admirable supervisor and facilitator who always find solutions to assist us in completing this website. We would also like to thank our mentors **Mr. Kea Kimleang** and **Mr. Kao Leangseng,** as well as other mentors, whose contributions are to stimulate suggestions and inspirations, and helped us to coordinate our project, especially in implementing this project.

**PREFACE**

This document is a technical report of **Khmer Anguli** project for our WEB group, named **Khmer Anguli**, which is developed and implemented by the 9th generation students of Korea Software HRD center during basic course. The vital part about this website is that students have studied the techniques and concept of implementing web applications by using Spring Boot (Spring Framework) along with ReactJS (JavaScript library).

## I INTRODUCTION

In the world of technology, computer has become an important device for various purposes such as document preparation, storing data, writing code, communicating and others. In those cases, typing skills are important to get the work done faster. Based on this reason, we have come up with a project to create a website called **Khmer Anguli**. **Khmer Anguli** is a website developed by the 9th generation students of Korea Software HRD during the basic course in 2021. It is a website that allows users to learn Khmer typing, by choosing categories, articles, and multiple levels such as practice, medium and advanced level. Moreover, the users can save their typing result, and chat with other users. On the other hand, the users can attend the event to join and compete with others. The user account, articles and categories are managed by administrators such as insert, update, and delete. Furthermore, the administrator can disable the users from the website. Especially when they go against the policy, the administrator will be able to disable them directly.

### 1.1 Goals

**Khmer Anguli** is created by our team to provide many useful features for people who are new to Khmer keyboard typing. The website's goal is to help people practice and learn Khmer typing skills on keyboard efficiency. Specifically, the primary focus of the website that matters is to help people choose the levels that are fit for their current skills. It will enable visitors to be able to play directly on the browser. Moreover, they can type with their favorite categories and articles in both solo challenge and event mode. Therefore, our website will distribute the best Khmer typing experiences and skills with practice, solo challenge, and event mode.

### 1.2 Motivation

To provide better work for users in Cambodia, our team has developed this **Khmer Anguli** website with useful features and clean UI that make users feel comfortable when they are using it. We create this document to help customers to understand the implementation process. This website, administrators can manage all users, articles, categories that are illegal by deleting them from the site itself. Users can view articles in all categories to type. Especially we have our chat platform for users to contact each other. Users can use this website anywhere anytime. They can use it through a smartphone or a computer just by connecting to the internet.

### 1.3 Contribution

**Khmer Anguli** is a website that provides an easy way to learn Khmer keyboard typing with many options. They can select practice mode for fresh start or solo challenge and event mode with error checking, word per minute, and correctness which are helpful to help them learn carefully. They can get in any typing mode, use chat to stay connected, and save typing results or view other top players. On this website, administrators will be able to delete, insert, and update categories, articles, and users. Also, visitors can get their typing mode, create communication with chat.

## II REQUIREMENT

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

### 2.1 Business Modeling

A business modeling refers to a company's plan for making a profit. It identifies the products or services the business plans to sell. Our web-based application called **Khmer Anguli** was created to provide services to consumers. Our web-based application focuses on two main contributors such as administrators and users.

**Figure 1**. Activity Diagram of how admin controls website's information: The administrators fully control on the website, such as categories, article, events, and users. In addition, the administrator can manage the categories and articles within the category such as insert, delete, and update. The administrators have to choose any level before adding a category or article. Moreover, the administrator is able to view or disable users who go against the policy of the website.

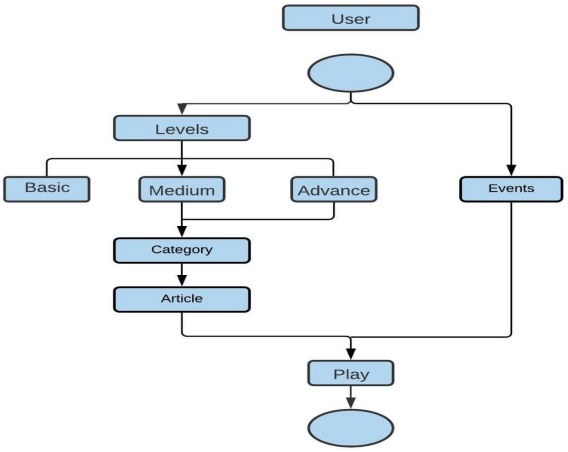



**Figure 2.** Activity Diagram of how users play on our website, and control their information:
Users can play on **Khmer Anguli** website directly, join events, or select levels to play, and choose category and article to play.

### 2.2 System Requirement Modeling

#### 2.2.1 Problem of Statement

As the requirements mentioned above, our administrator can do what they are authorized to do, such as search, update, and delete based on their authorization, whereas users can do what they are authorized to do, such as playing on our website directly, choosing categories, articles to type, saving account, and other functionalities. Nevertheless, there are still some problems related to the process of our website. For instance, our site doesn't involve any invited methods to compete between one and only one user; Therefore, it is hard when they want to play with their partner. Moreover, the website doesn't include team competition, so the users can be disappointed when they want to compete with their team members only.

#### 2.2.2 System Requirement

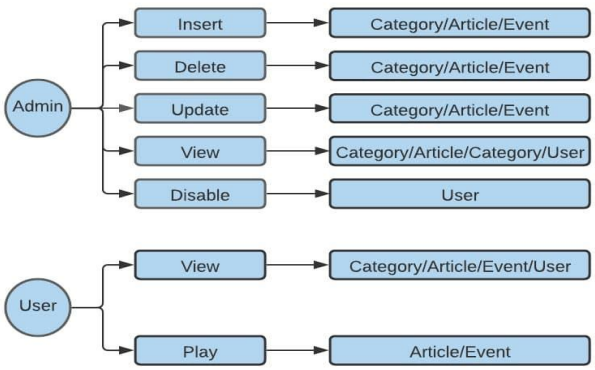



**Figure 3.** Use Case Diagram: The use case diagram provided above marks all of the contributors within the website; the contributors are given a number of authorizations to use in the website as the following:

**Admin**: can review users, count users, survey users, and disable users who go against the policy of the website. Besides, the administrator can add categories, articles within categories into any level, delete and update. Furthermore, the administrators can manage events on our website if any event comes, or it is over.
**User**: can play by choosing levels, categories, and articles. Moreover, users can save their typing information into histories. In case of users who don't have accounts in our system, the users have to sign up with their account to save their typing information. Furthermore, the users can join events in our system and view other top players in our system.

## III ANALYSIS

Analysis is the process, or profession of studying an activity such as a procedure, a business, or a physiological function. We do the analysis in order to define its goals or purposes and to discover operations and procedures for accomplishing them most efficiently. Analysis is the process of studying a procedure or business in order to identify its goals, 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 studying how well those component parts work and interact to accomplish their purpose.

### 3.1 Purpose

The Core purpose of developing **Khmer Anguli** is to provide ease for **Administrators** to:
- Control users, events, categories, and articles within categories.
- Can search, view, and disable users.
- Can insert, delete, update, view categories and articles.
- Can insert events that are suitable.
- Can delete events directly.

Provide ease for **Users** to:
- Choose levels, categories, and articles to type.
- Save their played records to their histories by signing up with their accounts if they don't have accounts in our system.
- Can join and compete in events directly.
- Can search and chat to other users.
- Can view top players who got highest scores.

### 3.2 Scope

There are some scope's plans that we have to develop this web application such as:
- Manage the categories and articles such as create new categories and add articles.
- Manage the user's information such as register and save results.
- Manage event, chat platform and notification.

### 3.3 Risk Management

Risk management is the process of identifying, assessing and controlling threats to an application's process. These threats, or risks, could stem from a wide variety of sources, including implementation, maintenance, strategic management errors, accidents and natural disasters.
Normally, every project always has risks. Therefore, we have to identify and manage the risk of project because it

is an essential part of building project.

| Risk | Description |
|---|---|
| Server Down | When many users access at the same time. |
| Chat Platform | Difficult to manage any process when a user contacts with other users. |

# IV PLANNING

## 4.1 Tree Structure View

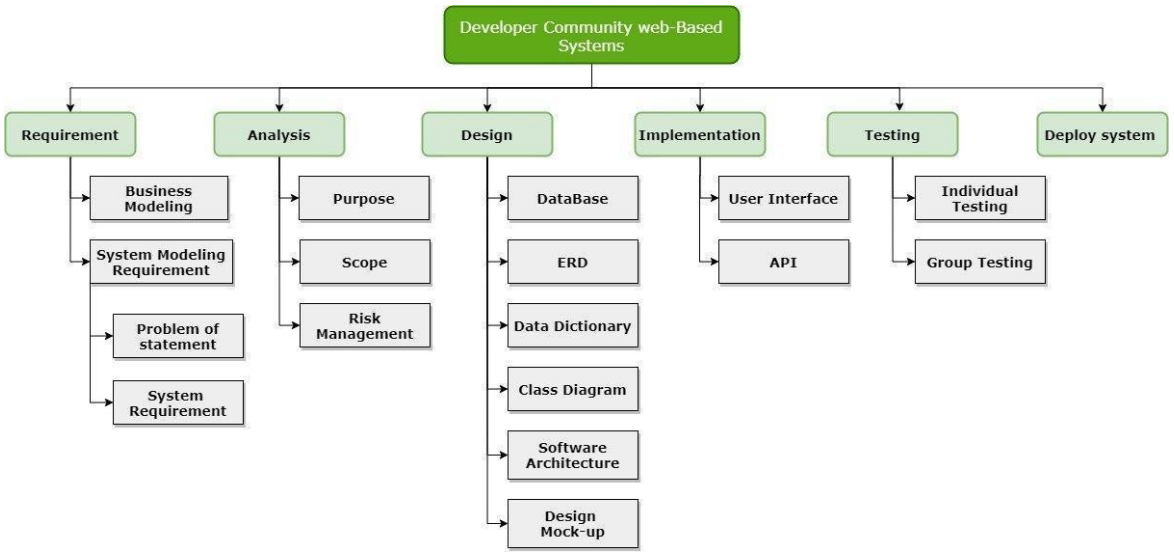



**Figure 4.** Tree Structure View, in this picture show about our scope planning

## 4.2 Work Breakdown Structure (WBS)

| Task Name | Start Date | End Date | Assigned To | Duration |
|---|---|---|---|---|
| 1. Requirement | | | | |
| 1.1 Business Modeling | 14-June-2021 | 17-June-2021 | All Members | 4 days |
| 1.2 System Requirement Modeling | 17-June-2021 | 26-June-2021 | Chhayhong, Pich | 10 days |
| 1.2.1 Problem of Statement | 22-June-2021 | 23-June-2021 | Jork | 2 days |
| 1.2.2 System Requirement | 22-June-2021 | 28-June-2021 | Pich, Chhayhong, Jork | 7days |
| 2. Analysis | | | | |
| 2.1 Purpose | 24-June-2021 | 27-June-2021 | All Members | 4 days |
| 2.2 Scope | 24-June-2021 | 26-June-2021 | Chhayhong | 3 days |
| 2.3 Risk Management | 29-June-2021 | 30-June-2021 | Jork | 2 days |
| 3. Planning | | | | |
| 3.1 Tree Structure View | 29-June-2021 | 01-July-2021 | Pich | 3 days |
| 3.2 Work Breakdown Structure (WBS) | 10-July-2021 | 13-July-2021 | Jork | 4 days |
| 3.3 Documentation | | | | |
| 3.3.1 Research Paper | 24-June-2021 | 06-July-2021 | Pich, Chhayhong, Jork | 13 days |
| 3.3.2 Technical Report | 8-July-2021 | 19-July-2021 | Jork, Chhayhong | 12 days |
| 4. Design | | | | |
| 4.1 ERD | 07-July-2021 | 11-July-2021 | Pich | 5 days |

| | | | | |
|---|---|---|---|---|
| 4.2 Data Dictionary | 07-July-2021 | 08-July-2021 | Pich | 2 days |
| 4.3 Database Design | 1-July-2021 | 10-July-2021 | Pich, Chhayhong, Jork | 10 days |
| 4.4 Spring Boot MVC | 16-July-2021 | 18-July-2021 | Chhayhong, Pich | 3 days |
| 4.5 Class Diagram | 17-July-2021 | 19-July-2021 | Pich | 3 days |
| 4.6 Software Architecture | 12-July-2021 | 13-July-2021 | All Member | 2 days |
| 4.8 Interface Design | 1-July-2021 | 13-July-2021 | All Member | 13 days |
| 4.9 Source Code | 22-July-2021 | 25-August-2021 | All Member | 25 days |
| 4.10 Implement UI Design | 22-July-2021 | 10-August-2021 | Panha,Chantha, Vireak | 14 days |
| 5. Implement | | | | |
| 5.1 Front-End | | | | |
| 5.1.1 Dashboard | 22-July-2021 | 03-August-2021 | Pich | 7 days |
| 5.1.2 Manage User's Accounts Login | 22-July-2021 | 03-August-2021 | Chantha | |
| 5.1.3 Manage category | 22-July-2021 | 03-August-2021 | Panha | |
| 5.1.4 Manage Article | 22-July-2021 | 03-August-2021 | Panha | |
| 5.1.5 Manage Event | 22-July-2021 | 03-August-2021 | Chantha | |
| 5.1.6 Manage Chart Platform | 04-August-2021 | 11-August-2021 | Chhayhong | 7 days |
| 5.1.9 Manage Notification | 04-August-2021 | 11-August-2021 | All members | |
| 5.1.10 Merge UI | 04-August-2021 | 11-August-2021 | Chhayhong | |
| | | | | |
| 5.2.1 Login And Web Security | 31-July-2021 | 01-August-2021 | Chhayhong, Pich | 3 days |
| 5.2.3 Manage User | 04-August-2021 | 07-August-2021 | Chantha | 4 days |
| 5.2.4 Manage Category | 25-July-2021 | 29-August-2021 | Pich, Chhayhong | 5 days |
| 5.2.5 Manage Article | 22-July-2021 | 30-August-2021 | Pich, Chhhayhong | 6 days |
| 5.2.6 Manage Event | 27- July-2021 | 29-July-2021 | Vireak | 2 days |
| | | | | |
| 5.3.1 Create Database, Functions and Views | 22-July-2021 | 28-July-2021 | Pich, Chhayhong, Jork | 7 days |
| 6. Submit Assignment | 23-August-2021 | 25-August-2021 | All Member | 3 days |
| 7. Deploy | 25-August-2021 | 27-August-2021 | All Member | 3 days |

# V Design

## 5.1 POJO and Redux

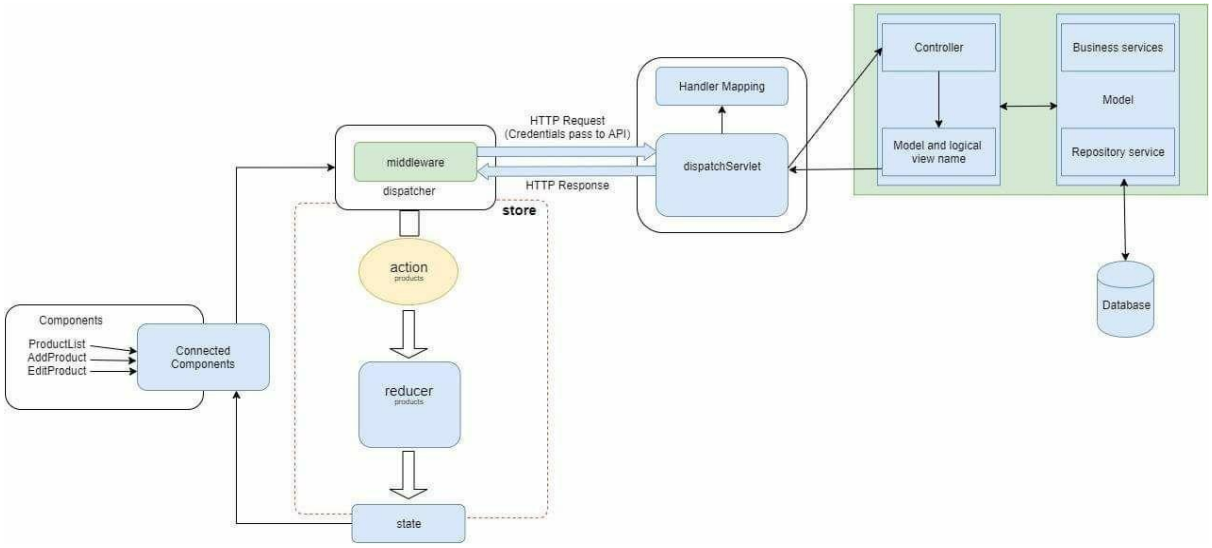



**Figure 5.** React Redux client with Spring Boot server

### 5.2 Class Diagram

We separate into two projects Spring Boot and ReactJS:
Spring RESTful APIs:
- business-login
- persistence
- web-service

View:
- ReactJS

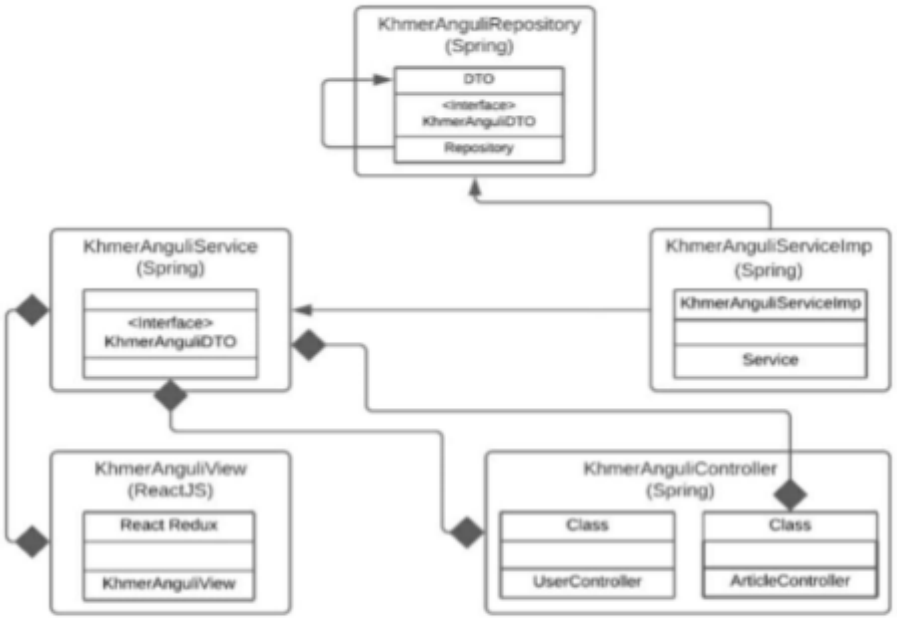



**Figure 6.** Class Diagram

### 5.3 Data Dictionary

| Table Name | Field Name | Datatype | Length | Required | PK/FK | Default | Reference | Description |
|---|---|---|---|---|---|---|---|---|
| roles | role_id | serial | 16 | Y | | | | Auto increment |
| | name | varchar | 100 | Y | | | | |
| users_roles | user_role_id | serial | 16 | Y | PK | | | Auto increment |
| | user_id | int4 | 32 | Y | FK | | users | |
| | role_id | Int4 | 32 | Y | FK | | roles | |
| users | user_id | serial | 16 | Y | PK | | | Auto increment |
| | firstname | varchar | 255 | | | | | |
| | lastname | varchar | 255 | | | | | |
| | username | varchar | 255 | Y | | | | |
| | password | varchar | 255 | Y | | | | |
| | gender | varchar | 7 | | | | | |
| | date_of_birth | date | | | | | | |
| | address | varchar | 255 | | | | | |
| | phone_number | varchar | 255 | | | | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | email | bool | 255 | | | | | 0 = active<br>1 = inactive |
| | disable | numeric | 1 | Y | | 1 | | 0=active, 1=inactive |
| | bio | varchar | 255 | | | | | |
| | profile_image | varchar | 255 | | | | | |
| categories | category_id | serial | 16 | Y | PK | | | Auto increment |
| | name | varchar | 255 | Y | | | | |
| | created_date | date | | | | | | |
| | updated_date | date | | | | | | |
| authors | author_id | serial | 16 | Y | PK | | | Auto increment |
| | name | varchar | 255 | Y | | | | |
| | gender | varchar | 7 | | | | | |
| articles | article_id | serial | 16 | Y | PK | | | Auto increment |
| | title | varchar | 255 | Y | | | | |
| | content | text | | Y | | | | |
| | created_date | date | | Y | | | | |
| | updated_date | date | | | | | | |
| | reference_url | varchar | 255 | | | | | |
| | level | enum | | Y | | | | |
| | category_id | int4 | 32 | Y | FK | | categories | |
| | author_id | int4 | 32 | Y | FK | | authors | |
| events | event_id | serial | 16 | Y | PK | | | Auto increment |
| | name | varchar | 255 | Y | | | | |
| | description | varchar | 255 | | | | | |
| | started_date | date | | Y | | | | |
| | end_date | date | | Y | | | | |
| | updated_date | date | | | | | | |
| | article_id | int4 | 32 | Y | FK | | articles | |
| | created_date | date | | Y | | | | |
| messages | message_id | serial | 16 | Y | PK | | | Auto increment |
| | message_body | text | | Y | | | | |
| | message_date | date | | Y | | | | |
| | is_seen | numeric | 1 | | | 1 | | 0=unchecked, 1=seen |
| | is_delete | numeric | 1 | | | 1 | | 0=active, 1=deleted |
| | parent_message_id | int4 | 32 | Y | FK | | messages | |
| | user_sender_id | int4 | 32 | Y | FK | | users | |
| | user_receiver_id | int4 | 32 | Y | FK | | users | |
| reports | report_id | serial | 16 | | PK | | | Auto increment |
| | title | varchar | 255 | | | | | |
| | content | varchar | 255 | | | | | |
| | report_date | date | | | | | | |
| | is_reviewed | numeric | 1 | | | 1 | | 0=uncheck, 1= reviewed |
| | user_id | int4 | 32 | | FK | | users | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| user_played_records | record_id | serial | 32 | Y | PK | | | Auto increment |
| | wpm | float4 | 4 | Y | | | | |
| | correctness_percentage | float4 | 4 | Y | | | | |
| | played_date | date | | Y | | | | |
| | total_time | float4 | 4 | Y | | | | |
| | duration | float4 | 4 | Y | | | | |
| | record_type | enum | | Y | | | | |
| | article_id | int4 | 32 | Y | FK | | articles | |
| | user_id | int4 | 32 | Y | FK | | users | |

## 5.4 Database Diagram

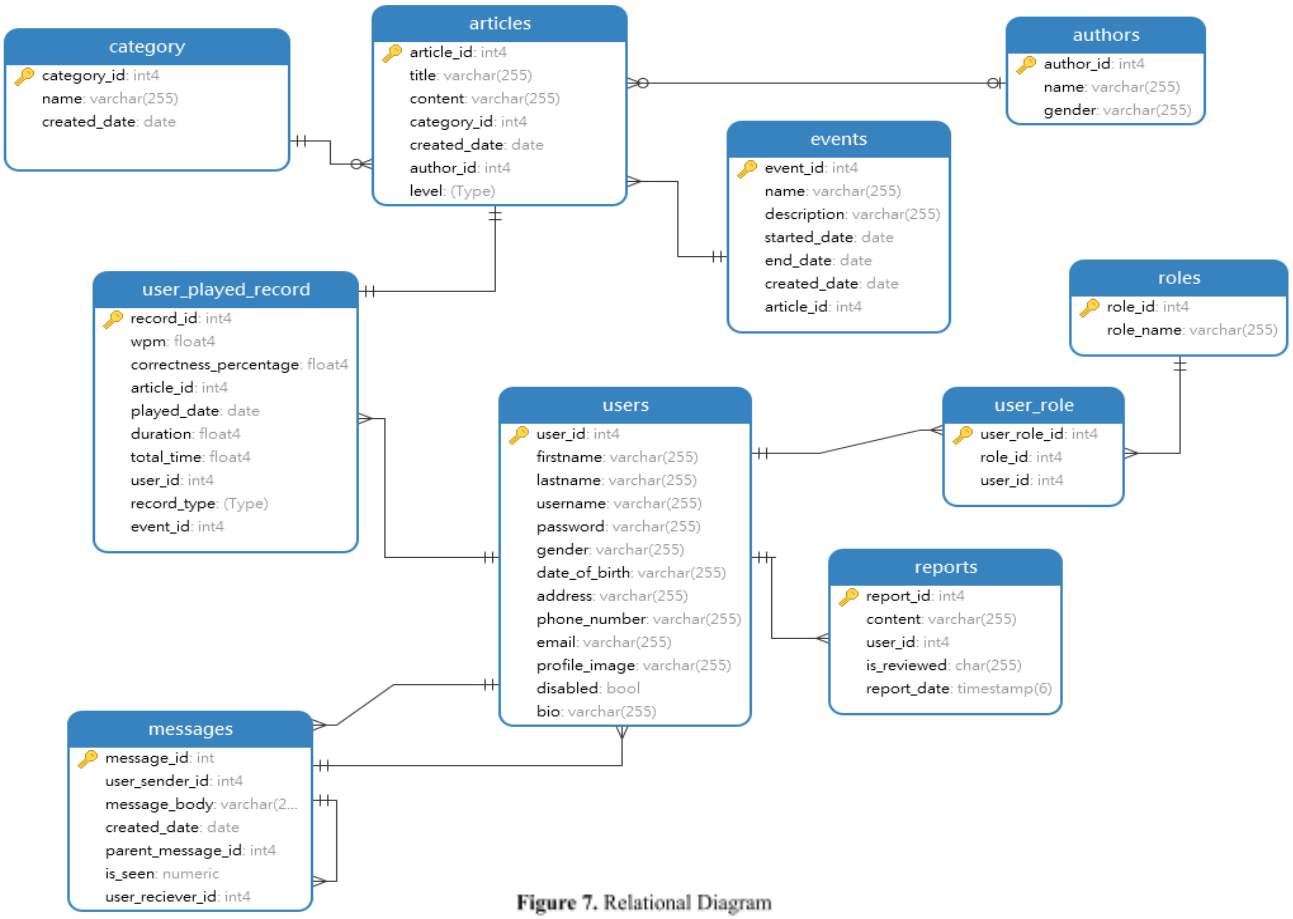
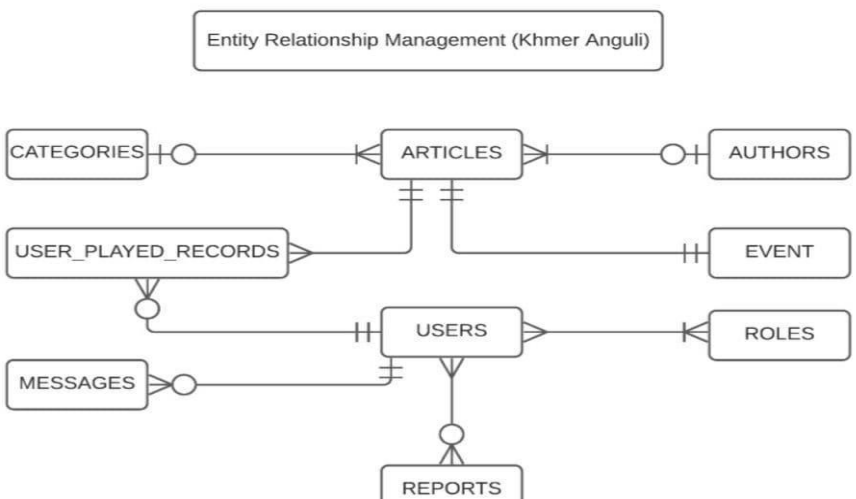



**Figure 7.** Relational Diagram

## 5.5 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 database



### 5.6 Software Architecture

Spring Web Controller to view web page, use ReactJS for view only

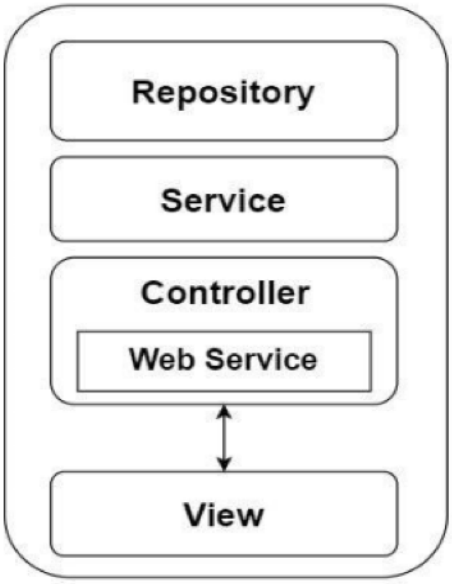



**Figure 9.** Software Architecture

As in the above figure, describe the flow of POJO

- 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:**

- POJO stand for Plain Old Java Object: is a pure data structure that has fields with getters and possibly setters, and may override some methods form Object or some other interface, but does not have behavior of its own.

**Client:**

- React: is a JavaScript library used in web development to build interactive elements on websites.
- Redux Pattern: Redux is really a fairly simple design pattern. We can write logic goes into a single function, and the only way to run that logic is to give Redux a plain object that describes something that has happened. Redux have 4 main concepts such as actions, store, reducers, and middleware.
- HTML: Hyper Text Markup Language is a set of markup tags used for creating web pages.
- CSS: Cascading Style Sheet is the style sheet language used for apply style to the HTML document.

Our application is based on the MVC pattern that applied with 4 main components to use in Spring MVC:

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

A Service: all your business logics are contained 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.

### 5.7 Interface Design

**Figure 10.** Sign up page

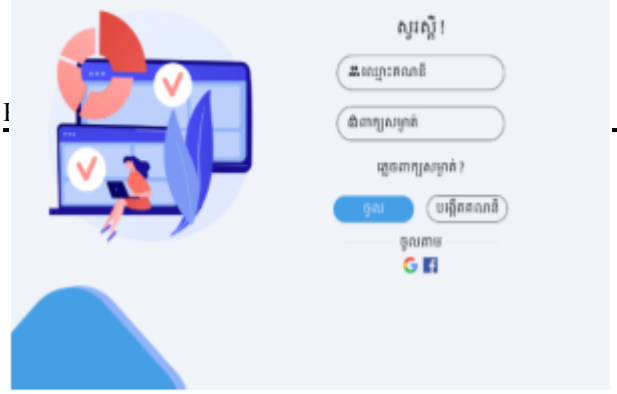


**Figure 13.** Login page

**Figure 18.** Chat

### 5.8 Source Code

- **ArticleRepository.java**

```
package
org.ksga.springboot.khmeranguliapi.repository;
import org.apache.ibatis.annotations.*;
import
org.ksga.springboot.khmeranguliapi.payload.dto.Art
icleDTO;
import
org.ksga.springboot.khmeranguliapi.payload.request
.ArticleRequest;
import
org.ksga.springboot.khmeranguliapi.repository.prov
ider.ArticleProvider;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface ArticleRepository {
    @SelectProvider(type =
ArticleProvider.class,method = "findAll")
    @Results(id = "articleMapping", value = {
            @Result(column = "article_id",property
= "id"),
            @Result(column = "category_id",
property = "category.id"),
            @Result(column =
"category_name",property = "category.name"),
            @Result(column =
"author_name",property = "author.name"),
            @Result(column =
"category_created_date", property =
"category.created_date"),
            @Result(column =
"category_updated_date", property =
"category.updated_date"),
            @Result(column = "author_id",property
= "author.id"),
            @Result(column = "gender",property =
"author.gender"),
    })
    public List<ArticleDTO> findAll();
    @SelectProvider(type =
ArticleProvider.class,method = "findOne")
    @ResultMap("articleMapping")

    public List<ArticleDTO> findOne(int id);
    @UpdateProvider(type =
ArticleProvider.class,method = "update")
    public boolean update(int id,ArticleRequest
articleRequest);
    @DeleteProvider(type =
ArticleProvider.class,method = "delete")
```

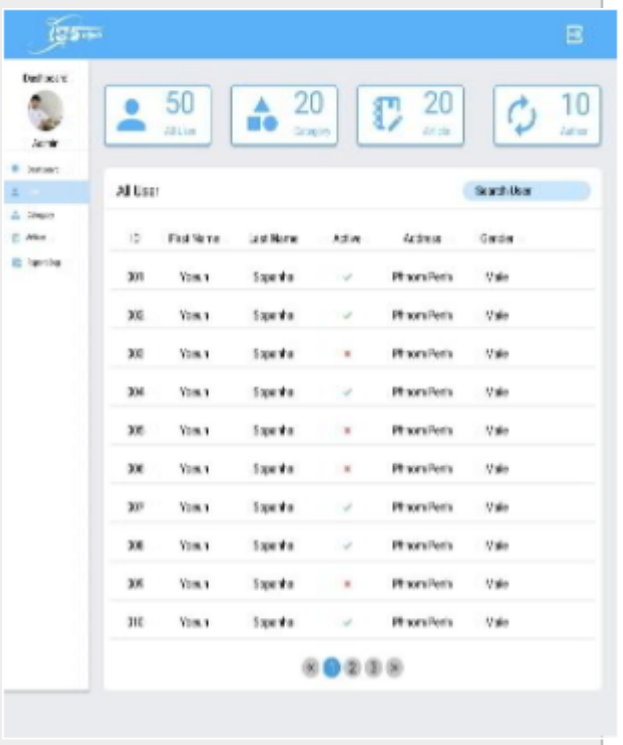


**Figure 19.** Admin dashboard

```
    public boolean delete(int id);
    @InsertProvider(type =
ArticleProvider.class,method = "insert")
    @ResultMap("articleMapping")
    public boolean save(@Param("articleRequest")
ArticleRequest articleRequest);

    @SelectProvider(type =
ArticleProvider.class,method = "findRandom")
    @ResultMap("articleMapping")
    public List<ArticleDTO>
findRandomArticle(String level);

    @Select("select * from articles order by
random() limit 1")
    public List<ArticleDTO>
findRandomArticleNoLevel();
}
```

- **CateogoryRepository.java**

```
package
org.ksga.springboot.khmeranguliapi.repository;

import org.apache.ibatis.annotations.*;
import
org.ksga.springboot.khmeranguliapi.model.Category;
import
org.ksga.springboot.khmeranguliapi.payload.dto.Cat
egoryDTO;
import
org.ksga.springboot.khmeranguliapi.payload.request
.CategoryRequest;
import
org.ksga.springboot.khmeranguliapi.repository.prov
ider.CategoryProvider;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface CategoryRepository {

    @SelectProvider(type = CategoryProvider.class,
method = "findAllCategory")
    @Results( id = "categoryMapping", value = {
            @Result(column =
"category_id",property = "id")}
    )
    public List<CategoryDTO> findAll();
    @SelectProvider(type = CategoryProvider.class,
method = "findOneCategory")
    @ResultMap("categoryMapping")
    public List<CategoryDTO> findOne(int id);
    @UpdateProvider(type = CategoryProvider.class,
method = "updateCategory")
    public boolean update(int id, CategoryRequest
category);
    @DeleteProvider(type = CategoryProvider.class,
method = "deleteCategory")
    public boolean delete(int id);
    @InsertProvider(type = CategoryProvider.class,
method = "insertCategory")
    public boolean save(@Param("category")
CategoryRequest categoryRequest);

}
```

- **EventRepository.java**

```
package
org.ksga.springboot.khmeranguliapi.repository;

import org.apache.ibatis.annotations.*;

import
org.ksga.springboot.khmeranguliapi.payload.dto.Eve
ntDTO;

import
org.ksga.springboot.khmeranguliapi.payload.request
.EventRequest;

import
org.ksga.springboot.khmeranguliapi.repository.prov
ider.EventProvider;

import org.springframework.stereotype.Repository;

import org.springframework.stereotype.Service;

import java.util.List;

@Repository

public interface EventRepository {

    @SelectProvider(type = EventProvider.class,
method = "findAll")

    @Results(id = "eventMapping",value = {

            @Result(column = "event_id",property =
"id"),

            @Result(column = "article_id",property
= "article.id"),

            @Result(column = "title",property =
"article.title"),

            @Result(column = "content",property =
"article.content"),

            @Result(column =
"article_created_date",property =
"article.created_date"),

            @Result(column =
"article_updated_date",property =
"article.updated_date"),

            @Result(column =
"reference_url",property =
"article.reference_url"),

            @Result(column = "level",property =
"article.level")

    })

    public List<EventDTO> findAll ();

    @SelectProvider(type = EventProvider.class,
method = "findOne")

    @ResultMap("eventMapping")
```

```
    public List<EventDTO> findById (int id);

    @UpdateProvider(type = EventProvider.class,
method = "update")

    public boolean update(int id, EventRequest
eventRequest);

    @DeleteProvider(type = EventProvider.class,
method = "delete")

    public boolean delete (int id);

    @InsertProvider(type = EventProvider.class,
method = "insert")

    public boolean save (@Param("event")
EventRequest eventRequest);

}
```

- **MessageRepository.java**

```
package
org.ksga.springboot.khmeranguliapi.repository;

import org.apache.ibatis.annotations.*;
import
org.ksga.springboot.khmeranguliapi.model.Message;
import
org.ksga.springboot.khmeranguliapi.payload.dto.Mes
sageDTO;
import
org.ksga.springboot.khmeranguliapi.payload.request
.MessageRequest;
import
org.ksga.springboot.khmeranguliapi.repository.prov
ider.MessageProvider;
import org.apache.ibatis.annotations.Mapper;
import
org.springframework.web.bind.annotation.RequestPar
am;
import java.util.List;
@Mapper
public interface MessageRepository {

    @Select("select DISTINCT * from messages where
user_receiver_id=#{id}")
    List<MessageDTO> findMessagedRecord(int id);

        @Select("select * from messages where
user_sender_id=#{id} or user_receiver_id=#{id}")
       List<MessageDTO>  selectMessageByUserId(int
id);

                        @InsertProvider(type      =
MessageProvider.class,method = "insert")
  boolean createMessage(@RequestParam("sender")
int sender,@RequestParam("receiver") int
receiver,@Param("messageRequest") MessageRequest
messageRequest);

    @UpdateProvider(type =
MessageProvider.class,method = "update")
    public Boolean updateMessage(int
id,@Param("message") Message message);

    @UpdateProvider(type =
MessageProvider.class,method = "delete")
    boolean deleteMessage(int id);
}
```

- **ReportRepository.java**

```
package
org.ksga.springboot.khmeranguliapi.repository;
import org.apache.ibatis.annotations.*;
import
org.ksga.springboot.khmeranguliapi.payload.dto.Rep
ortDTO;
import
org.ksga.springboot.khmeranguliapi.payload.request
.ReportRequest;
import
org.ksga.springboot.khmeranguliapi.repository.prov
ider.ReportProvider;
import org.springframework.stereotype.Repository;
import
org.springframework.web.bind.annotation.RequestPar
am;
import java.util.List;
@Repository
public interface ReportRepository {
    @InsertProvider(type = ReportProvider.class,
method = "insert")
    public boolean insert (@RequestParam("id")int
id, @Param("reportRequest") ReportRequest
reportRequest);
    @SelectProvider(type = ReportProvider.class,
method = "findAll")
    @Results(id = "reportMapping",value = {
            @Result(column = "report_id",property
= "id")
    })
    public List<ReportDTO> findAll();
    @SelectProvider(type = ReportProvider.class,
method = "findByUserId")
    public List<ReportDTO> findByUserId(int
user_id);
    @DeleteProvider(type = ReportProvider.class,
method = "delete")
    public boolean delete (int id);
}
```

- **RoleRepository.java**

```
package
org.ksga.springboot.khmeranguliapi.repository;

import org.apache.ibatis.annotations.Mapper;
import org.apache.ibatis.annotations.Select;
import
org.ksga.springboot.khmeranguliapi.model.auth.ERol
e;
import
org.ksga.springboot.khmeranguliapi.model.auth.Role
;

import java.util.Optional;

@Mapper
public interface RoleRepository {
    @Select("SELECT * FROM roles WHERE role_name =
#{name}")
    Optional<Role> findByName(ERole role);
}
```

- **SaveRecordRepository.java**

```
package
org.ksga.springboot.khmeranguliapi.repository;
import org.apache.ibatis.annotations.*;
import
org.ksga.springboot.khmeranguliapi.payload.dto.Sav
eRecordDTO;
import
org.ksga.springboot.khmeranguliapi.payload.request
.SaveRecordRequest;
```

```
import
org.ksga.springboot.khmeranguliapi.repository.prov
ider.SaveRecordProvider;
import org.springframework.stereotype.Repository;

import java.util.Date;
import java.util.List;
@Repository
public interface SaveRecordRepository {
    @SelectProvider(type =
SaveRecordProvider.class, method = "findAll")
    @Results(id = "saveRecordMapping",value = {
            @Result(column = "record_id",property
= "id"),
            @Result(column = "article_id",property
= "article.id"),
            @Result(column = "title",property =
"article.title"),
            @Result(column = "content",property =
"article.content"),
            @Result(column =
"created_date",property = "article.created_date"),
            @Result(column =
"updated_date",property = "article.updated_date"),
            @Result(column =
"reference_url",property =
"article.reference_url"),
            @Result(column = "level",property =
"article.level"),
            @Result(column = "user_id",property =
"userProfile.user_id"),
            @Result(column = "firstname",property
= "userProfile.firstname"),
            @Result(column = "lastname",property =
"userProfile.lastname"),
            @Result(column = "gender",property =
"userProfile.gender"),
            @Result(column = "username",property =
"userProfile.username")

    })
    public List<SaveRecordDTO> findAll ();
    @ResultMap("saveRecordMapping")
    @SelectProvider(type =
SaveRecordProvider.class,method = "findByUserId")
    public List<SaveRecordDTO>
findRecordByUserId(int user_id);
    @DeleteProvider(type =
SaveRecordProvider.class,method = "delete")
    public boolean deleteRecord (int id);
    @InsertProvider(type =
SaveRecordProvider.class,method = "insert")
    public boolean save(int
id,@Param("saveRecordRequest") SaveRecordRequest
saveRecordRequest);

    public List<SaveRecordDTO> totalRecordByDate
(Date startDate, Date endDate, int user_id);
}
```

- **UserProfileRepository.java**

```
package
org.ksga.springboot.khmeranguliapi.repository;

import org.apache.ibatis.annotations.*;
import
org.ksga.springboot.khmeranguliapi.model.Message;
import
org.ksga.springboot.khmeranguliapi.model.UserProfi
le;
import
org.ksga.springboot.khmeranguliapi.model.auth.Role
;
```

```
import
org.ksga.springboot.khmeranguliapi.payload.dto.Use
rProfileDTO;
import
org.ksga.springboot.khmeranguliapi.payload.request
.UserProfileRequest;
import
org.ksga.springboot.khmeranguliapi.payload.request
.UserSearchProfileRequest;
import
org.ksga.springboot.khmeranguliapi.repository.prov
ider.AccountProvider;
import
org.ksga.springboot.khmeranguliapi.repository.prov
ider.MessageProvider;
import
org.ksga.springboot.khmeranguliapi.repository.prov
ider.UserProfileProvider;
import org.springframework.stereotype.Repository;
import java.util.List;
import java.util.Set;
@Repository
public interface UserProfileRespository {
    @UpdateProvider(type =
AccountProvider.class,method = "disableAccount")
    boolean disableAccount(int id);

    @Select("SELECT * FROM users where
users.username=#{id}")
    @Results({
            @Result(property = "user_id", column =
"user_id"),
            @Result(property = "roles", column =
"user_id", many = @Many(select =
"findAllRolesByUserId")),
    })
    UserProfileDTO findOneAccount(String id);

    @Select("SELECT * FROM users where
users.user_id=#{id}")
    @Results({
            @Result(property = "user_id", column =
"user_id"),
            @Result(property = "roles", column =
"user_id", many = @Many(select =
"findAllRolesByUserId")),
    })
    UserProfileDTO findOwnedAccount(int id);
    @Select("SELECT * FROM users")
    @Results({
            @Result(property = "user_id", column =
"user_id"),
            @Result(property = "roles", column =
"user_id", many = @Many(select =
"findAllRolesByUserId")),
    })
    List<UserProfileDTO> findAll();
    @Select("SELECT * FROM user_roles ur INNER
JOIN roles r ON ur.role_id = r.role_id WHERE
ur.user_id = #{userId}")
    Set<Role> findAllRolesByUserId(Long userId);

    //    search user
                        @SelectProvider(type      =
UserProfileProvider.class,method                  =
"searchUserByUsername")
                    public     List<UserProfileDTO>
searchUserByUsername(@Param("search")
UserSearchProfileRequest search);

                        @UpdateProvider(type      =
UserProfileProvider.class,method = "update")
             public    Boolean    updateProfile(int
id,@Param("user")               UserProfileRequest
userProfileRequest);
```

```
}
```

- **UserRepository.java**

```
package
org.ksga.springboot.khmeranguliapi.repository;

import org.apache.ibatis.annotations.Insert;
import org.apache.ibatis.annotations.Many;
import org.apache.ibatis.annotations.Mapper;
import org.apache.ibatis.annotations.Options;
import org.apache.ibatis.annotations.Result;
import org.apache.ibatis.annotations.Results;
import org.apache.ibatis.annotations.Select;
import
org.ksga.springboot.khmeranguliapi.model.auth.Role
;
import
org.ksga.springboot.khmeranguliapi.model.auth.User
;

import java.util.Date;
import java.util.Optional;
import java.util.Set;

@Mapper
public interface UserRepository {

    @Select("SELECT * FROM users WHERE username =
#{username}")
    @Results({
            @Result(property = "user_id", column =
"user_id"),
            @Result(property = "roles", column =
"user_id", many = @Many(select =
"findAllRolesByUserId")),
    })
    Optional<User> findUserByUsername(String
username);

    @Select("SELECT * FROM user_roles ur INNER
JOIN roles r ON ur.role_id = r.role_id WHERE
ur.user_id = #{userId}")
    Set<Role> findAllRolesByUserId(Long userId);
    @Insert("INSERT INTO users (username,
password,firstname,lastname,gender,date_of_birth,a
ddress,phone_number,email) VALUES (#{username},
#{password},
#{firstname},#{lastname},#{gender},#{date_of_birth
},#{address},#{phone_number},#{email})")
    @Options(useGeneratedKeys = true, keyProperty
= "user_id", keyColumn = "user_id")
    boolean insertUser(User user);

    @Insert("<script>INSERT INTO user_roles
(user_id, role_id) VALUES "
            + "<foreach collection='roles'
item='role' separator=','>"
            + "(#{userId}, #{role.role_id})"
            + "</foreach>"
            + "</script>")
     boolean insertUsersRoles(int userId, Set<Role>
roles);

    default boolean save(User user) {
        if (insertUser(user)) {
                                                return
insertUsersRoles(user.getUser_id(),
user.getRoles());
        }
        return false;
    }

    @Select("SELECT CASE WHEN EXISTS " +
                "(SELECT user_id FROM users WHERE
username = #{username}) " +
              "THEN CAST(1 AS BIT) ELSE CAST(0 AS
BIT) END")
    boolean existsByUsername(String username);
}
```

- **ArticleRestController.java**

```
package
org.ksga.springboot.khmeranguliapi.controller.rest
;
import
org.ksga.springboot.khmeranguliapi.model.Article;
import
org.ksga.springboot.khmeranguliapi.payload.dto.Art
icleDTO;
import
org.ksga.springboot.khmeranguliapi.payload.mapper.
ArticleMapper;
import
org.ksga.springboot.khmeranguliapi.payload.request
.ArticleRequest;
import
org.ksga.springboot.khmeranguliapi.payload.respons
e.Response;
import
org.ksga.springboot.khmeranguliapi.service.Article
Service;
import
org.springframework.beans.factory.annotation.Autow
ired;
import org.springframework.web.bind.annotation.*;
import java.util.List;
@CrossOrigin(origins={"*","http://139.162.28.200:1
603/","http://localhost:3000/"})
@RestController
@RequestMapping("/api/v1/articles")
public class ArticleRestController {
    @Autowired
    private ArticleService articleService;
    @Autowired
    private ArticleMapper articleMapper;
    public ArticleRestController(ArticleService
articleService) {
        this.articleService = articleService;
    }
    @GetMapping("/displayAll")
    public Response<List<ArticleDTO>> findAll(){
        List<ArticleDTO> articles =
articleService.findAllArticle();
        if(!articles.isEmpty()){
            return
Response.<List<ArticleDTO>>ok().setPayload(article
s);
        }else{
            return Response.notFound();
        }
    }
    @GetMapping("/displayById/{id}")
    public Response<List<ArticleDTO>>
findById(@PathVariable int id){
        List<ArticleDTO> findOne =
articleService.findOneArticle(id);
        if(!findOne.isEmpty()){
            return
Response.<List<ArticleDTO>>ok().setPayload(findOne
);
        }else{
            return Response.notFound();
        }
    }
    @PutMapping("/update/{id}")
    public Response<ArticleDTO>
updateArticle(@PathVariable int id,@RequestBody
ArticleRequest articleRequest){
```

```
        boolean isUpdated =
articleService.updateArticle(id, articleRequest);
        if(isUpdated){
            return Response.<ArticleDTO>ok();
        }else {
            return Response.notFound();
        }
    }
    @DeleteMapping("/delete/{id}")
    public Response<ArticleDTO>
deleteArticle(@PathVariable int id){
        boolean isDeleted =
articleService.deleteArticle(id);
        if(isDeleted){
            return Response.<ArticleDTO>ok();
        }else {
            return Response.notFound();
        }
    }
    @PostMapping("/create")
    public Response<ArticleDTO>
createArticle(@RequestBody ArticleRequest
articleRequest) {
        boolean isCreated =
articleService.insertArticle(articleRequest);
        if (isCreated) {
            Article article =
articleMapper.articleRequestToArticle(articleReque
st);
            ArticleDTO articleDTO =
articleMapper.articleToArticleDto(article);

            return
Response.<ArticleDTO>ok().setPayload(articleDTO);
        } else {
            return
Response.<ArticleDTO>badRequest().setErrors("Check
your data again");
        }
    }
    @GetMapping("/random/{level}")
    public Response<List<ArticleDTO>>
getRandomArticleByLevel(@PathVariable String
level) {
        List<ArticleDTO> article =
articleService.findRandomlArticle(level);
        if(!article.isEmpty()){
            return
Response.<List<ArticleDTO>>ok().setPayload(article
);
        }else{
            return Response.notFound();
        }
    }
}
```

- **AuthorRestController.java**

```
package
org.ksga.springboot.khmeranguliapi.controller.rest
;

import
org.ksga.springboot.khmeranguliapi.model.Author;
import
org.ksga.springboot.khmeranguliapi.model.Category;
import
org.ksga.springboot.khmeranguliapi.payload.dto.Aut
horDTO;
import
org.ksga.springboot.khmeranguliapi.payload.dto.Cat
egoryDTO;
import
org.ksga.springboot.khmeranguliapi.payload.mapper.
AuthorMapper;
import
org.ksga.springboot.khmeranguliapi.payload.request
.AuthorRequest;
import
org.ksga.springboot.khmeranguliapi.payload.respons
e.Response;
import
org.ksga.springboot.khmeranguliapi.service.AuthorS
ervice;
import
org.springframework.beans.factory.annotation.Autow
ired;
import org.springframework.web.bind.annotation.*;
import java.util.List;
@RestController
@CrossOrigin(origins={"*","http://139.162.28.200:1
603/","http://localhost:3000/"})
@RequestMapping("api/v1/authors")
public class AuthorRestController {
    @Autowired
    private AuthorService authorService;
    @Autowired
    private AuthorMapper authorMapper;
    public AuthorRestController(AuthorService
authorService) {
        this.authorService = authorService;
    }
    @GetMapping("/displayAll")
    public Response<List<AuthorDTO>>
displayAllAuthor(){
        List<AuthorDTO> authors=
authorService.findAllAuthor();
        if(!authors.isEmpty()){
            return
Response.<List<AuthorDTO>>ok().setPayload(authors)
;
        }else{
            return Response.notFound();
        }
    }
    @GetMapping("/displayById/{id}")
    public Response<List<AuthorDTO>>
displayById(@PathVariable int id){
        List<AuthorDTO> findOne =
authorService.findOneAuthor(id);
        if(!findOne.isEmpty()){
            return
Response.<List<AuthorDTO>>ok().setPayload(findOne)
;
        }else{
            return Response.notFound();
        }
    }
    @PostMapping("/addNew")
    public Response<AuthorDTO>
addNewAuthor(@RequestBody AuthorRequest
authorRequest){
        boolean isCreated =
authorService.insertAuthor(authorRequest);
        if (isCreated){
            Author author =
authorMapper.authorRequestToAuthor(authorRequest);
            AuthorDTO authorDTO =
authorMapper.authorToCategoryDto(author);
            return
Response.<AuthorDTO>ok().setPayload(authorDTO);
        } else {
            return
Response.<AuthorDTO>badRequest().setErrors("Check
your data again");
        }
    }

    @PutMapping("/updateById/{id}")
```

```
    public Response<AuthorDTO>
updateAuthor(@PathVariable int id, @RequestBody
AuthorRequest author){
        boolean isUpdated =
authorService.updateAuthor(id,author);
        if(isUpdated){
            return Response.<AuthorDTO>ok();
        }else {
            return Response.notFound();
        }
    }

    @DeleteMapping("/deleteById/{id}")
    public Response<AuthorDTO>
deleteById(@PathVariable int id){
        boolean isDeleted =
authorService.deleteAuthor(id);
        if(isDeleted){
            return Response.<AuthorDTO>ok();
        }else {
            return Response.notFound();
        }
    }
}
```

- **AuthRestController.java**

```
package
org.ksga.springboot.khmeranguliapi.controller.rest
;
import
org.ksga.springboot.khmeranguliapi.model.auth.ERol
e;
import
org.ksga.springboot.khmeranguliapi.model.auth.Role
;
import
org.ksga.springboot.khmeranguliapi.model.auth.User
;
import
org.ksga.springboot.khmeranguliapi.payload.request
.LoginRequest;
import
org.ksga.springboot.khmeranguliapi.payload.request
.RegisterRequest;
import
org.ksga.springboot.khmeranguliapi.payload.respons
e.JwtResponse;
import
org.ksga.springboot.khmeranguliapi.payload.respons
e.Response;
import
org.ksga.springboot.khmeranguliapi.repository.Role
Repository;
import
org.ksga.springboot.khmeranguliapi.repository.User
Repository;
import
org.ksga.springboot.khmeranguliapi.security.jwt.Jw
tUtils;
import
org.ksga.springboot.khmeranguliapi.security.servic
e.UserDetailsImpl;
import
org.springframework.beans.factory.annotation.Autow
ired;
import
org.springframework.security.authentication.Authen
ticationManager;
import
org.springframework.security.authentication.Userna
mePasswordAuthenticationToken;
import
org.springframework.security.core.Authentication;
import
org.springframework.security.core.GrantedAuthority
;
import
org.springframework.security.core.context.Security
ContextHolder;
import
org.springframework.security.crypto.password.Passw
ordEncoder;
import
org.springframework.web.bind.annotation.CrossOrigi
n;
import
org.springframework.web.bind.annotation.PostMappin
g;
import
org.springframework.web.bind.annotation.RequestBod
y;
import
org.springframework.web.bind.annotation.RequestMap
ping;
import
org.springframework.web.bind.annotation.RestContro
ller;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import java.util.HashSet;
import java.util.Set;
import java.util.stream.Collectors;

@CrossOrigin(origins={"*","http://139.162.28.200:1
603/","http://localhost:3000/"})
@RestController
@RequestMapping("/api/v1/auth")
public class AuthRestController {
    @Autowired
    AuthenticationManager authenticationManager;

    @Autowired
    UserRepository userRepository;

    @Autowired
    RoleRepository roleRepository;

    @Autowired
    PasswordEncoder encoder;

    @Autowired
    JwtUtils jwtUtils;

    @PostMapping("/login")
    public Response<JwtResponse>
authenticateUser(@RequestBody LoginRequest
loginRequest) {
        Authentication authentication =
authenticationManager.authenticate(
                new
UsernamePasswordAuthenticationToken(loginRequest.g
etUsername(), loginRequest.getPassword()));

SecurityContextHolder.getContext().setAuthenticati
on(authentication);
        String jwt =
jwtUtils.generateJwtToken(authentication);
        UserDetailsImpl userDetails =
(UserDetailsImpl) authentication.getPrincipal();
        Set<String> roles =
userDetails.getAuthorities()
                .stream()

.map(GrantedAuthority::getAuthority)
                .collect(Collectors.toSet());
        return
Response.<JwtResponse>ok().setPayload(
```

```
                new
JwtResponse(jwt,userDetails.getUser_id(),
userDetails.getUsername(),userDetails.getProfile_i
mage())
        );
    }

    @PostMapping("/register")
    public Response<User>
registerUser(@RequestBody RegisterRequest
registerRequest) {
        try {
            Pattern pattern =
Pattern.compile("^[A-Za-z0-9 ]+$");
            Matcher matcher =
pattern.matcher(registerRequest.getUsername());
            boolean isNotAllowedChar =
matcher.find();
            System.out.println(isNotAllowedChar);
            if(!isNotAllowedChar){
                return
Response.<User>badRequest().setErrors("Special
Character is not allowed in Username");
            }
            if
(userRepository.existsByUsername(registerRequest.g
etUsername())) {
                return Response
                        .<User>badRequest()
                        .setErrors("Error:
Username is already taken!");
            }

            User user = new User(

registerRequest.getFirstname(),
                    registerRequest.getLastname(),
                    registerRequest.getGender(),

registerRequest.getDate_of_birth(),
                    registerRequest.getAddress(),

registerRequest.getPhone_number(),
                    registerRequest.getEmail(),
                    registerRequest.getUsername(),

encoder.encode(registerRequest.getPassword())
            );

            Set<String> strRoles =
registerRequest.getRoles();
            Set<Role> roles = new HashSet<>();

            if (strRoles == null) {
                Role userRole =
roleRepository.findByName(ERole.ROLE_USER)
                        .orElseThrow(() -> new
RuntimeException("Error: Role is not found."));
                roles.add(userRole);
            } else {
                strRoles.forEach(role -> {
                    if ("admin".equals(role)) {
                        Role adminRole =
roleRepository.findByName(ERole.ROLE_ADMIN)
                                .orElseThrow(() ->
new RuntimeException("Error: Role Admin is not
found."));
                        roles.add(adminRole);
                    } else {
                        Role userRole =
roleRepository.findByName(ERole.ROLE_USER)
                                .orElseThrow(() ->
new RuntimeException("Error: Role User is not
found."));
                        roles.add(userRole);
                    }
                });
            }
            user.setRoles(roles);
            userRepository.save(user);

            return Response
                    .<User>ok()
                    .setPayload(user);
        } catch (RuntimeException ex) {
            return Response
                    .<User>exception()
                    .setSuccess(false)
                    .setErrors(ex.getMessage());
        }
    }
}
```

- **CategoryRestController.java**

```
 package
org.ksga.springboot.khmeranguliapi.controller.rest
;
import
org.ksga.springboot.khmeranguliapi.model.Category;
import
org.ksga.springboot.khmeranguliapi.payload.dto.Art
icleDTO;
import
org.ksga.springboot.khmeranguliapi.payload.dto.Cat
egoryDTO;
import
org.ksga.springboot.khmeranguliapi.payload.mapper.
CategoryMapper;
import
org.ksga.springboot.khmeranguliapi.payload.request
.CategoryRequest;
import
org.ksga.springboot.khmeranguliapi.payload.respons
e.Response;
import
org.ksga.springboot.khmeranguliapi.service.Categor
yService;
import
org.springframework.beans.factory.annotation.Autow
ired;
import org.springframework.web.bind.annotation.*;
import java.util.List;
@RestController
@CrossOrigin(origins={"*","http://139.162.28.200:1
603/","http://localhost:3000/"})
@RequestMapping("/api/v1/categories")
public class CategoryRestController {
    @Autowired
    private  CategoryService categoryService;
    @Autowired
    private CategoryMapper categoryMapper;
    public CategoryRestController(CategoryService
categoryService) {
        this.categoryService = categoryService;
    }
    @GetMapping("/displayAll")
    public Response<List<CategoryDTO>>
displayAllCategory(){
      List<CategoryDTO> categories =
categoryService.findAllCategory();
        if(!categories.isEmpty()){
            return
Response.<List<CategoryDTO>>ok().setPayload(catego
ries);
        }else{
            return Response.notFound();
        }
    }
    @GetMapping("/displayById/{id}")
```

```
    public Response<List<CategoryDTO>> findById
(@PathVariable int id){
        List<CategoryDTO> findOne =
categoryService.findOneCategory(id);
        if(!findOne.isEmpty()){
            return
Response.<List<CategoryDTO>>ok().setPayload(findOn
e);
        }else{
            return Response.notFound();
        }
    }
    @PostMapping("/addNew")
    public Response<CategoryDTO>
addNewCategory(@RequestBody CategoryRequest
categoryRequest){
        boolean isCreated =
categoryService.insertCategory(categoryRequest);
        if (isCreated){
            Category category =
categoryMapper.categoryRequestToCategory(categoryR
equest);
            CategoryDTO categoryDTO =
categoryMapper.categoryToCategoryDto(category);
            return
Response.<CategoryDTO>ok().setPayload(categoryDTO)
;
        } else {
            return
Response.<CategoryDTO>badRequest().setErrors("Chec
k your data again");
        }
    }
    @PutMapping("update/{id}")
    public Response<CategoryDTO>
updateCategory(@PathVariable int id,@RequestBody
CategoryRequest category){
        boolean isUpdated =
categoryService.updateCategory(id,category);
        if(isUpdated){
            return Response.<CategoryDTO>ok();
        }else {
            return Response.notFound();
        }
    }
    @DeleteMapping("delete/{id}")
    public Response<CategoryDTO>
deleteCategory(@PathVariable("id") int id){
        boolean isDeleted =
categoryService.deleteCategory(id);
        if(isDeleted){
            return Response.<CategoryDTO>ok();
        }else {
            return Response.notFound();
        }
    }
}
```

- **EventRestController.java**

```
package
org.ksga.springboot.khmeranguliapi.controller.rest
;
import
org.ksga.springboot.khmeranguliapi.model.Event;
import
org.ksga.springboot.khmeranguliapi.payload.dto.Eve
ntDTO;
import
org.ksga.springboot.khmeranguliapi.payload.mapper.
EventMapper;
import
org.ksga.springboot.khmeranguliapi.payload.request
.EventRequest;
import
org.ksga.springboot.khmeranguliapi.payload.respons
e.Response;
import
org.ksga.springboot.khmeranguliapi.service.EventSe
rvice;
import
org.springframework.beans.factory.annotation.Autow
ired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@CrossOrigin(origins={"*","http://139.162.28.200:1
603/","http://localhost:3000/"})
@RestController
@RequestMapping("/api/v1/events")
public class EventRestController {

    private final EventService eventService;
    private EventMapper eventMapper;
    @Autowired
    public EventRestController(EventService
eventService) {
        this.eventService = eventService;
    }
    @PostMapping("/addNew")
    public Response<EventDTO> create (@RequestBody
EventRequest eventRequest){
        boolean isCreated =
eventService.saveEvent(eventRequest);
        if (isCreated) {
            return Response.ok();
        } else {
            return
Response.<EventDTO>badRequest().setErrors("Check
your data again");
        }
    }

    @GetMapping("/findALl")
    public Response<List<EventDTO>> displayAll(){
        List<EventDTO> events =
eventService.findAll();
        if(!events.isEmpty()){
            return
Response.<List<EventDTO>>ok().setPayload(events);
        }else{
            return Response.notFound();
        }
    }
    @GetMapping("/findById/{id}")
    public Response<List<EventDTO>>
displayById(@PathVariable int id){
        List<EventDTO> event =
eventService.findById(id);
        if(!event.isEmpty()){
            return
Response.<List<EventDTO>>ok().setPayload(event);
        }else{
            return Response.notFound();
        }
    }
    @PutMapping("/update/{id}")
    public Response<EventDTO> update(@PathVariable
int id,@RequestBody EventRequest eventRequest){
        boolean isUpdated =
eventService.update(id,eventRequest);
        if(isUpdated){
            return Response.<EventDTO>ok();
        }else {
            return Response.notFound();
        }
    }
```

```
    @DeleteMapping("/delete/{id}")
    public Response<EventDTO> delete(@PathVariable
int id){
        boolean isDeleted =
eventService.delete(id);
        if(isDeleted){
            return Response.<EventDTO>ok();
        }else {
            return Response.notFound();
        }
    }
}
```

- **FileStorageRestController.java**

```
package
org.ksga.springboot.khmeranguliapi.controller.rest
;

import org.springframework.core.io.Resource;
import org.springframework.core.io.UrlResource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.*;
import
org.springframework.web.multipart.MultipartFile;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

import static java.nio.file.Files.copy;
import static java.nio.file.Paths.get;
import static
java.nio.file.StandardCopyOption.REPLACE_EXISTING;
import static
org.springframework.http.HttpHeaders.CONTENT_DISPO
SITION;
@CrossOrigin(origins={"*","http://139.162.28.200:1
603/","http://localhost:3000/"})
@RestController
@RequestMapping("/api/file")
public class FileStorageRestController {
    // define a location
    public static final String DIRECTORY =
System.getProperty("user.home") +
"\\Downloads\\uploads";

    // Define a method to upload files, return
image name
    @PostMapping("/upload")
    public ResponseEntity<List<String>>
uploadFiles(@RequestParam("files")List<MultipartFi
le> multipartFiles) throws IOException {
        List<String> filenames = new
ArrayList<>();
        for(MultipartFile file : multipartFiles) {
            UUID uuid = UUID.randomUUID();
            String filename =
StringUtils.cleanPath(uuid+file.getOriginalFilenam
e());
            Path fileStorage =
get(DIRECTORY,filename).toAbsolutePath().normalize
();
            copy(file.getInputStream(),
fileStorage,REPLACE_EXISTING);
//
filenames.add(DIRECTORY+"\\"+filename);
            filenames.add(filename);
        }
        return
ResponseEntity.ok().body(filenames);
    }

    // Define a method to download files
    @GetMapping("download/{filename}")
    public ResponseEntity<Resource>
downloadFiles(@PathVariable("filename") String
filename) throws IOException {
        Path filePath =
get(DIRECTORY).toAbsolutePath().normalize().resolv
e(filename);
        if(!Files.exists(filePath)) {
            throw new
FileNotFoundException(filename + " was not found
on the server");
        }
        Resource resource = new
UrlResource(filePath.toUri());
        HttpHeaders httpHeaders = new
HttpHeaders();
        httpHeaders.add("File-Name", filename);
        httpHeaders.add(CONTENT_DISPOSITION,
"attachment;File-Name=" + resource.getFilename());
        return
ResponseEntity.ok().contentType(MediaType.parseMed
iaType(Files.probeContentType(filePath)))

.headers(httpHeaders).body(resource);
    }
}
```

- **MessageRestController.java**

```
package
org.ksga.springboot.khmeranguliapi.controller.rest
;

import
org.ksga.springboot.khmeranguliapi.model.Message;
import
org.ksga.springboot.khmeranguliapi.payload.dto.Mes
sageDTO;
import
org.ksga.springboot.khmeranguliapi.payload.mapper.
MessageMapper;
import
org.ksga.springboot.khmeranguliapi.payload.request
.MessageRequest;
import
org.ksga.springboot.khmeranguliapi.payload.respons
e.Response;
import
org.ksga.springboot.khmeranguliapi.service.Message
Service;
import
org.springframework.beans.factory.annotation.Autow
ired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@CrossOrigin(origins={"*","http://139.162.28.200:1
603/","http://localhost:3000/"})
@RequestMapping("/api/v1/message")
public class MessageRestController {
    @Autowired
    private MessageService messageService;
    @Autowired
    private MessageMapper messageMapper;
    @PostMapping("/send/{sender}/{receiver}")
    public Response<MessageDTO>
sendMessage(@PathVariable int sender,@PathVariable
int receiver, @RequestBody MessageRequest
messageRequest) {
```

```
        boolean isSend =
messageService.insert(sender,receiver,messageReque
st);
        if (isSend) {
            Message message =
messageMapper.messageRequestToMessage(messageReque
st);
            MessageDTO messageDTO =
messageMapper.messageToMessageDto(message);
            return
Response.<MessageDTO>ok().setPayload(messageDTO);
        } else {
            return
Response.<MessageDTO>badRequest().setErrors("Check
your data again");
        }
    }
    @GetMapping("/connected/{id}")
    public Response<List<MessageDTO>>
findConnectedMessage(@PathVariable int id) {
        List<MessageDTO> message =
messageService.findMessagedRecord(id);
        if(!message.isEmpty()){
            return
Response.<List<MessageDTO>>ok().setPayload(message
);
        }else{
            return Response.notFound();
        }
    }

    //Todo:need Recursive to filter relationship
    @GetMapping("/view/{id}")
    public Response<List<MessageDTO>>
findByOwnedID(@PathVariable int id) {
        List<MessageDTO> message =
messageService.findOwnedMessage(id);
        if(!message.isEmpty()){
            return
Response.<List<MessageDTO>>ok().setPayload(message
);
        }else{
            return Response.notFound();
        }
    }
    //Todo:Display all messaged user profile by
select distinct
    // by your own id.
    @PostMapping("/delete/{id}")
    public Response<MessageDTO>
deleteMessage(@PathVariable int id) {
        boolean isDeleted =
messageService.delete(id);
        if(isDeleted){
            return Response.<MessageDTO>ok();
        }else {
            return Response.notFound();
        }
    }
}
```

- **ReportRestController.java**

```
package
org.ksga.springboot.khmeranguliapi.controller.rest
;

import
org.ksga.springboot.khmeranguliapi.model.Event;
import
org.ksga.springboot.khmeranguliapi.model.Report;
import
org.ksga.springboot.khmeranguliapi.model.SaveRecor
d;
import
org.ksga.springboot.khmeranguliapi.payload.dto.Rep
ortDTO;
import
org.ksga.springboot.khmeranguliapi.payload.dto.Sav
eRecordDTO;
import
org.ksga.springboot.khmeranguliapi.payload.mapper.
ReportMapper;
import
org.ksga.springboot.khmeranguliapi.payload.request
.ReportRequest;
import
org.ksga.springboot.khmeranguliapi.payload.respons
e.Response;
import
org.ksga.springboot.khmeranguliapi.security.servic
e.UserDetailsImpl;
import
org.ksga.springboot.khmeranguliapi.service.ReportS
ervice;
import
org.springframework.beans.factory.annotation.Autow
ired;
import
org.springframework.security.core.annotation.Authe
nticationPrincipal;
import
org.springframework.security.core.parameters.P;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@CrossOrigin(origins={"*","http://139.162.28.200:1
603/","http://localhost:3000/"})
@RestController
@RequestMapping("/api/v1/reports")
public class ReportRestController {
    @Autowired
    private final ReportService reportService;
    @Autowired
    private ReportMapper reportMapper;
    public ReportRestController(ReportService
reportService) {
        this.reportService = reportService;
    }
    @PostMapping("/addReport")
    public Response<ReportDTO> addReport
(@AuthenticationPrincipal UserDetailsImpl
user,@RequestBody ReportRequest reportRequest){
        try{
            boolean isAdded =
reportService.insert(user.getUser_id(),reportReque
st);
            if (isAdded) {
                Report report =
reportMapper.reportRequestToReport(reportRequest);
                ReportDTO reportDTO =
reportMapper.reportToReportDto(report);
                return
Response.<ReportDTO>ok().setPayload(reportDTO);
            } else {
                return
Response.<ReportDTO>badRequest().setErrors("Check
your data again");
            }
        } catch (Exception e) {
            return Response.unauthorized();
        }
    }
    @GetMapping("/findALL")
    public Response<List<ReportDTO>> displayAll(){
        List<ReportDTO> reports =
reportService.findAll();
        if(!reports.isEmpty()){
```

```
            return
Response.<List<ReportDTO>>ok().setPayload(reports)
;
        }else{
            return Response.notFound();
        }
    }
    @GetMapping("/findByUserId/{user_id}")
    public Response<List<ReportDTO>>
displayByUserId(@PathVariable int user_id){
        List<ReportDTO> report =
reportService.findByUserId(user_id);
        if(!report.isEmpty()){
            return
Response.<List<ReportDTO>>ok().setPayload(report);
        }else{
            return Response.notFound();
        }
    }
    @DeleteMapping("/delete/{id}")
    public Response<ReportDTO>
delete(@PathVariable int id){
        boolean isDelete =
reportService.delete(id);
        if(isDelete){
            return Response.<ReportDTO>ok();
        }else {
            return Response.notFound();
        }
    }

}
```

- **TestRestController.java**

```
package
org.ksga.springboot.khmeranguliapi.controller.rest
;

import
org.ksga.springboot.khmeranguliapi.model.auth.User
;
import
org.ksga.springboot.khmeranguliapi.repository.User
Repository;
import
org.ksga.springboot.khmeranguliapi.security.servic
e.UserDetailsImpl;
import
org.springframework.beans.factory.annotation.Autow
ired;
import
org.springframework.security.access.prepost.PreAut
horize;
import
org.springframework.security.core.annotation.Authe
nticationPrincipal;
import
org.springframework.web.bind.annotation.CrossOrigi
n;
import
org.springframework.web.bind.annotation.GetMapping
;
import
org.springframework.web.bind.annotation.RequestMap
ping;
import
org.springframework.web.bind.annotation.RestContro
ller;

import java.security.Principal;
import java.util.Optional;

@CrossOrigin(origins={"*","http://139.162.28.200:1
603/","http://localhost:3000/"})
@RestController
@RequestMapping("/api/test")
public class TestRestController {
    @Autowired
    UserRepository userRepository;

    @GetMapping("/all")
    public String allAccess() {
        return "Public Content.";
    }

    @GetMapping("/user")
    @PreAuthorize("hasRole('USER') or
hasRole('MODERATOR') or hasRole('ADMIN')")
    public int userAccess(@AuthenticationPrincipal
UserDetailsImpl user) {
        return user.getUser_id();
    }

    @GetMapping("/admin")
    @PreAuthorize("hasRole('ADMIN')")
    public String adminAccess(Principal principal)
    {
         Optional<User>
userID=userRepository.findUserByUsername(principal
.getName());

System.out.println(userID.get().getUser_id());
         return "Hello ID: "
+userID.get().getUser_id()+ " " +
principal.getName();
    }
}
```

- **UserProfileRestController.java**

```
package
org.ksga.springboot.khmeranguliapi.controller.rest
;
import
org.ksga.springboot.khmeranguliapi.payload.dto.Use
rProfileDTO;
import
org.ksga.springboot.khmeranguliapi.payload.mapper.
UserProfileMapper;
import
org.ksga.springboot.khmeranguliapi.payload.request
.UserProfileRequest;
import
org.ksga.springboot.khmeranguliapi.payload.request
.UserSearchProfileRequest;
import
org.ksga.springboot.khmeranguliapi.payload.respons
e.Response;
import
org.ksga.springboot.khmeranguliapi.repository.User
Repository;
import
org.ksga.springboot.khmeranguliapi.security.servic
e.UserDetailsImpl;
import
org.ksga.springboot.khmeranguliapi.service.UserPro
fileService;
import
org.springframework.beans.factory.annotation.Autow
ired;
import
org.springframework.security.access.prepost.PreAut
horize;
import
org.springframework.security.core.Authentication;
import
org.springframework.security.core.annotation.Authe
nticationPrincipal;
import
org.springframework.security.core.context.Security
ContextHolder;
```

```
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@CrossOrigin(origins={"*","http://139.162.28.200:1
603/","http://localhost:3000/"})
@RequestMapping("/api/v1/user")
public class UserProfileRestController {
    @Autowired
    private UserProfileService userProfileService;
    @Autowired
    UserProfileMapper userProfileMapper;
    @Autowired
    UserRepository userRepository;
//    Get all user mainly for admin to control
    @GetMapping("/all")
    public Response<List<UserProfileDTO>>
findAll() {
        List<UserProfileDTO> users=
userProfileService.findAll();
        System.out.println(users);
        if(!users.isEmpty()){
            return
Response.<List<UserProfileDTO>>ok().setPayload(use
rs);
        }else{
            return Response.notFound();
        }
    }
//    Disable user account if they are violete the
application term of service

    @PostMapping("/disable/{id}")
    @PreAuthorize("hasRole('ADMIN')")
    public Response<UserProfileDTO>
disableAccount(@PathVariable int id) {
        boolean isDisabled =
userProfileService.disable(id);
        if(isDisabled){
            return
Response.<UserProfileDTO>ok().setSuccess(true);
        }else {
            return Response.notFound();
        }
    }


    //    requrie header
    @GetMapping("/findByUserName/{id}")
    public Response<UserProfileDTO>
findAccountUsername(@PathVariable String id) {
        UserProfileDTO userProfileDTO =
userProfileService.findOne(id);
        if(userProfileDTO!=null){
            return
Response.<UserProfileDTO>ok().setPayload(userProfi
leDTO);
        }else {
            return Response.notFound();
        }
    }
    //    requrie header
    @GetMapping("/findOwnAccount")
    public Response<UserProfileDTO>
findOwnedAccount(@AuthenticationPrincipal
UserDetailsImpl user) {
            try{
                UserProfileDTO userProfileDTO
=userProfileService.findOwnerAccount(user.getUser_
id());
//
System.out.println(userProfileDTO);
                if(userProfileDTO!=null){
                    return
Response.<UserProfileDTO>ok().setPayload(userProfi
leDTO);
                }else {
                    return Response.notFound();
                }
            } catch (Exception e) {
                return Response.unauthorized();
            }
    }

//    Search User
    @GetMapping("/search")
    public List<UserProfileDTO>
searchUserByUserName( UserSearchProfileRequest
search){
        return
userProfileService.searchUserByUsername(search);
    }
    //Update user profile by user id from header
    @PostMapping("/update")
    public Response<UserProfileDTO>
updateProfile(@AuthenticationPrincipal
UserDetailsImpl user, @RequestBody
UserProfileRequest userProfileRequest) {
        try{
            Authentication authentication =
SecurityContextHolder.getContext().getAuthenticati
on();
            String
currentUser=authentication.getName();
            try{
                if(currentUser.isEmpty()){
                    return Response.notFound();
                }else {
                    boolean isUpdated =
userProfileService.update(user.getUser_id(),
userProfileRequest);
                    if (isUpdated) {
                        return
Response.<UserProfileDTO>ok();
                    } else {
                        return
Response.notFound();
                    }
                }
            } catch (Exception e) {
                return Response.unauthorized();
            }
        } catch (Exception e) {
           return Response.unauthorized();
        }
    }
}
```

- **SaveRecordController.java**

```
package
org.ksga.springboot.khmeranguliapi.controller.rest
;

import
org.ksga.springboot.khmeranguliapi.model.SaveRecor
d;
import
org.ksga.springboot.khmeranguliapi.payload.dto.Sav
eRecordDTO;
import
org.ksga.springboot.khmeranguliapi.payload.mapper.
SaveRecordMapper;
import
org.ksga.springboot.khmeranguliapi.payload.request
.SaveRecordRequest;
import
org.ksga.springboot.khmeranguliapi.payload.respons
e.Response;
```

```
import
org.ksga.springboot.khmeranguliapi.security.servic
e.UserDetailsImpl;
import
org.ksga.springboot.khmeranguliapi.service.SaveRec
ordService;
import
org.springframework.beans.factory.annotation.Autow
ired;
import
org.springframework.security.core.Authentication;
import
org.springframework.security.core.annotation.Authe
nticationPrincipal;
import
org.springframework.security.core.context.Security
ContextHolder;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@CrossOrigin(origins={"http://localhost:3000/"})
@RestController
@RequestMapping("/api/v1/records")
public class SaveRecordRestController {
    @Autowired
    private SaveRecordService saveRecordService;
    @Autowired
    private SaveRecordMapper saveRecordMapper;
    public
SaveRecordRestController(SaveRecordService
playRecordService) {
        this.saveRecordService =
playRecordService;
    }
    @GetMapping("/displayAll")
    public Response<List<SaveRecordDTO>> findAll
(){
        List<SaveRecordDTO> playRecord =
saveRecordService.findAll();
        if(!playRecord.isEmpty()){
            return
Response.<List<SaveRecordDTO>>ok().setPayload(play
Record);
        }else{
            return Response.notFound();
        }
    }
    @GetMapping("/displayByUserHeader")
    public Response<List<SaveRecordDTO>>
findById(@AuthenticationPrincipal UserDetailsImpl
user){
        try{
            Authentication authentication =
SecurityContextHolder.getContext().getAuthenticati
on();;
            String
currentUser=authentication.getName();
            try{
                List<SaveRecordDTO> findByUserId =
saveRecordService.findById(user.getUser_id());
                if(!findByUserId.isEmpty()){
                    return
Response.<List<SaveRecordDTO>>ok().setPayload(find
ByUserId);
                }else{
                    return Response.notFound();
                }
            } catch (Exception e) {
                return Response.unauthorized();
            }
        } catch (Exception e) {
            return Response.unauthorized();
        }
    }
```

```
    @GetMapping("/displayByUserId/{id}")
    public Response<List<SaveRecordDTO>>
findById(@PathVariable int id){
        List<SaveRecordDTO> findByUserId =
saveRecordService.findById(id);
        if(!findByUserId.isEmpty()){
            return
Response.<List<SaveRecordDTO>>ok().setPayload(find
ByUserId);
        }else{
            return Response.notFound();
        }
    }

    @DeleteMapping("/delete/{id}")
    public Response<SaveRecordDTO>
delete(@PathVariable int id){
        boolean isDeleted =
saveRecordService.delete(id);
        if(isDeleted){
            return Response.<SaveRecordDTO>ok();
        }else {
            return Response.notFound();
        }
    }

    @PostMapping("/addRecord")
    public Response<SaveRecordDTO>
updateProfile(@AuthenticationPrincipal
UserDetailsImpl user, @RequestBody
SaveRecordRequest saveRecordRequest) {
            try{
                boolean isAdded =
saveRecordService.save(user.getUser_id(),
saveRecordRequest);
                if (isAdded) {
                    SaveRecord saveRecord =
saveRecordMapper.saveReocordRequestToSaveRecord(sa
veRecordRequest);
                    SaveRecordDTO saveRecordDTO =
saveRecordMapper.saveRecordToSaveRecordDTO(saveRec
ord);
                    return
Response.<SaveRecordDTO>ok().setPayload(saveRecord
DTO);
                } else {
                    return
Response.<SaveRecordDTO>badRequest().setErrors("Ch
eck your data again");
                }
            } catch (Exception e) {
                return Response.unauthorized();
            }
    }
    }
```

- **AuthController.java**

```
package
org.ksga.springboot.khmeranguliapi.controller;

import org.springframework.stereotype.Controller;
import
org.springframework.web.bind.annotation.GetMapping
;

@Controller
public class AuthController {

    @GetMapping("/auth/login")
    public String login() {
        return "auth/login";
    }

}
```

- **HomeController.java**

```
package
org.ksga.springboot.khmeranguliapi.controller;

import
org.springframework.security.access.prepost.PreAut
horize;
import org.springframework.stereotype.Controller;
import
org.springframework.web.bind.annotation.GetMapping
;

@Controller
public class HomeController {
    @PreAuthorize("hasRole('ADMIN')")
    @GetMapping("/")
    public String index() {
        return "index";
    }
}
```

## VI EVALUATION

**Khmer Anguli** is a web application that helps people enhance their Khmer typing skill. Our team improved and added some features, which are useful and popular for users to their work. We also provide a new UI (user interface).

### 6.1 Project Strength

**Khmer Anguli** has some strength points such as:

- User can play on our website directly without having account.
- Admin fully controls on the website
- User can chat to other users.
- Consumer can report users who go against the policy on the website.

### 6.2 Project Weakness

Even though we have a lot of strengths but we also have some weaknesses:

- User cannot invite their partner or team members to type
- In chat feature has no block function
- User cannot send voice in chat feature.

## VII CONCLUSION

**Khmer Anguli** is a Khmer typing platform with multiple beneficial features to help users improve their Khmer typing skill. It provides lots of features that users need, such as multiple levels, multiple categories and articles, clear guideline, accuracy, and chat feature that all of them 5are so useful and popular.

## I REFERENCES

1. Feature: www.facebook.com
2. User Interface: www.best.aliexpress.com
3. System Architecture: www.geeksforgeeks.org
   www.github.com