# Spring Fundamentals Exam

# Dictionary Application

Exam for the ["Spring Fundamentals" course @ SoftUni](https://judge.softuni.org/Contests/4407/Spring-Fundamentals-Exam-21-Oct-2023).

**Dictionary App** is designed to help individuals and language enthusiasts effortlessly access and explore the vast world of words and their meanings. It provides a user-friendly interface with a comprehensive dictionary that allows users to search for words, browse definitions, and enhance their vocabulary.

Dictionary App is a valuable tool for students, writers, professionals, and anyone seeking to enrich their language skills. It provides an extensive database of words and their meanings, making it a reliable companion for German, French, Spanish and Italian.

There are several requirements you must follow in the implementation:

## Database Requirements

The **Database** of the **Dictionary** application needs to support **3 entities**:

### User

* Has an **Id –** "**UUID-String**" **or Long**
* Has a Username (unique, not null)
  + Username length must be between 3 and 20 characters (inclusive of 3 and 20).
* Has a Password (not null)
  + Password length must be between 3 and 20 characters (inclusive of 3 and 20).
* Has an Email (unique, not null)
  + Must contain '@'.
* Has addedWords
  + The addedWords is a collection that contains all words that the user added. One user may have many word and one word can be added by only one user.

### Word

* Has an **Id –** "**UUID-String**" **or Long**
* Has a Term (not null)
  + Term length must be between 2 and 40 characters (inclusive of 2 and 40).
* Has a Translation (not null)
  + Translation length must be between 2 and 80 characters (inclusive of 2 and 80).
* Has an Example
  + Example length must be between 2 and 200 characters (inclusive of 2 and 200).
* Has an inputDate – date (not null)
  + The input date must be a date in the past or present.
* Has a Language (not null)
  + One word has one language and one language can have many words.
* Has an addedBy
  + The user who added the word in the dictionary. One word has one user, but one user may have many words.

### Language

* Has an **Id** –"**UUID-String**" **or Long**
* Has a **Language name** (unique, not null)
  + an option between (GERMAN, SPANISH, FRENCH, ITALIAN)
* Has a Description (not null)
  + For GERMAN - "A West Germanic language, is spoken by over 90 million people worldwide. Known for its complex grammar and compound words, it's the official language of Germany and widely used in Europe."
  + For SPANISH – "A Romance language, is spoken by over 460 million people worldwide. It boasts a rich history, diverse dialects, and is known for its melodious sound, making it a global cultural treasure."
  + For FRENCH – "A Romance language spoken worldwide, known for its elegance and cultural richness. It's the official language of France and numerous nations, famed for its cuisine, art, and literature."
  + For ITALIAN - "A Romance language spoken in Italy and parts of Switzerland, with rich cultural heritage. Known for its melodious sounds, it's a gateway to Italian art, cuisine, and history."
* Has collection of **Words**
  + One language may **have many** words, but one word has **only one** language.

Implement the entities with the **correct data types** and implement **repositories** for them.

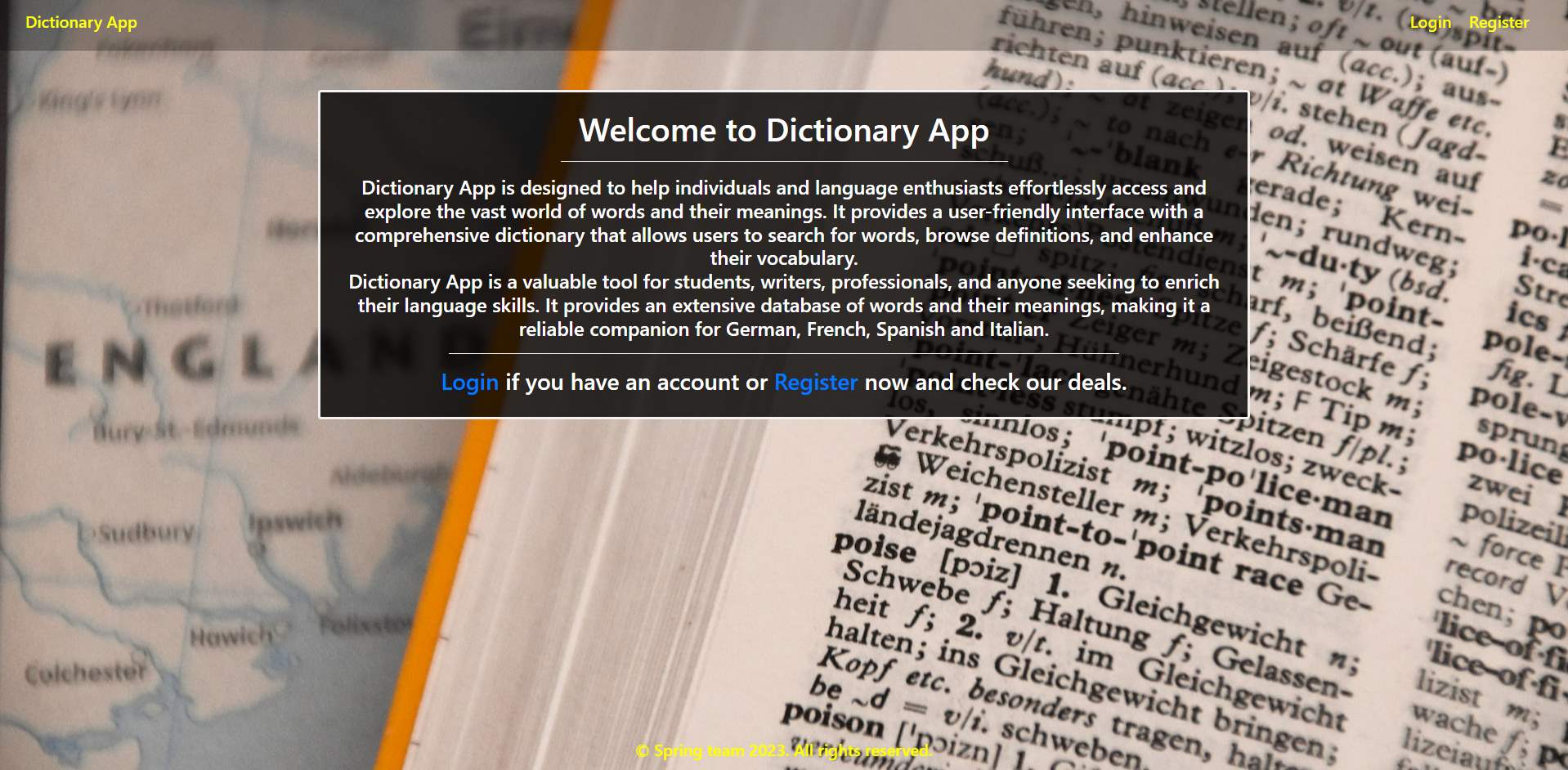
Here is the ER Diagram:

## Initialize languages

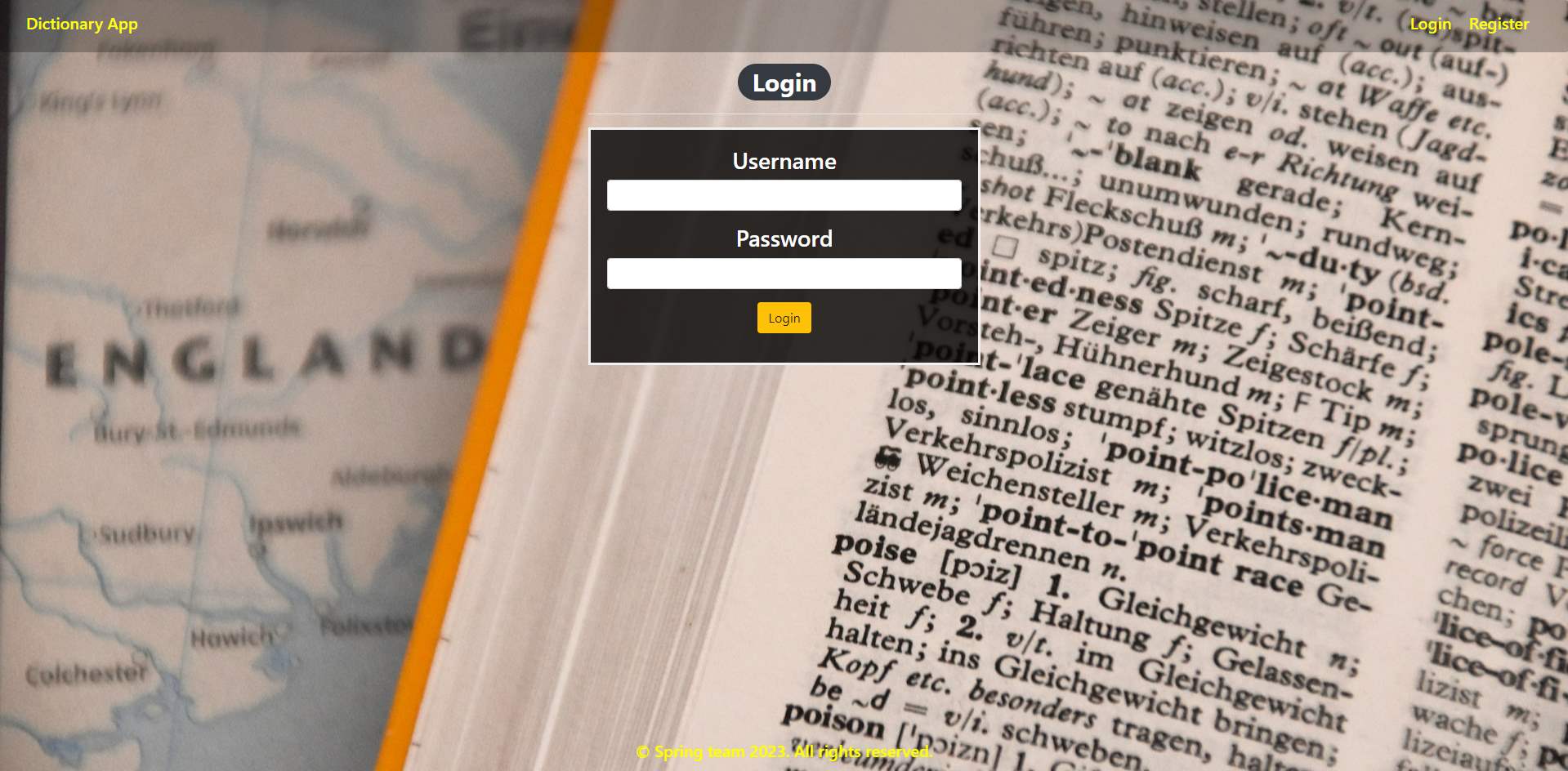
* Implement a method that checks (when the app started) if the database does not have **languages** and   
  initialize them
  + **You are free to do it in different ways.**

## Page Requirements

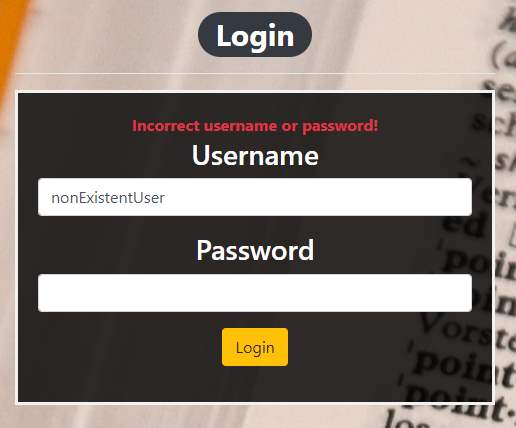
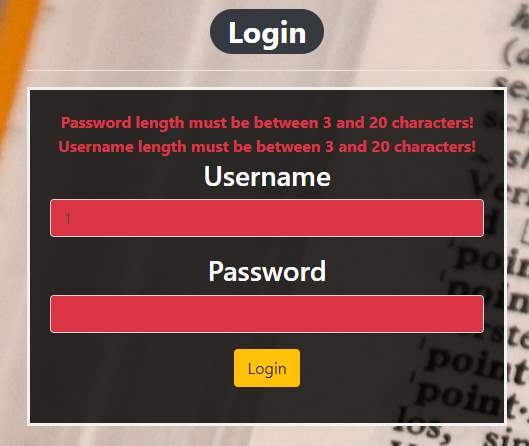
### Index Page (logged out user)



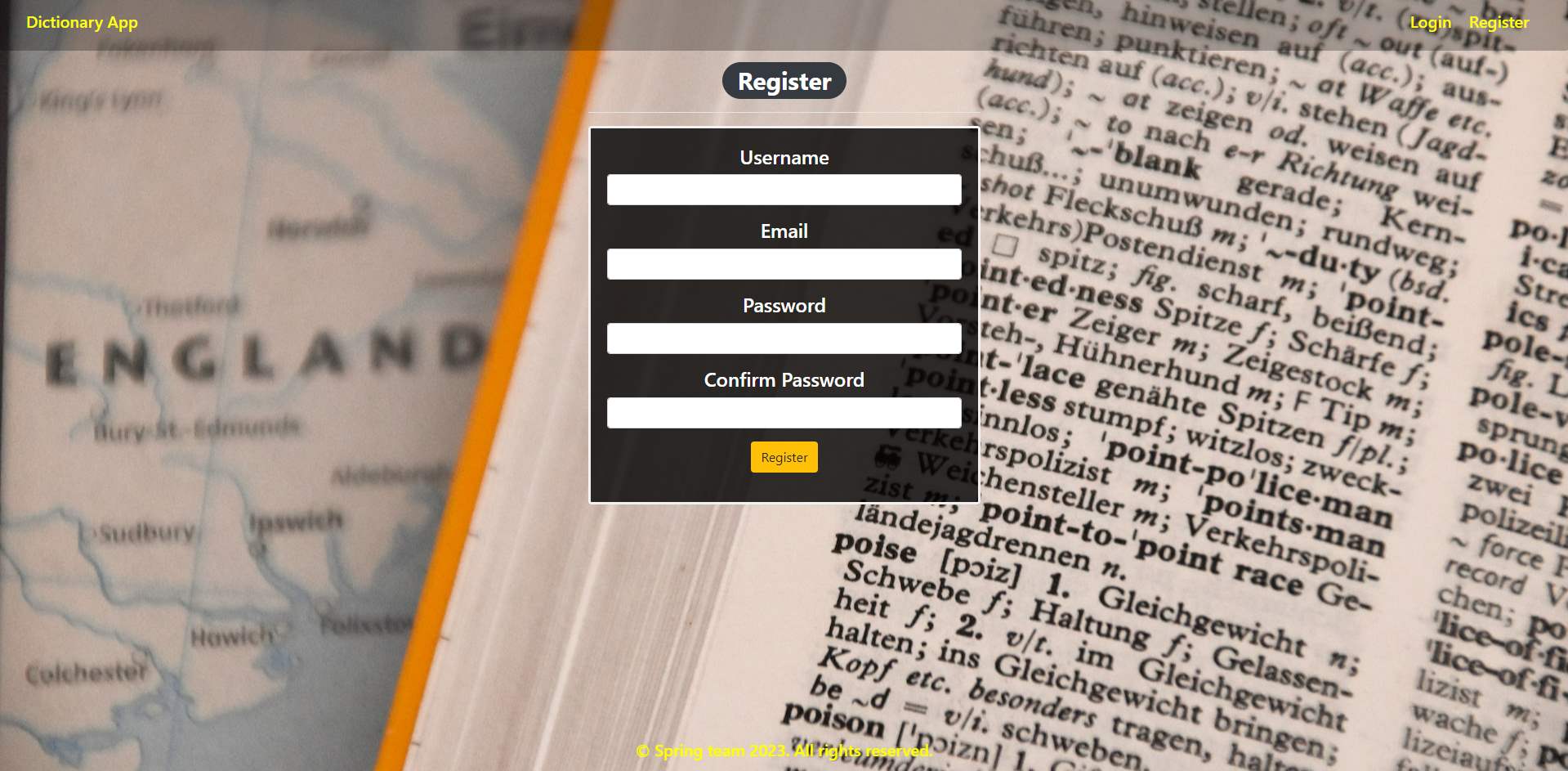
### Login Page (logged out user)



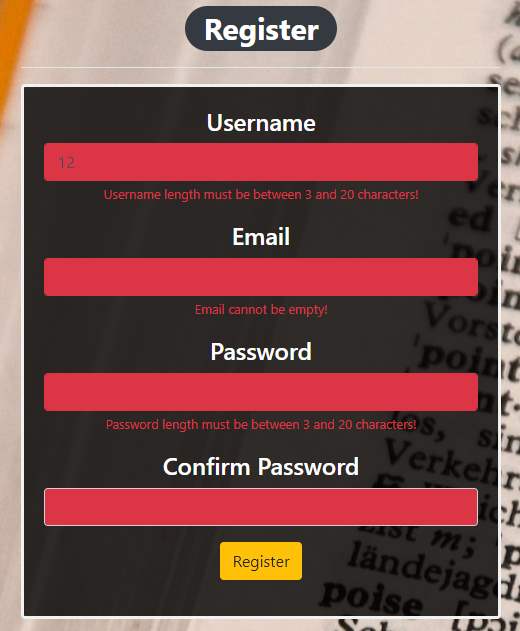
### Login Page validations

### Register Page (logged out user)

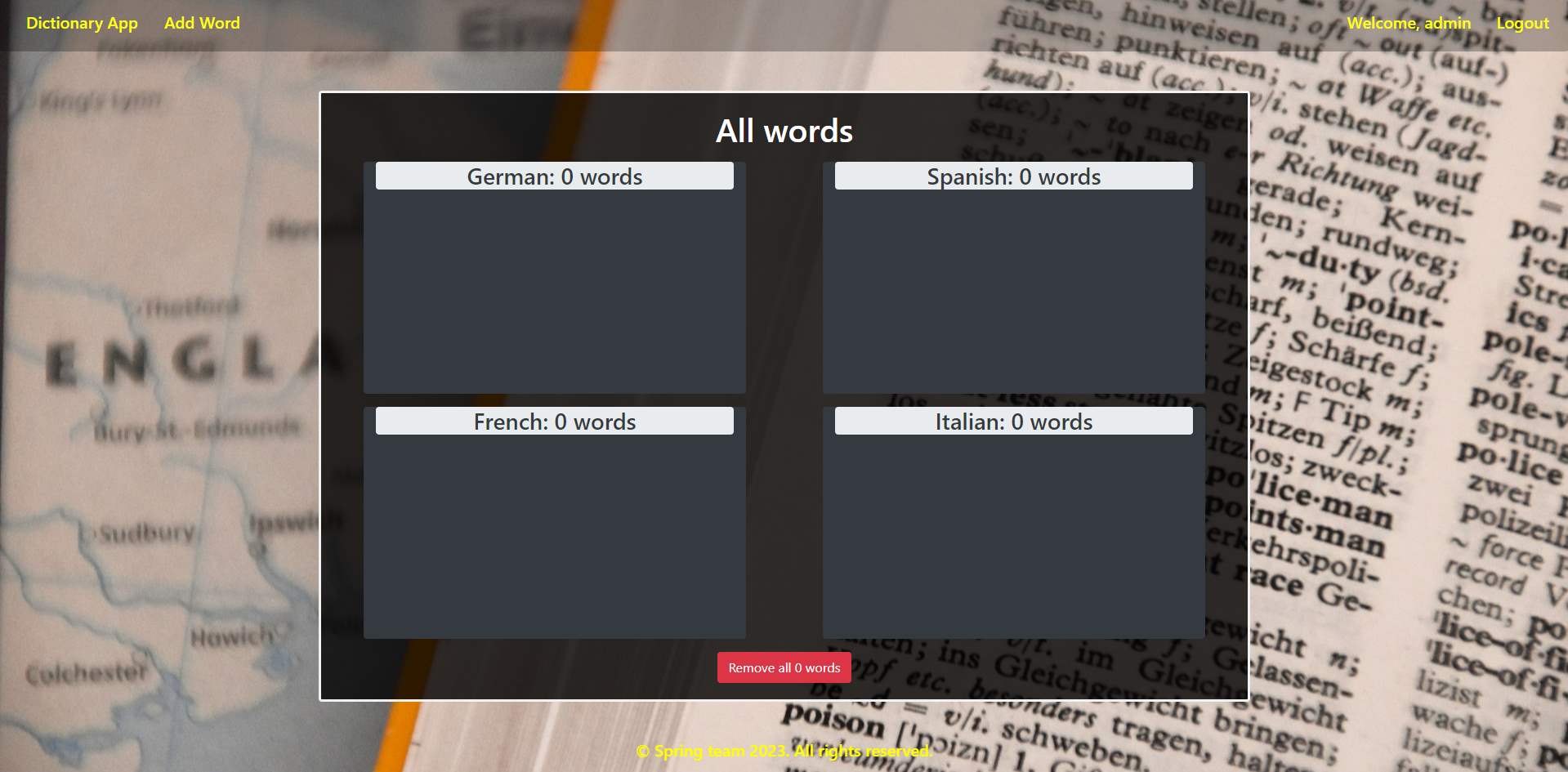


### Register Page validations

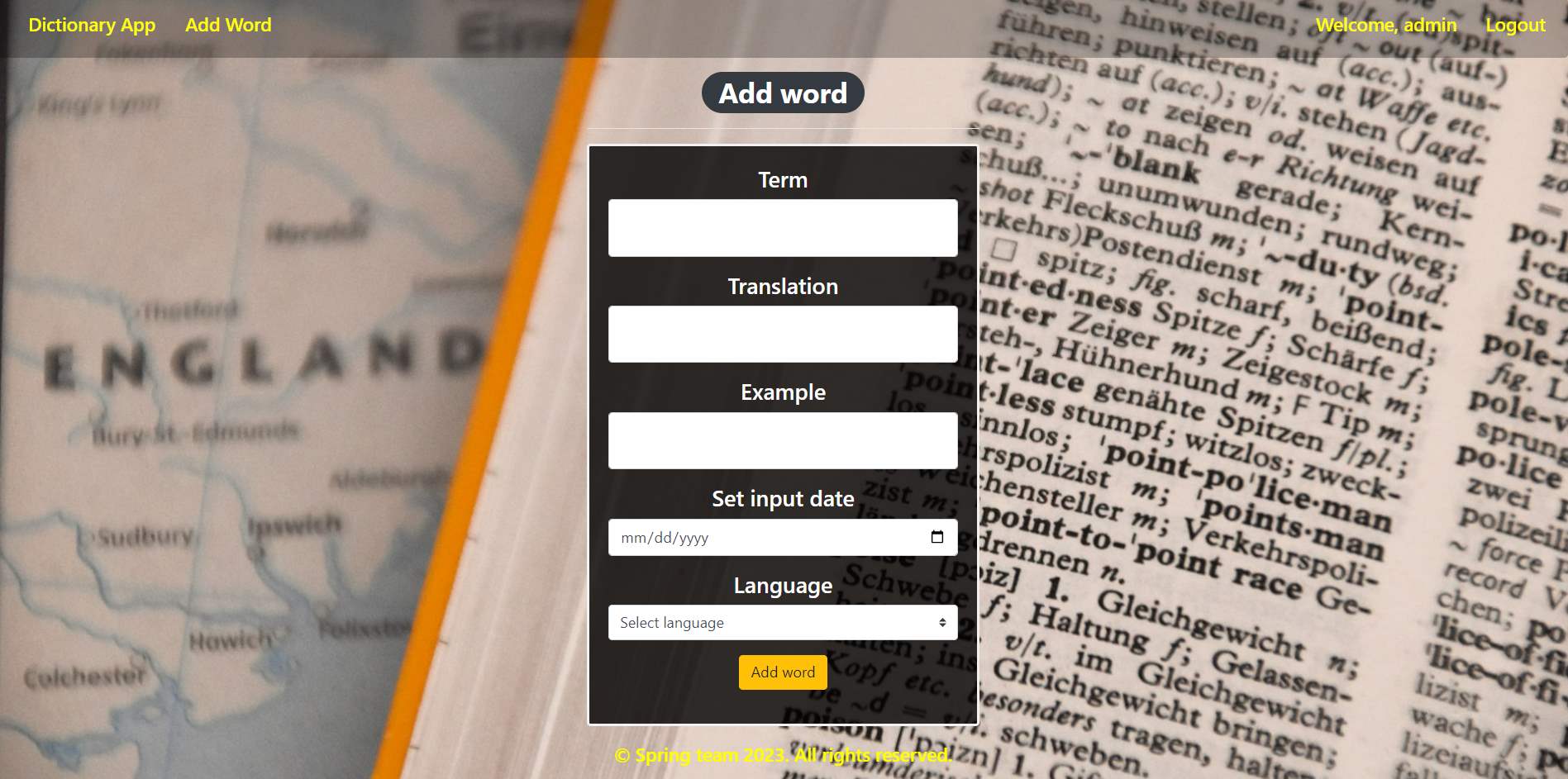


### Home Page (without having any words)

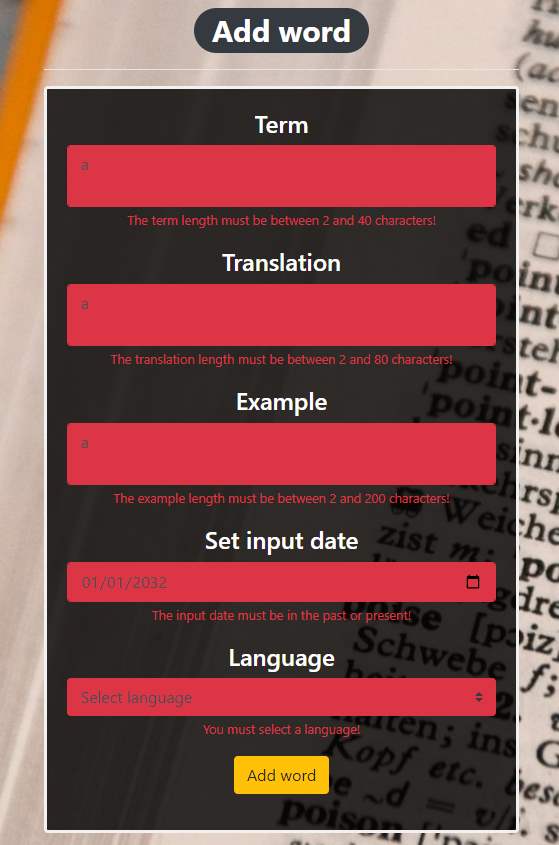
* The upper left section (German) of the page should visualize **all of the added German words** from the database from all users.
* The upper right section (Spanish) of the page should visualize **all of the added Spanish words** from the database from all users.
* The bottom left section (French) of the page should visualize **all of the added French words** from the database from all users.
* The bottom right section (Italian) of the page should visualize **all of the added Italian words** from the database from all users.



### Add word



### Add word validation



### Home Page (with words)



The templates have been given to you in the application skeleton, so make sure you implement the pages correctly.

**NOTE**: The templates should look **EXACTLY** as shown above.

**NOTE**: The templates do **NOT** **require** **additional** **CSS** for you to write. Only **bootstrap** and the **given CSS** are enough.

## Functional Requirements

The Functionality Requirements describe the functionality that the application must support.

The application should provide **Guest** (not logged-in) users with the functionality to log in, register, and view the Index page.

The application should provide **Users** (logged in) with the functionality to **log out, add a new word (Add Word page), view all words (Home page), remove one word with the "Remove" button, or remove all words from the dictionary with the "Remove all words" button.**

Dictionary Application in the navbar should **redirect** to the appropriate URL depending on if the **user is logged in**.

The application should provide functionality for **adding words**with language of **German, Spanish, French, or Italian**.

The words should be separated into their own categories. All German words in the German section, all French words in the French section, etc.

The **Remove** button next to the word **should remove the word**from DB.

Below the language sections is located a button "Remove all words {total count of words}" with an info that shows the **total count of all words**.

Below each word is located an info bar that shows the **username of the user who added the word and the input date** (in the format yyyy-MM-dd).

The application should store its data in a MySQL database.

## Security Requirements

The Security Requirements are mainly access requirements. Configurations about which users can access specific functionalities and pages.

* Guest (not logged in) users can access the Index page.
* Guest (not logged in) users can access the Login page.
* Guest (not logged in) users can access the Register page.
* Users (logged in) can access the Home page.
* Users (logged in) can access Add Word page.
* Users (logged in) can access Logout functionality.

## Submitting

* When submitting a solution it is necessary to make an **archived file** of **src + pom.xml**/**build.gradle** and **settings.gradle**.
* The judge's usual behavior is to return a "Compile time error" as a response when a solution is successfully uploaded.
* Only the **last**file submitted will be checked and evaluated.

## Scoring

### Database – 10 points.

### Pages – 25 points.

### Functionality – 35 points.

### Security – 5 points.

### Validations – 15 points.

### Code Quality – 10 points.