Welcome to the Korean Learning Service.

This service is created for self-teaching and self-learning.

This application is designed for users to practice Korean words that they want to learn. Users have the ability to add and remove words as desired. Default words are provided, however they may not be removed. Words that a user adds in are private only to that user, different accounts may have different words.

This is a simple application not intended for full purpose learning, but rather to demonstrate how to create a web-service with an MVC approach. However, this service is extremely useful for learning easy phrases in Korean. Enjoy ! :)

Site: ec2-54-190-55-152.us-west-2.compute.amazonaws.com/HackProject/index.php

This is a 3-tier web application.

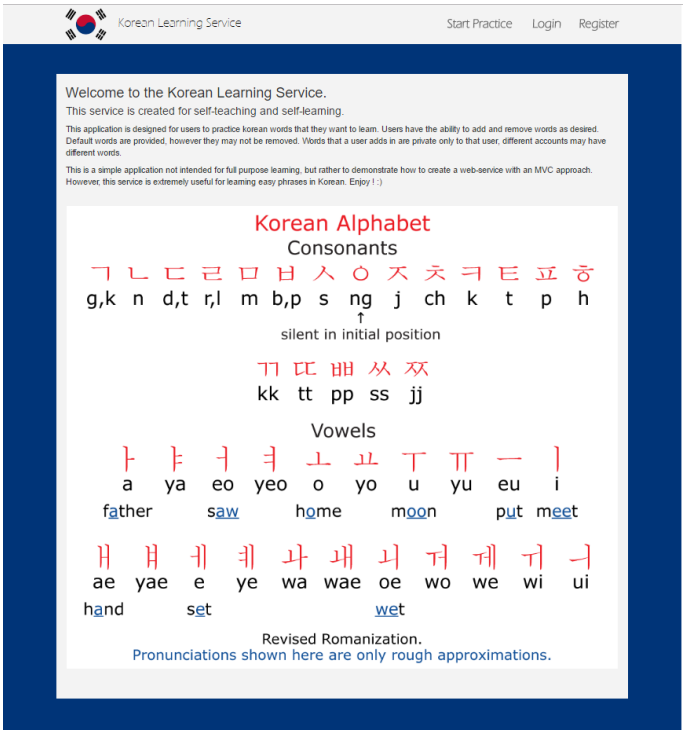
Web Application Layer: Display and the Multiple Choice Game was made using PHP, jQuery, and Ajax.

Web Service Layer: I wrote a RESTful service in PHP. The calls to this restful service was done via Ajax.

Database Layer: I used a MySQL Server for the database.

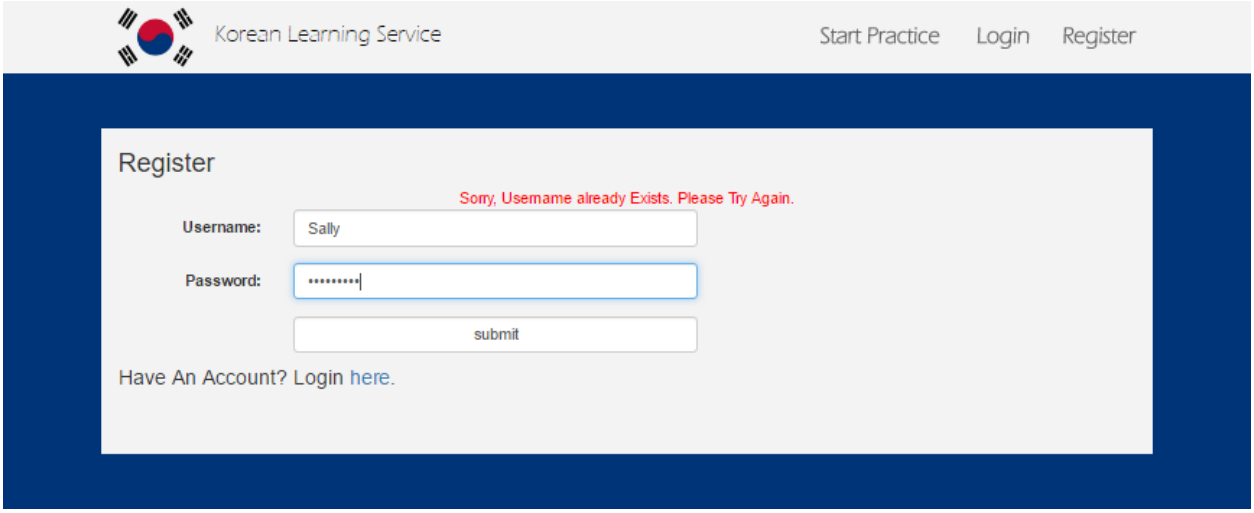
Stepping Through the Site.

The user is first presented with the alphabet of the Korean Language, as well as a description for what the site entails. Note the simple layout so users will not get lost.



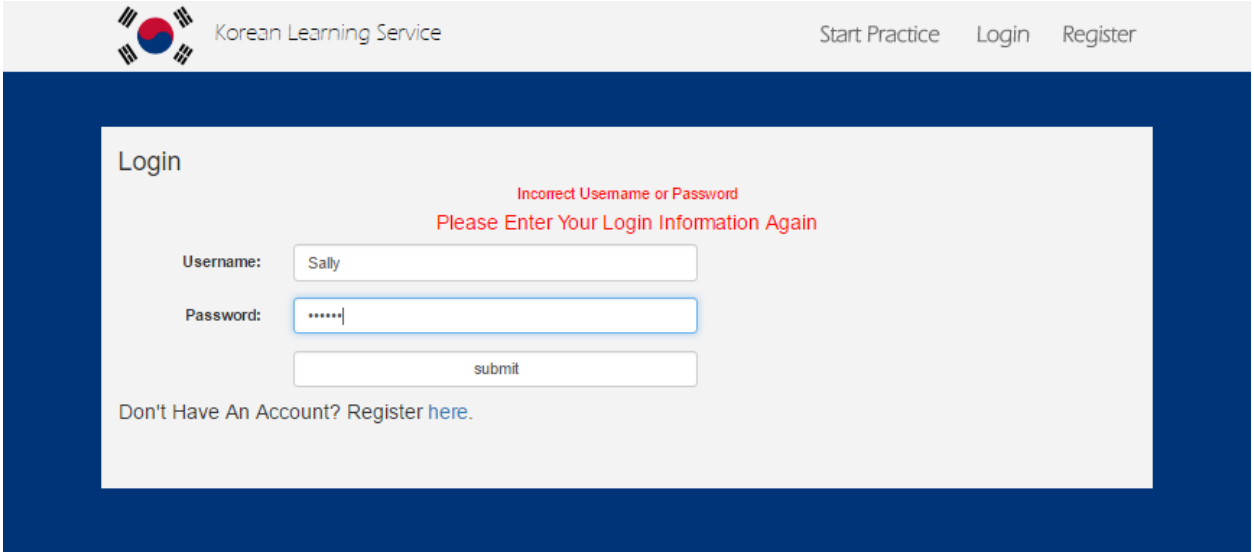
The Login

The login page was done using session variables and speaking with the MySQL database. Note the fabulous error checking! PHP was the core to the login system. Files can be found in /php.



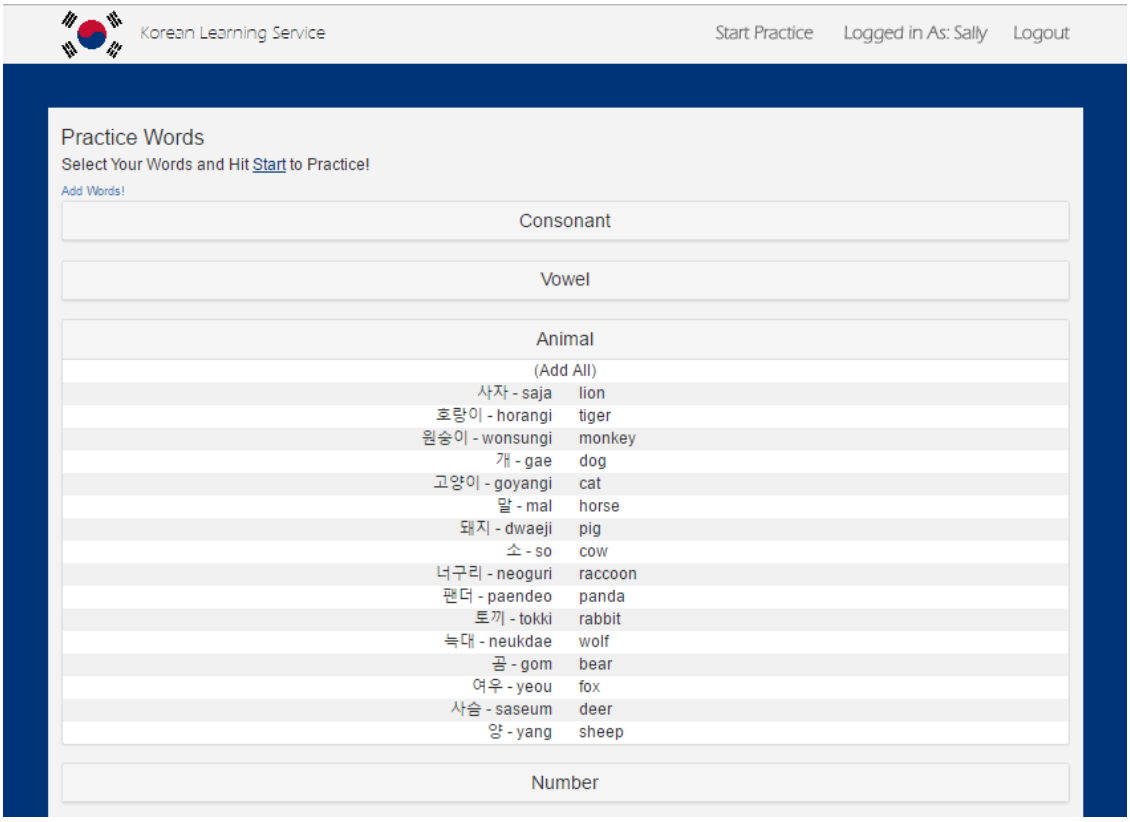
The Registration

You can't have Login without the Registration! It's pretty very similar to the Login Page.



The Meat and Bones. The Practice Section.

This page it jammed pack with useful words translated. Note the format is Korean Symbols – Pronunciation, English Translation. These words are populated using a GET call to our web service, which returns a JSON. This JSON is parsed and presented as below.

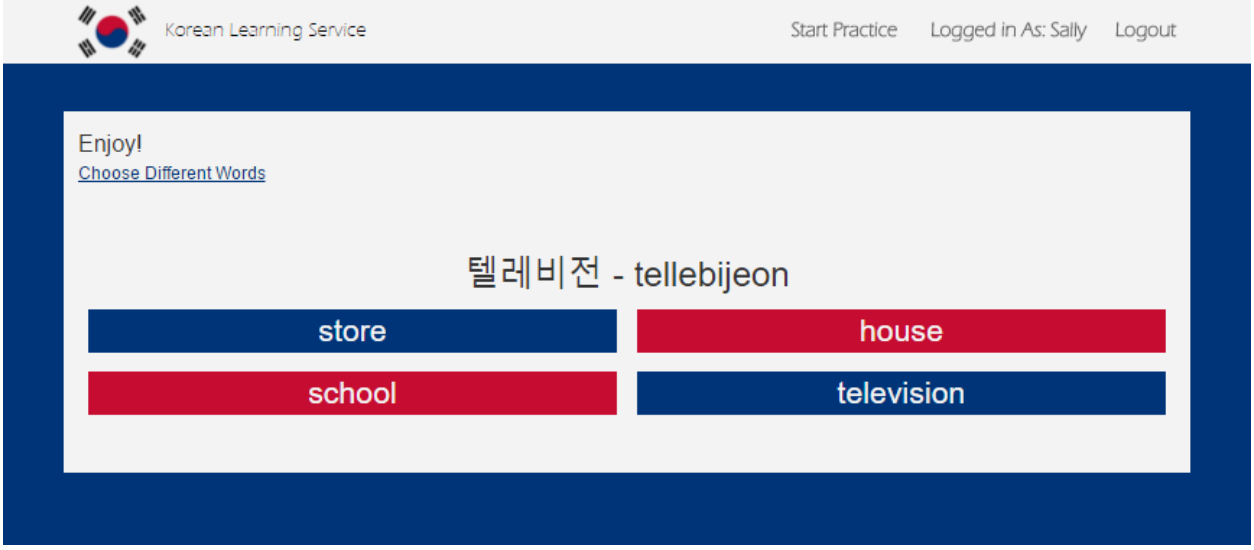


This is the Ajax GET call to our Restful service. Note the address were are targeting, "api.php/{user-id}. With this, we can rgab not only the default words, but the custom words belonging to the logged in user as well. The code below can be found in js/restful. The Restful service can be found in api.php (Will talk about later.

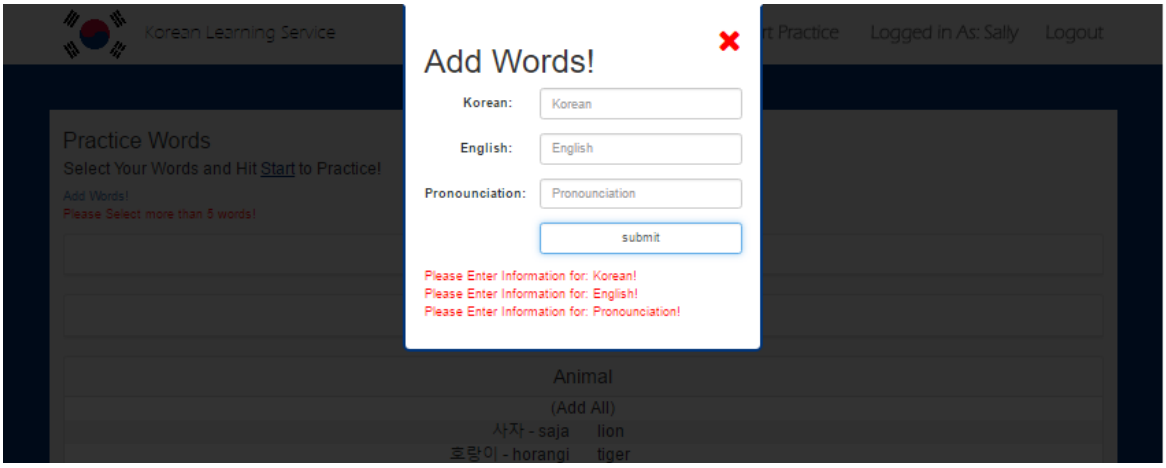
```
function getWords(){
    //Make our Rest GET Call to our Service, Get JSON back.
    $.get("api.php/"+$('#user-id').html(), function(data) {
        $('#words-container').html("");
        var words = jQuery.parseJSON(data);
        $.each(words, function(key,word) {

            //Each Word's Contents
            var category = word.category;
            var koreanWord = word.koreanWord;
            var englishWord = word.englishTranslation;
            var romanization = word.romanization;
            var wordID = word.id;

            //Look at Word's Type, if it doesn't exist, make a new
            if($('#' + category + '-container').length < 1){
                $('#words-container').append($('<div id="'+categor
                toggle="collapse" href="#'+category+'-collapse"><h
                id="'+category+'-collapse" class="panel-collapse c
                </ul></div></div></div>'));

            }

            var workingContainer = $('#' + category + '-container'
            if(category == "MyWords"){
                workingContainer.find('.words-group').append($('<l
                class="koreanCol"><em style="color:red;" class="fa
                '+koreanWord+' - '+romanization+'</span></div><div
                class="englishCol">'+englishWord+'</span></div></d
            }
            else{
                workingContainer.find('.words-group').append($('<l
                class="koreanCol">'+koreanWord+' - '+romanization+
                class="englishCol">'+englishWord+'</span></div></d
            }

        });
```

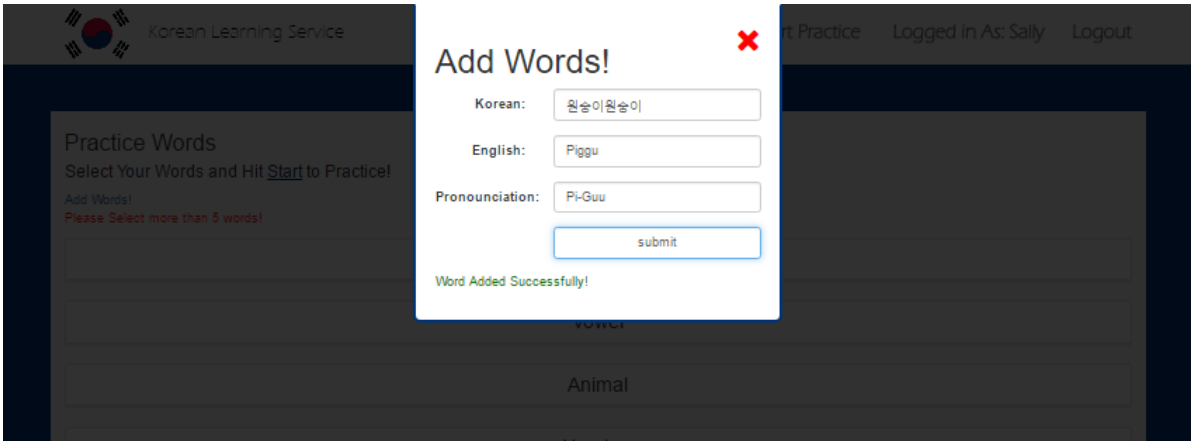
Users may select words to practice! These words will be tossed into a simple multiple choice game to strengthen the user's memorization and knowledge of the Korean Language. Note the minimum of 5 words. (Don't fret, there are plenty of default words built in for the default user).

Users may also go back to add new words and remove them without having to reselect all words. Code for Multiple Choice game can be found at js/gameManager.js.



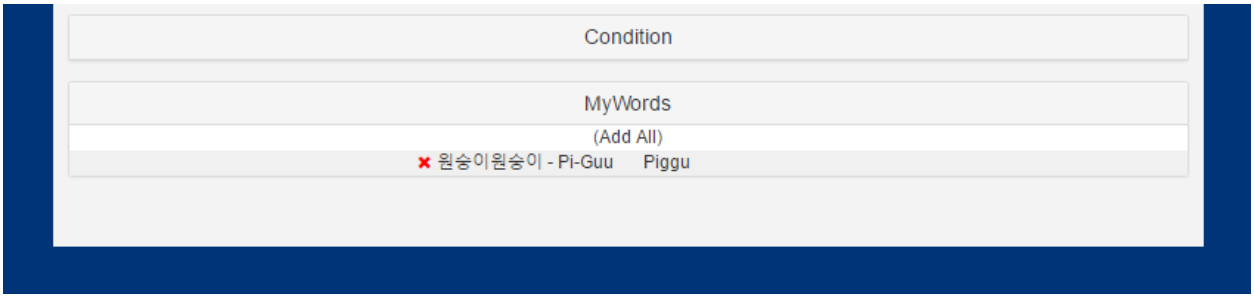
A neat feature users have is the ability to add their own custom words!!! Of course, these has a lot of error checking built into it using jQuery and PHP. On a successful submit, a POST call is made to our restful service, which will attempt to add the new word into our database!






```
if(isValid == true){
    //Make our Rest POST Call to our Service to store new word.
    $.ajax({
      type: "POST",
      url: "api.php/"+$('#user-id').html(),
      data: JSON.stringify ({username: $('#user-id').html(), wordType: "MyWords", category: "MyWords", koreanWord: $('#Korean').val(),
      englishTranslation: $('#English').val(), romanization: $('#Pronounciation').val()}),
      dataType: 'json',
      success: function(data, textStatus ){
          $('#errors-container').html("<p style='color:green;'>Word Added Successfully!</p>");
          getWords();
       },
       error: function(xhr, textStatus, errorThrown){
          $('#errors-container').html("<p style='color:red;'>Word Already Added, Try a different word!</p>");
       }
    });
}
return isValid;
```

Note the category added specifically for user's custom words. Also note users have the ability to remove the custom words that they have added!



This, once again, goes through our Restful API. We make a DELETE call for this. Also note, the url we make our call to is: "api.php/{user-id}/{id}". This where we can delete the user's specific word and ensure we do not affect other user's words who have similar deintitions!

```
//Handle Delete Word Events
$('.delete-word').click(function(){
    var id = $(this).attr('value');
    //Make our Rest DELETE Call to our Service to remove our word.
    $.ajax({
      type: "DELETE",
      url: "api.php/"+$('#user-id').html()+"/"+id,
      data: JSON.stringify ({username: $('#user-id').html(), wordType: "MyWords", category: "MyWords", koreanWord: $('#Korean').val(),
      englishTranslation: $('#English').val(), romanization: $('#Pronounciation').val()}),
      dataType: 'json',
      success: function(data, textStatus ){
          $('#start-errors-container').html("<p style='color:green;'>Word Deleted Successfully!</p>");
          getWords();
       },
       error: function(xhr, textStatus, errorThrown){
          $('#start-errors-container').html("<p style='color:red;'>Word could not be Deleted!</p>");
       }
    });
});
```

The Restful API.

When we get our call to our Restful service, we grab the kind of request it was (method). Then we grab the request and well as the input if any exist (i.e. the JSON we sent via POST when adding a new word to the database).

```
//Get the HTTP method, path and body of the request
$method = $_SERVER['REQUEST_METHOD'];
$request = explode('/', trim($_SERVER['PATH_INFO'],'/'));
$input = json_decode(file_get_contents('php://input'),true);

//Connect to DB
include("secure/database.php");
$dbconn = connectDB();
mysql_select_db('wordTables');

//Sample Path would be: http://localhost/api.php/{$user}
//Sample Path would be: http://localhost/api.php/{$user}/${wordID}
//Retrieve the user from the path
$user= $request[0];

if(sizeof($request) >= 2){
    $wordID = $request[1];
}
```

Depending on the kind of call will depend on the SQL that is generated.

```
//Create SQL based on HTTP method
switch ($method) {
  case 'GET':
    $sql = "SELECT * FROM words WHERE (username='defaultUser' OR userName='$user')"; break;

  case 'POST':
    if($input !=null){
        $username = $input['username'];
        $wordType = $input['wordType'];
        $category = $input['category'];
        $koreanWord = $input['koreanWord'];
        $englishTranslation = $input['englishTranslation'];
        $romanization = $input['romanization'];
    }
    $sql = "INSERT INTO words(username, wordType, category, koreanWord, englishTranslation, romanization) VALUES('$username', '$wordType',
    '$category', '$koreanWord', '$englishTranslation', '$romanization');"; break;

  case 'DELETE':
    $sql = "DELETE FROM words WHERE (id='$wordID' AND username='$user')"; break;
}
```

This SQL is ran through a query and returns accordingly with what the user orders.

Database

A quick snapshot of the database. Pretty straightforward. This file can be found in secure/populateDatabase.sql.

```sql
DROP TABLE IF EXISTS userTables.authentication;
DROP TABLE IF EXISTS userTables.user_info;
DROP SCHEMA IF EXISTS userTables;

CREATE SCHEMA userTables;

CREATE TABLE userTables.user_info (
    username            VARCHAR(30) PRIMARY KEY,
    registration_date   TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    description         VARCHAR(500),
    is_admin            boolean DEFAULT false
);


CREATE TABLE userTables.authentication (
    username           VARCHAR(30) PRIMARY KEY,
    password_hash      CHAR(40) NOT NULL,
    salt               CHAR(40) NOT NULL,
    FOREIGN KEY (username) REFERENCES userTables.user_info(username)
);

DROP TABLE IF EXISTS wordTables.words;
DROP SCHEMA IF EXISTS wordTables;

CREATE SCHEMA wordTables;


CREATE TABLE wordTables.words(
    id                  SERIAL,
    username            VARCHAR(30) NOT NULL REFERENCES userTables.user_info,
    wordType            VARCHAR(30),
    category            VARCHAR(30),
    koreanWord          NVARCHAR (100),
    englishTranslation  VARCHAR(100),
    romanization        VARCHAR(100)
);
```