DESIGN AND IMPLEMENTATION OF A WORD MATCH GENERATOR

by

E. M. GERTIS

(Under the Direction of Dr. Daniel Liang)

ABSTRACT

Word Match Generator is an effective teaching tool for helping students develop new vocabulary. It enables instructors to create *word matching interactives*. A word matching interactive is a web-based exercise that enables students to drag and drop key terms to their corresponding descriptions. Our word matching interactives are accessible through any internet browser. Thus, providing students with the ability to practice their vocabulary from anywhere in the world.

INDEX WORDS: Online learning,Computer Science Education, Word Matching

DESIGN AND IMPLEMENTATION OF A WORD MATCH GENERATOR

by

E. M. GERTIS

B.S. Physics, University of North Carolina at Chapel Hill, 2017

A Thesis Submitted to the Graduate Faculty of Georgia Southern University in Partial

Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

DESIGN AND IMPLEMENTATION OF A WORD MATCH GENERATOR

by

E. M. GERTIS

Electronic Version Approved:
May 2022

DEDICATION

I dedicate my work to my professors at Georgia Southern University, nothing that I do is of my own fruition.

ACKNOWLEDGMENTS

I would like to sincerely thank Dr. Daniel Liang for his mentorship and support throughout this process. Dr. Florin, for his willingness to provide feedback on my work. Dr. Allen, for motivating me with positive reinforcement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

This chapter will cover the purpose of the research conducted for this thesis. It will concisely describe the research history, significance of the field, and the specific problem that this research will solve. It will also familiarize the focus of the problem and the contributions of this research.

1.1 Historical Background

A variety of concepts from biology to computer science can be taught through word matching games. Specifically, subjects that use a hierarchical structure of vocabulary can be taught with word matching exercises. Forma Curran introduced the concept of word matching games in 1994 when they showed vocabulary was an essential part of effective communication [3]. Further, Al Masri and Majeda Al Najar proved that word matching games were effective in several ways. They showed that games give engagement pleasure for students, thus supporting them and helping them memorize new words. Gains in conceptual knowledge were reported for interactive courses, regardless of whether the course was high school, college, or university level. The average post-test score in the experimental group was higher than the average post-test score in the controlled group [14]. Their studies revealed post test scores of 80.40 for their experimental group and 77.20 for the controlled group. The measured p-value of the post-test was less than the significance value of 0.05. Therefore, the null hypothesis was rejected and the alternative hypothesis was accepted.

One larger study by Hake examined student performance with interactive engagement and traditional lecture methods in introductory physics courses. In a comparison of 14 classes using traditional methods with 48 classes using interactive engagement the performance of interactive engagement and traditional lecture methods in introductory physics courses was measured [9]. A persuasive argument that resonates with the theoretical rationale for using classroom activities was supported by the positive effect of engagement from 6,500 students examined in Hake's study. However, the relationship between cause and effect cannot be completely isolated in this non-equivalent group design.

Masari and Najar designed an experiment that involved matching pairs of words, cards, or pictures. In their experiments students had to find a partner with the appropriate card or picture. For example, students shuffled 20 word cards, 10 word matches, in random order. Then each student was tasked with looking for a matching pair of words or pictures within a certain time until all of the cards had the right pair [14]. The reported significance value from the experiment was 0.023 which was less than the p-value of 0.05. An additional study *The effect of using word matching games on primary stage students achievement in English language vocabulary in Jordan*, showed that word matching exercises do not have an impact as far as gender is concerned, but they concluded that there were statistically significant differences in the post-test between the control and experimental groups. Essentially, the treatment had the same effect on male and female students. The experimental group managed to significantly improve their English vocabulary. However, the control group improvement was not statistically significant.

Nikmah showed that there was a statistically significant difference between pre-test and post-test groups after they participated in *make a match*. Their experiments involved a technique where students were split into two groups, A and B, each group received topic cards. After playing the game, students ended up having more discussions with teachers which lead to an overall improvement in their vocabulary. Students who participated in *make a match* received a mean score of 18.67 in their pre-test and 25.30 in their post-test. The p-value associated with *make a match* technique was less than 0.05 [15]. Therefore, the null hypothesis was rejected which supported that *make a match* was effective at helping students improve their communication skills.

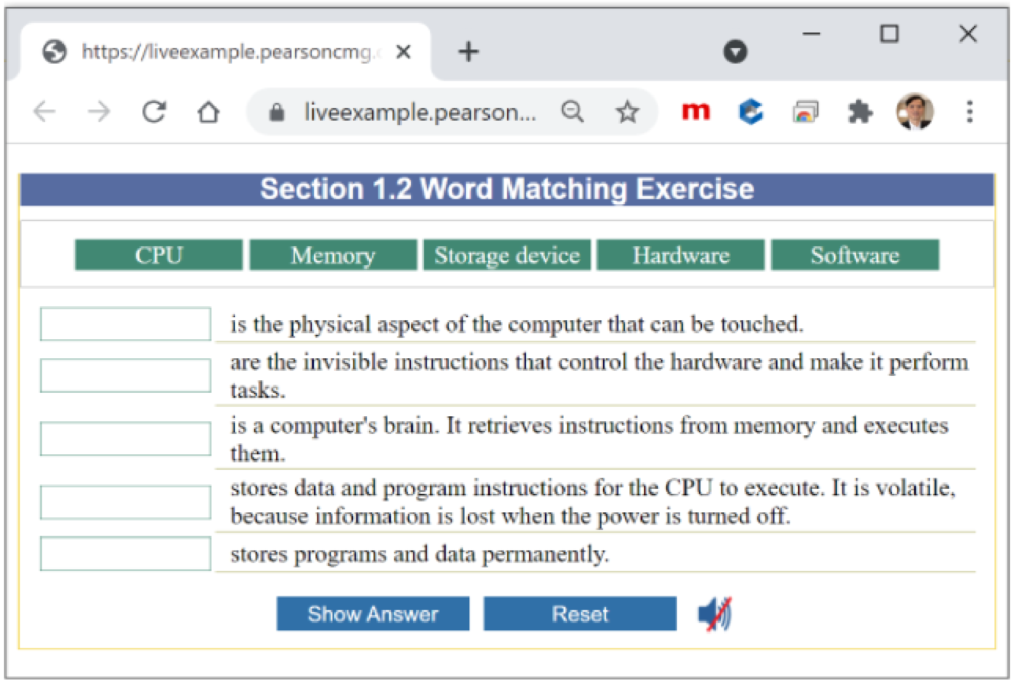
1.2 Problem Statement

There is a need to address a difficulty that computer science students tend to face when they are learning new terminology. In *Introduction to Java Programming and Data Structures* students first learn the key terms for speaking the language of computer science. For example, they are taught there is difference between the terms hardware and software. Knowing the difference between these terms is important for beginner computer science students. A computer includes both hardware and software, in general, hardware is the physical aspect of the computer that can be seen, and software is the invisible instructions that control the hardware and make it work. The hardware of a computer consists of a CPU, cache, memory, hard disk, floppy disk, monitor, printer, and communication devices [11].

In order for students to be successful they need to be able to understand what these differences. Vocabulary development is an essential part of the learning process. Our tool, Word Match Generator, can be used to help computer science students develop their academic vernacular through word matching exercises. Given a set of key terms and corresponding definitions, students can use word matching exercises to match each key term with its definition.

At this time education is shifting towards online platforms. Many in-person classes now feature a remote component. COVID-19 forced individuals into isolation. The findings from *Effects of COVID-19 in E-learning on Higher Education Institution Students: The Comparison Between Male and Female* showed that the consequences of the pandemic were unstoppable and uncontrollable for many industries of the world. Most High Education systems are now operating through E-learning platforms [4]. At least 120 countries have stopped face-to-face learning and approximately one billion students education were effected world wide with COVID-19.

Now there is a need for online resources that can help students learn autonomously. Previous researchers developed many successful methods for utilizing word matching games



Section 1.2 Word Matching Exercise

CPU | Memory | Storage device | Hardware | Software

	is the physical aspect of the computer that can be touched.
	are the invisible instructions that control the hardware and make it perform tasks.
	is a computer's brain. It retrieves instructions from memory and executes them.
	stores data and program instructions for the CPU to execute. It is volatile, because information is lost when the power is turned off.
	stores programs and data permanently.

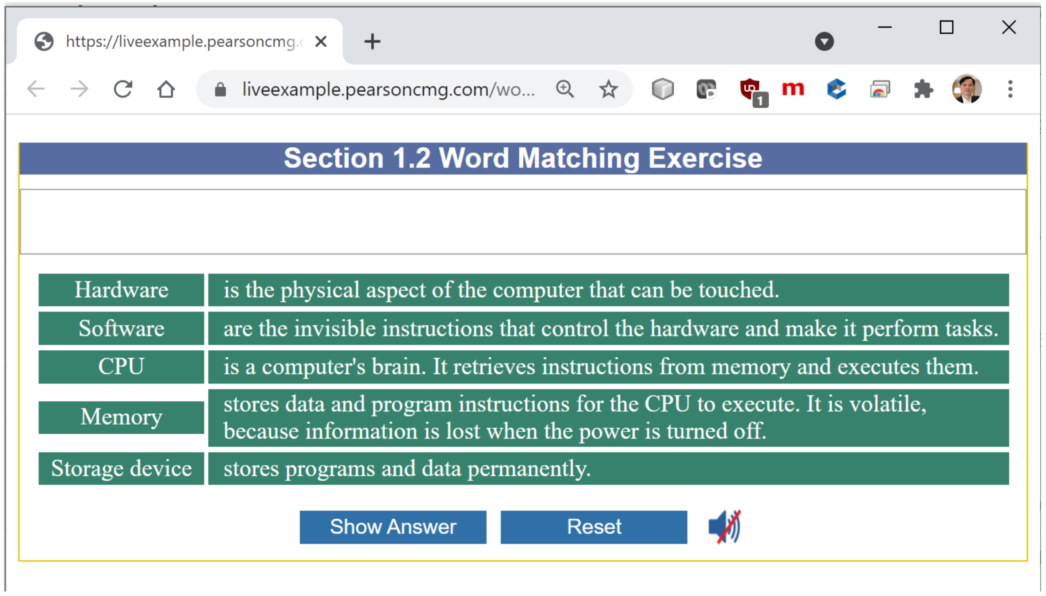
Figure 1.1: Word Match Generator Before Dragging Boxes

in the classroom. However, they did not develop a method for enabling students to learn vocabulary from anywhere in the world. Our tool provides students with an opportunity to improve their communication skills in computer science from anywhere where there is a stable internet connection.

Figure 1.2 shows an example of a word matching exercise, which can be viewed from `https://liveexample.pearsoncmg.com/wordmatch/Section1_2.html`.

Figure 1.2 shows the result after the user drags the key terms to match their descriptions. A congratulations dialog (see Figure 1.3) is displayed when all of the key terms are matched to their descriptions.

We have developed more than 60 word matching exercises. Each word matching exercise is an *interactive*. The *interactives* are embedded within interactive ebooks [11], [12], [10]. These interactives have received good reviews [7], [6]. They help students learn and grasp key terms. Previously, each of the word matching interactives were programmed

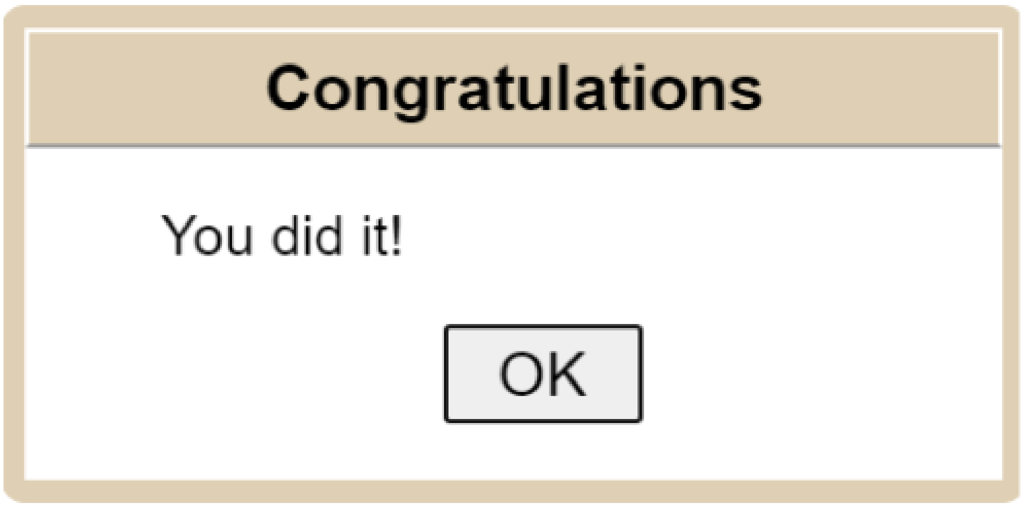


https://liveexample.pearsoncmg.c
liveexample.pearsoncmg.com/wo...

Section 1.2 Word Matching Exercise

Hardware	is the physical aspect of the computer that can be touched.
Software	are the invisible instructions that control the hardware and make it perform tasks.
CPU	is a computer's brain. It retrieves instructions from memory and executes them.
Memory	stores data and program instructions for the CPU to execute. It is volatile, because information is lost when the power is turned off.
Storage device	stores programs and data permanently.

Figure 1.2: Word Match After Dragging Boxes



Congratulations

You did it!

Figure 1.3: Congratulations Dialog Box

manually. The effort required to develop Word Match Generator involved advanced programming skills and a significant time investment. We have now empowered instructors with the ability to create word matching exercises through a simple web based tool. Using our tool instructors can enter key terms and their descriptions. Then they can automatically generate the HTML code for a word matching interactives. Thus, saving them precious time and resources. Our system can be utilized by instructors to create exercises from any academic discipline. These exercises will help students learn the language of their academic discipline. In the subsequent sections we will describe the design and implementation of Word Match Generator, it's usage in the field, and the results from our research.

CHAPTER 2

LITERATURE REVIEW

2.1 Improved Reading Skills by Students at PPEP TEC High School who used Fast ForWord® Products

Computer software products that focus on learning and cognitive skills, and provide an optimal learning environment have been proven by university based research studies to help build memory, attention, processing, and sequencing skills which are critical for success. FastFor products used progress tracker reports to study how phonemic awareness and the acoustic properties of speech impacted the rapid development of language and reading skills.

The games used in FastFor products vary in their nature. In *Matches and Bug Out!/Laser Match* students chose a square on a grid and hear a sound or a word. The goal is to find each square's match and clear the grid. Students who participated in *Bear Bags and Bear Bags: More Lunch* developed an understanding of alphabetic principles (phonics) by helping "Momma Bear" sort words (on pieces of toast) into phoneme-based categories. A squirrel mail carrier pulls words out of a mailbag and participants sort them into different categories by clicking on the appropriate mailbox in *Quail Mail*. Students develop their understanding of words meanings, auditory recognition of phonemes, and sound processing by clicking on pictures that matches words that they hear in *Cards*. In *Ant Antics* participants improved their vocabulary by picking one of the four alternatives that represented a picture. As students played *Canine Crew* they developed their vocabulary, decoding, and automatic word recognition by matching pairs of words together on the basis of a criteria in a grid. Finally, in *Twisted Pictures* students built their sentence comprehension by developing syntax, working memory, logical reasoning, and vocabulary by selecting sentences that more accurately described a picture based on a set of alternatives.

Students who used FastFor products noticed a significant improvement in their reading comprehension skills. Their improvement was measured by the Brigance Comprehensive inventory of Basic Skills. The results showed that students gained three and one-half years in reading grade level. Their average letter-word identification improved by 14 months and their average gain on the passage Comprehension subtest of the Woodcock-Johnson III Tests of Achievement was 2 years.

FastFor products took a previously successful concept, "computer software that focuses on developing learning and cognitive skills" and improved up on it. They added 7 word matching games to their software. Their contribution to lead to the overall improvement of reading comprehension skills of the participants. They noted that the longer the students used the product the more their skill level improved.

Word Match Generator provides instructors with the capability to create *word matching interactives* which are analogous to *Canine Crew*, *Matches and Bug Out!/Laser Match*, *Bear Bags and Bear Bags: More Lunch*, *Quail Mail*, *Cards*, *Ant Antics*, *Canine Crew*, and *Twisted Pictures* in that they get students to take a set of words and match them to another set of words. Fundamentally, word matching games help students learn the relationship between words and their meanings.

Word matching games have had success not just for improving vocabulary, but also for improving reading comprehension. For example, Canine Crew, had a measurable impact on students reading comprehension. The effects f student participation in this game as measured by the Brigance Comprehensive Inventory of Basic Skills are shown in Figure 2.1.

Games like Canine Crew use word matching to train vocabulary, decoding, and automatic word recognition. Merzenich and Tallal showed that an optimal learning environment, a focus on reading, cognitive skills, resulted in dramatic improvements in auditory processing, and language skills of school children who were experiencing academic read-

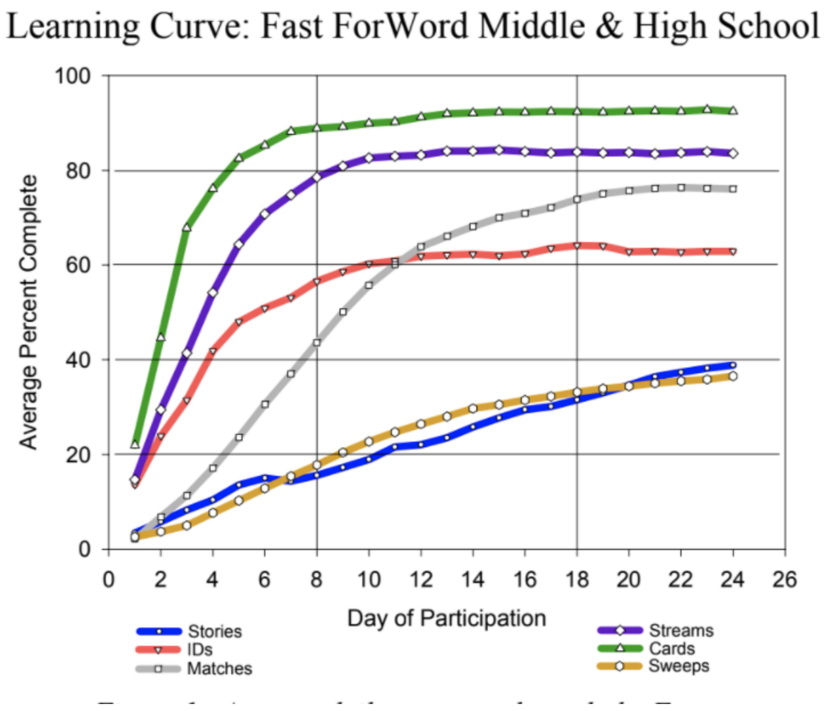


Figure 2.1: Learning Curve: Fast ForWord Middle & High School

ing failure. [2]

Word matching games increase student participation by providing them with an opportunity to get away from traditional learning activities. In Canine Crew they were forced to change the way that they saw relationships between the words and their meanings. The daily practice of matching different pairs of words on a grid had a measurable effect on their reading comprehension skills.

2.2 Effectiveness Of Make a Match in Teaching Vocabulary

Language is an essential component of learning vocabulary. As stated in the introduction, games give engagement pleasure for students. They can provide them with an outlet. Especially, for students who struggle learning new vocabulary. Games help them learn new words without having to spend time reading vast amounts of literature. Nikmah showed that games that involve matching pairs of words, cards, or pictures can be used to teach a variety of academic subjects. The statistical significance of Husein and Nikamah's work supports the hypothesis that students who participate in word matching games have a measurable effect on improving student vocabulary. The disadvantage of their methods is that it they require the guidance of teachers [15].

They identified that vocabulary is closely related to the four language skills of reading, writing, listening, and speaking. They employed a method "*Make a Match*" which encourages students to practice their vocabulary by dealing with cards each other to explain the meaning of words. The difference between in *Make a Match* and the work of Curran is that students use pictures in a addition to words in their word matching games. In *Make a Match* students are split into two groups. Then they get together with partners and try to match words with corresponding pictures. Arifah and Kusumarasdyati defined *Make a Match* as one of the co-operative learning techniques that is used with pairs. The disadvantages of *Make a Match* is that it requires guidance from the teachers, time restrictions, and the organization groups of students. However, the advantages of *Make a Match* is that it encourages students to cooperate, helps them avoid boredom, leads to daily participation and a more interesting classroom dynamic.

2.3 The Effect of Using Word Matching Games on Primary Stage Students Achievement in English Language Vocabulary in Jordan

The role of success of games in student development cannot be denied. Games bring relaxation and help students learn new words. However, they require lots of effort on behalf of the instructor. Masari and Najar designed a game that has been statistically proven to help students their English vocabulary. In their experiments students used pairs of cards and words to match colors, shapes, numbers, and word definitions. In each case the experimental group subjects managed to significantly improve their English vocabulary meanwhile the control group did not. What they showed is that games promote knowledge transfer. This is most likely because they require student participation and active involvement with the material.

Over the last decade books for teachers and students have focused on ways of organizing, practicing, and processing new vocabulary to accessible and memorable to students. It is well known that vocabulary is a major problem for students, Al Masari and Majed Al Najar decided to investigate the effect of using word matching games as a strategy on the achievement of primary stage male and female student learning English as a foreign language in Amman, Jordan. They applied an analytical method for measuring the effectiveness of word matching exercises. Few studies have actually measured the impact of games on student learning. Masari and Najar study that utilized 158 students showed that there was a statistically significant difference between the control group and the experimental group.

Word Match Generator follows a similar strategy as prescribed by Masari and Najar in that they both utilize word matching strategies to help improve students vocabulary and reading comprehension. The difference is that Word Match Generator does not require a physical presence for student participation. It can be accessed from anywhere in the world.

2.4 Effects of COVID-19 in E-learning on higher education institution students: the group comparison between male and female

The findings of the study reveal that males and females have a different level of usage towards E-learning portals in Malaysian Universities. In previous literature, the DM model is tested on the overall population, like banks and other financial sectors. In this study, the whole population is divided into males and female categories to hope for different theoretical contributions by having a division of the population. In a DM model (DeLone and McLean) developed in 1992 the success of an information system was primarily based on the following three aspects: technical (the accuracy of the information system), semantic (success of the right information conveyed to the receiver), and effectiveness (influence of the information on the receiver).

CHAPTER 3
METHODOLOGY

3.1 DESIGN AND IMPLEMENTATION

The specific problem that our research aims to address is the development of vocabulary through word matching interactives. From a technical perspective one of the most optimal ways to approach this problem was to develop an online teaching tool. Currently, E-learning is enhancing students' knowledge, even the academic staff and professional and industry people skills through the internet [1]. Most higher education universities are providing online courses for their students [16]. Among the many challenges that schools have faced in this transition to online learning is accessibility for students with disabilities and/or their parents or caregivers with disabilities [5]. The unique difference between our research and that of previous researchers is that our tool can be accessed from anywhere in the world. Its specific purpose was to help students develop their vocabulary for a particular academic topic. We have designed an intuitive user interface that any instructor can use to generate a word matching interactive based on a key terms and descriptions. By default, two entries for key terms and their descriptions are displayed in Word Match Generator as shown in Figure 3.1. An instructor can click the Add More button to display more entries for creating additional key terms and descriptions.

The design of Word Match Generator was based on web MVC (Model View Controller) architecture. Our MVC application utilizes HTML5, CSS, JavaScript, and JavaServer Pages supported by the Spring Boot framework. In this type of web based architecture a client submits a requests to a server which then processes the request and saves the content. Instructors can rapidly create word matching exercises without having to wait for their requests too be fully processed. An XMLHttpRequest was designed to send a payload consisting of static HTML code to be saved on the server. This type of request provides our

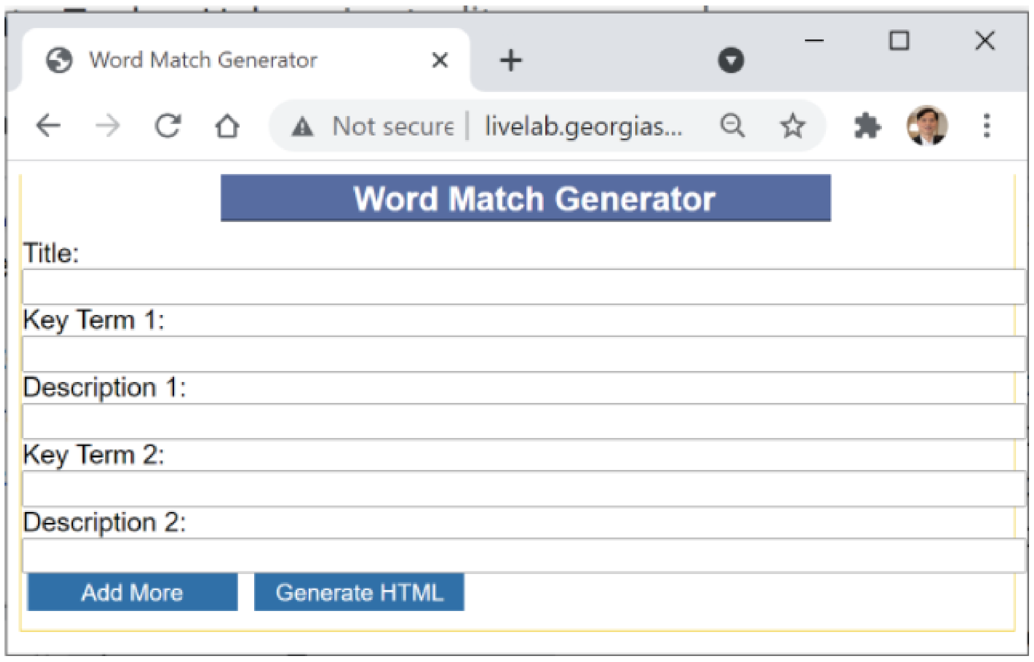


Figure 3.1: Initial Screen for Word Match Generator

application with the asynchronicity needed to enable instructors to create word matching exercises without delay. The specific HTTP method used for this implementation was a POST request. The reasoning behind choosing a POST request was because they are typically used for interactions where data is stored. It is a method that requests that a web server accepts the data enclosed in the body of the request message for storing it. This is shown in Figure 3.3.

The application that we have designed can be deployed on any web server that supports a Java virtual machine. It was designed to be completely self contained. All of the file storage and generation takes place on a single server, as shown in Figure 3.4. The source code for the application is located in the appendix. The purpose of WordMatchController.java is to process the incoming requests from the client. The methods within the Controller class then use the methods from WordMatchService.java to save the content to the server. Once the content is saved a new wordmatch.jsp file is created content stored in the HTTP POST request. The service methods utilize the model structure from View.java. The Controller to service interaction is captured in Figure 3.4. The rendered view of wordmatch.jsp is shown in Figure 3.8.

One of the first obstacles in creating Word Match Generator was to design a method for generating the static HTML code. The HTML generating function that we implemented uses a JavaScript method. The source code for the method is shown in the appendix under wordmatch.js. The primary function of this method is to retrieve content from HTML inputs, key terms, and descriptions. Then concatenate them into a string. The key terms and descriptions are then randomly shuffled before they are concatenated. Then they are processed into string which represents a word matching interactive. After this process is completed the entire HTML string is then displayed in text area box as shown in Figure 3.6. At this point an instructor can post the HTML code to an external web server for students to access. When the HTML page for the exercise is displayed, the descriptions appear in a random order as shown in Figure 3.8.

The application that we have designed can be deployed on any web server that supports a Java virtual machine. It was designed to be completely self contained. All of the file storage and generation takes place on a single server.

3.1.1 The Use of Word Match Generator

An example of word match generator can be seen at `http://livelab.georgiasouthern.edu/wordmatchgenerator`http://livelab.georgiasouthern.edu/wordmatchgenerator as shown in Figure 3.1.

The functionality of our application can be described in a simple sequence of steps. In step one an instructor can simply enter a title, Key Term 1, Description for Key Term 1, Key Term 2, and Description for Key Term 2. In step 2, they can click the Add More button to create more entries for key terms and their descriptions. For example, an instructor

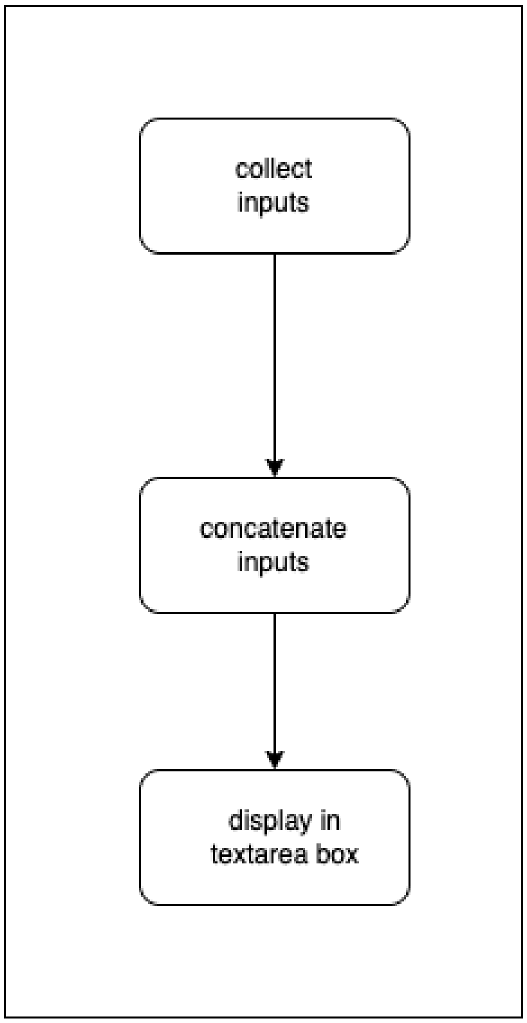


Figure 3.2: Flow Diagram for Generating HTML

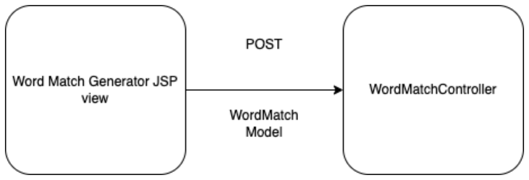


Figure 3.3: Model View Controller Interaction for Word Match Generator

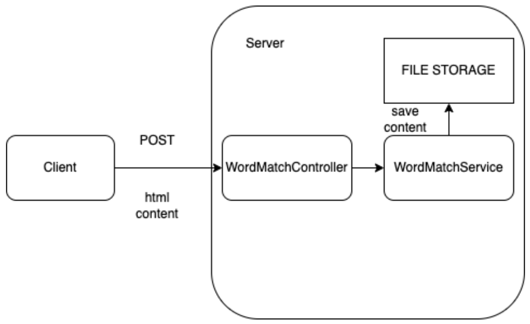
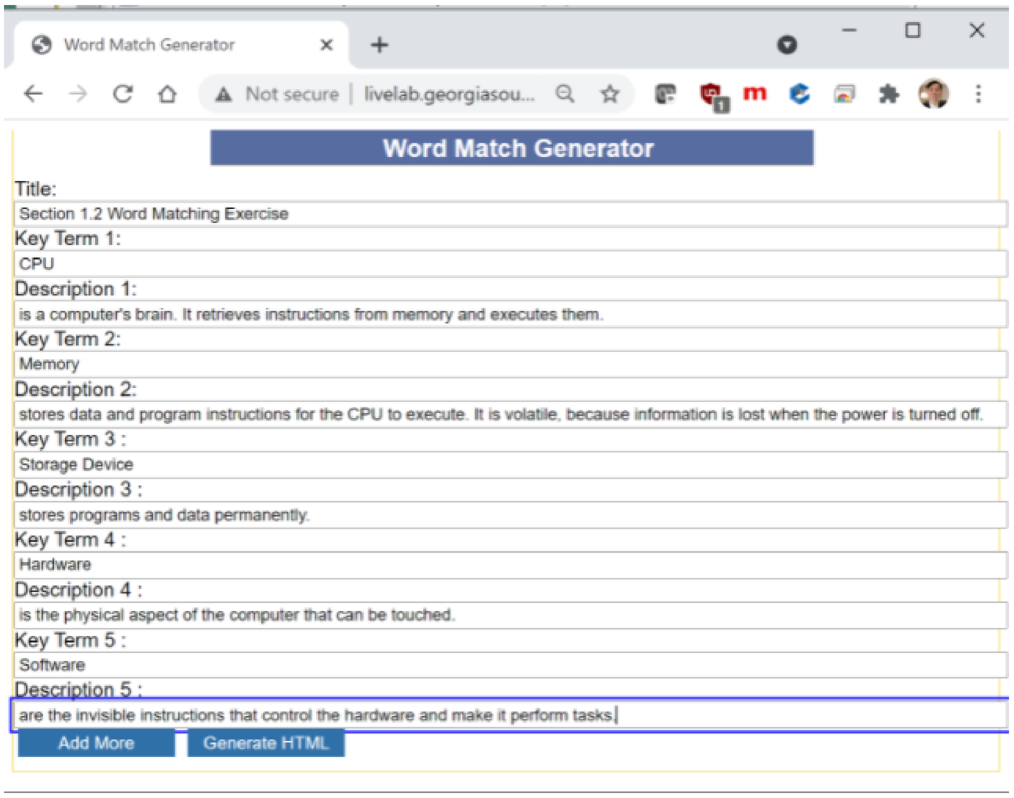


Figure 3.4: Technical Diagram For Word Match Generator

can enter the following entries inputs and descriptions shown in Figure 3.8. Word Match Generator then displays the HTML code as shown in Figure 3.6. An instructor can click the Post button to send generated HTML code for the word matching interactive to server as shown in 3.7.

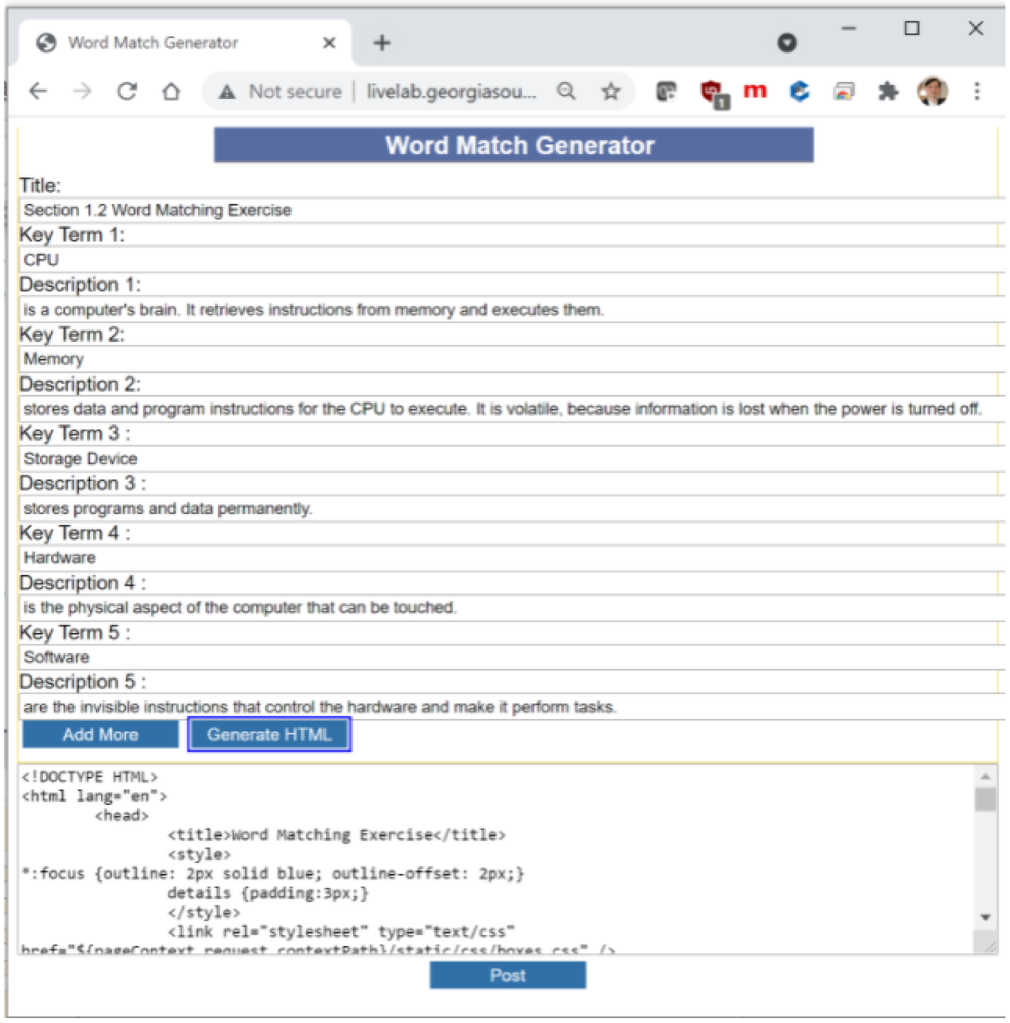
After clicking the Post button to post the word match interactive to the server. The server saves the generated HTML file for the word matching interactive. It then creates a URL for the exercise. After the generated HTML file is posted, a View button is displayed, as shown in Figure 3.7. An instructor can access any of the exercises by clicking the View button. The View button serves two purposes. First, it renders the HTML code for the exercise. Second, it shows the URL for the exercise on the server. Clicking the View button displays the exercise using the URL, as shown in Figure 3.8. The instructor can give this URL to the student. An example of the created word matching interactive is shown in Figure 3.8.



Word Match Generator

Title:	Section 1.2 Word Matching Exercise
Key Term 1:	CPU
Description 1:	is a computer's brain. It retrieves instructions from memory and executes them.
Key Term 2:	Memory
Description 2:	stores data and program instructions for the CPU to execute. It is volatile, because information is lost when the power is turned off.
Key Term 3 :	Storage Device
Description 3 :	stores programs and data permanently.
Key Term 4 :	Hardware
Description 4 :	is the physical aspect of the computer that can be touched.
Key Term 5 :	Software
Description 5 :	are the invisible instructions that control the hardware and make it perform tasks.

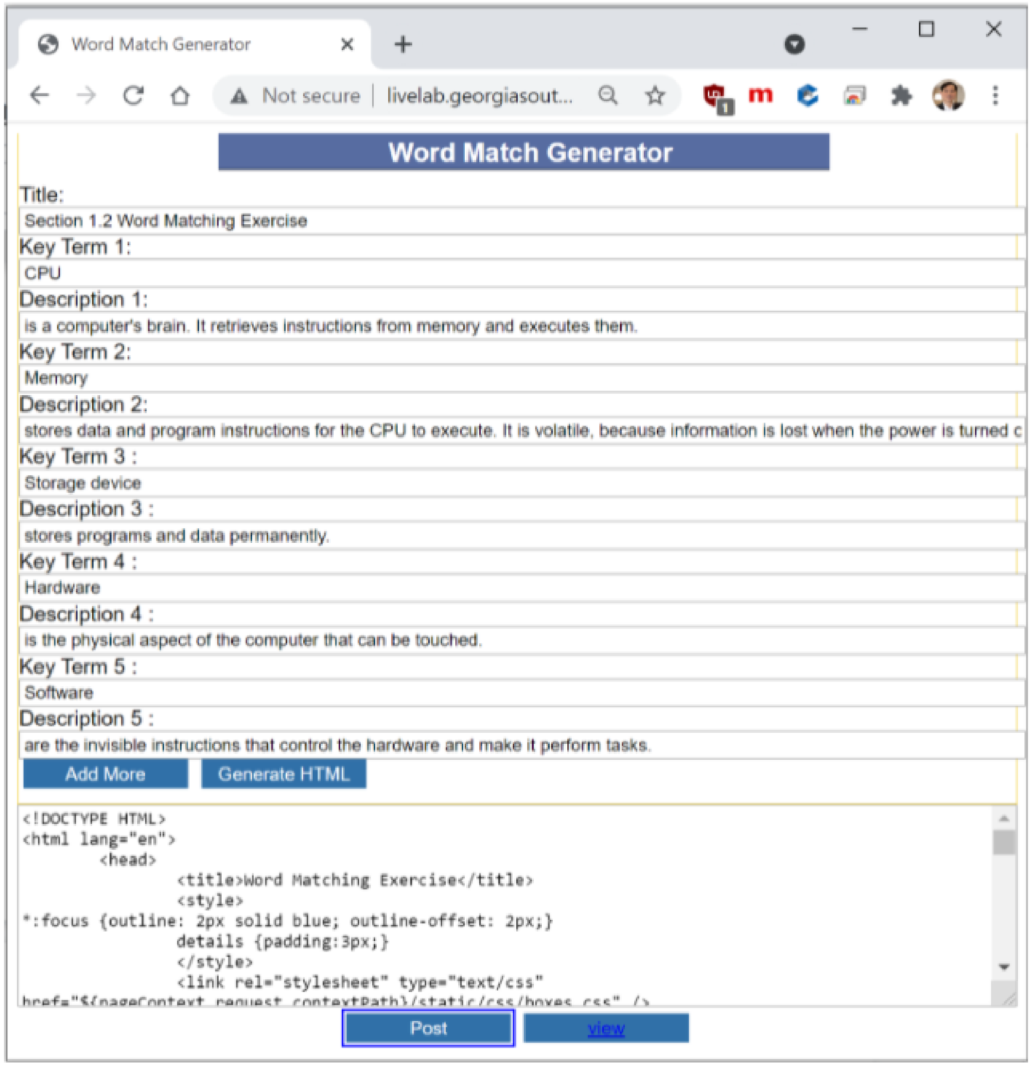
Figure 3.5: Key Terms and Description Inputs



Title:
Section 1.2 Word Matching Exercise
Key Term 1:
CPU
Description 1:
is a computer's brain. It retrieves instructions from memory and executes them.
Key Term 2:
Memory
Description 2:
stores data and program instructions for the CPU to execute. It is volatile, because information is lost when the power is turned off.
Key Term 3 :
Storage Device
Description 3 :
stores programs and data permanently.
Key Term 4 :
Hardware
Description 4 :
is the physical aspect of the computer that can be touched.
Key Term 5 :
Software
Description 5 :
are the invisible instructions that control the hardware and make it perform tasks.

```
<!DOCTYPE HTML>
<html lang="en">
        <head>
                <title>Word Matching Exercise</title>
                <style>
*:focus {outline: 2px solid blue; outline-offset: 2px;}
                details {padding:3px;}
                </style>
                <link rel="stylesheet" type="text/css"
```

Figure 3.6: Generated HTML Code

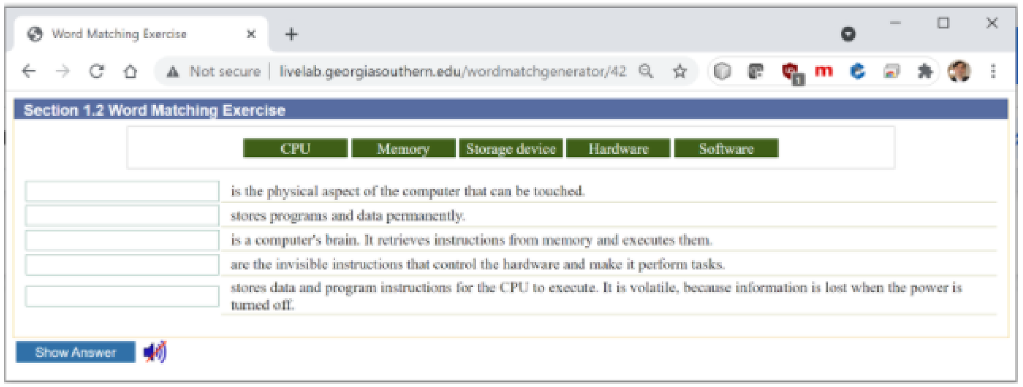


Word Match Generator

Title:
Section 1.2 Word Matching Exercise
Key Term 1:
CPU
Description 1:
is a computer's brain. It retrieves instructions from memory and executes them.
Key Term 2:
Memory
Description 2:
stores data and program instructions for the CPU to execute. It is volatile, because information is lost when the power is turned c
Key Term 3 :
Storage device
Description 3 :
stores programs and data permanently.
Key Term 4 :
Hardware
Description 4 :
is the physical aspect of the computer that can be touched.
Key Term 5 :
Software
Description 5 :
are the invisible instructions that control the hardware and make it perform tasks.

```
<!DOCTYPE HTML>
<html lang="en">
        <head>
                <title>Word Matching Exercise</title>
                <style>
*:focus {outline: 2px solid blue; outline-offset: 2px;}
                details {padding:3px;}
                </style>
                <link rel="stylesheet" type="text/css"
href="${pageContext.request.contextPath}/static/css/boxes.css" />
```

Figure 3.7: Clicking Post Button


Word Matching Exercise
Not secure | livelab.georgiasouthern.edu/wordmatchgenerator/42

Section 1.2 Word Matching Exercise

CPU Memory Storage device Hardware Software

is the physical aspect of the computer that can be touched.
stores programs and data permanently.
is a computer's brain. It retrieves instructions from memory and executes them.
are the invisible instructions that control the hardware and make it perform tasks.
stores data and program instructions for the CPU to execute. It is volatile, because information is lost when the power is turned off.

Show Answer

Figure 3.8: Rendered HTML Code

CHAPTER 4
RELATED WORKS

Al Masair and Majeda showed that word matching games gave students pleasure and enjoyment. Their hypothesis supports the argument that games provide students with an interactive experience that can help them learn new words. Nikmah and Husein's work supports the justification that word matching games are effective at helping students improve their communication skills. Introductory text books such as *Introduction to Java Programming and Data Structures* emphasize the importance of learning key terms at the beginning of the learning cycle. In introduction of [11], [12], and [10] key terms and definitions are presented to the student.

The importance of learning key terms and definitions at the beginning of the text is critical for future understanding of subsequent chapters. This is supported by the results from FastFor products which showed that the consistent exposure to word matching games over an extended period of time improved student reading comprehension skill. [14] and [15] showed that post-test vocabulary retention was higher than pre-test groups. Further, [3], [8], [13], [14], [15] showed that the measured p-value was less than the specified significance value. Therefore, the null hypothesis for these studies was rejected which implies that each of the hypotheses presented in studies was accepted. These results from these works support the argument that students who participated in word matching exercises ended up improving their retention of new words.

CHAPTER 5

RESULTS

Word Match Generator removes the pain that instructors typically facce when they have to create word matching games. Before implementing Word Match Generator all of the word matching interactives had to be developed manually. The process of manually creating these exercises was a large waste of time for instructors. To create each exercise an individual file with specific content had to be created, updated and maintained. Word Match Generator removes the pain of this manual process.

Our tool provides instructors with an ability to create a word matching interactives without writing any code. The first iteration of our tool required instructors to at least copy and paste their code from the text area on to the server. The manual effort required to copy and paste resulted in poor adoption so we added a Post button to save the generated HTML code. Once the content is saved onto the server create a URL is created for the instructor to access the exercise directly without any extra work. In retrospect, we should have created this tool earlier to save hundreds of hours of writing word matching interactives manually.

CHAPTER 6

CONCLUSION

Our tool addresses the need for computer science instructors be able to teach new vocabulary from anywhere in the world. Its primary purpose serves to help students improve their computer science vocabulary through word matching interactives. The contribution from our research is a web-based tool for automatically generating a word matching interactives. Now instructors can enter their terms and descriptions to generate a word matching interactive and share the URL with students. The tool is freely available from `http://livelab.georgiasouthern.edu/wordmatchgenerator`. It enables the instructors to guide the students in need, consequently leading to a better learning experience. The effectiveness of our application is supported by the research shown in [3], [8], [13], [14], [15].

REFERENCES

[1] Donnie Adams, Bambang Sumintono, Ahmed Mohamed, and Nur Syafika Mohamad Noor. E-learning readiness among students of diverse backgrounds in a leading malaysian higher education institution. *Malaysian Journal of Learning and Instruction*, 15(2):227–256, 2018.

[2] MELISSA M Agocs, MARTHA S Burns, LOGAN E De Ley, STEVEN L Miller, and BARBARA M Calhoun. Fast forword language. *Treatment of language disorders in children*, pages 471–508, 2006.

[3] M. Arifah and Kusumarasdyati. The effectiveness of make a match technique for teaching writing descriptive text to the seventh graders of smpn 1 karang binangun lamongan. *UNESA*, 1(1):1–8, 2013.

[4] Kaliope Azzi-Huck and Tigran Shmis. Managing the impact of covid-19 on education systems around the world: How countries are preparing, coping, and planning for recovery. *World Bank Blogs*, 18, 2020.

[5] Joanne L Badge, Emma Dawson, Alan J Cann, and Jon Scott. Assessing the accessibility of online learning. *Innovations in Education and Teaching International*, 45(2):103–113, 2008.

[6] Candace Cooney. Revel educator study assesses quiz, exam, and final course grades at central michigan university. *http://www.pearsoned.com/results/revel-educator-study-assesses-quiz-exam-final-course-grades-central-michigan-university*, Fall 2015.

[7] Candace Cooney. Revel™ educator study observes homework and exam grades at university of louisiana. *http://www.pearsoned.com/results/revel-educator-study-observes-homework-exam-grades-university-louisiana/*, Spring 2016.

[8] M. Dewi. The impact of the application of a match technique towards students' vocabulary mastery. 2014.

[9] Richard R Hake. Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. *American journal of Physics*, 66(1):64–74, 1998.

[10] Yong Daniel Liang. Revel™ for introduction to python programming and data structures 2e. 2018.

[11] Yong Daniel Liang. Revel™ for introduction to java programming and data structures 12e. 2020.

[12] Yong Daniel Liang. Revel™ for introduction to c++ programming and data structures 5e. 2021.

[13] Sondang Manik and May Christiani. Teaching vocabulary using matching word on computer assisted language learning. *International Journal of English Language Teaching*, 4(7):1–26, 2016.

[14] A. A. Masri and M. A. Najar. The effectivness of using word games on primary stage students achievement in english language vocabulary in jordan. *American International Journal of Contemporary Research*, 4(9):22, 2014.

[15] Busmin Gurning Ria Dhatun Nikmah and Rahmad Husein. The effectiveness of make a match technique in teaching vocabulary. *ACM Transactions on Computing Education*, 10(3):22, 2010.

[16] Arfan Shahzad, Rohail Hassan, Adejare Yusuff Aremu, Arsalan Hussain, and Rab Nawaz Lodhi. Effects of covid-19 in e-learning on higher education institution students: the group comparison between male and female. *Quality & quantity*, 55(3):805–826, 2021.

Appendix A
FIRST APPENDIX

First appendix introduction..

A.1 APP

```
package com.company.app;


import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;


@SpringBootApplication

public class Application {


    public static void main(String[] args) {

        SpringApplication.run(Application.class, args);

    }

}

package com.frugalis.Spring.Boot.Resources;


import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.

    ResourceHandlerRegistry;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;


@Configuration

public class ResourceConfigs implements WebMvcConfigurer

{
    private static final String[] CLASSPATH_RESOURCE_LOCATIONS =
    {
        "classpath:/htmlFiles/",
        "classpath:/static/",
        "classpath:/static/images"
    };

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry)
    {
        registry.addResourceHandler("/**")
                .addResourceLocations(CLASSPATH_RESOURCE_LOCATIONS)
                .setCachePeriod(3000);
    }
}
```

A.2 SERVICE

```
package com.company.app.service;

/*
 * WordMatchService.java
 * Author: Evan Gertis
 */
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.company.app.model.WordMatch;
import java.io.File;
import java.io.IOException; // Import the IOException class to handle
   errors
import java.io.FileWriter; // Import the FileWriter class
import java.io.IOException; // Import the IOException class to handle
   errors
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.stream.Stream;
import java.nio.file.Files;
import java.util.Base64;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;


@Service
public class WordMatchService {
        private static final Logger logger = LogManager.getLogger(
           WordMatchService.class);

        @Autowired
        WordMatchService(){
        }

        public Long saveJSP(WordMatch wordMatch){
                logger.info(wordMatch);
```

```
Long numberOfFiles = (long) 0;
try {
       File file = new File("./src/main/webapp/view/
          word_match0.jsp");
       logger.info("Decoding String");
       String cleanedHTML = wordMatch.toString().replace("
          WordMatch(content=","").replace(")","");
       logger.info(cleanedHTML);
       byte[] decodedBytes = Base64.getDecoder().decode(
          cleanedHTML.getBytes());
       String html = new String(decodedBytes, "UTF-8");
       logger.info(html);
       if (file.createNewFile()) {
              System.out.println("File created: " + file.
                 getName());
              try {
                     FileWriter myWriter = new FileWriter
                        ("./src/main/webapp/view/
                        word_match0.jsp");
                     myWriter.write(html);
                     myWriter.close();
              } catch (IOException e) {
                     System.out.println("An error occurred
                        .");
                     e.printStackTrace();
              }
       } else {
              String fileName = file.getName().toString();
String index = fileName.substring(fileName.
   indexOf("h") + 1);
index = index.substring(0, index.indexOf("."))
   ;
Integer parsedInt = Integer.parseInt(index);
System.out.println(parsedInt);
Stream<Path> files = Files.list(Paths.get("./
   src/main/webapp/view/"));
numberOfFiles = files.map(Path.class::cast)
                        .filter(path -> path.
                           getFileName().
                           toString().
                           startsWith("
                           word_match"))
                        .count();
fileName = fileName.replace(index,
   numberOfFiles.toString());
System.out.println(numberOfFiles);
System.out.println("fileName should have been
   printed by now");
file = new File(fileName);
String JSPfileName = "./src/main/webapp/view
   /"+file;
FileWriter myWriter = new FileWriter(
   JSPfileName);
myWriter.write(html);
myWriter.close();
// Write JSP file to HTML
// File path is passed as parameter
File jspFile = new File(JSPfileName);

// Note: Double backquote is to avoid compiler
// interpret words
// like \test as \t (ie. as a escape sequence)

// Creating an object of BufferedReader class
BufferedReader br
= new BufferedReader(new FileReader(jspFile));

// Declaring a string variable
String st;
// Condition holds true till
// there is character in a string
String htmlFileName = JSPfileName.replace("jsp
   ","html");
File htmlFile = new File(htmlFileName);
String content = "Writing To File";
if (!htmlFile.exists()) {
       htmlFile.createNewFile();
}
try {
       FileWriter fw = new FileWriter(htmlFile
          .getAbsoluteFile());
       BufferedWriter bw = new BufferedWriter(
          fw);
       while ((st = br.readLine()) != null) {
```

```
                    System.out.println(st);
                    bw.write(st);
                }
                bw.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                System.out.println("Done");
                // try
                // {
                //  JSPtoHTML(JSPfileName);
                // }
                // catch (IOException e) {
                //  System.out.println("An error occurred
                  .");
                //  e.printStackTrace();
                // }
            }
    } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
    }
    return numberOfFiles;
}

public void JSPtoHTML(String fileNameForJSP) throws Exception {
    // File path is passed as parameter
    File file = new File(fileNameForJSP);
```

```
// Note: Double backquote is to avoid compiler
// interpret words
// like \test as \t (ie. as a escape sequence)

// Creating an object of BufferedReader class
BufferedReader br
= new BufferedReader(new FileReader(file));

// Declaring a string variable
String st;
// Condition holds true till
// there is character in a string
String htmlFileName = fileNameForJSP.replace("jsp","html");
File htmlFile = new File(htmlFileName);
String content = "Writing To File";
if (!htmlFile.exists()) {
       htmlFile.createNewFile();
}
try {
       FileWriter fw = new FileWriter(htmlFile.
          getAbsoluteFile());
       BufferedWriter bw = new BufferedWriter(fw);
       while ((st = br.readLine()) != null) {
              System.out.println(st);
              bw.write(st);
}
bw.close();
```

```
        } catch (IOException e) {
                e.printStackTrace();
        }
        System.out.println("Done");
}

public Long saveHTML(WordMatch wordMatch){
        logger.info(wordMatch);
        Long numberOfFiles = (long) 0;
        try {
                File file = new File("./src/main/webapp/view/
                    word_match0.html");
                logger.info("Decoding String");
                String cleanedHTML = wordMatch.toString().replace("
                    WordMatch(content=","").replace(")","");
                logger.info(cleanedHTML);
                byte[] decodedBytes = Base64.getDecoder().decode(
                    cleanedHTML.getBytes());
                String html = new String(decodedBytes, "UTF-8");
                logger.info(html);
                if (file.createNewFile()) {
                        System.out.println("File created: " + file.
                            getName());
                        try {
                                FileWriter myWriter = new FileWriter
                                    ("./src/main/webapp/view/
                                    word_match0.html");
                                myWriter.write(html);
                myWriter.close();
        } catch (IOException e) {
                System.out.println("An error occurred
                    .");
                e.printStackTrace();
        }
} else {
        String fileName = file.getName().toString();
        String index = fileName.substring(fileName.
            indexOf("h") + 1);
        index = index.substring(0, index.indexOf("."))
            ;
        Integer parsedInt = Integer.parseInt(index);
        System.out.println(parsedInt);
        Stream<Path> files = Files.list(Paths.get("./
            src/main/webapp/view/"));
        numberOfFiles = files.map(Path.class::cast)
                                .filter(path -> path.
                                    getFileName().
                                    toString().
                                    startsWith("
                                    word_match"))
                                .count();
        fileName = fileName.replace(index,
            numberOfFiles.toString());
        System.out.println(numberOfFiles);
        System.out.println("fileName should have been
            printed by now");
```

```
                             file = new File(fileName);
                             FileWriter myWriter = new FileWriter("./src/
                                main/webapp/view/"+file);
                             myWriter.write(html);
                             myWriter.close();
                      }
              } catch (IOException e) {
                      System.out.println("An error occurred.");
                      e.printStackTrace();
              }
              return numberOfFiles;
       }
}
```

A.3 MODEL

```
package com.company.app.model;

/*
 * WordMatch.java
 * Author: Evan Gertis
 */

import lombok.Data;


@Data
public class View {
        public Long id;


    public void setId(Long Id) {
        this.id = Id;
    }
}

package com.company.app.model;


/*
 * WordMatch.java
 * Author: Evan Gertis
 */


import lombok.Data;



@Data
public class WordMatch {

    public String content;
}
```

A.4 CONTROLLER

```
package com.company.app.controller;
```

```
//WordMatchController.java
//Author: Evan Gertis 10/11/2021

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;

import com.company.app.model.View;
import com.company.app.model.WordMatch;
import com.company.app.service.WordMatchService;

@Controller
public class WordMatchController {
       private static final Logger logger = LogManager.getLogger(
          WordMatchController.class);
       private final WordMatchService wordMatchService;

       @Autowired
       public WordMatchController(WordMatchService wordMatchService) {
              logger.info("visiting word match");
```

```
        this.wordMatchService = wordMatchService;
}

@PostMapping("/wordmatchgenerator")
public ResponseEntity<View> saveWordMatchJSP(@RequestBody WordMatch
    wordMatch) {
        logger.info("Processing word match from client");
        logger.info(wordMatch);
        Long Id = wordMatchService.saveJSP(wordMatch);
View view = new View();
logger.info("New view created with id {}",Id);
logger.info("View object before {}",view);
view.setId(Id);
logger.info("View object after {}",view);
        return new ResponseEntity<View>(view, HttpStatus.CREATED);
}

@PostMapping("/wordmatchgeneratorXML")
public ResponseEntity<HttpStatus> saveWordMatchXML(@RequestBody
    WordMatch wordMatch) {
        logger.info("Processing word match from client");
        logger.info(wordMatch);
        // Long Id = wordMatchService.saveHTML(wordMatch);
// View view = new View();
// logger.info("New view created with id {}",Id);
// logger.info("View object before {}",view);
// view.setId(Id);
// logger.info("View object after {}",view);
```

```
            return new ResponseEntity<HttpStatus>(HttpStatus.OK);
        }

        @RequestMapping("/wordmatchgenerator")
    public String getWordMatch(Model model) {
        return "word_match";
    }

        @RequestMapping("/wordmatchgenerator/{id}")
    public String getWordMatch(@PathVariable String id ,Model model) {
        return "word_match"+id;
    }

}
```

A.5 VIEW

```
    <%@ page contentType="text/html;charset=UTF-8" language="java" %>

<!DOCTYPE HTML>
<html lang="en">
  <head>
    <title>Word Match Generator</title>
    <style>
    *:focus {outline: 2px solid blue; outline-offset: 2px;}
    details {padding:3px;}
    </style>
    <link rel="stylesheet" type="text/css" href="${pageContext.request.
```

```
        contextPath}/static/css/boxes.css" />
    <link rel="stylesheet" type="text/css" href="${pageContext.request.
        contextPath}/static/css/style.css" />
    <script type="text/javascript" src="${pageContext.request.contextPath}/
        static/js/event1.js"></script>

<!-- Global Site Tag (gtag.js) - Google Analytics -->
<script async src="https://www.googletagmanager.com/gtag/js?id=UA
    -89940905-27"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments)};
  gtag('js', new Date());
  gtag('config', 'UA-89940905-27');
</script>

<!-- <script type="text/javascript" src="../logging.js"></script> -->
</head>


  <body>
    <div id="maincontentstyle">
        <div id="boxstyle">
          <h3 id= "title">Word Match Generator</h3>
          <div>
            <div id="inputs">
                  <div id="inputBoxes">
                      <title>Input:</title>
          <div>
            Title: <input id= "title_input" type="text">
          </div>


          <div>
            Key Term 1: <input id="el1" type="text" value="">
          </div>
          <div>
            Description 1: <input id="dl1" type="text" value="">
          </div>
          <div>
            Key Term 2: <input id="el2" type="text" value="">
          </div>
          <div>
            Description 2: <input id="dl2" type="text" value="">
          </div>
        </div>
        <span style="padding: 3px">
          <button id ="add_more" class="button" type="button"
              onClick="add_more()">Add More</button>
        </span>
        <span style="padding: 3px">
          <button id ="one" class="button" type="button" onClick="
              generate_html()">Generate Html</button>
        </span>
    </div>
    </div>
  </div>
```

```
          <div id="results" class="row">
          </div>
          <div id="renderedHTML" class="row">
          </div>
        </div>



<script src="${pageContext.request.contextPath}/static/js/jquery-1.7.2.min.
   js"></script>

<script src="${pageContext.request.contextPath}/static/js/jquery-ui.min.js
   "></script>
<script src="${pageContext.request.contextPath}/static/js/jquery.ui.touch-
   punch.min.js"></script><script src="${pageContext.request.contextPath}/
   static/js/jquery.alerts.js"></script><link href="${pageContext.request.
   contextPath}/static/js/jquery.alerts.css" rel="stylesheet" type="text/
   css" media="screen" />

<script type="text/javascript">
$(init);
$( window ).unload(function() {
  removeStorage.removeItem("someVarKey1");
});
function init() {
      document.getElementById('resetButton').style.display = 'none';
document.getElementById("resetButton").style.visibility = "hidden";
if (false && sessionStorage.getItem("someVarKey1")) // No focus for the
```

```
    first time
  $("#one").focus();
  var numbers = [3, 4, 5, 1, 2];
  initialize(numbers);



}
</script>

<script type="text/javascript" src="${pageContext.request.contextPath}/
    static/js/word_match.js"></script>
<script type="text/javascript" src="${pageContext.request.contextPath}/
    static/js/GetElementPosition3.js"></script>
<script>



audioOn = false;
$(function() {
  $('.menulink').click(function(){
    if (audioOn) {
      $("#bg").attr('src',"${pageContext.request.contextPath}/static/images/
          audioOff.png");
      audioOn = false;
    }
    else {
      $("#bg").attr('src',"${pageContext.request.contextPath}/static/images/
          audioOn.png");
      audioOn = true; speak(" ");
    }
    return false;
  });
});
   </script>

   </body>
</html>
<script src="//cdnjs.cloudflare.com/ajax/libs/highlight.js/10.7.1/highlight
    .min.js"></script>
```

A.6 STATIC

A.6.1 JS

```
    // window.load = main()

// function main(){
    // initially html is not generated.
    var htmlGenerated = false;
    // number of inputs start out as 2.
    var numberOfInputs = 2;
    // initially no addtional inputs have been added.
    var addMore = false;
    // initialize the answer.
    var answer = '';

    // saved variables
```

```
var saved = false
var saved_id = 0

function reset() {
// reset the htmGenerated to false.
htmlGenerated = false;
numberOfInputs = 2;
var someVarName = true;
sessionStorage.setItem("someVarKey1", someVarName);
window.location.reload();
}

function populate_numbers_array(footer, dlArray){
    console.log("populating numbers")
    dlArray.forEach(i => {
        footer += i.replace ( /[^\d.]/g, '' );
        footer += ',';
        console.log("adding "+i+" to the numbers array")
    })
    return footer
}

function show_answer() {
    jAlert(answer, 'Correct Match');
}

function generate_html() {
```

```
// retrieve the keys and descriptions. Then load them into their
   respective arrays.
const e_inputs = document.querySelectorAll("[id^='el']");
const d_inputs = document.querySelectorAll("[id^='dl']");
let elArray = [];
let dlArray = [];
const title = document.getElementById('title_input').value;
e_inputs.forEach( i => { if(i.value) elArray.push(i.value) });
d_inputs.forEach( i => { if(i.value) dlArray.push(i.value) });



//has the html already been generated?
if(!htmlGenerated){
   //fetch the results box
   results = document.getElementById("results");

   //create textarea
   textarea = document.createElement("textarea");
   textarea.setAttribute("id","generated_html_textarea");

   // initialize blank html
   header = '<!DOCTYPE HTML>\n<html lang=\"en\">\n\t<head>\n\t\t<title>
      Word Matching Exercise</title>\n\t\t<style>\n*:focus {outline: 2
      px solid blue; outline-offset: 2px;}\n\t\tdetails {padding:3px
      ;}\n\t\t</style>\n\t\t<link rel=\"stylesheet\" type=\"text/css\"
       href=\"${pageContext.request.contextPath}/static/css/boxes.css
      \" />\n\t\t<script type=\"text/javascript\" src=\"${pageContext.
      request.contextPath}/static/js/event1.js\">';
```

```
header += "<link rel=\"stylesheet\" type=\"text/css\" href=\"${
   pageContext.request.contextPath}/static/css/style.css\" />"
header += '</'
header += 'script>\n'
header += '<script async src=\"https://www.googletagmanager.com/gtag
   /js?id=UA-89940905-27\">'
// header += '</'
// header += 'script>\n<script>\n\t window.dataLayer = window.
   dataLayer || [];\n\t function gtag(){dataLayer.push(arguments)
   };\tgtag(\"js\", new Date());\tgtag(\"config\", \"UA
   -89940905-27\");\n'
header += '</'
header += 'script>\n'
header += '<script src="${pageContext.request.contextPath}/static/js
   /jquery-1.7.2.min.js">'
header += '</'
header += 'script>\n'
header += '<script src="${pageContext.request.contextPath}/static/js
   /jquery-ui.min.js">'
header += '</'
header += 'script>\n'
header += '<script src="${pageContext.request.contextPath}/static/js
   /jquery.ui.touch-punch.min.js">'
header += '</'
header += 'script>\n'
header += '<script src="${pageContext.request.contextPath}/static/js
   /event1.js">'
header += '</'
header += 'script>\n'
header += '<script src="${pageContext.request.contextPath}/static/js
   /jquery.alerts.js">'
header += '</'
header += 'script>\n'
header += '<link href="${pageContext.request.contextPath}/static/js/
   jquery.alerts.css" rel="stylesheet" type="text/css" media="
   screen">'
header += '<script type=\"text/javascript\" src=\"${pageContext.
   request.contextPath}/static/js/logging.js\">'
header += '</'
header += 'script>\n</head>\n\t\t<body>';
let html = '';
html += header;
html += '<div id=\'maincontentstyle\'>\n'
html += '\t<center>\n'
html += '\t\t<div id=\'boxstyle\'>\n'
html += '\t\t\t<h3 id=\'title\'>'+title+"</h3>\n";
//create key inputs
html += '\t\t\t\t<center>\n'
html += '\t\t\t\t\t<div class=\'source\'>\n'
console.log("The value of the dlArray is")
console.log(dlArray)
console.log("The value of the elArray is")
console.log(elArray)
dlArray = shuffleDescriptions(dlArray);
for (let i = numberOfInputs; i < elArray.length+numberOfInputs; i++)
   {
   html += '\t\t\t\t\t\t\t<div id=\'s';
       id  = i-numberOfInputs+1;//elArray[i-numberOfInputs].replace
            ( /[^\d.]/g, '' );
   console.log("id "+id)
       html += id;
       html +='\' class=\'draggyBox-small ui-draggable\'>\n';
       html += '\t\t\t\t\t\t\t\t'
       html += elArray[i-numberOfInputs]
       html += '\n';
       html +='\t\t\t\t\t\t\t</div>\n';
}
elArray = shuffleKeys(elArray);
console.log("The value of the dlArray is")
console.log(dlArray)
console.log("The value of the elArray is")
console.log(elArray)
html += '\t\t\t\t\t\t</div>\n'
html += '\t\t\t\t\t\t</center>\n'

//create description inputs
html += '\t\t\t\t\t\t<table id=\'tablestyle\'>\n'
for (let i = numberOfInputs; i < dlArray.length+numberOfInputs; i++)
   {
   html +='\t\t\t\t\t\t\t<tr>\n'
   html += '\t\t\t\t\t\t\t<td id=\'row';
   id  =  i-numberOfInputs+1;//dlArray[i-numberOfInputs].replace (
       /[^\d.]/g, '' );
   console.log("id "+id);
   html += id;
   html +='\'>\n';
   html += '\t\t\t\t\t\t\t\t<div id=\'t';
   html += id;
   html +='\' class=\'ltarget ui-droppable\'>'
   html +='</div>\n'
   html +='\t\t\t\t\t\t\t</td >\n'
   html +='\t\t\t\t\t\t\t<td id=\'d'
   html += id
   html += '\'>\n'
   html += '\t\t\t\t\t\t\t\t';
   html += dlArray[i-numberOfInputs];
   html += '\n';
   html +='\t\t\t\t\t\t\t\t</td >\n'
   html +='\t\t\t\t\t\t\t</tr>\n';
}
html += '\t\t\t\t\t\t</table>\n';
html += '\t\t\t\t\t</center>\n'
html += '\t\t</div>\n'
html += '\t</center>\n'
html += '</div>'
html += '<span style=\"padding: 3px\"> <button id =\"one\" class=\"
   button\" type=\"button\" onClick=\"show_answer'
html += '()'
html += '"'
html += ">"
html += 'Show Answer'
html += '</'
html += 'button> <button id = \"resetButton\" class=\"button\" type
   =\"button\" onClick=\"reset'
html += '()'
html += '"'
html += '>'
html += 'Reset'
html += '</'
html += 'button>'
html += '</span>'
html += '<span id="audio" style="">'
html += '<a href="" title="Turns Text-to-Speech Output On or Off"
   class="menulink" style="text-decoration: none;">'
html += '<img id="bg" src="${pageContext.request.contextPath}/static
   /images/audioOff.png" height="30" width="30" style="margin-
   bottom:-10px; padding-bottom:-20px;">'
html += '</a>'
html += '</span>'
footer = '\n\t\t</body>\n</html>\n';
footer += ''
footer += '<script type="text/javascript">'
footer += '$(init);'
footer += '$( window ).unload(function() {'
footer += 'removeStorage.removeItem("someVarKey1");'
footer += '});'
footer += 'function reset() {'
footer += ' var someVarName = true;'
footer += 'sessionStorage.setItem("someVarKey1", someVarName);'
footer += 'window.location.reload();'
footer += '}'
footer += 'function init() {'
footer += '   document.getElementById(\'resetButton\').style.
   display = \'none\';'
footer += 'document.getElementById("resetButton").style.visibility =
    "hidden";'
footer += 'if (false && sessionStorage.getItem("someVarKey1"))'
footer += '$("#one").focus();'
console.log('var numbers = [');
footer += 'var numbers = ['
for (let i = numberOfInputs; i < dlArray.length+numberOfInputs; i++)
   {
   footer += dlArray[i-numberOfInputs].replace ( /[^\d.]/g, '' );
   console.log(dlArray[i-numberOfInputs].replace ( /[^\d.]/g, '' ))
   footer += ',';
   console.log(',')
}
console.log('];')
footer += '];'
footer += 'initialize(numbers);'
footer += '}'
footer += '</script>'
footer += ' <script>'
footer += ' answer = '
footer += '\"'
answer = '';
for (let i = numberOfInputs; i < dlArray.length+numberOfInputs; i++)
    {
```

```
    answer += elArray[i-numberOfInputs];
    answer += ':';
    answer += dlArray[i-numberOfInputs];
    answer += ' '
}
footer += answer
console.log(answer)
footer += '\"'
footer += ';'
// footer += '\n'
// footer += ' Iteration: is one time execution of the loop body.'
// footer += '\n'
// footer += 'Loop Continuation Condition: is a Boolean expression
    that controls the execution of the loop.'
// footer += '\n'
// footer += 'Infinite Loop: is a loop that runs forever due to an
    error in the code.'
// footor += '\n'
// footer += 'Off-by-one: is an error in the program that causes the
     loop body to be executed one more or less time."'
footer += ' function show_answer() {'
footer += '     jAlert(answer, \'Correct Match\');'
footer += ' }'
footer += '</script>'
footer += ' '
footer += '<script type="text/javascript" src="${pageContext.request
    .contextPath}/static/js/GetElementPosition3.js"></script>'
footer += ' <script>'
footer += '  $(function(){'
footer += ' if (\'speechSynthesis\' in window) {'
footer += '  speechSynthesis.onvoiceschanged = function() {'
footer += '    var $voicelist = $(\'#voices\');'
footer += ''
footer += '    if($voicelist.find(\'option\').length == 0) {'
footer += '      speechSynthesis.getVoices().forEach(function(voice,
     index) {'
footer += '        var $option = $(\'<option>\')'
footer += '        .val(index)'
footer += '        .html(voice.name + (voice.default ? \' (default)
   \' :\'\'));'
footer += '        $voicelist.append($option);'
footer += '      });'
footer += ''
footer += '      $voicelist.form_select();'
footer += '    }'
footer += '  }'
footer += ' } '
footer += '});   '
footer += 'audioOn = false;'
footer += '$(function() {'
footer += '$(\'.menulink\').click(function(){'
footer += ' if (audioOn) {'
footer += '   $("#bg").attr(\'src\',"${pageContext.request.
   contextPath}/static/images/audioOff.png"); '
footer += '   audioOn = false;'
footer += ' }'
footer += ' else {'
footer += '    $("#bg").attr(\'src\',"${pageContext.request.
   contextPath}/static/images/audioOn.png");'
footer += '    audioOn = true; speak(" ");'
footer += ' }'
footer += ' return false;'
footer += '});'
footer += '});'
footer += ' </script> '
html += footer;

// html generation is done.
htmlGenerated = true;
textarea.value = html;
results.replaceChildren(textarea);

// Generate reset, show answer, , and render html buttons
controls = document.createElement("div");
controls.setAttribute("id","program1");
controls.setAttribute("style","border: 1px solid #EB0D1B; width: 450
   px; font-family: courier; font-size: 100.5%; margin: 0px auto;
   border: 1px; text-align: center; margin-top: 5px;");
controls.innerHTML += '<button id = "renderHTMLButton" class="button
   " type="button" onClick="render_html()">Render html</button>\n';
controls.innerHTML += '<button id = "submit" class="button" type="
   button" onClick="saveContent()"> Save </button>\n';
controls.innerHTML += `<button id=\"view_button\" class=\"button\"
   style=\" display: none;\"><a href=\"${window.location.href}/${
        saved_id}\"> view</a> </button>\n`;
    if(document.getElementById("renderHTMLButton"))
        results.parentNode.replaceChild(controls);
    results.parentNode.appendChild(controls);
}
}



function saveContent(){
    console.log("calling save content");
    var html_content = document.getElementById("generated_html_textarea
        ");
    var b64_string = btoa(html_content.value)
    console.log(b64_string)
    var xhr = new XMLHttpRequest();
    xhr.open("POST", "/wordmatchgenerator", true);
    xhr.setRequestHeader("Content-Type", "application/json;charset=UTF
        -8");
    xhr.onreadystatechange = function()
    {
        if(xhr.readyState == 4 && xhr.status == 201) {
            console.log(xhr.status)
            console.log("content saved");
            saved = true;
            view_button = document.getElementById("view_button");
            view_button.style.display = "inline";
            console.log('JSON.parse(xhr.response).id ' + JSON.parse(xhr.
                response).id)
            saved_id = JSON.parse(xhr.response).id
            console.log('saved_id ' +saved_id)
            view_button.children[0].href = `${window.location.href}/${
                saved_id}`
        }
        else{
            console.log(xhr.status)
            console.log(xhr.response)
            console.log("content was not save successfully");
        }
    }
    console.log('{"content":\"'
            +b64_string+'\"}');
    xhr.send(JSON.stringify({content: b64_string}));
}

function add_more() {
// we've added more inputs.
addMore = true;

// set html generated to false, because new inputs have been added.
htmlGenerated = false;

// increment the number of inputs.
numberOfInputs++;

//fetch the input boxes.
inputs = document.getElementById("inputBoxes");
```

```
// create newline
br = document.createElement("br");

//create a new row for a key term.
row = document.createElement("div");

// set the key term text.
row.innerHTML = "Key Term ";
row.innerHTML +=numberOfInputs;
row.innerHTML +=" :";

// create the input for the key.
key = document.createElement("input");
key.setAttribute("id","el"+numberOfInputs);

//add the key to the row.
row.appendChild(key);
row.after(br);

//create a row for the new description.
row2 = document.createElement("div");

// set the description text.
row2.innerHTML = "Description "
row2.innerHTML+=numberOfInputs;
row2.innerHTML+=" :";
row2.after(br);
```

```
// create the description input
description = document.createElement("input");
description.setAttribute("id","dl"+numberOfInputs);

// add the description to the row.
row2.appendChild(description);

// add the rows for the key and the description to the inputBoxes.
inputs.appendChild(row);
inputs.appendChild(row2);
}

function render_html(){

textarea = document.getElementById("generated_html_textarea");
// Set the generate html to the value from the textarea.
generated_html = textarea.value;
console.log(generated_html);
// Create a new tab.
var new_window = window.open('');
maincontentstyle = document.getElementById("maincontentstyle");
if(document.getElementById("rendered_html"))
    document.getElementById("rendered_html").remove();

rendered_html = document.createElement("div");
rendered_html.setAttribute("id","rendered_html");
rendered_html.setAttribute("style","border: 1px solid #EB0D1B; width:
    450px; font-family: courier; font-size: 100.5%; margin: 0px auto;
    border: 1px; text-align: center; margin-top: 5px;");
rendered_html.innerHTML += generated_html;
results = document.getElementById("results");

if(document.getElementById("rendered_html"))
    results.parentNode.appendChild(rendered_html);

// Append the rendered html to the results tab
results.parentNode.appendChild(rendered_html);
header = '<!DOCTYPE HTML>\n<html lang=\"en\">\n\t<head>\n\t\t<title>
    Word Matching Exercise</title>\n\t\t<style>\n*:focus {outline: 2px
    solid blue; outline-offset: 2px;}\n\t\tdetails {padding:3px;}\n\t\t
    </style>\n\t\t<link rel=\"stylesheet\" type=\"text/css\" href=\"
    static/css/boxes.css\" />\n\t\t<script type=\"text/javascript\" src
    =\"static/js/event1.js\">';
header += '</'
header += 'script>\n'
header += '<script async src=\"https://www.googletagmanager.com/gtag/js
    ?id=UA-89940905-27\">'
header += '</'
header += 'script>\n'
// header += '<script>\n\t window.dataLayer = window.dataLayer || [];\n
    \t function gtag(){dataLayer.push(arguments)};\tgtag(\"js\", new
    Date());\tgtag(\"config\", \"UA-89940905-27\");\n'
// header += '</'
// header += 'script>\n'
header += '</head>\n\t\t<body>';
```

```
new_tab_html = header;
new_tab_html += rendered_html.innerHTML;
footer = '\n\t\t</body>\n</html>\n';
footer += '<script type="text/javascript" src="static/js/
   GetElementPosition3.js">'
footer += '</'
footer += 'script>'
footer += '<script type=\"text/javascript\" src=\"static/js/word_match.
   js\">'
footer += '<script src="static/js/jquery-1.7.2.min.js">'
footer += '</'
footer += 'script>\n'
footer += '<script src="static/js/jquery-ui.min.js">'
footer += '</'
footer += 'script>\n'
footer += '<script src="static/js/jquery.ui.touch-punch.min.js">'
footer += '</'
footer += 'script>\n'
footer += '<script src="static/js/event1.js">'
footer += '</'
footer += 'script>\n'
footer += '<script src="static/js/jquery.alerts.js">'
footer += '</'
footer += 'script>\n'
footer += '<link href="static/js/jquery.alerts.css" rel="stylesheet"
   type="text/css" media="screen">'
footer += '<script type=\"text/javascript\" src=\"static/js/logging.js
   \">'
```

```
    footer += '</'
    footer += 'script>\n'
    new_tab_html += footer;
    console.log(new_tab_html);
    new_window.document.write(new_tab_html);
    }

    rand = Math.random();
    function shuffleDescriptions(a){
        for(let j,i=a.length;i>1;){
         j=Math.floor(rand*i--);
         if (i!=j) [a[i],a[j]]=[a[j],a[i]]
        }
        console.log("shuffled dlarray")
        console.log(a)
        return a
    }
    function shuffleKeys(a){
        for(let j,i=a.length;i>1;){
         j=Math.floor(rand*i--);
         if (i!=j) [a[i],a[j]]=[a[j],a[i]]
        }
        console.log("shuffled elarray")
        console.log(a)
        return a
    }
// }
```