Interactive Word Search Generator

CS-450-01

Christina Porter

01/10/2024

I.   INTRODUCTION.

WORD SEARCHES have been viable for learning vocabulary through semantic meaning and similarity between words since early times in multiple cultures. Typically, each word search revolves around a particular general topic, with terms relating to that topic as the words to seek. These words are typically hidden in a grid format, like an array, in straight lines going forwards, backwards, up, down, and diagonally in all four directions. Word searches can be customized to be appropriate for different vocabulary levels or difficulty. As a result, word searches have the potential to be an invaluable resource in the learning process, both inside and outside the classroom.

When the mention of a class comes up, the typical thought may be of a lecture. This is known as passive learning. In passive learning, you grasp the material as it was presented to you. As time progresses, classrooms are providing resources for multiple types of learning facilitated to the students. This is known as active learning. Described in [1], active learning is self-governed, or autonomous. Active learning allows students to choose what method they use to learn instead of a one-size-fits-all approach, whether this means they choose a passive learning method or a different method. A study conducted by L. Nirmal, M. MS, and M. Prasad found that, "Multicentric approach to learning is more effective than a single one," in dental education. [2] To support this claim in other fields, there are other studies that found graduate electrical engineering students find the learning activity level correlates with the perceived usefulness of the activity [3], and 90% of a class of social science university students in the UK agreed that an active learning approach made information more memorable and engaging than the alternative. [4]

Word searches have been applied in two different ways within research. There is one introductory version where the words are given, and a more advanced version where you obtain the word through filling in blanks based on surrounding context. [2] Within studies conducted with the general population 50 years or older and 2nd grade and nursing students, it was found that the frequency of completing word search puzzles had a positive correlation to cognitive function and the ability to recall words respectively. [5], [6], [7] In addition to vocabulary improvement, a study by J. Leach, M. Mancini, et al. found that there are even physical connections associated with word searches and cognitive function. [8]

Various word searches are difficult to find in stores as the physical age dwindles. In the digital age, the need for a digitized puzzle generator is needed that can bypass the challenge of manually creating topics and sub-words. There are three websites out of many that are comprehensive to available features in most digital word search generators: Word Search Labs [9], Worksheet Zone [10], and education.com [11]. Each of these websites includes the ability to create your own title and upload your own words and choose the orientation of the words in the puzzle, and the accessibility of an answer key. Some isolated features include size of the grid, shape of the grid, and the ability to save within a profile. One of the three even includes the ability to create puzzles through topics found by AI, albeit the words generated with this method often repeat themselves in multiple parts within the answer key.
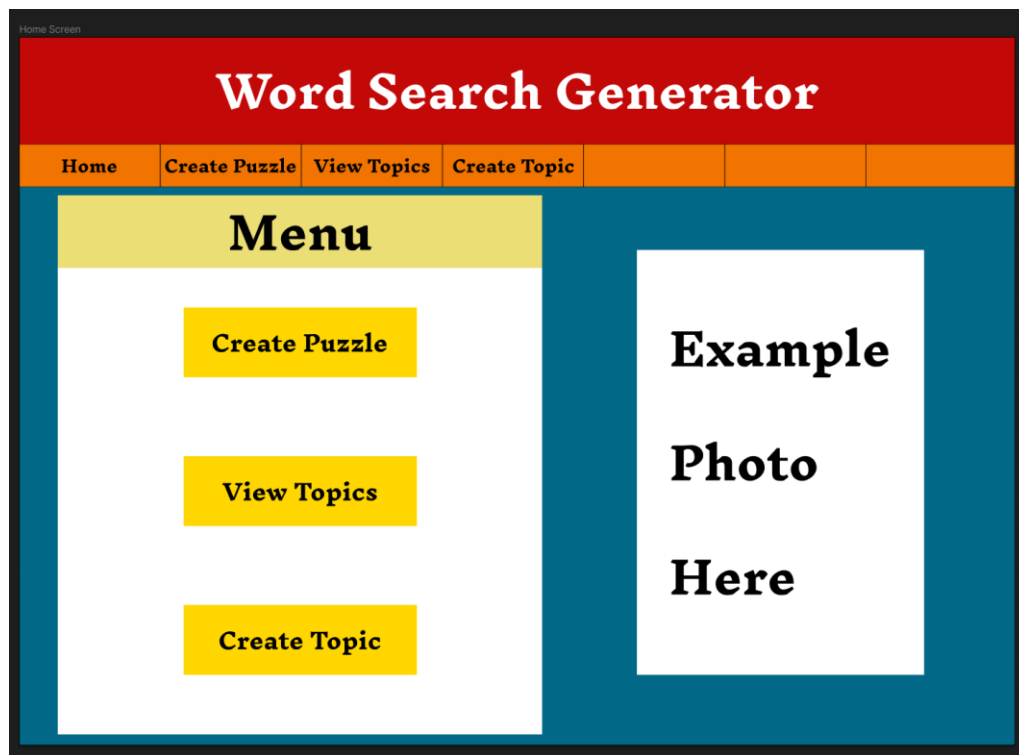
This project proposal is related to the creation of a new program in the web browser solely dedicated to the creation and storing of word search problems. For each of the sites mentioned above, the sites use a singular interface.
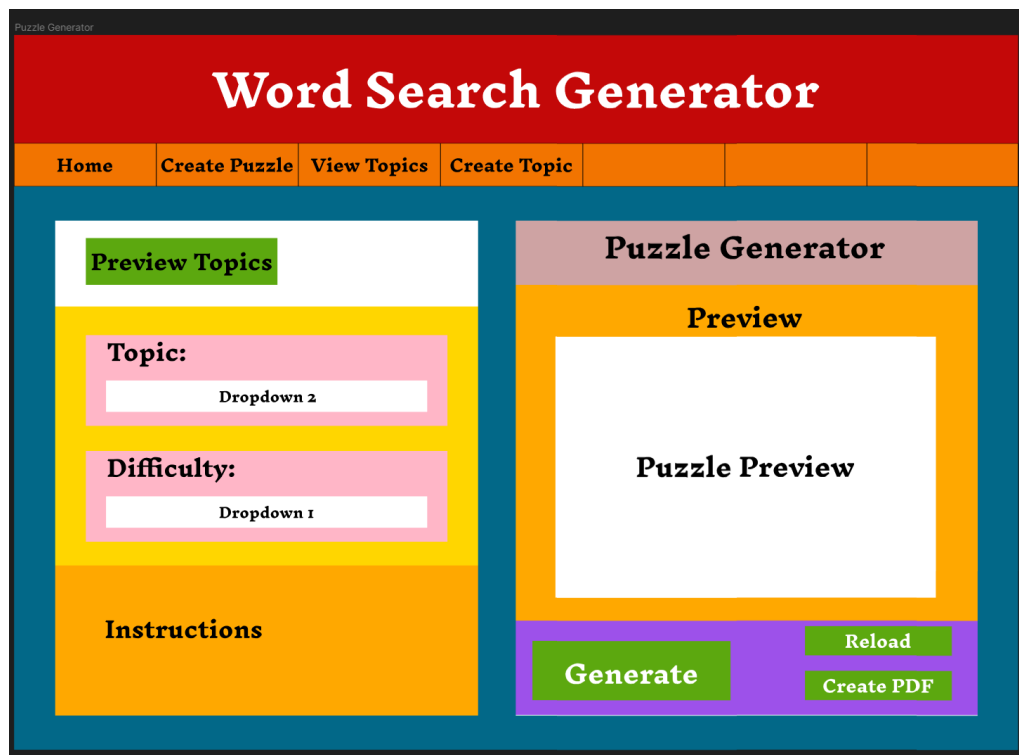
A. Project Methodology.

This proposal uses a fully functional multi-page website using React, which controls the website components and functionality with a backend of NodeJS to handle the logic within the website. This project will consist of several parts to ensure there is always a functional project.

1) *Part 1 – Initial Setup:* During Part 1, the focus will be: creating the front-end interface, and setting up the database, calling the database, setting up the logic of components. The methods that will be taken are as follows.

   a) *Interface:* The interface will be set up via React with CSS styling. The following images are planned layouts for the pages. Components are displayed in separate colors – this is not representative of the finished look.
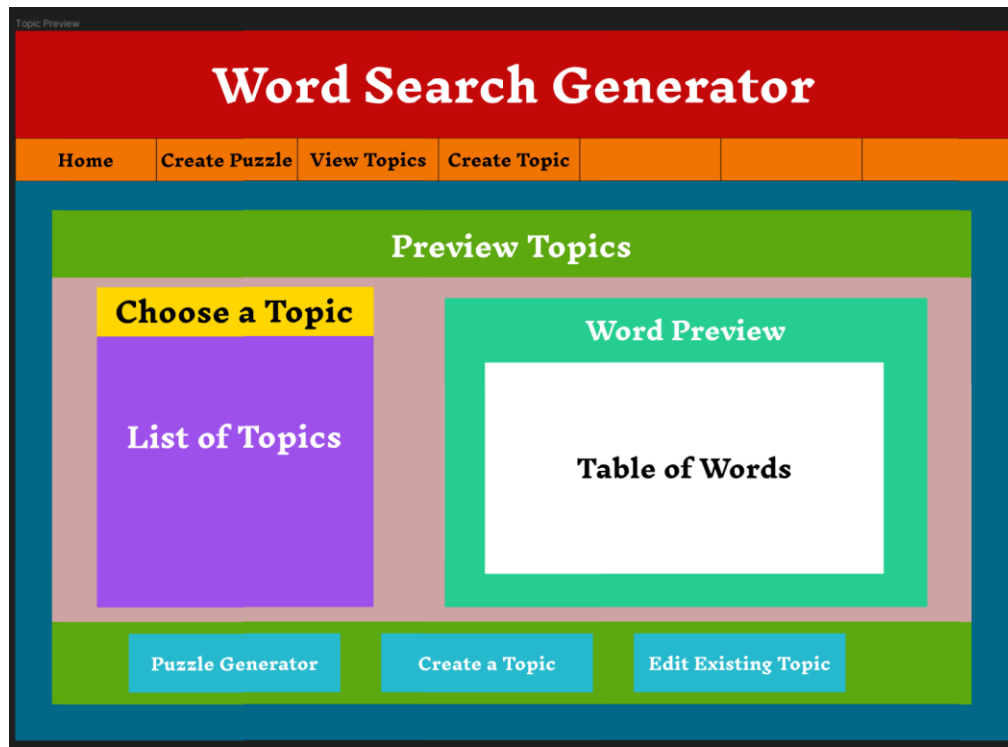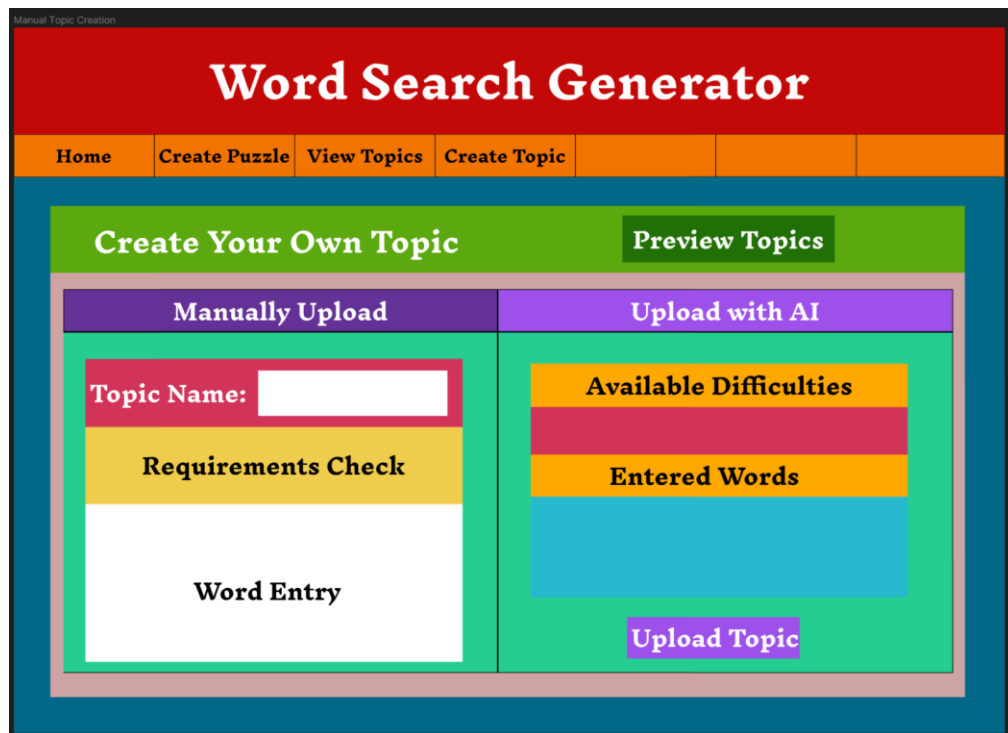
HOME PAGE/MAIN MENU

Word Search Generator

| Home | Create Puzzle | View Topics | Create Topic | | |

Menu

Create Puzzle

View Topics

Create Topic

Example

Photo

Here

PUZZLE CREATION PAGE

Word Search Generator

| Home | Create Puzzle | View Topics | Create Topic | | |

Preview Topics

Topic:

Dropdown 2

Difficulty:

Dropdown 1

Instructions

Puzzle Generator

Preview

Puzzle Preview

Generate

Reload

Create PDF

TOPICS PAGE

CREATE MANUAL TOPICS PAGE

b) *Backend* – The backend will be implemented with NodeJS with a database using MongoDB. The user case and logic are below, divided into pages.

I. *MAIN MENU*

  User Story: The user will click on the buttons under the menu redirect to the respective pages.

II. *PUZZLE CREATION PAGE*

  1) *User Story:* Instructions will be available at the bottom of the screen for the user to reference. The user will click on the 'Topic' dropdown to display topics available in the database. Once the topic is chosen, the user will click on the 'Difficulty' dropdown to display the available difficulties associated with the topic in the database. After the choices are selected in both fields, the user will click on the 'Generate' button to display the puzzle. Based on the parameters chosen, the puzzle will display in the window to the right. If the user does not like the generated puzzle, the user can click on the 'Reload' button to generate a new puzzle. If the user does like the generated puzzle, the user can click on the 'Generate PDF' button to download a PDF of the puzzle they can print. If the user would like to view the topics and associated words and difficulties available, they can click the 'Preview Topics' button at the top to redirect to the topics page.

III) PREVIEW TOPICS PAGE

  i. *User Story:* Users will see a displayed list of topics available in the database. Once a topic is clicked, all words associated with that topic in the database will preview on the right. From

there, the user can choose to go back to the puzzle generator or to create a new topic. Edit an existing topic will come at a later step.

IV)   CREATE YOUR OWN TOPIC PAGE

i. *User Story:* Users can upload their own topic if the topic they want to see is not available in the preview topics page. When they enter the page, they will automatically be taken to the 'Manually Upload' screen. The user will enter their desired topic name and puzzle title, then the words they want to be associated with that puzzle.

As they add words, requirement checks for each difficulty will be displayed above the word entry box. The requirements for at least one difficulty must be met to upload a topic. If any requirements are not met – a red x ⊗ will display next to the requirement. For any requirements that are met, the symbol will automatically change to a green ✓ ✅ .

The interface on the right side of the screen will update to display the accuracy of the entered information. The count of words with the characters for each difficulty will display under 'Available Difficulties.'

ex: *Number of words between 3-5 letters: x*

Number of words with 3-10 characters: y

Number of words with 3-15 characters: z

If the requirements are met for a difficulty, that difficulty will also display under 'Available Difficulties' above the word counts. All entered words will display under 'Entered Words' for

the user to check the accuracy of entered words and adjust as needed. Once the user is
satisfied with the detected parameters, they can click the 'Upload Topic' button. Once this
button is clicked, they will be redirected to the 'Preview Topics' screen.

2) *Part 1.5 - Database Setup*: A noSQL database will be established in MongoDB. This database will be used
to store topics and associated words. This database will not be used to store puzzle information, as the
information is intended to be different on every render of the program. Storage will be relatively simple -
the topic name will be stored as a string and the words will be stored within a string Realm Set. This will be
good for setting up initial values, as initially we won't have edit topic functionality to re-write to the
database. MongoDB automatically sets up unique IDs.

3) *Part 3 – Cloud Implementation*: To implement cloud functionality, the platform AWS Amplify will be used.
The first step will be to move the UI to the cloud platform, connect the database to GraphQL, then ensure
the program is rendering and functioning properly. Once this is established, an authentication service will
be implemented that allows cookies, temporary authentication keys, and browser session storage. To take it
further, Amazon Cognito will be used to establish a sign-up, sign-in, and sign-out function using MFA. An
account recovery process will be added.

4) *Part 4 – Additional Feature Implementation:* A page that will read what is currently in the database and
alter the contents will be implemented in the form of an Edit Existing Topic page.
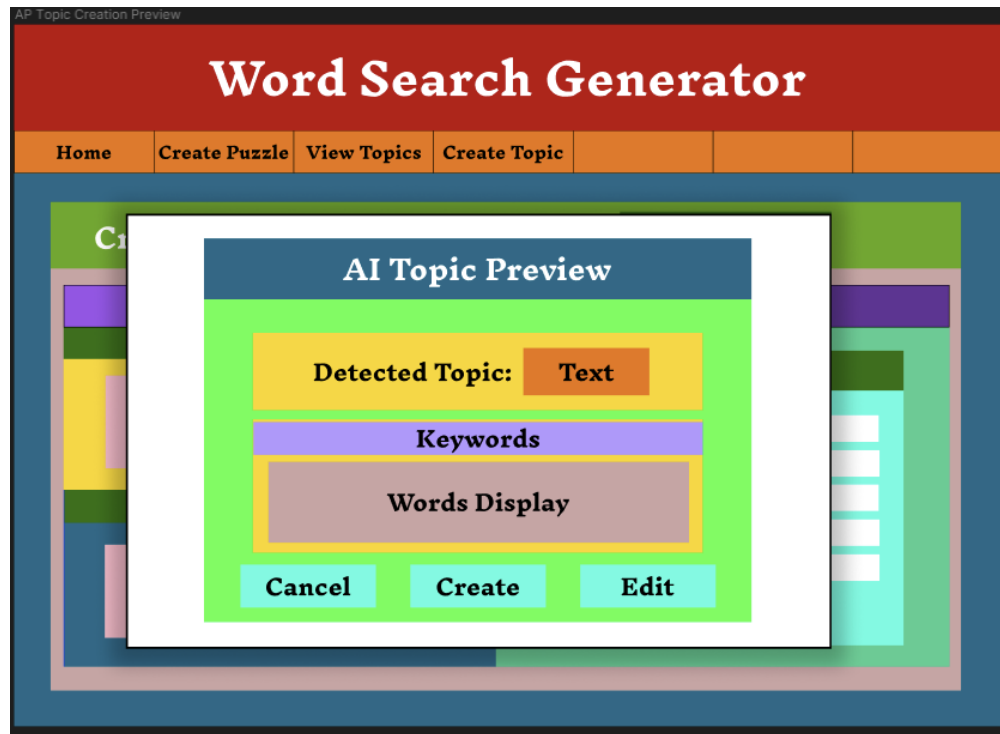
a) *Interface:* This will be a subpage accessed within the 'Preview Topics' page. The interface of the 'Edit Existing Topics' page will be like the 'Preview Topics' and 'Create Topic' pages. On the left-hand side, there is a Choose Topic section like the 'Preview Topics' page and the word editor available in the 'Create Topic' page. On the left hand, you can view the auto-populated information that is available in the 'Create Topic' pages. At the bottom, there are buttons to navigate critical database changes.

b) *User Story:* The user will be able to use this page to edit existing topics in the database. When loading in, no information will be pre-populated except for the list of topics. Once the user clicks on the topic they would like to edit, the list of current words will show in a text box designated the 'Existing Words Editor' and the current met requirements will show above the text box with populated fields such as number of characters counts for words, difficulties, and detected words. Users simply must remove or add words in the text box, then click on 'Make Changes'. Once this button is clicked, users will be redirected to the 'View Topics' page.

Other functions available include the ability to change the topic name and delete the topic by pressing the designated buttons. Users can view topics, words, and difficulties in this page, so there isn't a need for a 'View Topics' button directly on this page. Once a topic is entered or other function performed, the user will be redirected to the 'View Topics' page. If they would like to navigate to another page, they can use the menu at the top. If the user decides to leave the page before the submit button is clicked, they will get a notification asking if they're sure because their changes won't be saved.

c) *Part 1 - Generate a Topic with AI:* The first AI implementation takes a word model trained by articles and/or websites provided by the user. In this implementation, the subject matter will be extracted through NLTK in Python, BeautifulSoup, and requests (for URLs). Another method that could be used is Genism's Word2Vec. The user will have the option to input a word document or a website, depending on the button clicked. Documents will be accepted through word documents or PDFs. If a PDF if submitted, the Python package, PyPDF will be used to extract the words and turn it into text before using NLTK to parse the subject matter. After the subject is parsed, word embeddings will be used to implement the Common Bag of Words (CBoW) model to parse

associated words with the topic. The output will then be placed into a set to avoid duplicate words.
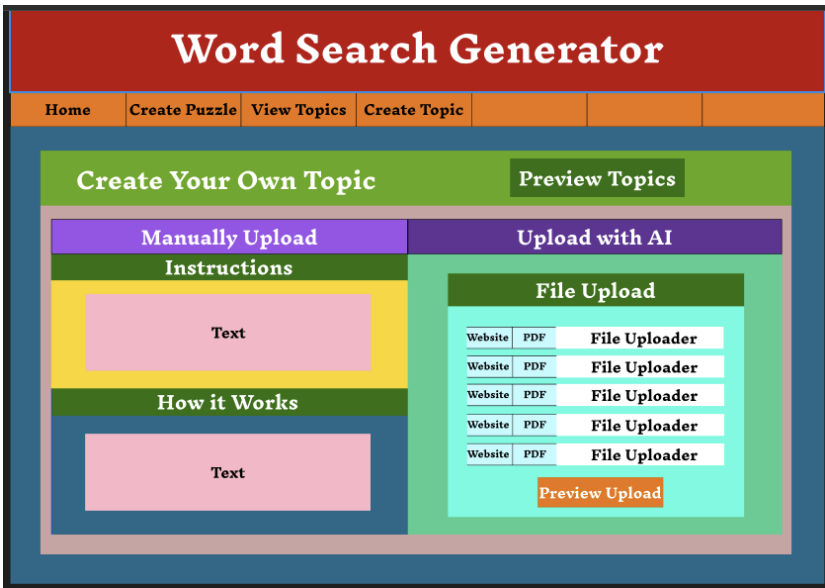
On the page, instructions and a quick explanation of the topic will be provided. Once the user submits the articles, a preview of the puzzle will be implemented. Users will be able to Edit the results before adding to the database. In both cases, the user will be taken to the 'Create Your Own Topic' page with the fields pre-populated to confirm results. From here, the user can check to make sure it meets difficulty requirements and preview the words one more time before uploading to the database. As mentioned previously, once the topic is created, they will be moved to the 'View Topics' page.

Word Search Generator

Home | Create Puzzle | View Topics | Create Topic

AI Topic Preview

Detected Topic: Text

Keywords

Words Display

Cancel | Create | Edit

d) *Part 2 – Future Implementation:* In further construction of this project, there would be an implementation of extracting the topic from the text as mentioned above but using word embeddings to generate the vectors and using reverse cosine similarity to find the similar words and creating a plane around the resulting vector to gather a corpus of words.

There may need to be a process to compare results from the CBoW model.

B. Proposed Project Schedule.

Date	Task
1/16	**Turn in proposal**
1/29	Identify structure and create React components structure.
2/5	• Put together elements of UI and ensure display is viable. • Set up initial DB and fill with 15 initial entries to test.
2/8	**Status Report Due**
2/12	Connect DB to project.
2/19	• Create display components and logic for the Preview Topics and Create Your Own Topic.
2/22	**Status Report Due**
2/26	Implement logic for creating puzzle and add to components.
3/4	Set up cloud configuration and DB
3/7	• **Status Report Due** • **Poster Abstract Due**
3/11	Add authentication and ensure each account contains separate states.

Date	Task
3/19	**Poster Due**
3/25	Create program to scrape documents and websites
4/1	Implement topic and word generator for scraper.
4/4	**Status Report Due**
4/10	Add topic and word page to website.
4/16	**Presentation Date**
4/24	**Final Project Due** **Final Report Due** **Final Video Due**
If there's time:	Create edit functionality of AI generated topics.

C. Initial Useful Resources.

[1]   "AWS Educate." Amazon. https://aws.amazon.com/education/awseducate/ (accessed Jan. 2024).

[2]   "AWS Amplify Resources." Amazon. https://aws.amazon.com/amplify/resources/?nc=sn&loc=6&dn=3 (accessed Jan. 2024).

[3]   "Build a Full-Stack React Application." Amazon. https://aws.amazon.com/getting-started/hands-on/build-react-app-amplify-graphql/module-one/ (accessed Jan. 2024).

[4]   "Introduction to GraphQL." GraphQL. https://graphql.org/learn/ (accessed Jan. 2024).

[5]   baeldung. "Topic Modeling with Word2Vec." Baeldung. https://www.baeldung.com/cs/ml-word2vec-topic-modeling (accessed Jan. 2024).

[6]   C. E. Moody. "Ida2vec – flexible & interpretable NLP models." Ida2vec. https://lda2vec.readthedocs.io/en/latest/?badge=latest (accessed Jan. 2024).

[7]   "Quick Start." React. https://react.dev/learn (accessed Jan. 2024).

[8]   "What is MongoDB?" MongoDB. https://www.mongodb.com/docs/manual/ (accessed Jan. 2024).

[9]   "JavaScript." mdn web docs. https://developer.mozilla.org/en-US/docs/Web/JavaScript (accessed Jan. 2024).

[10]   M. Dutta. "Word2Vec for Word Embeddings – A Beginner's Guide." Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/07/word2vec-for-word-embeddings-a-beginners-guide/ (accessed Jan. 2024).

[11]   M. Suri. "A Dummy's Guide to Word2Vec." Medium. https://medium.com/@manansuri/a-dummys-guide-to-word2vec-456444f3c673 (accessed Jan. 2024).

[12]   I. Logunova. "Word2Vec: Why Do We Need Word Representations?" Serokell. https://serokell.io/blog/word2vec (accessed Jan. 2024).

[13]   A. Sha. "Represent word vectors in Natural Language Processing." Medium. https://medium.com/@AlyssaSha/represent-word-vectors-in-natural-language-processing-13ae6230976d (accessed Jan. 2024).

[14]   S. Porter. "Understanding Cosine Similarity and Word Embeddings." Medium. https://spencerporter2.medium.com/understanding-cosine-similarity-and-word-embeddings-dbf19362a3c (accessed Jan. 2024).

[15]   TechClaw. "Cosine similarity between two arrays for word embeddings." Medium. https://medium.com/@techclaw/cosine-similarity-between-two-arrays-for-word-embeddings-c8c1c98811b# (accessed Jan. 2024).

[16]   H. Boukkouri. "Arithmetic Properties of Word Embeddings." Dataiku. https://blog.dataiku.com/arithmetic-properties-of-word-embeddings (accessed Jan. 2024).

[17]   P. Singh. "Deep Dive Into Word2Vec." Medium. https://medium.com/analytics-vidhya/deep-dive-into-word2vec-7fcefa765c17 (accessed Jan. 2024).

[18]   T. Arora. "Vector Algebra for Natural Language Processing." Medium.
         https://towardsdatascience.com/from-linear-algebra-to-text-representation-for-natural-language-processing-
         239cd3ccb12f (accessed Jan. 2024).

[19]   "JavaScript Comparison and Logical Operators." W3schools.
         https://www.w3schools.com/js/js_comparisons.asp (accessed Dec. 2023).

[20]   "JavaScript Promises." W3schools. https://www.w3schools.com/js/js_promise.asp (accessed Dec. 2023).

[21]   L. Criswell. "Destructuring Props in React." Medium. https://medium.com/@lcriswell/destructuring-props-
         in-react-b1c295005ce0 (accessed Dec. 2023).

[22]   "Create React App." React. https://create-react-app.dev/ (accessed Dec. 2023).

[23]   "Learn React." Codecademy. https://create-react-app.dev/ (accessed Nov. 2023).

[24]   "JSX." Codecademy. https://www.codecademy.com/learn/react-101/modules/react-101-jsx-u/cheatsheet
         (accessed Nov. 2023).

[25]   E. Herrera. "useState in React: A complete guide." https://blog.logrocket.com/guide-usestate-react/
         (accessed Dec. 2023).

[26]   "Callback function." mdn web docs." https://developer.mozilla.org/en-US/docs/Glossary/Callback_function
         (accessed Dec. 2023).

[27]   "Thinking in React." React. https://react.dev/learn/thinking-in-react (accessed Dec. 2023).

[28]   Jerry. "Thinking in React (Visualized)." Dev. https://dev.to/jareechang/thinking-in-react-visualized-g4p
         (accessed Jan. 2024).

[29]   "React Fundamentals." Make school. https://makeschool.org/mediabook/oa/tutorials/react-fundamentals-
         vm0/thinking-in-components/ (accessed Jan. 2024).

[30]   A. Singh. "Thinking in React – few tips." Medium. https://amnsingh.medium.com/thinking-in-react-few-
         tips-6b32fbe835a3 (accessed Jan. 2024).

[31]   A. Crosson. "Extract Subject Matter of Documents Using NLP." Medium.
         https://medium.com/@acrosson/extract-subject-matter-of-documents-using-nlp-e284c1c61824 (accessed
         Jan. 2024)

APPENDIX

LOGIC

e) *Backend* – The backend will be implemented with NodeJS with a database using MongoDB. The user case and logic are below, divided into pages.

I. *MAIN MENU*

User Story: The user will click on the buttons under the menu redirect to the respective pages.

II. *PUZZLE CREATION PAGE*

1) *Logic:* Topic names will be pulled from the database and input into the topics dropdown. Once a topic is clicked on, the difficulties associated with that topic will be pulled from the object in the database and input into the difficulties dropdown. When the 'Generate' button is clicked, the following will occur:

1. The words in the chosen topic will be separated into different linked lists that correspond to the number of characters in the word.

2. The difficulty chosen will determine which linked lists to combine.

   a. Easy: 3-5 letter words

   b. Intermediate: 3-10 letter words

   c. Hard & Expert: 3-15 letter words

   Each linked list corresponding to the number of needed letters will be combined into a single linked list, called *final*.

3. A function using random will choose which indices of *final* will be displayed in the puzzle. The number of words to choose will depend on the difficulty. Let's call this number *needed*.

   a. Easy: choose 10 words

   b. Intermediate: choose 20 words

   c. Hard: choose 30 words

   d. Expert: choose 40 words

      • Create a function that generates a random number between 0 to len($final$) – 1. Let this number be called *random*.

- Create a new linked list, ***words***.

- Let *i* be the number of words that have already been chosen. For *i* = 0, *i* <

  needed, i+1: Generate *random* to determine which index within *final* contains

  the chosen word. Add the string of the element at *final[random]* to *words*.

  Remove *final[random]* from linked list and repeat until the for loop is complete.

4. A square 2-dimensional array of size of the ceil((max number of letters for the

corresponding difficulty) x 1.5) will be established to create the array size. This array will

be initially filled with 0's and called ***puzzle***.

   a. Easy: [8][8]

   b. Intermediate: [15][15]

   c. Hard & Expert: [23][23]

5. A set will be created to store the letters used in the puzzle. This will be called ***letters***.

6. A linked list will be created to store the directions for words. This will be called

   directions.

   a. [North, Northeast, East, Southeast, South, Southwest, West, Northwest]

7. An object of strings will be created to designate the opposite of each direction. This will

be called ***oppositeDir***.

   a. *oppositeDir = {*
      N: South
      NE: Southwest
      E: West
      SE: Northwest
      S: North
      SW: Northeast
      W: East
      NW: Southeast
      }

8. An object of booleans representing the directions occupying each index will be created.

   Let this be ***availability***.

   a. *availability = {*
      N: False
      NE: False
      E: False
      SE: False
      S: False
      SW: False

W: False
NW: False
}

9. An array with the same dimensions as *puzzle* will be created with each index storing an initial *availability* object to represent which directions are occupied by the space. This array will be known as ==*availArr*==.

10. A random number generator will be created between the numbers 1-1 to 8-1 (or 0 to 7) to determine an index in *directions,* which will be the direction the chosen word will take. This will be known as ==*dir*==.

    Ex: dir -> directions[4] = South

11. A random number generator will be created between the numbers 1-1 (0) to the width of *puzzle* – 1.

    a. Easy: 0-7

    b. Intermediate: 0-14

    c. Hard & Expert: 0-22

    Two numbers from the generator will be created to determine the placement of the first letter at the beginning of each word. Let these numbers be designated ==*row*== and ==*column*==. The resulting starting index will be *[row][column]*.

12. A counter will be created to count the number of letters intersected in the same direction. This will be called ==*intersected*==.

13. The number of letters will be stored in a variable, ==*lett*==.

14. First, *puzzle* needs to be checked at *[row][column]* to ensure the starting spot is empty. If the value is 0, then the next letter in the direction will be checked. The indices of *[row][column]* will change by direction as follows, where *i* is the index of the character in the word:

    a. North: [row-*i*][column]

    b. Northeast: [row-*i*][column+*i*]

    c. East: [row][column+*i*]

    d. Southeast: [row+*i*][column+*i*]

e. South: [row+*i*][column]

f. Southwest: [row+*i*][column-*i*]

g. West: [row][column-*i*]

h. Northwest: [row-*i*][column-*i*]

For (*j*=0, *j* < *lett*, *j*++), *j* will be input for *i* above. There will be a check to make sure each index is >= 0 and < width of *puzzle*. If both constraints are not met, remove *dir* from *directions*. Regenerate a direction from a random number generator from 0 to the size of *directions*-1.

If both constraints are met:

- Check if *puzzle[newRowIndex][newColumnIndex]* is 0.
    - If it is, move on to the next letter.

      If it is not, check to see if the value of *puzzle[newRowIndex][newColumnIndex]* is the same as *l*.
        - If it is, check the directions stored in *availArr[newRowIndex][newRowColumn]*. If *dir* or *oppositeDir.dir* are marked as T, then increment *intersected*.
- Repeat the blue for the remainder of the letters in the word.
    - If *intersected* > 1, remove *dir* from *directions*. Regenerate a direction from a random number generator from 0 to the size of *directions*-1.
- If *directions* ends up empty, then regenerate the starting index and restart at the beginning of *12*.

If all the letters are successfully checked without constraint issues or *intersected* > 1, then replace the 0's in *puzzle* with the corresponding letters and update the same indices in *availArr* with the value *dir* as *True*. As each letter is placed, add the letter to the set *letters*.

15. Repeat for all elements in *words*.
16. Once all words are placed, create an array copy of *letters.*

17. Check the value of all indices in *puzzle*. If any indices have a value of 0, generate a random number from 0 to the length of the array of *letters*. Choose a random index in *letters* to replace the 0.

V) PREVIEW TOPICS PAGE

i. *User Story:* Users will see a displayed list of topics available in the database. Once a topic is clicked, all words associated with that topic in the database will preview on the right. From there, the user can choose to go back to the puzzle generator or to create a new topic. Edit an existing topic will come at a later step.

VI) CREATE YOUR OWN TOPIC PAGE

i. *Logic:* The user entry fields can be filled in any order. An onChange event will check the fields as characters are entered in both the title entry and word entry to make sure they meet requirements. If an illegal character is detected, a red dynamic error message will pop up as well as highlighting the entry box in red. Illegal entries are as follows:

1. *Title Name*

    a. More than 75 characters.

    b. Symbols or characters not in the English alphabet.

2. *Word Entry*

    a. Words less than 3 characters.

    b. Words more than 15 characters.

    c. Spaces.

    d. Symbols or characters not in the English alphabet.

    e. Words will not be recognized as separate unless they are entered on a new line.

The onChange will also update the status on the right as words are entered. There will be a counter for words between 3-7 characters, 8-10 characters, and 11-15 characters. Any time a character is entered, the number of characters in each word is re-evaluated. To count each difficulty, three separate equations will be used to determine the difficulty of words. 3-7 words will always remain stagnant for easy mode. If 8-10 characters is greater than 0, then intermediate level will take the number of *8-10 characters + 3-7 characters*. Likewise, if 11-

15 characters is greater than 0, then the hard & expert level will take the number of *11-15 characters + 8-10 characters + 3-7 characters*. As this is being calculated, the current words will be separated and rendered into the 'Entered Words' sections.

5) *Part 4 – Additional Feature Implementation:* A page that will read what is currently in the database and alter the contents will be implemented in the form of an Edit Existing Topic page.

    a) *Logic:* When the page is loaded, the topic component will load the list of topics available in the database. No other component will show information at this time. Once a topic is clicked, the database will be queried as to the word list, and print it out in a string to the text box below. The requirements will be a reused component as will the available difficulties and words list from the 'Create Your Own Topic' page. This page works by making a call to the database. The database is *not* changed until changes are submitted. At this point, the current element will be removed and re-uploaded into a new object.

    To edit the topic name, the words given a topic will be saved. Since topic names need to be unique, the original topic will be removed from the database and re-added with the new topic name. Deleted topics will be removed entirely from the database. Before editing and/or deleting a topic, a pop-up will display asking if you're sure before completing the operation.

REFERENCES

[1]   P. S. Minhas, A. Ghosh, and L. Swanzy, "The effects of passive and active learning on student preference and performance in an undergraduate basic science course," *Anatomical Sciences Education*, vol. 5, no. 4, pp. 200-207, Mar. 2012, doi: 10.1002/ase.1274.

[2]   L. Nirmal, M. MS, M. Prasad, "Use of Puzzles as an Effective Teaching–Learning Method for

Dental Undergraduates," *International Journal of Clinical Pediatric Dentistry,* vol. 13, no. 6, pp. 606-610, Nov-Dec 2020, doi: 10.5005/jp-journals-10005-1834.

[3]   A. J. Magana, C. Vieira, M. Boutin, "Characterizing Engineering Learners' Preferences for Active and Passive Learning Methods," *IEEE Transactions on Education,* vol. 61, no. 1, pp. 46-54, Sept. 2017, doi: 10.1109/TE.2017.2740203.

[4]   R. Howell, "Engaging students in education for sustainable development: The benefits of active learning, reflective practices and flipped classroom pedagogies," *Journal of Cleaner Production,* vol. 325, Nov 2021, doi: 10.1016/j.jclepro.2021.129318.

[5]   H. Brooker, et al., "An online investigation of the relationship between the frequency of word puzzle use and cognitive function in a large sample of older adults," *International Journal of Geriatric Psychiatry,* vol. 34, no. 7, pp. 921-931, July 2019, doi: 10.1002/gps.5033.

[6]   T. Fitria, "Using word search puzzle in improving students' English vocabulary: A systematic literature review," *Journal of English in Academic and Professional Communication,* vol. 9, no. 2, pp. 53-71, July 2023, doi: 10.25047/jeapco.v9i2.3927.

[7]   N. Kalkan, S. Güler, H. Bulut, A. Ay, "Views of students on the use of crossword and word search puzzle as a teaching technique in nursing education: A mixed-method study," *Nurse Education Today,* vol. 119, Sept. 2022, doi: 10.1016/j.nedt.2022.105542.

[8]   J. M. Leach, M. Mancini, J. A. Kaye, T. L. Hayes, F. B. Horak, "Day-to-Day Variability of Postural Sway and Its Association With Cognitive Function in Older Adults: A Pilot Study," *Frontiers in Aging Neuroscience,* vol. 10, May 2018, doi: 10.3389/fnagi.2018.00126.

[9]   M. Jones. Word Search Labs. https://wordsearchlabs.com/ (accessed Jan. 2024).

[10]   "Word Search Maker." Worksheet Zone. https://worksheetzone.org/worksheet-maker#create (accessed Jan. 2024).

[11]   "Word Search." Education.com. https://www.education.com/worksheet-generator/reading/word-search/ (accessed Jan. 2024).