KOLEJ PROFESIONAL MARA BERANANG

DIPLOMA IN COMPUTER SCIENCE

COURSE NAME	:	**OBJECT ORIENTED PROGRAMMING**
COURSE CODE	:	**CSC2744**
ACADEMIC SESSION	:	**SESSION 1 2023/2024**
TYPE OF ASSESSMENT	:	**FINAL ASSIGNMENT**
DURATION	:	**20/6/2023-10/07/2023**

CLO 3: Employ third party data in object oriented application development using graphical user interface (GUI) application framework

INSTRUCTION TO CANDIDATES:

1. Late submissions after given due date will not be accepted.
2. Report should be written using:
   Font type: Arial
   Size: 12 pts
   Line Spacing: 1.5
3. Coding format:
   Font type: Consolas
   Size: 10 pts
   Line Spacing: Single

Personal Details	
Name	DYAN BINTI AZIZOL
I/D Number	BCS2207-069
Class	DCS 4A
Lecturer	PUAN NINI ANIZA

Section / Question No.	**Marks**
Total	**/ 50**

Question

Electron is a powerful framework that enables developers to create cross-platform applications using web technologies such as HTML, CSS, and JavaScript. You need to choose one of the applications below to develop a desktop application using Electron that integrates the given API. The application needs to be developed with specific requirements or functionalities.

Name of Application	Description	Requirements
Dictionary and Thesaurus	An app that lists words in groups of synonyms and related concept.	• Word search. • Information Output: give the meaning of the searched word for different part of speech (noun/adjective), antonyms and the example of word usage, sounds and related URL for the searched word. • CRUD words of the day https://api.dictionaryapi.dev/api/v2/entries/en/digital
Meal planner	An app that displays suggestion of recipe based on food item entered by user	• Suggest recipe based on food item. • Information Output: One recipe suggestion that comprises of ingredients, instruction on how to cook and URL of the related site for the food and the link on how to prepare the food. • CRUD meal planner https://www.themealdb.com/api/json/v1/1/search.php?s=shawarma
Makeup Box	An app that finds makeup products based on brand and category entered.	• Display the product info according to search criteria. • Information Output: product description based on brand and name, product image, product website and related link for the searched product. • CRUD makeup top 5 list http://makeup-api.herokuapp.com/api/v1/products.json?brand=maybelline

Tasks:

1. Create desktop app using electron and apply third party data fetched from API and the requirements given. Your application should have at least 2 pages and you may add extra functionality or features of your choice to the application.
2. Implement CRUD (create, read, update, delete) process to the application as given in the requirements.
3. Apply HTML and CSS for user interface and provide evidence for application.
4. GUI Elements:
   i. Apply GUI elements that assist users in using application.
   ii. The application's 'look and feel' is attractive and informative.
5. Produce a report on your application functionalities and features. Include the following:
   i. Overview of your application with a brief description.
   ii. Screenshots of the application with explanations on how to use it.
   iii. Program codes of your system
6. Submit files in GitHub.

Assessment Rubric

ATTRIBUTES	CRITERIA	POOR (1 mark)	FAIR (2 marks)	GOOD (3 marks)	VERY GOOD (4 marks)	EXCELLENT (5 marks)	Mark Obtaine d
Reproduce and Process Information	1. Create desktop app using electron and apply third party data fetched from API and the requirements given. You may add extra functionality or features of your choice to the application.	The application is an extensive collection and rehash of other people's ideas, products, and images. There is no evidence of new thought.	The application is somewhat a collection and rehash of other people's ideas, products, and images. There is little evidence of new thought or inventiveness.	The application is a minimal collection or rehash of other people's ideas, products, and images. There is a few evidence of new thought or inventiveness.	The application shows a lot of evidence of originality and inventiveness.	The application shows significant evidence of originality and inventiveness. Most of the content and many of the ideas are fresh,original, and inventive.	
		Able to display part of the data from the API and does not fulfill the requirements.	Able to display sufficient data from the API that meet with the requirements together with the description.	Able to display sufficient data from the API that meet with the requirements together with the description.	Able to display extra data from the API beyond the application requirements together with no description.	Able to display extra data from the API beyond the application requirements together with the description.	
		The data from the API does not reflect the whole purpose of the application developed.	The data from the API is sufficient but does not reflect the whole purpose of the application developed.	The data from the API is meaningful but does not reflect the purpose of the application developed.	The data from the API is meaningful and reflect the purpose of the application developed.	The data from the API is meaningful to come up with extra idea for the application developed.	
		The developed application does not fulfill the requirements stated for the chosen	The developed application fulfills all the requirements stated for the chosen application with	The developed application fulfills all the requirements stated for the chosen application.	The developed application fulfills all the requirements stated for the chosen application.	The developed application fulfills all the requirements stated for the chosen application that	

		application.	no extra functionalities.	Add extra functionality or features to the application.	Add extra functionality or features to the application that enhances the user experience or adds value to the application.	utilizes the data from the API Add extra functionality or features to the application that enhances the user experience or adds value to the application.	
	2. Implement CRUD (create, read, update, delete) process to the application as given in the requirements.	• Able to create only 2 of the CRUD processes according to the requirements. • No feedback for the CRUD process. • The design for data input is poor	• Able to create only 3 of the CRUD processes according to the requirements. • No feedback for the CRUD process. • The design for data input is good with some room for improvement	• Able to perform all the CRUD processes according to the requirements. • No feedback for the CRUD process. • Well-designed data input for CRUD process.	• Able to perform all the CRUD processes according to the requirements. • No feedback for the CRUD process. • Well-designed data input for CRUD process.	• Able to perform all the CRUD processes according to the requirements. • Appropriate feedback for the CRUD process. • Well-designed and user-friendly data input for CRUD process.	
	3. Apply HTML and CSS for user interface and provide evidence for application.	• **Text -** All text used is too small to view or the font type is wrongly chosen. • **Graphics** - Graphics seem randomly chosen, are of low quality, OR distract the reader.	• **Text** – Some of the text used is too small to view or the font type is wrongly chosen. • **Graphics -** Graphics seem randomly chosen, are	• **Text** - Most text used is clear but does not describe the content well. • **Graphics** - Graphics are related to the theme/purpose of the application	• **Text** - All text used is clear but does not describe the content well. • **Graphics -** Graphics are related to the theme/purpose of the	• **Text** - All text used is clear and able to describe the content well. • **Graphics** - Graphics are related to the theme/purpose of the application, are thoughtfully	

			of low quality, OR distract the reader.	and are of excellent quality.	application, are of excellent quality and enhance reader interest or understanding	cropped, are of high quality and enhance reader interest or understanding.	
Curate	4.GUI Elements: i. Apply GUI elements that assist users in using application. i.	Not able to curate for required content. • **Layout** - The HTML elements in the application are cluttered looking or confusing. • **Navigation** Links do not take the reader to the sites/ pages described. User typically feels lost.	Limited curation for required content. • **Layout** - The HTML elements in the application is messy, may appear busy or boring. • **Navigation** Links seem to be missing and don't allow the user to easily navigate.	Satisfactory curation for required content. • **Layout** - The HTML elements are suitable. • **Navigatio n** Links allow the reader to move from page to page, but some links seem to be missing.	Good curation for required content. • **Layout** - The HTML elements are suitable and usable. • **Navigation** Links are labelled and allow the user to easily move from page to page.	Excellent curation for required content. • **Layout** - The HTML elements are well structured, attractive, and usable layout. • **Navigation** Links are clearly labelled, consistently placed, and allow the user to easily move from page to page.	
	ii. The application's 'look and feel' is attractive and informative.	• The application is in need of polish in its visual design and is not appropriate for the target audience. • **Color**	• The application is in need of polish in its visual design, but it is still appropriate for the target audience. • **Color**	• The application mostly follows good visual design principles (e.g.: alignment, contrast,	• The application demonstrate s good visual design principles (e.g.: alignment, contrast,	• The application clearly demonstrate s good visual design principles (e.g.: alignment,	

		Choice of colors and combinations are not suitable.	Choice of colors and combinations do not match the concept of the application.	easily read text) and is appropriate for the target audience. • **Color** Choice of colors and combinations match the concept of the application.	easily read text) and is appropriate for the target audience. • **Color** Appropriate colors used to produce an atmosphere that expresses the concept of the application.	contrast, easily read text) and is appropriate for the target audience. • **Color** Appropriate colors used to produce an atmosphere that expresses the concept of the application.	
Convey	5. Produce a report on your application functionalities and features that includes: i. Overview of the application. ii. Screenshots of the application with explanations on how to use it.	The overview of the application is vague. The user guide is incomplete and cannot be recognized as a user guide.	The overview of the application is very brief and does not describe the whole functionalities of the application. The user guide provides limited information with no screenshots of the application.	The overview of the application is clearly described the whole application and its functionalities. The user guide provides basic information with limited screenshots of the application.	The overview of the application is clearly described the whole application and its functionalities. The user guide provides adequate information with complete screenshots of the application.	The overview of the application is clearly described the whole application and its functionalities. The user guide provides extensive information with complete screenshots and labelling of the application.	
	iii. Program codes of the system	HTML, CSS and JavaScript codes attached are not complete. The codes are hardly read.	HTML, CSS and JavaScript codes attached are complete. The codes are hardly read.	HTML, CSS and JavaScript codes attached are complete. The codes are readable but not organized.	HTML, CSS and JavaScript codes attached are complete. The codes are readable and	HTML, CSS and JavaScript codes attached are complete and include comments for the important parts of the codes.	

		Does not submit complete electron files in GitHub	Completely submit all the electron file in GitHub.	Completely submit all the electron file in GitHub.	organized. Completely submit all the electron file in GitHub.	The codes are readable and organized. Completely submit all the electron file in GitHub.	
Total Marks Earned							**/50**
Total Percentage (40%)							**/40%**

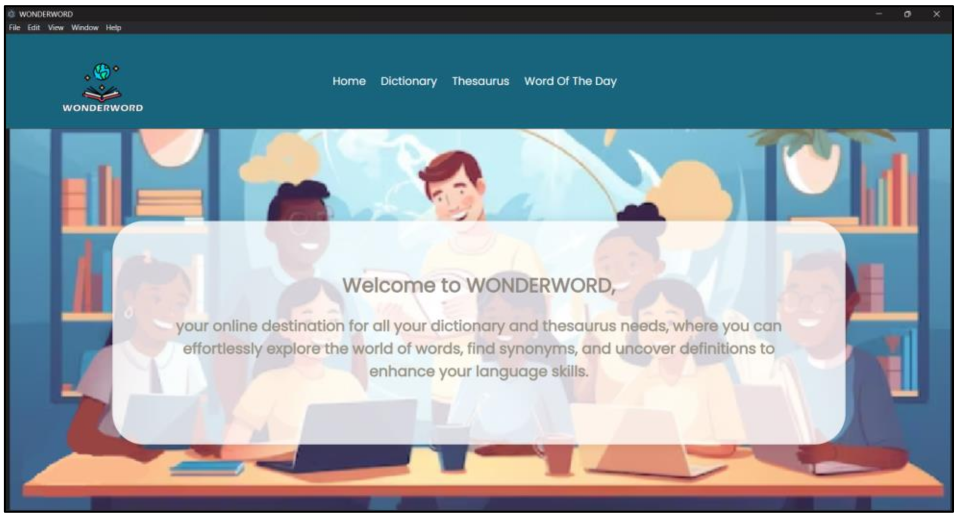
I. Overview of the application.

“Wonderword” is a user-friendly system that empowers to delve into world of words,discover synonyms and antonyms and unveil definitions to enhance language skills.This platform boasts two main features: firstly: it enables users to explore words in the dictionary or thesaurus.After searching for a word,users can seamlessly add their searched word to the Word Of The Day.Secondly,it allows users to perform essential operations such as create,read,update and delete entries of Word Of The Day.In conclusion, “Wonderword” serves as a user-friendly tool for language enthusiasts,providing a seamless and enriching experience.

II. Screenshots of the application with explanations on how to use it (User Guide).

index.html

This is Home pages (index.html) where users are warmly welcomed. Besides,there’s a menu navigation to other pages.



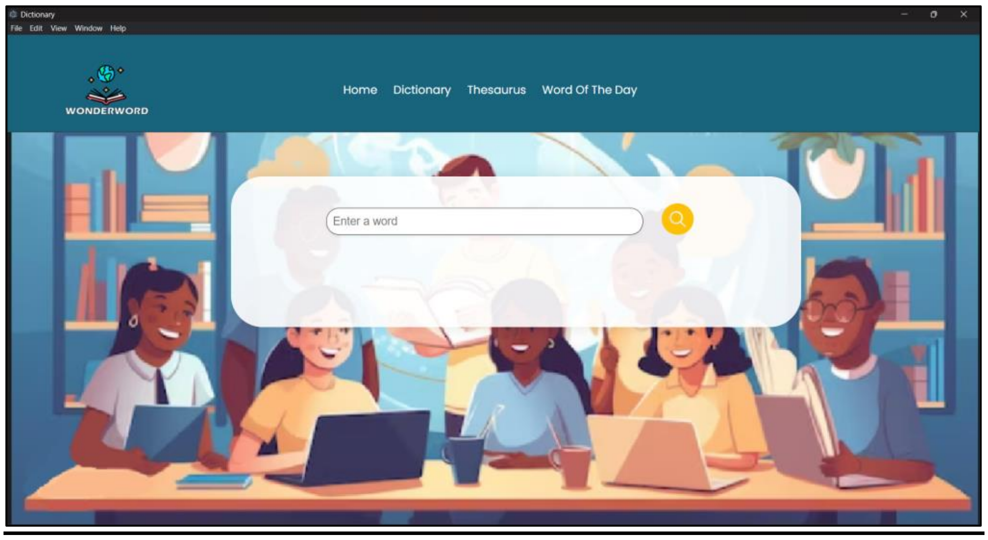
Welcome to WONDERWORD,

your online destination for all your dictionary and thesaurus needs, where you can effortlessly explore the world of words, find synonyms, and uncover definitions to enhance your language skills.

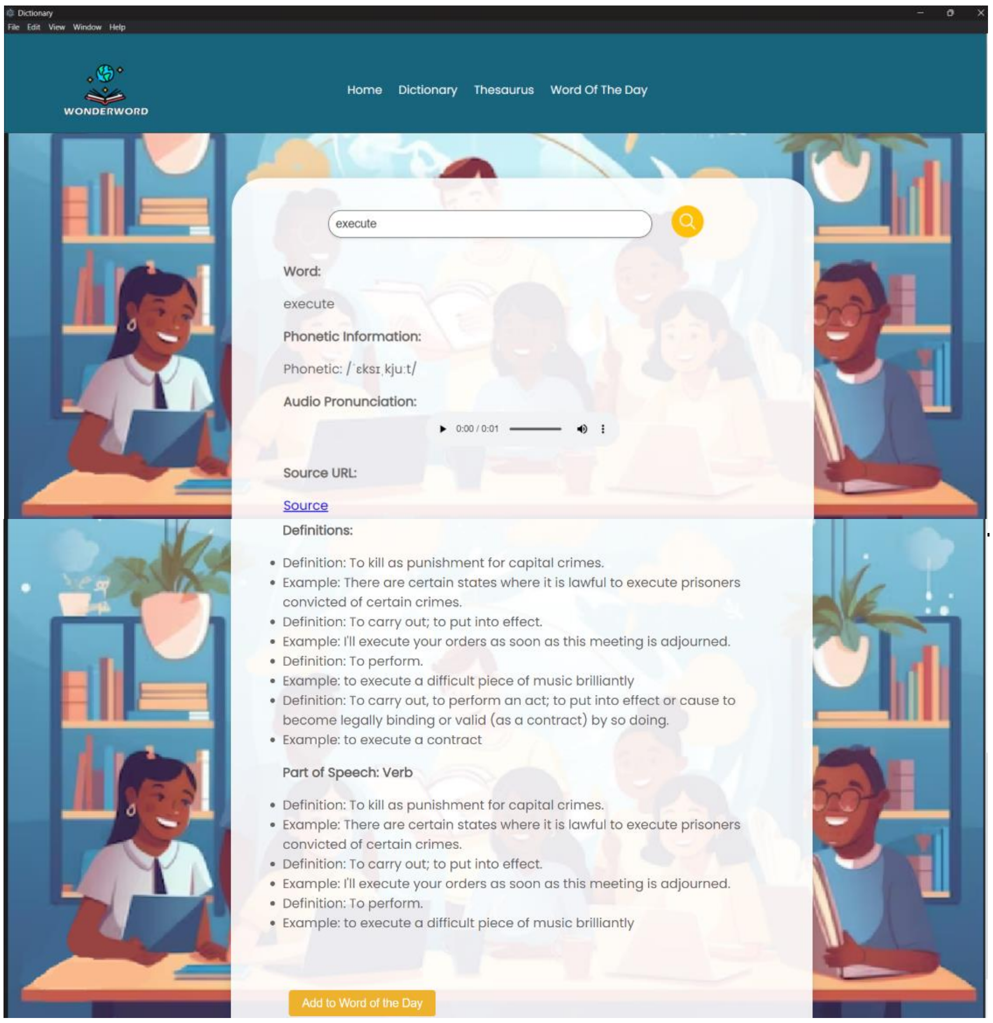
(Users click on ‘Dictionary’ navigation)

<u>dictionary.html</u>

This is Dictionary pages (dictionary.html) where users can search for any word they wish to explore. If users enter words and click on search icon,”ADD TO WORD OF THE DAY” button will appear.If users click on the ,”ADD TO WORD OF THE DAY” button, they will be directed to the wod.html, enabling users to add the searched word to the Word of The Day.



(Users enter word and click search icon)


Dictionary
File Edit View Window Help
WONDERWORD
Home Dictionary Thesaurus Word Of The Day
execute
0:00 / 0:01
Add to Word of the Day

Word:

execute

Phonetic Information:

Phonetic: /ˈɛksɪ kjuːt/

Audio Pronunciation:

Source URL:

Source

Definitions:

- Definition: To kill as punishment for capital crimes.
- Example: There are certain states where it is lawful to execute prisoners convicted of certain crimes.
- Definition: To carry out; to put into effect.
- Example: I'll execute your orders as soon as this meeting is adjourned.
- Definition: To perform.
- Example: to execute a difficult piece of music brilliantly
- Definition: To carry out, to perform an act; to put into effect or cause to become legally binding or valid (as a contract) by so doing.
- Example: to execute a contract

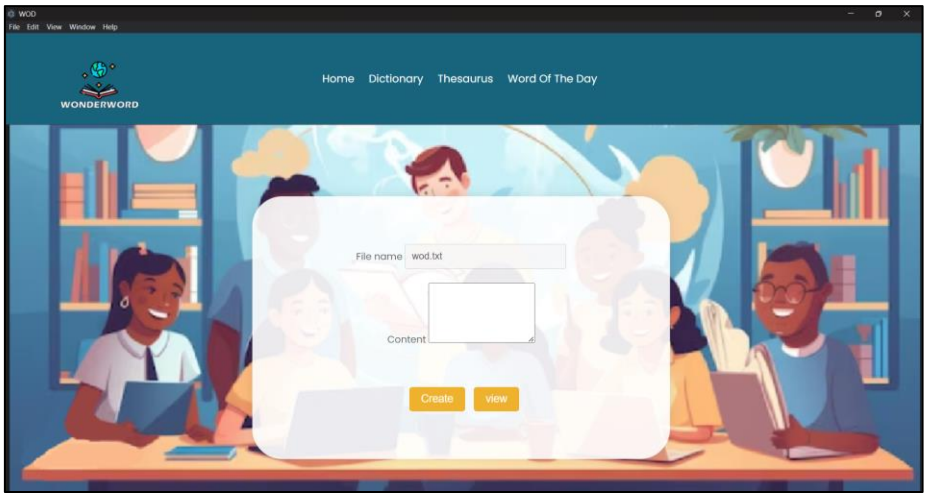
Part of Speech: Verb

- Definition: To kill as punishment for capital crimes.
- Example: There are certain states where it is lawful to execute prisoners convicted of certain crimes.
- Definition: To carry out; to put into effect.
- Example: I'll execute your orders as soon as this meeting is adjourned.
- Definition: To perform.
- Example: to execute a difficult piece of music brilliantly

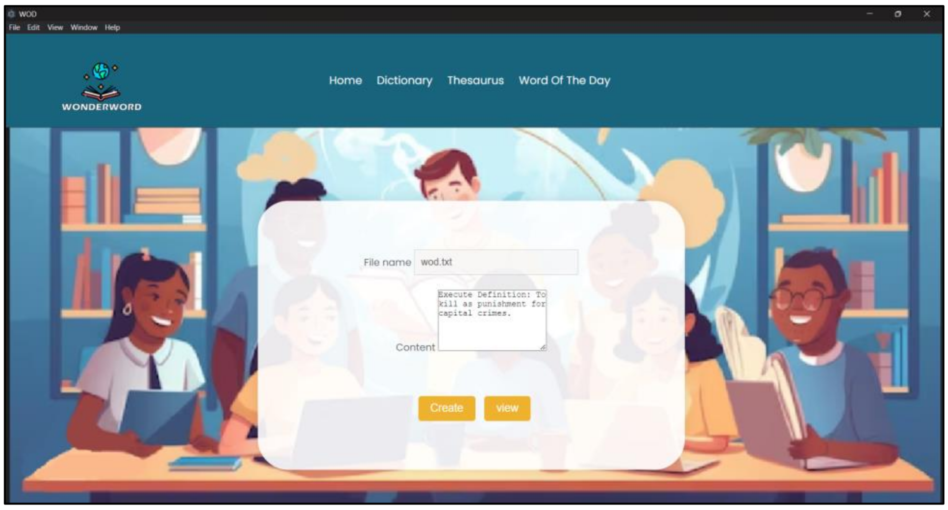
(Users click "ADD TO WORD OF THE DAY" button)

<u>wod.html</u>

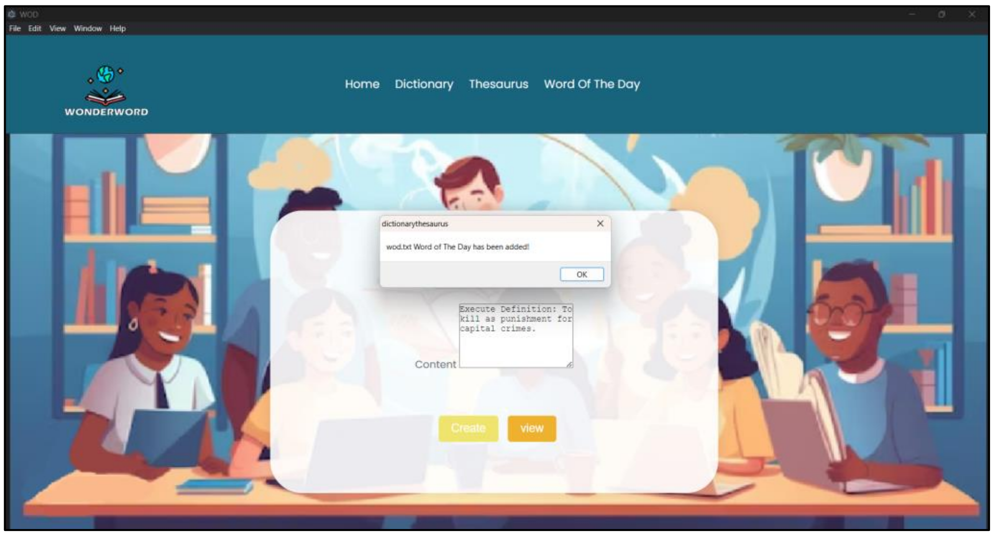
This is wod pages (wod.html) where users can add the searched word to the Word of The Day.If users click on the "Create" button,the word will be added to wod.txt files and popout messages will appear to confirm the successful addition. Besides that,if users click on the "view" button, they will be directed to the Word Of The Day page.



(Users add word and click "Create" button)



```
Execute Definition: To
kill as punishment for
capital crimes.
```



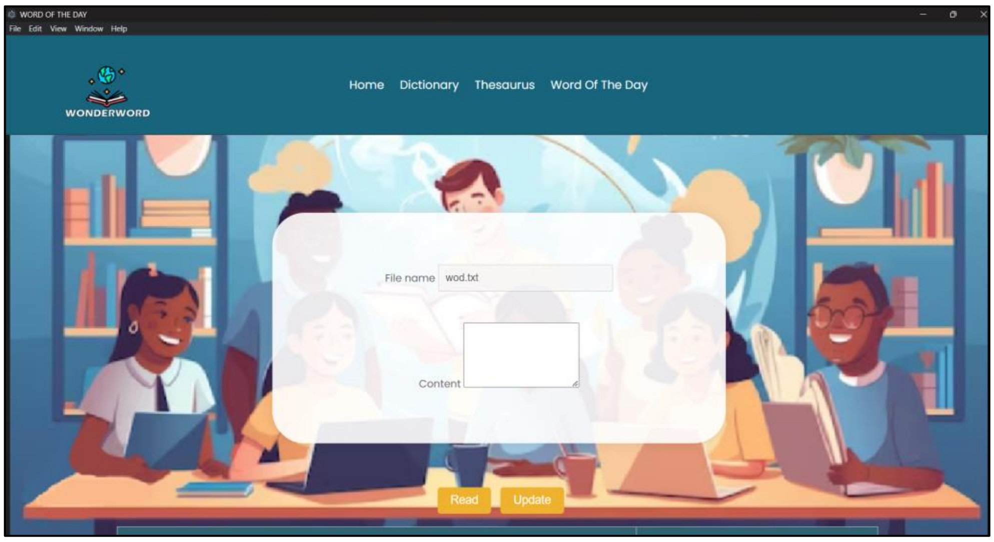
wod.txt Word of The Day has been added!

```
Execute Definition: To
kill as punishment for
capital crimes.
```

(Users click “view” button)

<u>CRUD.html</u>

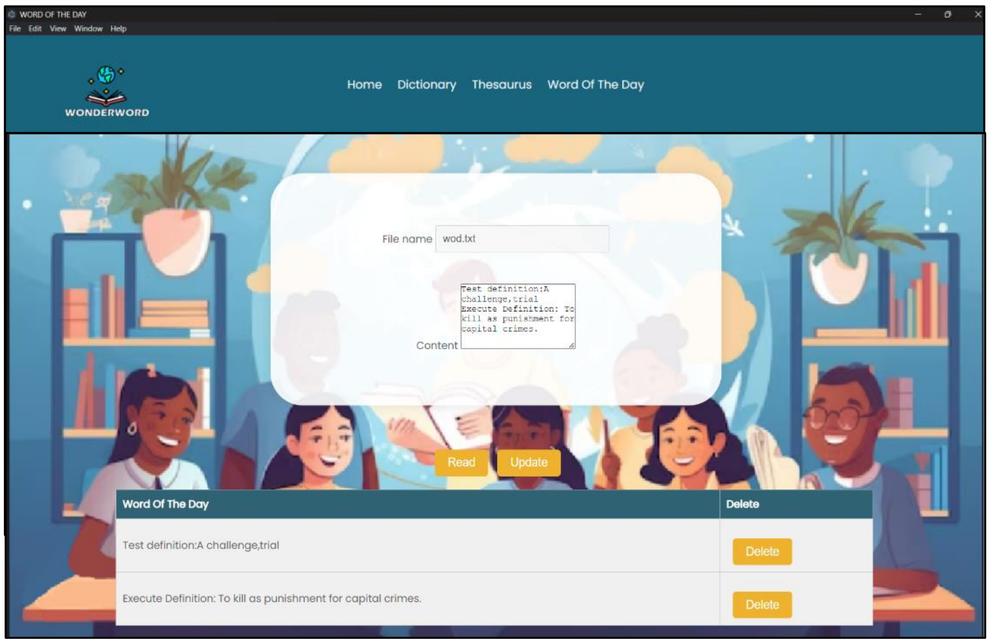
This is Word Of The Day pages (CRUD.html) where users can read,update and delete their Word Of The Day entries.If users click on the “Read” button,the word will be display in the Content form and table. Besides that,if users click on the “Update” button the word will be display on the Content form,allowing users to make updates and popout messages will appear to confirm the successful update . Apart from that, if users click on “delete” button, it will delete the selected word.



File name: wod.txt

Content:

(Users click “Read” button)



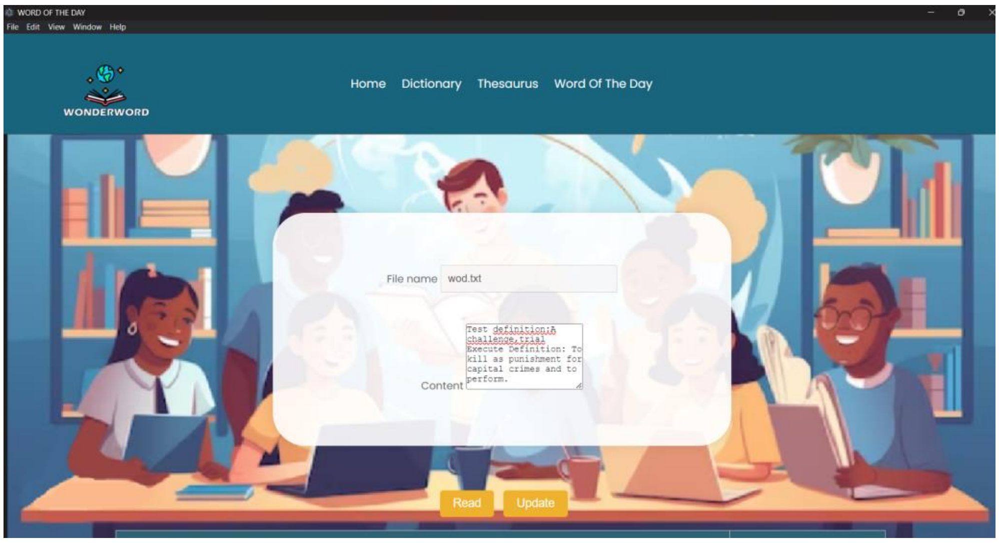
File name: wod.txt

Content:

```
Test definition:A
challenge,trial
Execute Definition: To
kill as punishment for
capital crimes.
```

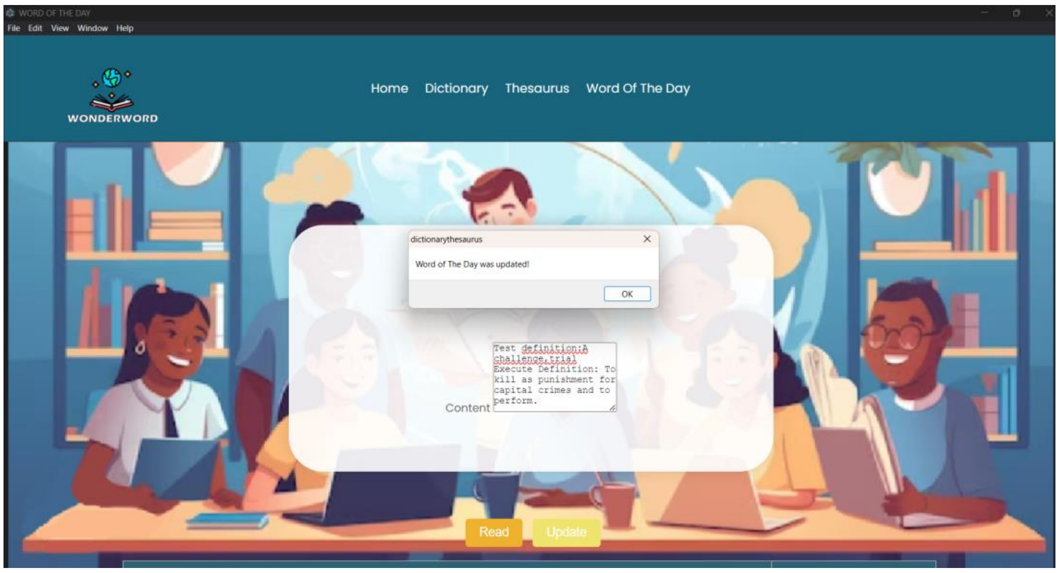
Word Of The Day	Delete
Test definition:A challenge,trial	Delete
Execute Definition: To kill as punishment for capital crimes.	Delete

(Users click “Update” button)



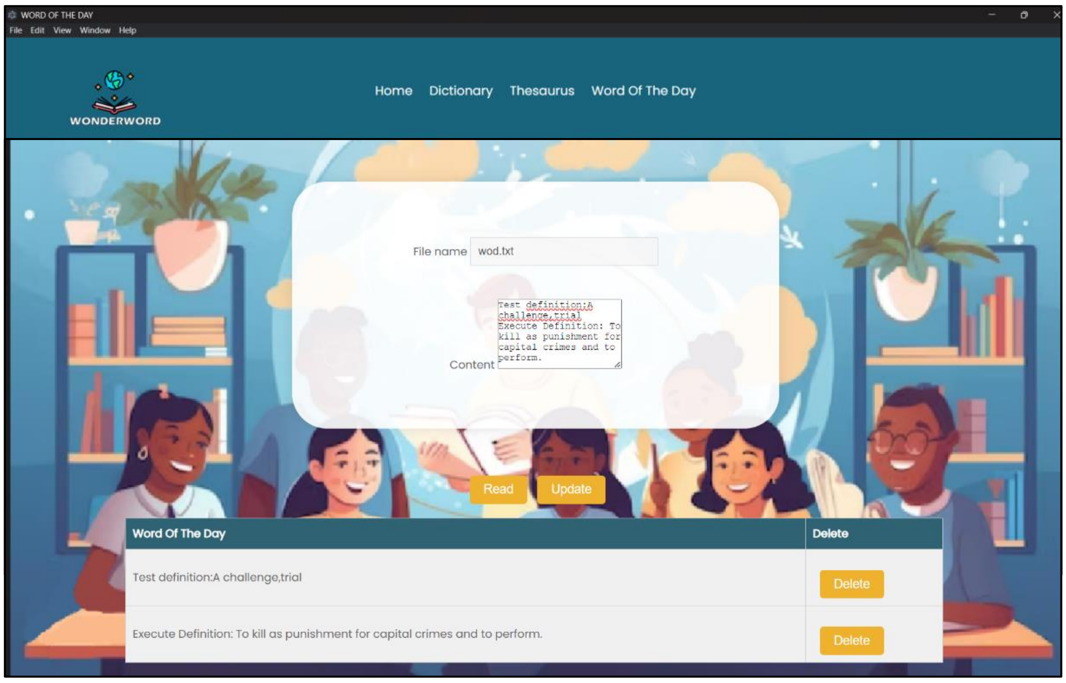
File name wod.txt

Content
```
Test definition:A
challenge,trial
Execute Definition: To
kill as punishment for
capital crimes and to
perform.
```



Word of The Day was updated!

Content
```
Test definition:A
challenge,trial
Execute Definition: To
kill as punishment for
capital crimes and to
perform.
```


WORD OF THE DAY
File Edit View Window Help
WONDERWORD
Home Dictionary Thesaurus Word Of The Day
File name
Content
Read
Update

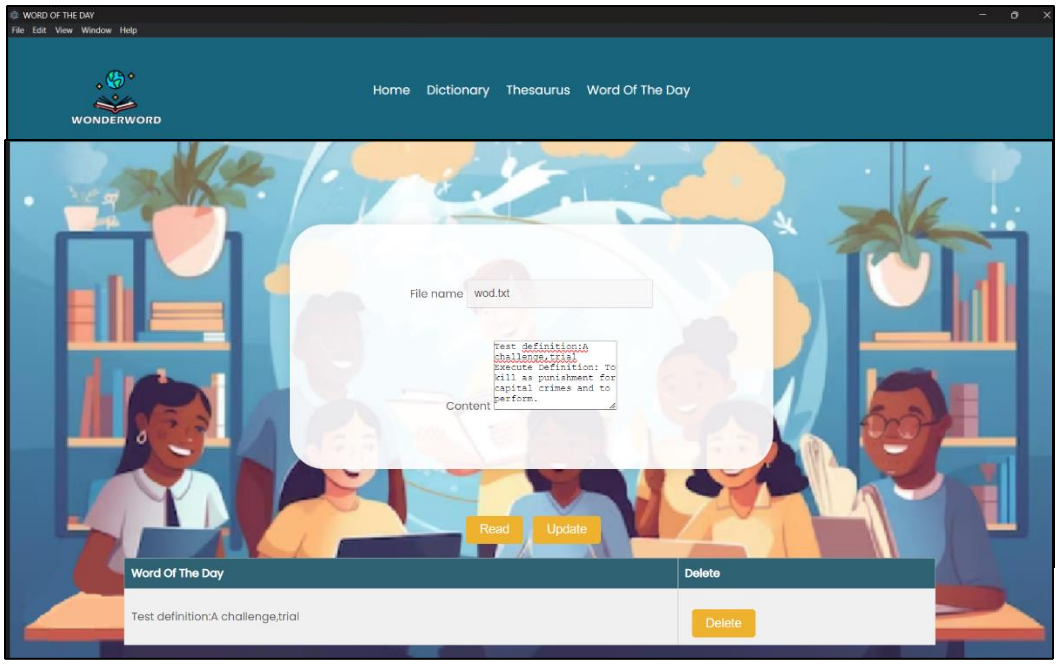
File name: wod.txt

Content:

```
Test definition:A
challenge,trial
Execute Definition: To
kill as punishment for
capital crimes and to
perform.
```

Word Of The Day	Delete
Test definition:A challenge,trial	Delete
Execute Definition: To kill as punishment for capital crimes and to perform.	Delete

(Users click "delete" button)


WORD OF THE DAY
File Edit View Window Help
WONDERWORD
Home Dictionary Thesaurus Word Of The Day
File name
Content
Read
Update

File name: wod.txt

Content:

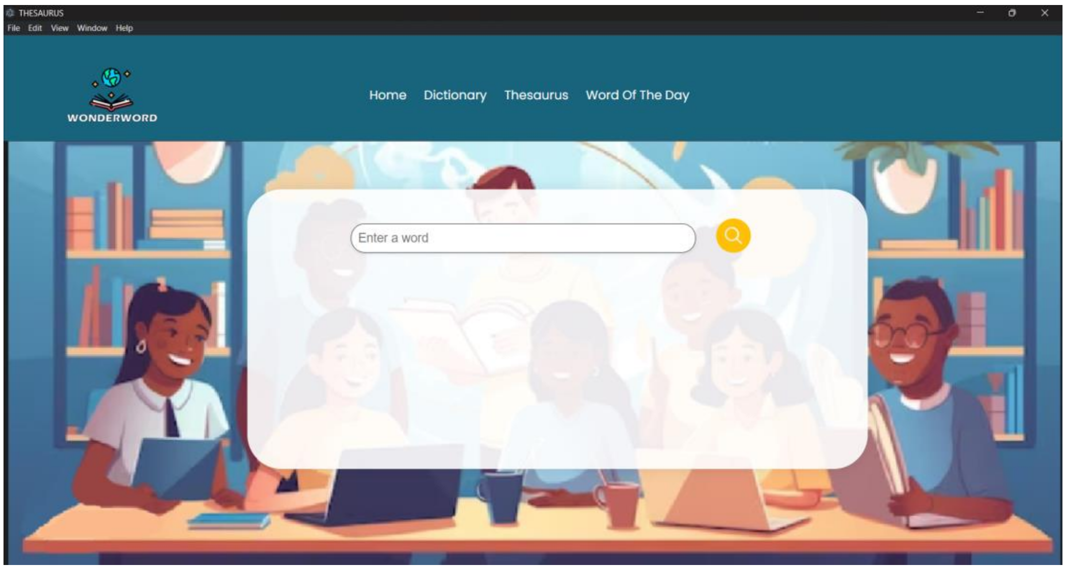
```
Test definition:A
challenge,trial
Execute Definition: To
kill as punishment for
capital crimes and to
perform.
```

Word Of The Day	Delete
Test definition:A challenge,trial	Delete

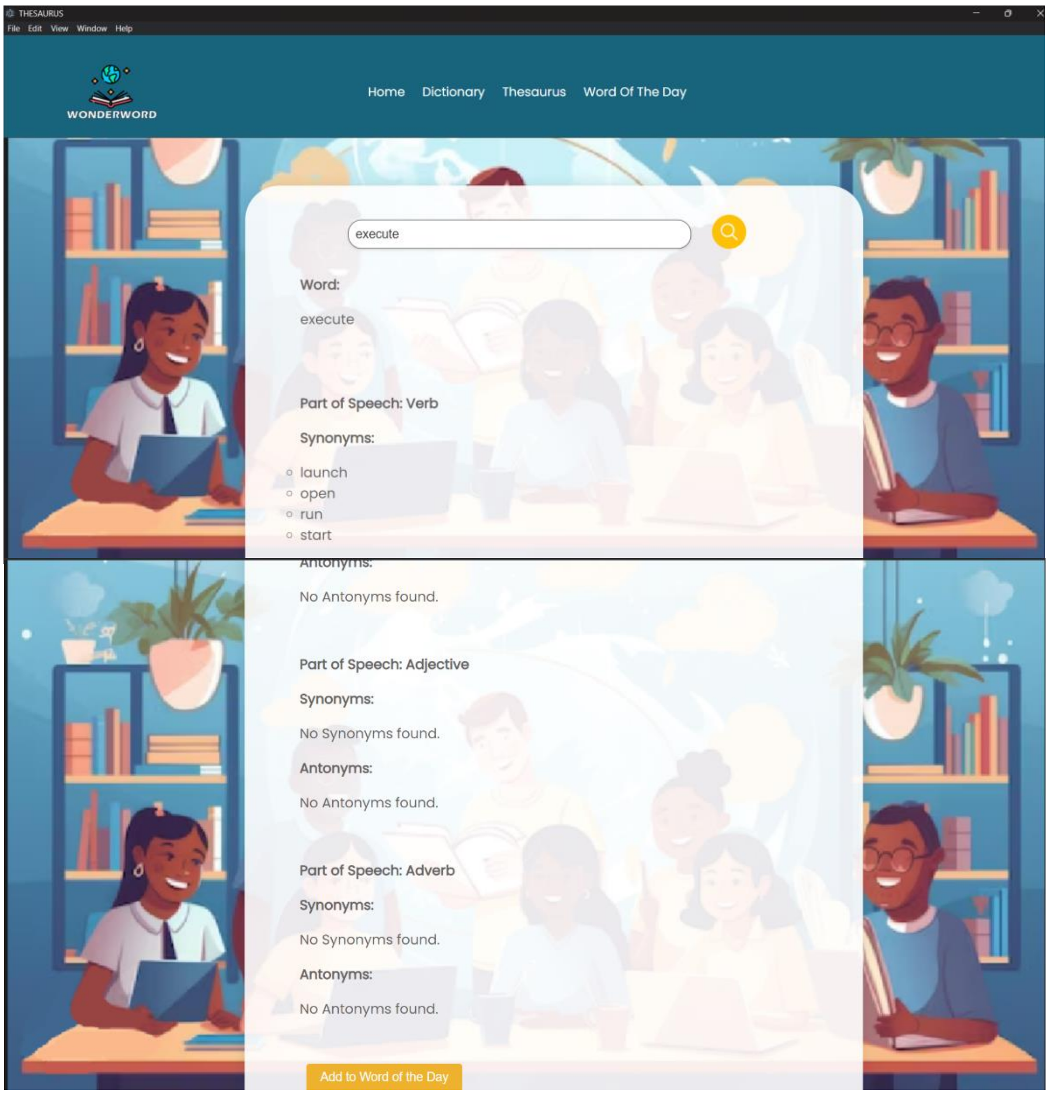
(Users click on 'Thesaurus' navigation)

thesaurus.html

This is Thesaurus pages (thesaurus.html) where users can search for any word they wish to explore. If users enter words and click on search icon,"ADD TO WORD OF THE DAY" button will appear. If users click on the ,"ADD TO WORD OF THE DAY" button, they will be directed to the wod.html, enabling users to add the searched word to the Word of The Day.



(Users enter word and click search icon)



execute

Word:

execute

Part of Speech: Verb

Synonyms:

- launch
- open
- run
- start

Antonyms:

No Antonyms found.

Part of Speech: Adjective

Synonyms:

No Synonyms found.

Antonyms:

No Antonyms found.

Part of Speech: Adverb

Synonyms:

No Synonyms found.

Antonyms:

No Antonyms found.

III. <u>Program codes of the system</u>

<u>index.js</u>

```
const { app, BrowserWindow } = require('electron');
const fs = require('fs')
const path = require('path')


// Handle creating/removing shortcuts on Windows when installing/uninstalling.
if (require('electron-squirrel-startup')) {
  // eslint-disable-line global-require
  app.quit();
}
```

```
const createWindow = () => {
  // Create the browser window.
  const mainWindow = new BrowserWindow({
    width: 800,
    height: 600,
    webPreferences: {
      nodeIntegration: true,
      contextIsolation: false,
    }

  });

  // and load the index.html of the app.
  mainWindow.loadFile(path.join(__dirname, 'index.html'));

  // Open the DevTools.
  //mainWindow.webContents.openDevTools();
};

// This method will be called when Electron has finished
// initialization and is ready to create browser windows.
// Some APIs can only be used after this event occurs.
app.on('ready', createWindow);

// Quit when all windows are closed, except on macOS. There, it's common
// for applications and their menu bar to stay active until the user quits
// explicitly with Cmd + Q.
app.on('window-all-closed', () => {
  if (process.platform !== 'darwin') {
    app.quit();
  }
});

app.on('activate', () => {
  // On OS X it's common to re-create a window in the app when the
  // dock icon is clicked and there are no other windows open.
  if (BrowserWindow.getAllWindows().length === 0) {
    createWindow();
  }
     });
```

preload.js

```
// See the Electron documentation for details on how to use preload scripts:
// https://www.electronjs.org/docs/latest/tutorial/process-model#preload-scripts
```

Index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">

    <!--TITTLE-->
    <title>Home</title>

    <!--CSS-->
    <link rel="stylesheet" type="text/css" href="home.css">

    <!--LINK FONTS-->
    <link href="https://fonts.googleapis.com/css?family=Poppins&display=swap"
     rel="stylesheet">

</head>
<body>

    <!--NAVIGATION-->
    <header class="top-nav">
        <img src="logo.png" class="logo">
            <nav>
                <ul>
                    <li><a href="#">Home</a></li>
                    <li><a href="dictionary.html">Dictionary</a></li>
                    <li><a href="thesaurus.html">Thesaurus</a></li>
                    <li><a href="CRUD.html">Word Of The Day</a></li>
                </ul>
            </nav>
    </header>
    <main>

        <!--CONTAINER-->
        <div class="welcome-container">
            <h1>Welcome to WONDERWORD,</h1>
            <br>

            <h2>your online destination for all your dictionary and thesaurus needs,
     where you can effortlessly explore the world of words, find synonyms, and
     uncover definitions to enhance your language skills.</h2>
        </div>
    </main>
</body>
</html>
```

home.css

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}



/*BODY SETTINGS*/
body {
  font-family: 'Poppins', Arial, sans-serif;
  margin: 0;
  padding: 0;
  background-image: url('background.png');
  background-size: cover;
  background-position: center;
  background-attachment: fixed;
  color: #67645f;
}

/*NAVIGATION SETTINGS*/
```

```
.top-nav {
  background-color: rgb(23, 100, 124);
  color: #fff;
  display: flex;
  align-items: center;
  padding: 20px;
  border: 1px solid rgb(59, 56, 56);
}

.top-nav nav ul {
  list-style: none;
  text-align: center;
  margin-right: 20px;
}

.top-nav nav ul li {
  display: inline;
  margin-right: 20px;
}

.top-nav nav ul li a {
  text-decoration: none;
  color: #fff;
  font-size: 18px;
  position: relative;
  transition: color 0.3s;
}

.top-nav img.logo {
  margin-right: 300px;
  width: 150px;
  margin-left: 60px;
}


.top-nav nav ul li a::after {
  content: "";
  display: block;
  width: 0;
  height: 2px;
  background-color: #fff;
  position: absolute;
  left: 0;
  bottom: 0;
  transition: width 0.3s, background-color 0.3s;
}

.top-nav nav ul li a:hover::after {
  width: 100%;
  background-color: rgb(255, 191, 0);
}


/*HEADER SETTINGS*/
header h1 {
  font-size: 36px;
  margin-top: 40px;
}


/*MAIN SETTINGS*/
main {
  width: 80%;
  margin: 0 auto;
  padding: 20px;
  text-align: center;
}


/*CONTAINER SETTINGS*/
.welcome-container {
  padding: 80px;
  border-radius: 50px;
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
  background-color: rgba(255, 255, 255, 0.8);
  margin-top: 130px;
```

```
}

.welcome-container h2 {
  color: #9b968a;
  font-size: 24px;
  margin-bottom: 20px;
}



/*FONTS SETTINGS*/
h1{
  color: #918a7e;
}
```

Index.css

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}



/*BODY SETTINGS*/
body {
  font-family: 'Poppins', Arial, sans-serif;
  margin: 0;
  padding: 0;
  background-image: url('background.png');
  background-size: cover;
  background-position: center;
  background-attachment: fixed;
  color: #67645f;
}



/*NAVIGATIONS SETTINGS*/
.top-nav {
  background-color: rgb(23, 100, 124);
  color: #fff;
  display: flex;
  align-items: center;
  padding: 20px;
  border: 1px solid rgb(59, 56, 56);
}

.top-nav nav ul {
  list-style: none;
  text-align: center;
  margin-right: 20px;
}

.top-nav nav ul li {
  display: inline;
  margin-right: 20px;
}

.top-nav nav ul li a {
  text-decoration: none;
  color: #fff;
  font-size: 18px;
  position: relative;
  transition: color 0.3s;
}

.top-nav img.logo {
  margin-right: 300px;
  width: 150px;
  margin-left: 60px;
}


.top-nav nav ul li a::after {
  content: "";
  display: block;
  width: 0;
  height: 2px;
  background-color: #fff;
  position: absolute;
  left: 0;
  bottom: 0;
  transition: width 0.3s, background-color 0.3s;
}

.top-nav nav ul li a:hover::after {
  width: 100%;
  background-color: rgb(255, 191, 0);
}



/*HEADER SETTINGS*/
header h1 {
  font-size: 36px;
  margin-top: 40px;
}



/*MAIN SETTINGS*/
main {
  width: 80%;
  margin: 0 auto;
  padding: 20px;
  text-align: center;

}



/*CONTAINER SETTINGS*/
.container {
  padding: 80px;
    border-radius: 50px;
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
    background-color: rgba(255, 255, 255, 0.9);
    margin-top: 50px;
    margin-left: 180px;
    max-width: 900px;
}



/*FONTS SETTINGS*/
h1{
  color: rgb(255, 255, 255);
}

.error {
  color: #d9534f;
}

/*SEARCH BAR SETTINGS*/
.search-container {
  display: flex;
  margin-top: -140px;
  justify-content: center;
  align-items: center;
  width: 600px;
}

.search-icon {
  width: 50px;
  height: 50px;
}

#searchData {
  flex: 1;
  width: 50px;
  padding: 10px;
  font-size: 18px;
  border: none;
  border-radius: 20px;
  background-color: #fff;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  margin-top: 20px;
  color: #333;
  border: 1px solid rgb(59, 56, 56);
  transition: background-color 0.3s, box-shadow 0.3s;
}

#searchData:hover {
  background-color: #f7f7f7;
}

#searchData:focus {
  background-color: #fff;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
  outline: none;
}

/*BUTTON SETTINGS*/
button {
  background-color:rgb(238, 178, 47);
  color: #fff;
  border: none;
  border-radius: 5px;
  padding: 10px 20px;
  font-size: 18px;
  margin-top: 20px;
  margin-left: 10px;
  cursor: pointer;
  transition: background-color 0.3s;
}

button:hover {
  background-color: rgb(239, 229, 110);
}

.button {
  border: none;
  border-radius: 60px;
  padding: 10px 20px;
  font-size: 18px;
  margin-top: 20px;
  margin-left: 10px;
  cursor: pointer;
  transition: background-color 0.3s;
}


/*INPUT FORM SETTINGS*/
input[type="text"] {
  width: 70%;
  padding: 10px;
  font-size: 16px;
}


/*FONTS SETTINGS*/
h2, h1, p ,h3 , ul{
  font-size: 20px;
  margin-top: 20px;
  text-align: left;
}


/*CONTAINER SETTINGS*/
#wordContainer,
#dictionaryMeaning,
#verbContainer,
#nounContainer,
#adjectiveContainer,
#synonymContainer,
#antonymContainer,
#synonymNounContainer,
#antonymNounContainer,
#synonymVerbContainer,
#antonymVerbContainer,
#synonymAdjectiveContainer,
#antonymAdjectiveContainer,
#addToWordOfDayContainer,
#Back,
#error {
  margin-top: 20px;
  text-align: left;
}
```

dictionary.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">


    <!--TITTLE-->
    <title>Dictionary</title>


    <!--CSS-->
    <link rel="stylesheet" type="text/css" href="index.css">


    <!--LINK FONTS-->
    <link href="https://fonts.googleapis.com/css?family=Poppins&display=swap"
     rel="stylesheet">


</head>
<body>


    <!--NAVIGATION-->
    <header class="top-nav">
        <img src="logo.png" class="logo">
            <nav>
                <ul>
                    <li><a href="index.html" >Home</a></li>
                    <li><a href="#">Dictionary</a></li>
                    <li><a href="thesaurus.html">Thesaurus</a></li>
                    <li><a href="CRUD.html">Word Of The Day</a></li>
                </ul>
            </nav>
    </header>


    <main>
```

```
   <div class="container">
   <br>
   <div class="search-bar">



       <br><br>


       <!--SEARCH BAR-->
       <center>
       <div class="search-container">
           <input type="text" id="searchData" placeholder="Enter a word">
           <div class="button" onclick="buttonClicked()"><img src="search.png"
alt="Search" class="search-icon"></div>
       </div>
       </center>


   </div>




       </h1>







       <!-- WORD -->
       <div id="wordContainer" style="display: none">
           <h2>Word:</h2>
           <p id="word"></p>
       </div>


       <!-- PHONETIC -->
       <div id="phoneticContainer" style="display: none">
           <h2>Phonetic Information:</h2>
           <p id="phoneticText"></p>
       </div>


       <!-- Audio -->
       <div id="audioContainer" style="display: none">
           <h2>Audio Pronunciation:</h2>
```

```
    <audio controls id="audio"></audio>
</div>

<!-- Source URL -->
<div id="sourceUrlContainer" style="display: none">
    <h2>Source URL:</h2>
    <p><a id="sourceUrl" target="_blank"></a></p>
</div>

<!-- DEFINITION -->
<div id="dictionaryMeaning" style="display: none">
    <h2>Definitions:</h2>
    <ul id="dictionaryDefinitions"></ul>
</div>

<!-- PART OF SPEECH VERB -->
<div id="verbContainer" style="display: none;">
    <h2>Part of Speech: Verb</h2>
    <div id="verbDefinitions"></div>
</div>

<!-- PART OF SPEECH NOUN -->
<div id="nounContainer" style="display: none;">
    <h2>Part of Speech: Noun</h2>
    <div id="nounDefinitions"></div>
</div>

<!-- PART OF SPEECH ADJECTIVE -->
<div id="adjectiveContainer" style="display: none;">
    <h2>Part of Speech: Adjective</h2>
    <div id="adjectiveDefinitions"></div>
</div>

<br><br>

<!--ADD TO WORD OF THE DAY BUTTON-->
<div id="addToWordOfDayContainer" style="display: none">
    <button onclick="navigateToCRUDPage()">Add to Word of the
```

```
     Day</button>
            </div>



            <!--ERROR MESSAGE-->
            <div id="error" style="display: none">
                <p id="errorMessage" class="app-error"></p>
            </div>




        </div>




</main>


    <!--JS-->
    <script src="dictionary.js"></script>

</body>
</html>
```

dictionary.js

```
// DICTIONARY


// BUTTON
function buttonClicked() {
    document.getElementById("dictionaryMeaning").style.display = "none";
    document.getElementById("error").style.display = "none";
    document.getElementById("audioContainer").style.display = "none";
    document.getElementById("sourceUrlContainer").style.display = "none";
    document.getElementById("phoneticContainer").style.display = "none";
    document.getElementById("nounContainer").style.display = "none";
    document.getElementById("verbContainer").style.display = "none";
    document.getElementById("adjectiveContainer").style.display = "none";

    var word = document.getElementById("searchData").value;




    displayWordOfTheDayButton();

    // FETCH API
    fetch(`https://api.dictionaryapi.dev/api/v2/entries/en/${word}`)
        .then((response) => response.json())
        .then((data) => {
            if (data.length > 0) {
                displayWord(word);
                displayAudio(data[0]);
                displaySourceUrl(data[0]);
                displayPhonetic(data[0]);
                displayPartOfSpeechNoun(data[0]);
                displayPartOfSpeechVerb(data[0]);
                displayPartOfSpeechAdjective(data[0]); // Add this line
                displayDictionaryData(data[0]);
            } else {
                displayError("Word not found");
            }
        })
        .catch((error) => {
            console.error("Error fetching dictionary data: ", error);
            displayError("Word not found");
        });
}

// FETCH WORD
function displayWord(word) {
    document.getElementById("wordContainer").style.display = "block";
    document.getElementById("word").textContent = word;
}
```

```
// FETCH PHONETIC
function displayPhonetic(data) {
    const phoneticContainer = document.getElementById("phoneticContainer");
    const phoneticTextElement = document.getElementById("phoneticText");

    if (data.phonetics && data.phonetics.length > 0) {
        const phonetics = data.phonetics[0];
        if (phonetics.text) {
            phoneticContainer.style.display = "block";
            phoneticTextElement.textContent = `Phonetic: ${phonetics.text}`;
        }
    }
}

// FETCH AUDIO
function displayAudio(data) {
    const audioContainer = document.getElementById("audioContainer");
    const audioElement = document.getElementById("audio");

    if (data.phonetics && data.phonetics.length > 0) {
        const phonetics = data.phonetics[0];
        if (phonetics.audio) {
            audioContainer.style.display = "block";
            audioElement.src = phonetics.audio;
        }
    }
}

// FETCH SOURCE URL
function displaySourceUrl(data) {
    const sourceUrlContainer = document.getElementById("sourceUrlContainer");
    const sourceUrlElement = document.getElementById("sourceUrl");

    if (data.phonetics && data.phonetics.length > 0) {
        const phonetics = data.phonetics[0];
        if (phonetics.sourceUrl) { // Use sourceUrl, not text
            sourceUrlContainer.style.display = "block";
            sourceUrlElement.href = phonetics.sourceUrl;
            sourceUrlElement.textContent = "Source";
        }
    }
}

// FETCH DEFINITION
function displayDictionaryData(data) {
    const dictionaryDefinitions = document.getElementById("dictionaryDefinitions");
    dictionaryDefinitions.innerHTML = "";

    document.getElementById("dictionaryMeaning").style.display = "block";

    for (let i = 0; i < 4; i++) {
        const definition = data.meanings[0].definitions[i];
        if (definition) {
            const li = document.createElement("li");
            li.textContent = `Definition: ${definition.definition}`;
            dictionaryDefinitions.appendChild(li);

            if (definition.example) {
                const exampleLi = document.createElement("li");
                exampleLi.textContent = `Example: ${definition.example}`;
                dictionaryDefinitions.appendChild(exampleLi);
            }
        }
    }
}




// FETCH PART OF SPEECH NOUN
function displayPartOfSpeechNoun(data) {
    const nounContainer = document.getElementById("nounContainer");
    const nounDefinitions = document.getElementById("nounDefinitions");
    const nounMeanings = data.meanings.find((meaning) => meaning.partOfSpeech ===
     "noun");

    if (nounMeanings) {
        nounContainer.style.display = "block";
        nounDefinitions.innerHTML = "";

        const ul = document.createElement("ul");

        for (let i = 0; i < 3 && i < nounMeanings.definitions.length; i++) {
            const definition = nounMeanings.definitions[i];
            const li = document.createElement("li");
            li.textContent = `Definition: ${definition.definition}`;

            if (definition.example) {
                const exampleLi = document.createElement("li");
                exampleLi.textContent = `Example: ${definition.example}`;
                li.appendChild(exampleLi);
            }

            ul.appendChild(li);
        }

        nounDefinitions.appendChild(ul);
    }
}


displayPartOfSpeechNoun(data);



// FETCH PART OF SPEECH VERB
function displayPartOfSpeechVerb(data) {
    const verbContainer = document.getElementById("verbContainer");
    const verbDefinitions = document.getElementById("verbDefinitions");
    const verbMeanings = data.meanings.find((meaning) => meaning.partOfSpeech ===
     "verb");

    if (verbMeanings) {
        verbContainer.style.display = "block";
        verbDefinitions.innerHTML = "";

        const ul = document.createElement("ul");

        for (let i = 0; i < 3 && i < verbMeanings.definitions.length; i++) {
            const definition = verbMeanings.definitions[i];
            const li = document.createElement("li");
            li.textContent = `Definition: ${definition.definition}`;

            if (definition.example) {
                const exampleLi = document.createElement("li");
                exampleLi.textContent = `Example: ${definition.example}`;
                li.appendChild(exampleLi);
            }

            ul.appendChild(li);
        }

        verbDefinitions.appendChild(ul);
    }
}

// FETCH PART OF SPEECH ADJECTIVE
function displayPartOfSpeechAdjective(data) {
    const adjectiveContainer = document.getElementById("adjectiveContainer");
    const adjectiveDefinitions = document.getElementById("adjectiveDefinitions");

    const adjectiveMeanings = data.meanings.find((meaning) => meaning.partOfSpeech
     === "adjective");

    if (adjectiveMeanings) {
        adjectiveContainer.style.display = "block";
        adjectiveDefinitions.innerHTML = "";
        const ul = document.createElement("ul");

        for (let i = 0; i < 3 && i < adjectiveMeanings.definitions.length; i++) {
            const definition = adjectiveMeanings.definitions[i];
            const li = document.createElement("li");
            li.textContent = `Definition: ${definition.definition}`;

            if (definition.example) {
                const exampleLi = document.createElement("li");
                exampleLi.textContent = `Example: ${definition.example}`;
                li.appendChild(exampleLi);
            }

            ul.appendChild(li);
        }

        adjectiveDefinitions.appendChild(ul);
    }
}


// ERROR MESSAGES
function displayError(message) {
    document.getElementById("error").style.display = "block";
    document.getElementById("errorMessage").textContent = message;

}



// NAVIGATE TO WOD.HTML
function navigateToCRUDPage() {
    window.location.href = 'wod.html';
}

// ADD TO WORD OF THE DAY BUTTON
function displayWordOfTheDayButton() {
    const addToWordOfDayContainer =
     document.getElementById("addToWordOfDayContainer");
    addToWordOfDayContainer.style.display = "block";
}
```

Thesaurus.html

```
<!DOCTYPE html>
<html>
<head>

    <!--TITTLE-->
    <title>THESAURUS</title>

    <!--CSS-->
    <link rel="stylesheet" type="text/css" href="index.css">

    <!--LINK FONTS-->
    <link href="https://fonts.googleapis.com/css?family=Poppins&display=swap"
     rel="stylesheet">

</head>
<body>

    <!--NAVIGATION-->
    <header class="top-nav">
        <img src="logo.png" class="logo">

        <nav>
            <ul>
                <li><a href="index.html" onclick="changeColor(this)">Home</a></li>
                <li><a href="dictionary.html"
     onclick="changeColor(this)">Dictionary</a></li>
                <li><a href="#" onclick="changeColor(this)">Thesaurus</a></li>
                <li><a href="CRUD.html" onclick="changeColor(this)">Word Of The
     Day</a></li>
            </ul>
        </nav>
</header>

<main>
```

```
<div class="container">
    <br>

    <div class="search-bar">


        <br><br>

        <!--SEARCH BAR-->
        <center>
        <div class="search-container">
            <input type="text" id="searchData" placeholder="Enter a word">
            <div class="button" onclick="searchThesaurus()"
 id="thesaurusData"><img src="search.png" alt="Search" class="search-
 icon"></div>
        </div>
        </center>



        <!-- WORD -->
        <div id="wordContainer" style="display: none">
            <h2>Word:</h2>
            <p id="word"></p>
        </div>

        <br>





        <!-- SYNONYM NOUN -->
        <div id="synonymNounContainer" style="display: none">
            <h2>Part of Speech: Noun </h2>
            <h3>Synonyms: </h3>
            <ul id="synonymNounList"></ul>
        </div>

        <!-- ANTONYM NOUN-->
```

```
<div id="antonymNounContainer" style="display: none">
    <h3>Antonyms:</h3>
    <ul id="antonymNounList"></ul>
</div>


<br><br>



<!-- SYNONYM VERB -->
<div id="synonymVerbContainer" style="display: none">
    <h2>Part of Speech: Verb </h2>
    <h3>Synonyms: </h3>
    <ul id="synonymVerbList"></ul>
</div>

<!-- ANTONYM VERB-->
<div id="antonymVerbContainer" style="display: none">
    <h3>Antonyms:</h3>
    <ul id="antonymVerbList"></ul>
</div>


<br><br>


<!-- SYNONYM ADJECTIVE-->
<div id="synonymAdjectiveContainer" style="display: none">
    <h2>Part of Speech: Adjective </h2>
    <h3>Synonyms: </h3>
    <ul id="synonymAdjectiveList"></ul>
</div>

<!-- ANTONYM ADJECTIVE-->
<div id="antonymAdjectiveContainer" style="display: none">
    <h3>Antonyms:</h3>
    <ul id="antonymAdjectiveList"></ul>
```

```
            </div>

            <br><br>

            <!-- SYNONYM ADVERB -->
            <div id="synonymAdverbContainer" style="display: none">
                <h2>Part of Speech: Adverb</h2>
                <h3>Synonyms:</h3>
                <ul id="synonymAdverbList"></ul>
            </div>

            <!-- ANTONYM ADVERB -->
            <div id="antonymAdverbContainer" style="display: none">
                <h3>Antonyms:</h3>
                <ul id="antonymAdverbList"></ul>
            </div>

            <br>

            <div id="addToWordOfDayContainer" style="display: none">
                <button onclick="navigateToCRUDPage()">Add to Word of the
     Day</button>
            </div>


            <!--ERROR MESSAGE-->
            <div id="error" style="display: none">
                <p id="errorMessage"></p>
            </div>


</div></div>

</main>

    <!--JS-->
    <script src="thesaurus.js"></script>
</body>
```

```
</html>
```

Thesaurus.js

```
// THESAURUS

// BUTTON
function searchThesaurus() {
    document.getElementById("error").style.display = "none";
    document.getElementById("synonymNounContainer").style.display = "none";
    document.getElementById("antonymNounContainer").style.display = "none";
    document.getElementById("synonymVerbContainer").style.display = "none";
    document.getElementById("antonymVerbContainer").style.display = "none";
    document.getElementById("synonymAdjectiveContainer").style.display = "none";
    document.getElementById("antonymAdjectiveContainer").style.display = "none";
    document.getElementById("synonymAdverbContainer").style.display = "none";
    document.getElementById("antonymAdverbContainer").style.display = "none";



    var word = document.getElementById("searchData").value;

    displayWordOfTheDayButton();

    //FETCH API
    fetch(`https://api.dictionaryapi.dev/api/v2/entries/en/${word}`)
        .then((response) => response.json())
        .then((data) => {
            if (data.length > 0) {
                displayWord(word);
                displaySynonymNoun(data[0]);
                displayAntonymNoun(data[0]);
                displaySynonymVerb(data[0]);
                displayAntonymVerb(data[0]);
                displaySynonymAdjective(data[0]);
                displayAntonymAdjective(data[0]);
                displaySynonymAdverb(data[0]);
                displayAntonymAdverb(data[0]);
            } else {
                displayError("Word not found");
            }
        })
        .catch((error) => {
            console.error("Error fetching dictionary data: ", error);
            displayError("Word not found");
        });
}

// FETCH WORD
function displayWord(word) {
    document.getElementById("wordContainer").style.display = "block";
    document.getElementById("word").textContent = word;
}



//FETCH SYNONYM NOUN
function displaySynonymNoun(data) {
    const synonymNounContainer = document.getElementById("synonymNounContainer");
    const nounMeanings = data.meanings.find((meaning) => meaning.partOfSpeech ===
     "noun");

    if (nounMeanings) {
        const synonyms = nounMeanings.synonyms;

        if (synonyms && synonyms.length > 0) {
            synonymNounContainer.style.display = "block";
            const ul = document.createElement("ul");

            for (let i = 0; i < 3 && i < synonyms.length; i++) {
                const synonym = synonyms[i];
                const li = document.createElement("li");
                li.textContent = `${synonym}`;
                ul.appendChild(li);
            }

            const synonymNounListContainer =
     document.getElementById("synonymNounList");
            synonymNounListContainer.innerHTML = "";
```

```
                synonymNounListContainer.appendChild(ul);
            }

         else {
            const li = document.createElement("li");
            li.textContent = "No Synonym found.";
            ul.appendChild(li);
        }
        }
    }


    displaySynonymNoun(data);



    // FETCH ANTONYM PART OF SPEECH NOUN
    function displayAntonymNoun(data) {
        const antonymNounContainer = document.getElementById("antonymNounContainer");
        const antonymNounList = document.getElementById("antonymNounList");

        const nounMeanings = data.meanings.find((meaning) => meaning.partOfSpeech ===
         "noun");

        if (nounMeanings) {
            antonymNounContainer.style.display = "block";
            antonymNounList.innerHTML = "";



            if (nounMeanings.antonyms.length > 0) {
                const antonymsUl = document.createElement("ul");

                for (const antonyms of nounMeanings.antonyms) {
                    const li = document.createElement("li");
                    li.textContent = `${antonyms}`;
                    antonymsUl.appendChild(li);
                }

                antonymNounList.appendChild(antonymsUl);
            } else {
                const noantonymsUl = document.createElement("ul");
                noantonymsUl.textContent = "No Antonyms found.";
                antonymNounList.appendChild(noantonymsUl);
            }
        }
    }

    displayAntonymNoun(data);


    // FETCH SYNONYM PART OF SPEECH VERB
    function displaySynonymVerb(data) {
        const synonymVerbContainer = document.getElementById("synonymVerbContainer");
        const synonymVerbList = document.getElementById("synonymVerbList");

        const verbMeanings = data.meanings.find((meaning) => meaning.partOfSpeech ===
         "verb");

        if (verbMeanings) {
            synonymVerbContainer.style.display = "block";
            synonymVerbList.innerHTML = "";

            if (verbMeanings.synonyms.length > 0) {
                const synonymsUl = document.createElement("ul");

                for (let i = 0; i < 4 && i < verbMeanings.synonyms.length; i++) {
                    const synonym = verbMeanings.synonyms[i];
                    const li = document.createElement("li");
                    li.textContent = ` ${synonym}`;
                    synonymsUl.appendChild(li);
                }

                synonymVerbList.appendChild(synonymsUl);
            } else {
                const noSynonymsUl = document.createElement("ul");
                noSynonymsUl.textContent = "No Synonyms found.";
                synonymVerbList.appendChild(noSynonymsUl);
            }
        }
    }


    displaySynonymVerb(data);

    // FETCH ANTONYM PART OF SPEECH VERB
    function displayAntonymVerb(data) {
        const antonymVerbContainer = document.getElementById("antonymVerbContainer");
        const antonymVerbList = document.getElementById("antonymVerbList");

        const verbMeanings = data.meanings.find((meaning) => meaning.partOfSpeech ===
         "verb");

        if (verbMeanings) {
            antonymVerbContainer.style.display = "block";
            antonymVerbList.innerHTML = "";

            if (verbMeanings.antonyms.length > 0) {
                const antonymsUl = document.createElement("ul");

                for (const antonyms of verbMeanings.antonyms) {
                    const li = document.createElement("li");
                    li.textContent = `${antonyms}`;
                    antonymsUl.appendChild(li);
                }

                antonymVerbList.appendChild(antonymsUl);
            } else {
                const noantonymsUl = document.createElement("ul");
                noantonymsUl.textContent = "No Antonyms found.";
                antonymVerbList.appendChild(noantonymsUl);
            }
        }
    }

    displayAntonymVerb(data);
```

```
}




// FETCH SYNONYM PART OF SPEECH ADJECTIVE
function displaySynonymAdjective(data) {
    const synonymAdjectiveContainer =
     document.getElementById("synonymAdjectiveContainer");
    const synonymAdjectiveList = document.getElementById("synonymAdjectiveList");

    const adjectiveMeanings = data.meanings.find((meaning) => meaning.partOfSpeech
     === "adjective");

    if (adjectiveMeanings) {
        synonymAdjectiveContainer.style.display = "block";
        synonymAdjectiveList.innerHTML = "";

        if (adjectiveMeanings.synonyms.length > 0) {
            const synonymsUl = document.createElement("ul");

            for (let i = 0; i < 4 && i < adjectiveMeanings.synonyms.length; i++) {
                const synonym = adjectiveMeanings.synonyms[i];
                const li = document.createElement("li");
                li.textContent = `${synonym}`;
                synonymsUl.appendChild(li);
            }

            synonymAdjectiveList.appendChild(synonymsUl);
        } else {
            const noSynonymsUl = document.createElement("ul");
            noSynonymsUl.textContent = "No Synonyms found.";
            synonymAdjectiveList.appendChild(noSynonymsUl);
        }
    } else {

        synonymAdjectiveContainer.style.display = "block";
        synonymAdjectiveList.innerHTML = "No Synonyms found.";
    }
}

displaySynonymAdjective(data);
```

```
}




// FETCH ANTONYM PART OF SPEECH ADJECTIVE
function displayAntonymAdjective(data) {
    const antonymAdjectiveContainer =
     document.getElementById("antonymAdjectiveContainer");
    const antonymAdjectiveList = document.getElementById("antonymAdjectiveList");


    const adjectiveMeanings = data.meanings.find((meaning) => meaning.partOfSpeech
     === "adjective");


    if (adjectiveMeanings) {
        antonymAdjectiveContainer.style.display = "block";
        antonymAdjectiveList.innerHTML = "";


        if (adjectiveMeanings.antonyms.length > 0) {
            const antonymsUl = document.createElement("ul");


            for (const antonyms of adjectiveMeanings.antonyms) {
                const li = document.createElement("li");
                li.textContent = `${antonyms}`;
                antonymsUl.appendChild(li);
            }


            antonymAdjectiveList.appendChild(antonymsUl);
        } else {
            const noAntonymsUl = document.createElement("ul");
            noAntonymsUl.textContent = "No Antonyms found.";
            antonymAdjectiveList.appendChild(noAntonymsUl);
        }
    } else {


        antonymAdjectiveContainer.style.display = "block";
        antonymAdjectiveList.innerHTML = "No Antonyms found.";
    }
}


displayAntonymAdjective(data);
```

```
// FETCH SYNONYM PART OF SPEECH ADVERB
function displaySynonymAdverb(data) {
    const synonymAdverbContainer =
     document.getElementById("synonymAdverbContainer");
    const synonymAdverbList = document.getElementById("synonymAdverbList");


    const adverbMeanings = data.meanings.find((meaning) => meaning.partOfSpeech ===
     "adverb");


    if (adverbMeanings) {
        synonymAdverbContainer.style.display = "block";
        synonymAdverbList.innerHTML = "";


        if (adverbMeanings.synonyms.length > 0) {
            const synonymsUl = document.createElement("ul");


            for (let i = 0; i < 4 && i < adverbMeanings.synonyms.length; i++) {
                const synonym = adverbMeanings.synonyms[i];
                const li = document.createElement("li");
                li.textContent = `${synonym}`;
                synonymsUl.appendChild(li);
            }


            synonymAdverbList.appendChild(synonymsUl);
        } else {
            const noSynonymsUl = document.createElement("ul");
            noSynonymsUl.textContent = "No Synonyms found.";
            synonymAdverbList.appendChild(noSynonymsUl);
        }
    } else {
        // Handle the case where adverbs are not found.
        synonymAdverbContainer.style.display = "block";
        synonymAdverbList.innerHTML = "No Synonyms found.";
    }
}


// FETCH ANTONYM PART OF SPEECH ADVERB
function displayAntonymAdverb(data) {
    const antonymAdverbContainer =
      document.getElementById("antonymAdverbContainer");
    const antonymAdverbList = document.getElementById("antonymAdverbList");


    const adverbMeanings = data.meanings.find((meaning) => meaning.partOfSpeech ===
      "adverb");


    if (adverbMeanings) {
        antonymAdverbContainer.style.display = "block";
        antonymAdverbList.innerHTML = "";


        if (adverbMeanings.antonyms.length > 0) {
            const antonymsUl = document.createElement("ul");


            for (const antonyms of adverbMeanings.antonyms) {
                const li = document.createElement("li");
                li.textContent = `${antonyms}`;
                antonymsUl.appendChild(li);
            }


            antonymAdverbList.appendChild(antonymsUl);
        } else {
            const noAntonymsUl = document.createElement("ul");
            noAntonymsUl.textContent = "No Antonyms found.";
            antonymAdverbList.appendChild(noAntonymsUl);
        }
    } else {
        // Handle the case where adverbs are not found.
        antonymAdverbContainer.style.display = "block";
        antonymAdverbList.innerHTML = "No Antonyms found.";
    }
}

// NAVIGATE TO WOD.HTML
function navigateToCRUDPage() {
    window.location.href = 'wod.html';
}
```

```
// ADD TO WORD OF THE DAY BUTTON

function displayWordOfTheDayButton() {

    const addToWordOfDayContainer =
     document.getElementById("addToWordOfDayContainer");

    addToWordOfDayContainer.style.display = "block";

}
```

wod.html

```
<!DOCTYPE html>

<html>

<head>


    <!-- TITTLE-->

    <title>WOD</title>


    <!--CSS-->

    <link rel="stylesheet" type="text/css" href="wod.css">


    <!-- LINK FONTS-->

    <link href="https://fonts.googleapis.com/css?family=Poppins&display=swap"
     rel="stylesheet">


</head>

<body>


    <!-- NAVIGATION-->

    <header class="top-nav">

      <img src="logo.png" class="logo">

        <nav>

            <ul>

                <li><a href="index.html">Home</a></li>

                <li><a href="dictionary.html">Dictionary</a></li>

                <li><a href="thesaurus.html">Thesaurus</a></li>

                <li><a href="CRUD.html">Word Of The Day</a></li>

            </ul>

        </nav>

</header>
```

```
<main>



    <form>
      <div class="container">
        <div class="form-group">


          <!--FILE FORM-->
          <label>File name</label>
          <input id="fileName" type="text" class="form-control" value="wod.txt"
     disabled>
      </div>


        <br>


        <div class="form-group">


          <!--CONTENT FORM-->
          <label>Content</label>
          <textarea id="fileContents" class="form-control"rows="5"></textarea>
        </div>
      </form>


      <br><br>


      <!--BUTTON-->
      <button id="btnCreate" class="btn btn-default">Create</button> <button
     id="view"> <a href="CRUD.html" style="text-decoration: none;
     color:white;">view</a></button>
    </div>


    </div>
</main>


    <!--JS-->
    <script src="wod.js">
      </script>
</body>
</html>
```

wod.js

```
const { app, BrowserWindow } = require('electron');
const fs = require('fs');
const path = require('path');

//BUTTON
var btnAppend = document.getElementById('btnCreate');
var fileName = document.getElementById('fileName');
var fileContents = document.getElementById('fileContents');

let pathName = path.join(__dirname, 'Files');



// CREATE BUTTON
btnCreate.addEventListener('click', function () {
  let file = path.join(pathName, fileName.value);
  let contents = fileContents.value;

// APPEND TO FILE
  if (fs.existsSync(file)) {
    contents = '\n' + contents;
    fs.appendFile(file, contents, function (err) {
      if (err) {
        return console.log(err);
      }
      var txtfile = document.getElementById('fileName').value;
      alert(txtfile + ' Word of The Day has been added! ');
      console.log('Word of The Day has been added!');
    });
  } else {
    alert('File does not exist.');
  }
});
```

wod.css

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
  }


/*BODY SETTINGS*/
  body {
    font-family: 'Poppins', Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-image: url('background.png');
    background-size: cover;
    background-position: center;
    background-attachment: fixed;
    color: #67645f;
  }


  /*NAVIGATION SETTINGS*/
  .top-nav {
    background-color: rgb(23, 100, 124);
    color: #fff;
    display: flex;
    align-items: center;
    padding: 20px;
    border: 1px solid rgb(59, 56, 56);
  }

  .top-nav nav ul {
    list-style: none;
    text-align: center;
    margin-right: 20px;
  }

  .top-nav nav ul li {
    display: inline;
    margin-right: 20px;
  }
```

```
.top-nav nav ul li a {
  text-decoration: none;
  color: #fff;
  font-size: 18px;
  position: relative;
  transition: color 0.3s;
}

.top-nav img.logo {
  margin-right: 300px;
  width: 150px;
  margin-left: 60px;
}


.top-nav nav ul li a::after {
  content: "";
  display: block;
  width: 0;
  height: 2px;
  background-color: #fff;
  position: absolute;
  left: 0;
  bottom: 0;
  transition: width 0.3s, background-color 0.3s;
}

.top-nav nav ul li a:hover::after {
  width: 100%;
  background-color: rgb(255, 191, 0);
}


/*HEADER SETTINGS*/
header h1 {
  font-size: 36px;
  margin-top: 40px;
```

```
}


/*MAIN SETTINGS*/
main {
  width: 80%;
  margin: 0 auto;
  padding: 20px;
  text-align: center;
}

/*FONTS SETTINGS*/
h1{
  color: rgb(255, 255, 255);
}



/*BUTTON SETTINGS*/
button {
  background-color: rgb(238, 178, 47);
  color: #fff;
  border: none;
  border-radius: 5px;
  padding: 10px 20px;
  font-size: 18px;
  margin-top: 20px;
  margin-left: 10px;
  cursor: pointer;
  transition: background-color 0.3s;
}

button:hover {
  background-color: rgb(239, 229, 110);
}

.button {
  border: none;
```

```
}
    border-radius: 60px;
    padding: 10px 20px;
    font-size: 18px;
    margin-top: 20px;
    margin-left: 10px;
    cursor: pointer;
    transition: background-color 0.3s;
  }


  /*INPUT FORM SETTINGS*/
  input[type="text"] {
    width: 50%;
    padding: 10px;
    font-size: 16px;
  }


  /*CONTAINER SETTINGS*/
  .container {
    padding: 80px;
    border-radius: 50px;
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
    background-color: rgba(255, 255, 255, 0.9);
    margin-top: 100px;
    margin-left: 240px;
    max-width: 700px;
  }


/*TABLE SETTINGS*/
table {
  width: 100%;
  border-collapse: collapse;
  margin-top: 20px;
}

table th, table td {
  border: 1px solid #ccc;
  padding: 10px;
  text-align: left;
}


table tr {
  margin-bottom: 10px;
}


table th {
  background-color: rgb(47, 98, 114);
  color: #fff;
}


#fileTableBody {
  background-color: #f0f0f0;
}

#fileContents {
  min-height: 100px;
}
```

CRUD.html

```
<!DOCTYPE html>
<html>
<head>

    <!--TITTLE-->
    <title>WORD OF THE DAY</title>

    <!--CSS-->
    <link rel="stylesheet" type="text/css" href="wod.css">

    <!--LINK FONTS-->
    <link href="https://fonts.googleapis.com/css?family=Poppins&display=swap"
     rel="stylesheet">
```

```
</head>
<body>

    <!--NAVIGATION-->
    <header class="top-nav">
        <img src="logo.png" class="logo">
        <nav>
            <ul>
                <li><a href="index.html">Home</a></li>
                <li><a href="dictionary.html">Dictionary</a></li>
                <li><a href="thesaurus.html">Thesaurus</a></li>
                <li><a href="#">Word Of The Day</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <form>
            <div class="container">
                <div class="form-group">

                    <!--FILE FORM-->
                    <label>File name</label>
                    <input id="fileName" type="text" class="form-control"
     value="wod.txt" disabled>
                </div>

                <br><br>

                <div class="form-group">

                    <!--CONTENT FORM-->
                    <label for="fileContents">Content</label>
                    <textarea id="fileContents" class="form-control"
     rows="5"></textarea>
                </div>
            </div>
        </form>
        <br><br>



        <!--BUTTON-->
        <button id="btnRead" class="btn btn-default">Read</button>
        <button id="btnUpdate" class="btn btn-default">Update</button>

        <!--TABLE-->
        <table id="fileTable">
            <thead>
                <tr>
                    <th>Word Of The Day</th>
                    <th>Delete</th>
                </tr>
            </thead>
            <tbody id="fileTableBody"></tbody>
        </table>

    </main>



    <!--JS-->
    <script src="CRUD.js"></script>
</body>
</html>
```

CRUD.js

```
const { app, BrowserWindow } = require('electron');
const fs = require('fs')
const path = require('path')



//BUTTON
var btnRead = document.getElementById('btnRead')
var btnDelete = document.getElementById('btnDelete')
var btnUpdate = document.getElementById('btnUpdate')
var fileName = document.getElementById('fileName')
var fileContents = document.getElementById('fileContents')
```

```
let pathName = path.join(__dirname, 'Files')



//READ BUTTON
btnRead.addEventListener('click', function() {
    let file = path.join(pathName, fileName.value);

    fs.readFile(file, 'utf8', function(err, data) {
        if (err) {
            console.error(err);
            return console.log(err);
        }
        fileContents.value = data;



        const lines = data.split('\n');



        const tableBody = document.getElementById('fileTableBody');
        tableBody.innerHTML = '';



        lines.forEach(function(line) {
            addContentToTable(line);
        });

        console.log("Word of The Day has been Read!");
    });
});



//UPDATE BUTTON
btnUpdate.addEventListener('click', function () {
    let file = path.join(pathName, fileName.value);
    let contents = fileContents.value;

    fs.writeFile(file, contents, function (err) {
        if (err) {
            return console.log(err);
        }
        console.log("Word of The Day was updated!")
        alert( "Word of The Day was updated!")
        })
})


//TABLE
    function addContentToTable(content) {
        const newRow = document.createElement('tr');

        const contentCell = document.createElement('td');
        contentCell.textContent = content;

        const deleteCell = document.createElement('td');
        const deleteButton = document.createElement('button');
        deleteButton.textContent = 'Delete';

        deleteButton.addEventListener('click', function() {

            const row = this.parentNode.parentNode;
            const rowIndex = row.rowIndex;

            row.remove();

            const file = path.join(pathName, fileName.value);
            fs.readFile(file, 'utf8', function(err, data) {
                if (err) {
                    console.error(err);
                    return console.log(err);
                }
                const lines = data.split('\n');
                lines.splice(rowIndex - 1, 1);
                const updatedData = lines.join('\n');
                fs.writeFile(file, updatedData, 'utf8', function(err) {
                    if (err) {
                        console.error(err);
                    }
                });
            });
        });

        deleteCell.appendChild(deleteButton);

        newRow.appendChild(contentCell);
        newRow.appendChild(deleteCell);

        const tableBody = document.getElementById('fileTableBody');
        tableBody.appendChild(newRow);
    }
```

IV. <u>**Link GITHUB.**</u>

https://github.com/dyansasuke/dictionaryThesaurus.git