Spelling test final report

Abstract

In this paper I will document and create a spelling test platform designed for dyslexic primary school children in the UK for asymmetric learning.

Acknowledgements

I want to thank the external testers for testing my program and my friend Aaron James for proof reading.

I want to thank my supervisor Helen Phillips for the support I got from them for creating the website and writing this report.

Table of Contents

Contents

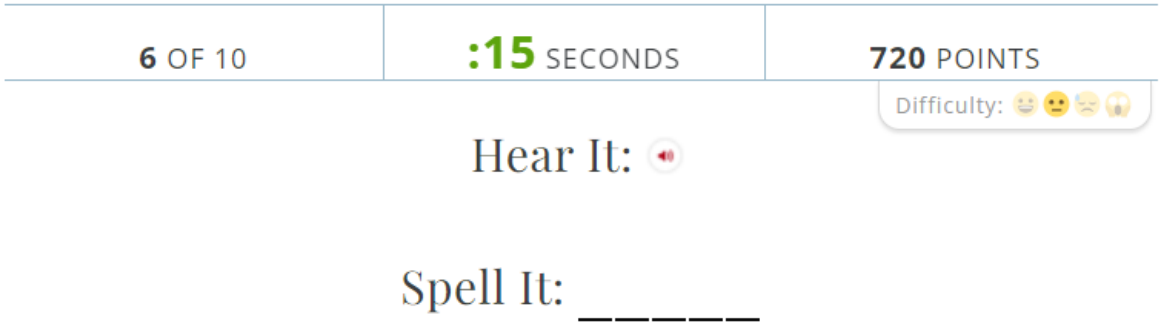
Introduction

The aim of this project is to create an automated/ semi-automated assessment tool to support children to undertake spelling tests. This can either be a part of synchronous or asynchronous learning materials. This will be tailored towards students with dyslexia as they will be able to change the font colour, style and background colour of the website. Additionally, it will allow the teacher to show more or less feedback depending on the severity of the case.

Background

Many spelling test websites and platforms exist. While looking at different existing ones there were two main patterns I noticed. One way was the websites use text to speech to read out the words that you are supposed to spell.

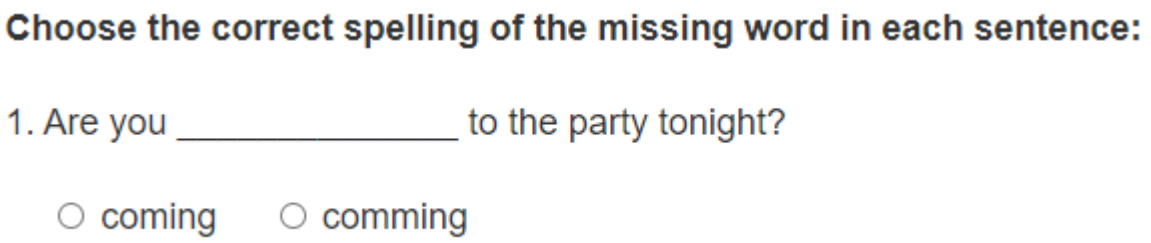
(**Figure 1:** screenshot of spelling test from Merriam Webster) [1]



The image above [1] shows what the spelling test looks like on one of the websites where they play audio sounds for text to speech. You also have a timer to limit you on how long you can think about the spelling. The website above is not just aimed at children but also adults who wish to improve their spelling.

The other way they do spelling tests is to have multiple radio button options in which you select the correct spelling for the word that fits the sentence.

(**Figure 2:** Screenshot of spelling test from englishclub) [2]



(**Figure 3:** Screenshot of spelling test from dailywritingtips) [3]

1. The car lost one of ___ wheels.

- ○ it
- ○ it's
- ○ its
- ○ its'

The two images above are from different websites with the same type of spelling tests. The second type of tests could be harder for children due to having to know how sentence structures work to get some of the correct answers.

I also asked a parent about what their children's school use to give them spelling homework and they directed me to Spelling Shed [4]. This website is designed for teachers teaching primary school children to learn to spell. The website shows you a word spelled out before it then takes you to another page where you enter in the word by either typing into a text box or clicking on a letter tile. There is also an option on the input page to hear a voice recording of the word.

(**Figure 4:** Screenshot of spelling test from Spelling Shed) [4]

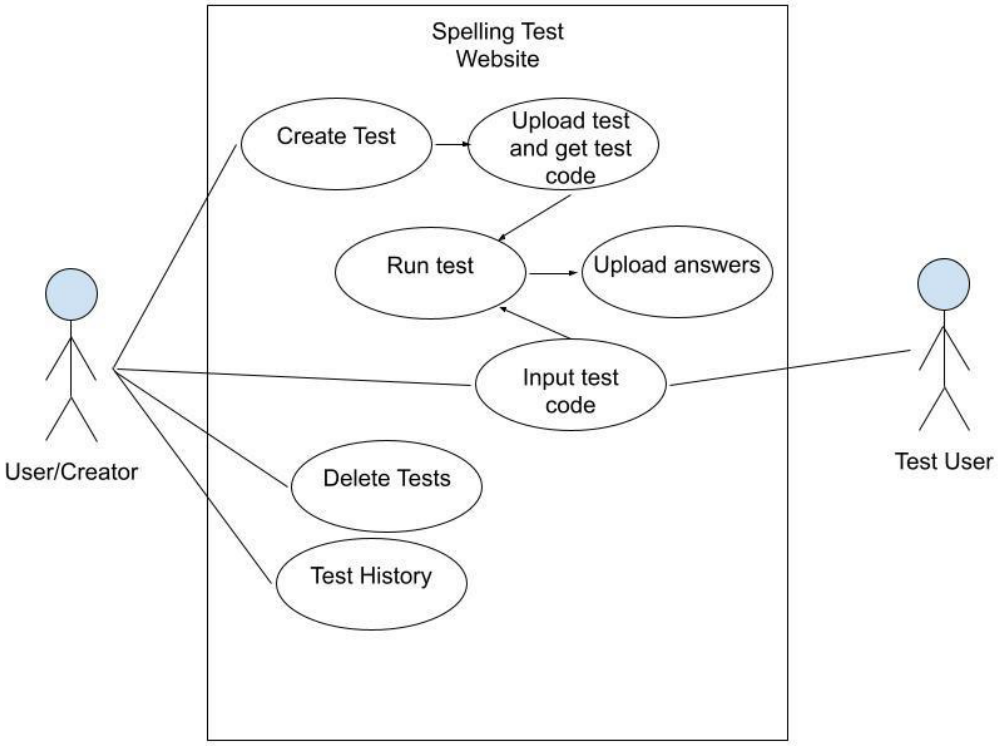
This website was the most comprehensive website that I looked at with clear gamification to entice students to do the tests. Sounds are used to make the website more interesting with scoreboards existing for each class to let students compete against each other. One of the downsides of this website is that it is a paid platform mostly aimed at whole schools or institutions. There is an option for parents as well but still at a monthly cost. The other downside was that when you are completing a test you cannot go back to a previous word if you think you might have got it wrong and wish to redo it.

The UK government has statutory requirements for what words should be known by what age. Along with the different definitions they give they also have example words which I will be using to create a revision mode. These guidelines give exact metrics on what the focus is on for primary school children learning to spell. [5]

Approach

Use Case Diagram

(**Figure 5:** use case diagram showing the actors and basic flow of the program)



The system is designed for two different types of users. Users who create tests and will need an account in order to save the created tests and to associate the test results with to allow them to see the corresponding results. The users who create tests also have options to see all their tests which will have infographics attached to them and the corresponding results in more detail The users can also delete their tests which will then cascade results removing them from the database to keep the overall size of the database to a minimum.

The other actor is the test user who is the person completing a test and submitting the result. The test user will only need to input a test code for the test to load with the correct words and settings. Once the test user has completed the test and clicked on submit the answers will be uploaded into the database.

Both actors can run tests and complete them. If the user is logged in, I plan to have result entries add results to the same entry for the test if the username is the same as a previous entry for that test. This will reduce the number of rows in the database table and allow for displaying of changes in results for each time they have completed a test in chronological order.

While conceptualising the use case diagram I separated out the user/ creator and the test user to show the two different sides to the system. Both actors could be one actor due to a test user being allowed to create tests however as you do not need to have an account to complete a test I have kept them separate to show this. If a test user does not have an account they can create tests but they will not be able to store the tests or see their results after seeing the results page upon test completion.

User stories

In week 4 I had a client meeting where we discussed different user stories and what it was that the client wanted from this project. Using a MuSCat methodology for must have, should have, could have and won't have to categorise the different ideas for the project. Through discussions we decided the focus for this project should be about how feedback and results are displayed alongside the different test creation options with less of an emphasis on the account system. Instead of having accounts assigned to a parent account, as discussed in the initial plan, tests will have a test code that can be entered in to complete the test with results being associated with the test and the test creator.

(**Table 1:** Table of results from interview with client about user stories)

ID	Stories	Additional Description	MuSCat
1	As a user I want to be able to login	To allow for a test creator to have tests associated with them they will need to have an account	Should
2	As a teacher I want to create spelling tests	Creating spelling tests should be clear and easy to understand/ follow	Must
3	As a teacher I want to create accounts for my students	To make sure all teachers can understand the results of the test users more accurately	Could
4	As a teacher I want to be able to see the results of my students		Must
5	As a teacher I want to see a history of spelling tests	To see all previously created spelling tests by the user	Must
6	As a student I want to be able to complete a spelling test		Must
7	As a user I want to be able to change font colour, size and website colour scheme	To allow users to choose what scheme is best for themselves to make completing the test as easy as possible.	Must
8	As a teacher I want to be able show feedback on tests	Suggests appropriate spelling for students, give badges.	Must
9	As a student I want to be encouraged to do more spelling		Should
10	Unlockables	e.g. when they have had 5 completely correct spelling tests they are able to access challenge mode - all a part of gameification	Should
11	different test modes	Including random/ revision modes	Must

Use Case Tables

The use case tables below are for what was decided as a "must" in user stories. These have been decided in a client meeting as being the key components that they would like to see in the created platform. These use cases will be used to test the system against later.

Use Case ID: 1	**Use Case Name: Spelling test creation**	**Rating:** Must
Description of Purpose: To allow for spelling tests to be created by a user and then used.		
Actors: Teacher/ User		
Pre-conditions: the user must be on the home page		
Post-condition:		
Basic Flow: Creating a custom spelling test 1. User clicks on create test 2. The user is asked for the length of the test. 3. User is then taken to another form page 4. User enters in the words manually into the text boxes 5. The user clicks confirm 6. The user is taken to the spelling test		
Alternative Flow: The user is logged in while creating a test **AC2:** 4.1 the user enters a name for the test 4.2 The user enters a name for the test 4.3 The user is then taken to the test screen and given a code for the test for later use		
Exceptions: 1. User clicks on other test creation options – see use case 7		
Date: 09/03/22		Version: 1.0

Use Case ID: 2	**Use Case Name: Test results page**	**Rating:** Must
Description of Purpose: After completing a test and clicking on submit the user will see the results of the test based on what the test creator has selected to show for the test user.		
Actors: Test user		
Pre-conditions: The user has completed a test and submitted		
Post-condition:		
Basic Flow: 1. The results page looks at the test settings 2. The test settings dictate what results are shown to the test user 3. An entry is made in the database where the results are saved		
Alternative Flow: **AC2:**		
Exceptions: 1. No results are shown to the user 1.1 Shows a message to notify that the test has been completed but no results are shown		
Date: 09/03/22		Version: 1.0

Use Case ID: 3	**Use Case Name: Test History**	**Rating:** Must

Description of Purpose: As a teacher/user there will be a way to see all the tests that you have made and are currently able to be used. This will also have charts of results with relevant information to make understanding results easier.
Actors: Teacher/ User
Pre-conditions: The teacher/ user have created a test and has gone to the test history page while logged in
Post-condition:
Basic Flow: 1. The user clicks on test history 2. The user is shown a page with all their tests on with relevant information and charts 3. The user can delete a test from here which will cascade and remove all result entries for that test
Alternative Flow: **AC2:** 2.1 The user clicks on a more detailed view 2.2 Charts are displayed showing each individual that has completed the test 2.3 Relevant information is displayed for each individual e.g. times the test has been completed by that individual
Exceptions:

Date: 10/03/22	Version: 1.0

Use Case ID: 4	**Use Case Name: Completing a spelling test**	**Rating:** Must
Description of Purpose: When a spelling test is completed the test creator needs to see the results that have been entered. As a test user they need to be able to click on a submit button and their results need to be stored for later use. A test user also needs to be able to find the test		
Actors: Teacher/ User/ Test User		
Pre-conditions: The test user has the test code		
Post-condition: The user has filled in the test and clicked submit		
Basic Flow: 1. The user enters in a test code 2. The system checks if the user is logged in 2.1 If the user is not logged in an option to enter a name appears at the top of the test page 3. The user clicks on submit 4. Either the username, if logged in, or the entered name (optional) is then stored into a database table with a link to the original test and test creator 4.1 If the user is logged in and has completed the test before no new entry is created but an additional score set is added into the existing database entry 5. Entries can then be pulled by the creator of the test when looking at test results and history		
Alternative Flow: **AC2:**		
Exceptions:		

Date: 10/03/22	Version: 1.0

Use Case ID: 5	**Use Case Name: Change Font colour, size, style**	**Rating:** Must
Description of Purpose: To increase the accessibility of the website to allow for dyslexic users to use the website with more ease.		
Actors: Teacher/ User/ Test User		
Pre-conditions: On the website		
Post-condition:		
Basic Flow: 1. On the menu bar at the top the user hovers over either colour scheme, font size or font style 2. A selection of options dropdown for the user to select 3. The user clicks on an option 4. The font or colours update on click without reloading of page 5. When going to a different page the selected settings will be retained		
Alternative Flow: **AC2:**		
Exceptions:		
Date: 13/03/22		Version: 1.0

Use Case ID: 6	**Use Case Name: Show feedback**	**Rating:** Must
Description of Purpose: To allow for the test creator to give varying amounts of feedback to the test users based on what the test creator would like		
Actors: Teacher/ User/ Test Creator		
Pre-conditions: Selected in test settings when creating the test		
Post-condition:		
Basic Flow: 1. The user has started to create a test 2. The user selects checkboxes with a few different options for feedback 3. Options are stored in test settings 4. When a test code is entered the settings will be retrieved 5. When test has been completed and submitted the chosen feedback is shown		
Alternative Flow: **AC2:**		
Exceptions: 2. No feedback is selected 2.1 Only a notice to say test has been submitted and completed will show		
Date: 13/03/22		Version: 1.0

Use Case ID: 7	**Use Case Name: Spelling test modes**	**Rating:** Must
Description of Purpose: Some different options for types of tests that the users will be able to choose from to get them started.		
Actors: Teacher/ User		
Pre-conditions: The user has selected a mode that is not 'custom tests'		
Post-condition:		
Basic Flow: If the user selects revision mode		

1. The user selects revision mode 2. The user is asked for what year(s) they want based on UK government guidelines [5] 3. The user selects a year(s) 4. The user is given different topics and selects one 5. The user sets the amount of words and test settings 6. The user clicks on 'random' 7. A test appears with words found in the selected category 7.1 if the chosen length of test is greater than the number of words in the selected category the test length will automatically shorten 8. Test can be completed and feedback displayed
Alternative Flow: If the user selects random mode **AC2:** The user selects random mode 1. The user inputs a test name if logged in and sets test settings 2. The user enters in how long the test should be 3. The user is given random words from the English dictionary 4. The users test is uploaded if logged in 5. The user submits their words 6. The user is shown the results page
Exceptions: for basic flow 6. The user clicks on custom 6.1 The user is taken to the custom test form 6.2 Any words found under the selected category appear in editable text boxes 6.3 The user clicks on submit

Date: 13/03/22	Version: 1.0

Why a website?

I chose to go with a website because a computer is often more accessible in a primary school than a mobile phone or other device. I thought a website would make it easier for a school child to access the spelling tests for asymmetric learning. As a website runs on a server there is also not chance of it being installed incorrectly for the user. There are also many options to easily allow users to change font styling and background colours to make it more accessible to children with learning difficulties.

Changes to Gantt chart plan

(**Figure 6:** New Gantt chart plan)

Work to be done	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12
Initial Planning												
Create a basic PHP website												
Text-to-speech												
Create spelling tests												
Client meeting												
Create word lists												
Options to change font and colour schemes												
Custom and automatic spelling test generation												
Challenge mode												
Practice mode												
Client meeting												
Beautify the website												
Database creation and account setup												
Statistical collection												
Client meeting												
Additional features if time is left and write-up												

After my client meeting in week 4 we decided to change the order of my plan to better reflect what the client wants. We decided that I will move the account creation and database systems behind it to a later week to make sure the key features of the spelling tests are done first. Due to this I have decided that instead of using a dictionary API I will use a JSON file containing the words and definitions of words from the English dictionary and upload it to the database.

Another change to note would be the statistical collection which has also been moved to a later week. This is due to the original plan for statistical collection being stored on a database linked to a teachers account. As the database has been moved to later the long-term statistical collection method must be done after the database has been implemented.

I also added in the additional entry of "text-to-speech" to better reflect the time it took for me to get the API working which ended up taking almost a whole week.

Fonts and Page Colour Scheme Research

On the website for the British Dyslexia Association [6] they have advice styles friendly to dyslexia [7]. Readable fonts are not limited to but include Arial, Comic Sans and Monaco. These fonts are good because they “can appear less crowded” [7]. For my website I will have Arial as the default font and the other two fonts as font options. The recommended font size is 12-14 point or 1-1.2rem however they state that “some dyslexic readers may request a larger font” [7]. Due to this I will have three options for font, 1rem, 1.2rem and 1.5rem. Having different word groupings clearly separated with extra space between items on the screen allows for the page to be more readable.

For colour schemes using single colour backgrounds with high contrast to the text colour is recommended. Having backgrounds with patterns can make content more distracting however as this is aimed at primary school children some more colourful options may be added. One of the options will be a pale orange background with dark blue font as this is the preferred way to view documents for a dyslexic friend of mine. Having a while background can be distracting because it “can appear too dazzling”[7] however an option for white will still be available.

Database design

I designed the database to allow tests, users, and test users to be linked together easily. By keeping these in separate tables it allows me to change one table without effecting the others. I will also use JSON to store parts of the information about tests, these include “settings” and “words” in the table

“spelling tests” and “answers” in the table “test user”. I decided to use JSON for this as it enables me to add additional options and information about tests easily as I continue to code.

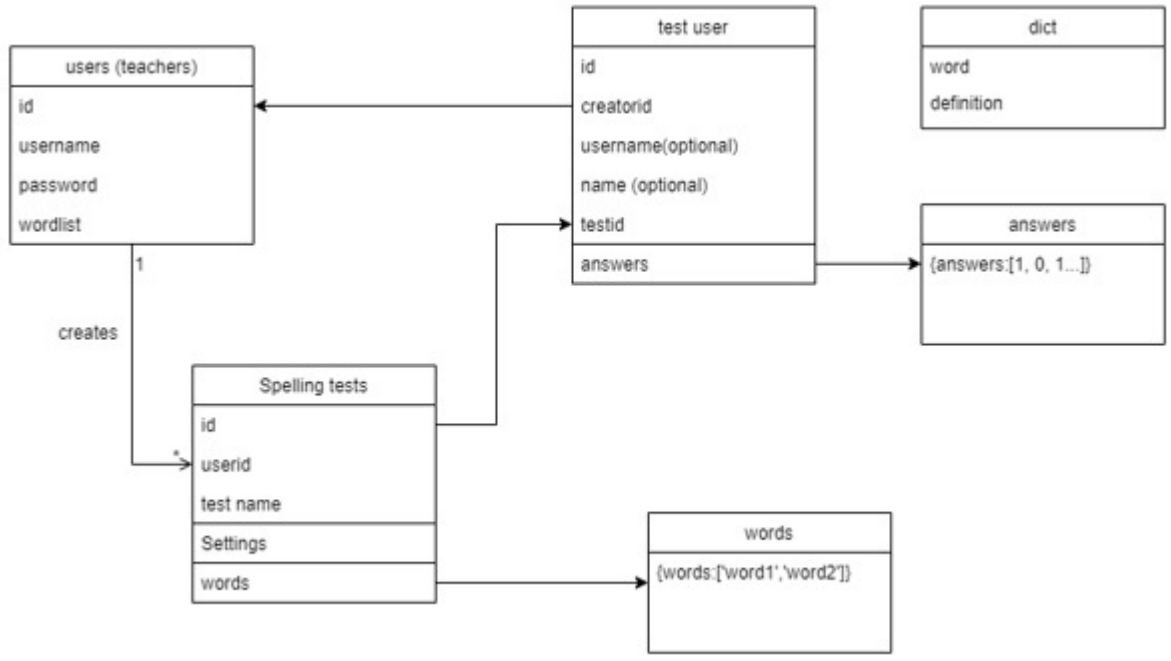
A good example would be “settings” in the table “spelling tests”. As settings should be mostly true or false statements with a key that says what the setting is for using JSON formatting I can very easily add in more test settings and add to the length of the JSON. JSON is very easily and quickly parsed allowing me to retain these settings across sessions without increasing processing time. This will also allow me to have the option of adding in a way to change tests and test settings after they have been uploaded to the database.

In the table “test user” I decided to use a JSON for storing answers to tests with the column names “answers”. I decided to use a JSON for this so that I could have one entry in “test user” have multiple answers to one test. This is so that I keep a record of each individual test allowing more options for displaying feedback and information about tests and how individuals performed on these tests. The entry “words” in the table “spelling tests” is very similar to “answers” however there will only ever be one list in “words”. I left “words” as a JSON in “spelling tests” so that I could have one row per test instead of having one word per row in the table and lots of repeated data as a new row would be needed for each word of the test.

The table “test user” has two foreign indexes named “testid” and “creatorid”. These are there to allow me get the relevant data about tests by either looking at the test and then finding the associated users or by looking at the user and looking at who has completed one of their tests. I could have this while only having the one foreign index “creatorid” but having two makes coding the displaying of feedback easier.

There is also a table called “dict”. This table is simply a dictionary which I will read and input into a table. This is so that I can easily get the definitions of words which will be useful for homophones and homonyms as a text to speech reader can be very difficult to differentiate between.

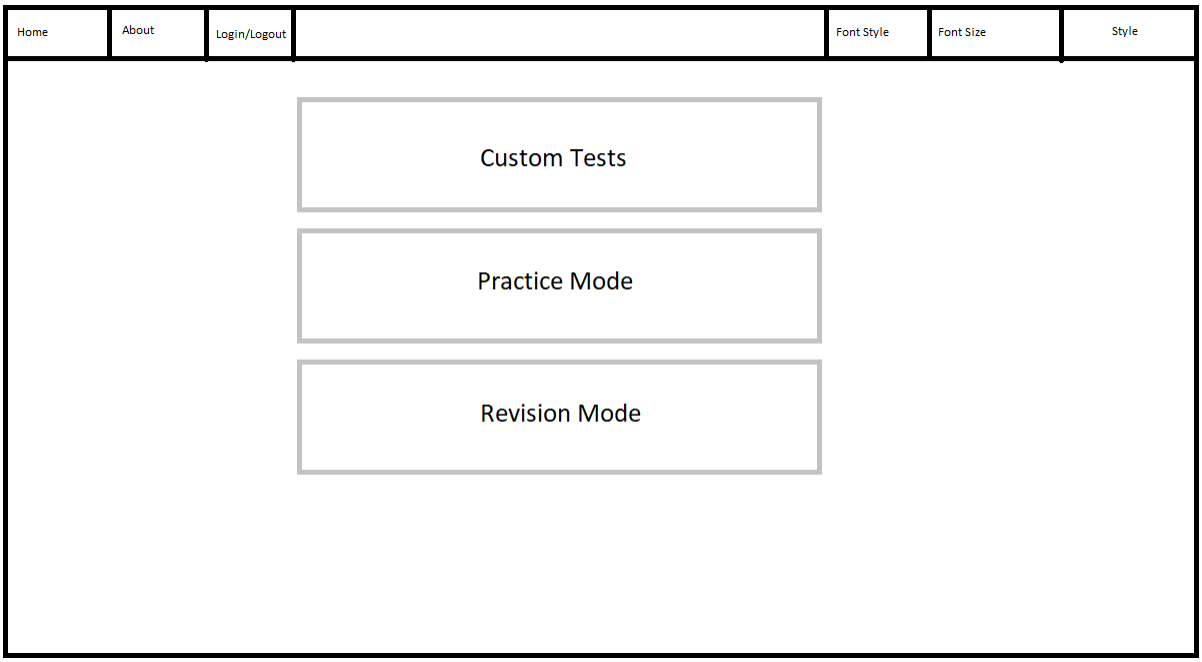
(**Figure 7:** Database design UML)



User Interface Design

Home Screen

(**Figure 7:** Initial design of landing page for the webside)

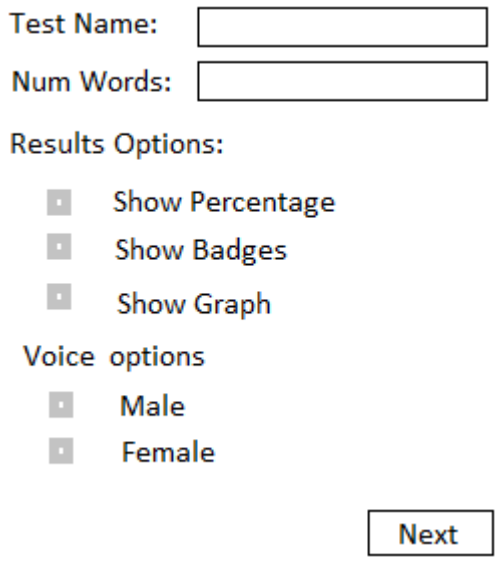


Above is the initial design of the home screen or landing page of the website. I chose to have the menu at the top due to it being at the edge making it easier to navigate the mouse to it. The menu will be present on all screens at the top. I grouped relevant items together on it with changes to the page style being on the right and page links on the left following the similarity and proximity principles of Gestalt [8].

In the centre of the screen I have three options for different test modes. Custom tests would allow you to enter in each word individually. Practice mode allows you to practice words that you get incorrect most often. Revision mode allows you to have a test premade or get word suggestion for the different topics from the UK governments spelling guidelines for key stages [5].

Test Creation Form

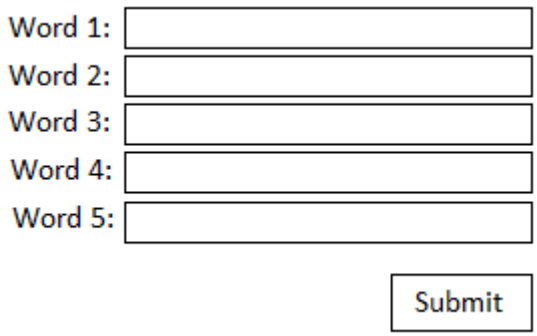
(**Figure 8:** Initial design of test setting form)



The screenshot above is the initial screen the user will see when creating a spelling test. The first two items on the form are to set a name for the test and set the length of the test, the length will be a number input only. Next, the form allows for you to set options for what results the person completing the test will see when they submit it. As the test creator you will also be able to see the results of everyone who has completed the test in a test history page when logged in.

The voice options change what voice that the text to speech reader will use when creating audio files for the selected words. Currently only male and female voices are on the list however I will add options for accents from different countries later.

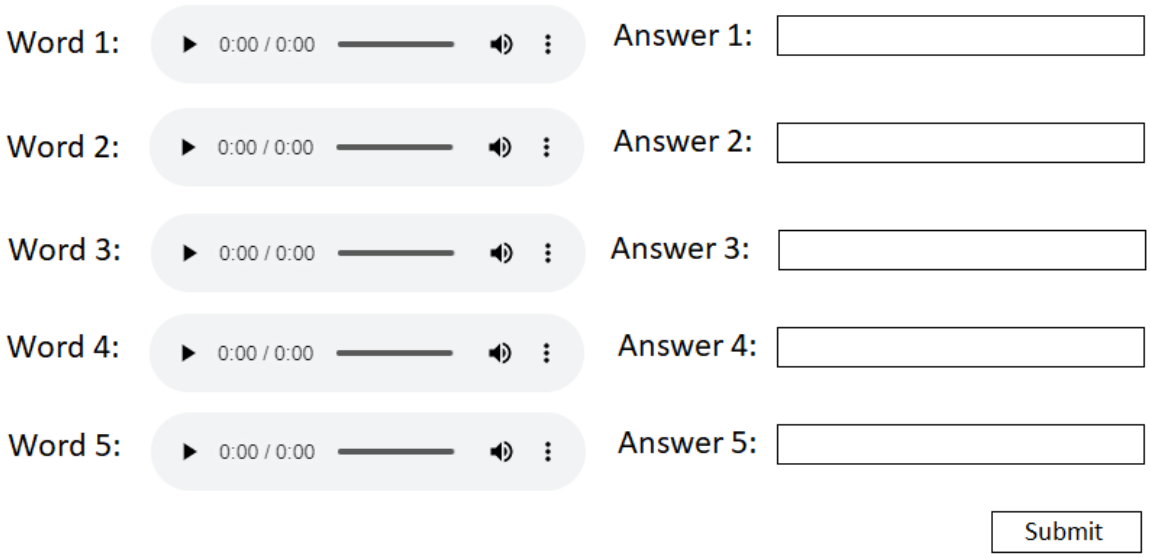
(**Figure 9:** Initial design of test word input form)



After clicking the “Next” button on the previous menu a menu like the one above will appear. In this menu it will take in the options of how many words you want to have and give you separate text boxes for inputting each word. Once all the words have been input you can click the submit button. Once the test has been created an entry in the database will be made and you will have a preview of the test where you will be able to hear each word and complete the test if you would like to.

Test Page

(**Figure 10:** Initial design spelling test page where the user completed the test)



Above is the test screen. I wanted to make it as clear and easy as possible to avoid any potential confusion. Having all the audio files and answer boxes on the same screen allows the user to check over their spellings before clicking submit. It also allows them to complete the test in any order and comeback to the text to speech audio files if they are unsure on the word the first time they hear it.

Once you click submit on this page you will be taken to the results page where the results will be displayed according to the selected options. The test results will also be stored in the database to allow for it to be referred to later by the test creator.

Implementation

WampServer and PHP

I decided on using WampServer as my PHP environment. WampServer is a package download that comes with useful PHP extensions installed. They have a built-in admin page where you can also access PHPMyAdmin to look at and create a database through a web browser UI. Extensions I plan to use that are pre-installed are password hashing and one of two different MySQL integration extensions, mysqli and pdo_mysql.

WampServer runs on Apache 2.4.51 and PHP 7.4.26. They do have a more recent version of PHP available but I have decided to use the default install to allow future installs potentially on other devices to be easier.

One of the limitations of PHP is when passing variables from a form for processing you need to link to a PHP page. This can be the same page that you were previously on by linking to the same page but processing becomes messy when trying to do this and often you have lots of similar pieces of code in one file. This caused me to need multiple PHP files or pages to have forms with different options. The form elements that are the same were put into functions to allow me to call this for forms in different files easily while trying to minimise duplicate code.

Another limitation is that with PHP there is no way to have an OnClick function like what is built into JavaScript. A way to get around this is to program an OnClick function in JavaScript then have it call a PHP function within the OnClick function call. The downside of this method is that finding errors

becomes very difficult as the JavaScript script hides PHP error messages as it assumes the error message is an incorrect variable input. The way you are supposed to do this is by using a PHP extension called AJAX [9] but I decided to not use it as there would not be many uses for it.

The final limitation is that PHP cannot parse through a JSON file as a JSON object. The JSON file has to be converted to an array for it to be parsed through making it slight more difficult than I initially thought.

One of the benefits of PHP is that you can easily call other PHP files within other PHP files and it will output the same. A good use for this for the header and footer elements which allow them to stay consistent across all pages. I decided to put the login, logout and create account in the header as well to allow for the user to login or logout from any page.

Another benefit is that coding HTML and PHP in one file is easy. PHP has a tag that symbolises when PHP code begins and ends allowing for different items on the webpage to have different values based on how PHP outputs it. This allows me to output repetitive lines of html, often making up a form, in one loop.

Font Styling and Colour Schemes

Initially I wanted to achieve this function using PHP. I managed to get the colour of the background to change consistently. However, when I introduced a function to change the colour of the text along with the changing background I could not get it to consistently read the inputs from the user across webpages. While researching on workarounds and planning for how I would change the font type and font size I decided to make the switch to JavaScript as they have inbuilt functions for changing specific elements of a webpage. After changing to JavaScript the options for changing the scheme and font styling of the webpage became consistent across the website.

I created two scripts to achieve the wanted function. The first script is what takes in the users input then processes the output, in this case the output being a change to colour scheme or font styling. The second script checks whenever a webpage is loaded whether there are previously saved options to the colour scheme or styling. If there are no saved settings then it outputs default settings to the console. In order to get this function working properly I had a choice between using sessionStorage or localStorage. sessionStorage will keep variables saved across the website until the tab is closed whereas localStorage will save the variable to the users computer. I decided to go with localStorage so that the settings for style and colour scheme will be saved and reproduced whenever the user accesses the website.

Google Cloud Services Text-To-Speech API with PHP

For the text-to-speech function of the website I decided to go with Google cloud services. The text-to-speech is available as part of their client libraries. Installation with PHP requires the use of composer [10]. The installation needs to be run in the rout directory of the project following the instruction guidelines for PHP from Google's client library documentation [11]. To get the text-to-speech to work you will need a Google service account. This account allows you to access Google APIs and do requests to them. In your code you need to reference the correct authentication token and define the system PATH. When I was installing the text-to-speech API I ran into an error where there was a guzzle client, which runs the backend of the API, had a request error where it would be unable to authenticate using my authentication token after it has been used once. To work around it I had to change the one of the client verifications in their code to false to skip the repeated authentications [12].

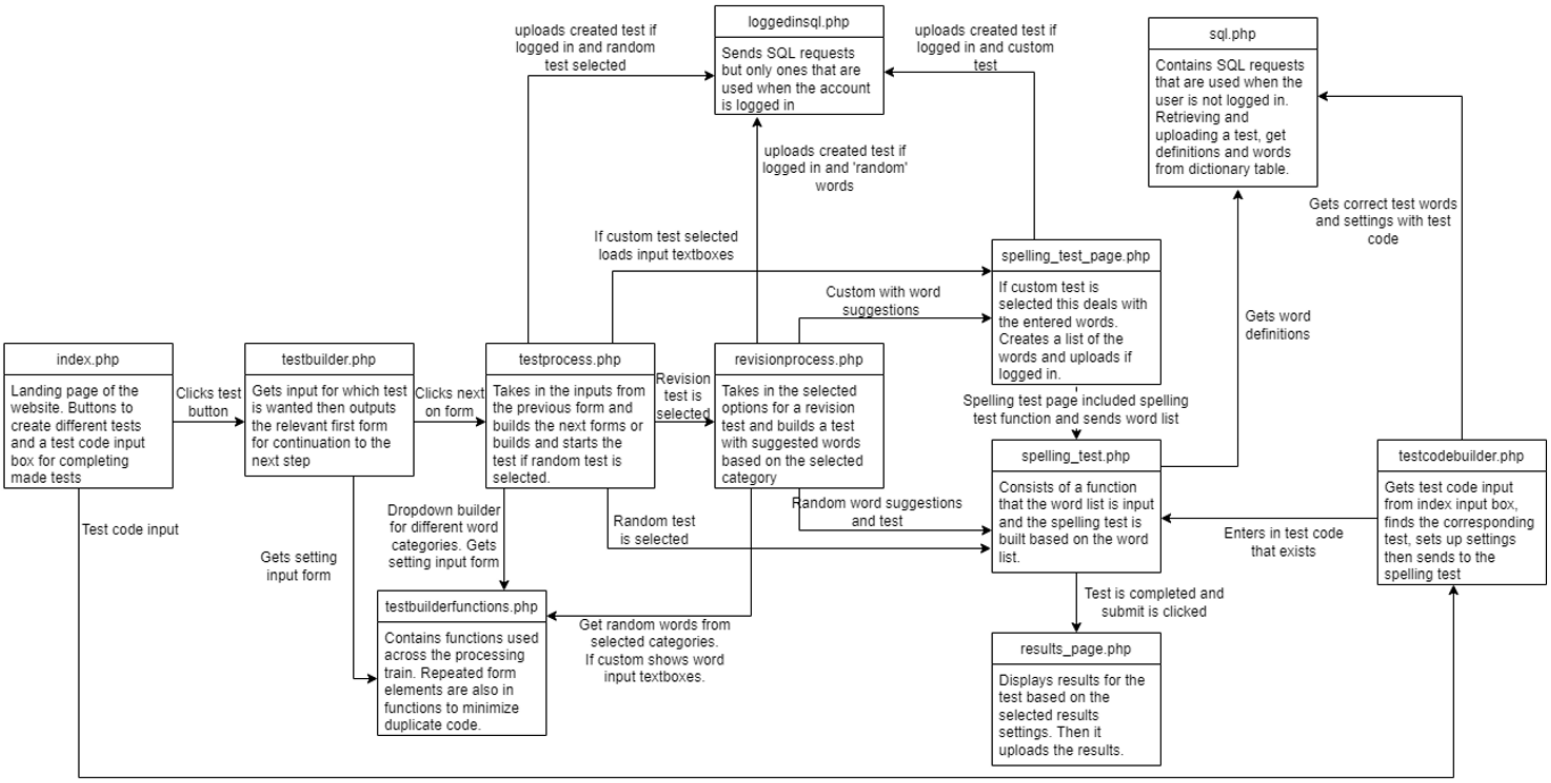
The text-to-speech API is a paid client that is offered by Google with free use of it for the first 4 million characters submitted. Although it is paid I did not think this would matter as through the creation of the website I was unlikely to request more than 4 million characters to be converted to an audio file. The main reason I went with the service was because it allowed me to have different accents. I thought having different accents was important because text-to-speech voices are not always easy to understand. When creating a test having different options will allow the test creator to select which accent they think would be best for the users completing the test.

Charts.Js

Charts.Js is what was used to create the different charts for displaying results and information. This is a free platform that allows you to display data in many different charts. I have only used the bar chart for this project as it was the most appropriate for the data collected. This uses JavaScript to build the charts. To get this to work I included a PHP function call when declaring a variable in JavaScript with the data I want being echoed into the JavaScript variable from the function. 'Echoed' is using the "echo" function of PHP to make the JavaScript thing that the output from PHP is the value of the variable [13].

Test Creation Structure

(**Figure 11:** Structure of how the tests are created)



Above is the architecture of how the tests are created, processed and then ran. All tests start at index.php, which is the landing page of the website, and end at results_page.php. There are three files which only contain functions group into their respective categories sql.php, loggedinsql.php and testbuilderfunctions.php. I decided to have these functions in separate PHP files and not included within the main flow because most functions are used in other places making it easy to repurpose the code that exists by changing the input values. sql.php includes all SQL functions and commands

that I allow to happen when the user is not logged in. This is to allow for a user to complete and submit a test without having to have an account. Loggedinsql.php contains all the SQL functions and commands which can only happen when a user is logged in such as uploading a created test. These SQL commands and functions are kept separate to add a little bit of security to what users can see.

Testbuilderfunctions.php contains functions that do not involve any SQL but contain key functions that allow for the main flow of the website to function properly. It also contains any repeated sections of forms such as the word inputs which remains constant across custom test forms. Due to custom word entry boxes not being present on all form pages and the different types of tests having different sections of forms this has allowed me to output the correct amount of textboxes with the right formatting easily across the flow of PHP files.

The main flow of the system, after clicking on the chosen type of test, will normally take you to the testbuilder.php page. This page is an initial set-up page of the test. On this page there is an option to name the test, only if you are logged in, an option for the number of words and the test settings options including different ways of displaying results and language options. If you have selected revision tests instead of asking for the length of the test and name it displays a dropdown with different school years in it alongside the test settings options.

The next page you go to after filling out the form is testprocess.php. This page also has a few different functions:

- If you had selected random mode this page will create a test of random words from the English dictionary and then load spelling_test.php with the randomly selected words. For you to complete.
- If you selected custom spelling test you will be shown input boxes that will allow you to enter in each word individually and click submit once the words have been entered. After clicking submit you will be taken to spelling_test_page.php which will process the inputs into the correct format then upload and run the test by passing it to spelling_test.php.
- If you clicked on year revision tests you will be taken to a second menu screen. In this menu you have an option to name the test if you are logged in, an option for the number of words on the test and two submit buttons, one named “Random words” and the other “Custom”. If you have selected “Years 1-6” on the previous page you cannot see any more options. If you selected a specific year then a dropdown with the different statutory requirements from the UK government for the selected year will appear. Clicking on “Random words” will cause the system to look at the statutory requirements for the selected year/ topic and create a test up to 10 words from the topics suggested words. After it will take you to spelling_test.php. If you selected “Custom” it will take you to the custom word input page where any found words relating to the selected category will appear prefilled into the text box. You can change the suggested words in the textboxes or click submit which will redirect to spelling_test_page.php for processing of word inputs before running the test.

Testcodebuilder.php allows for someone without an account to find and complete a spelling test. This was made with the thought that as a teacher you would want you student to easily be able to find and complete the spelling test you made for them. The test creator can see the test code for their test when they create it and by looking at their “all tests” page. This code is entered into a number box on the index.php page which then finds and builds the test for the user corresponding to the test code.

Spelling_test_page.php is a very simple page which is only used when a custom test is created. It will take in all the different entered words and format them into the correct order. Spelling_test.php consists of setting up the text-to-speech and a function that created the spelling test in the correct format with the audio and form element being next to and in line with each other. On the spelling test page if you are not logged in there will be an option to enter in a name for yourself so the creator of the test can identify who did it even without an account. When a spelling test has been completed you are sent to results_page.php where the results are sent to an entry in the database. If there is an entry with the same username already existing for the test it will append the new series of results into a JSON list stored for further data processing when displaying all results.

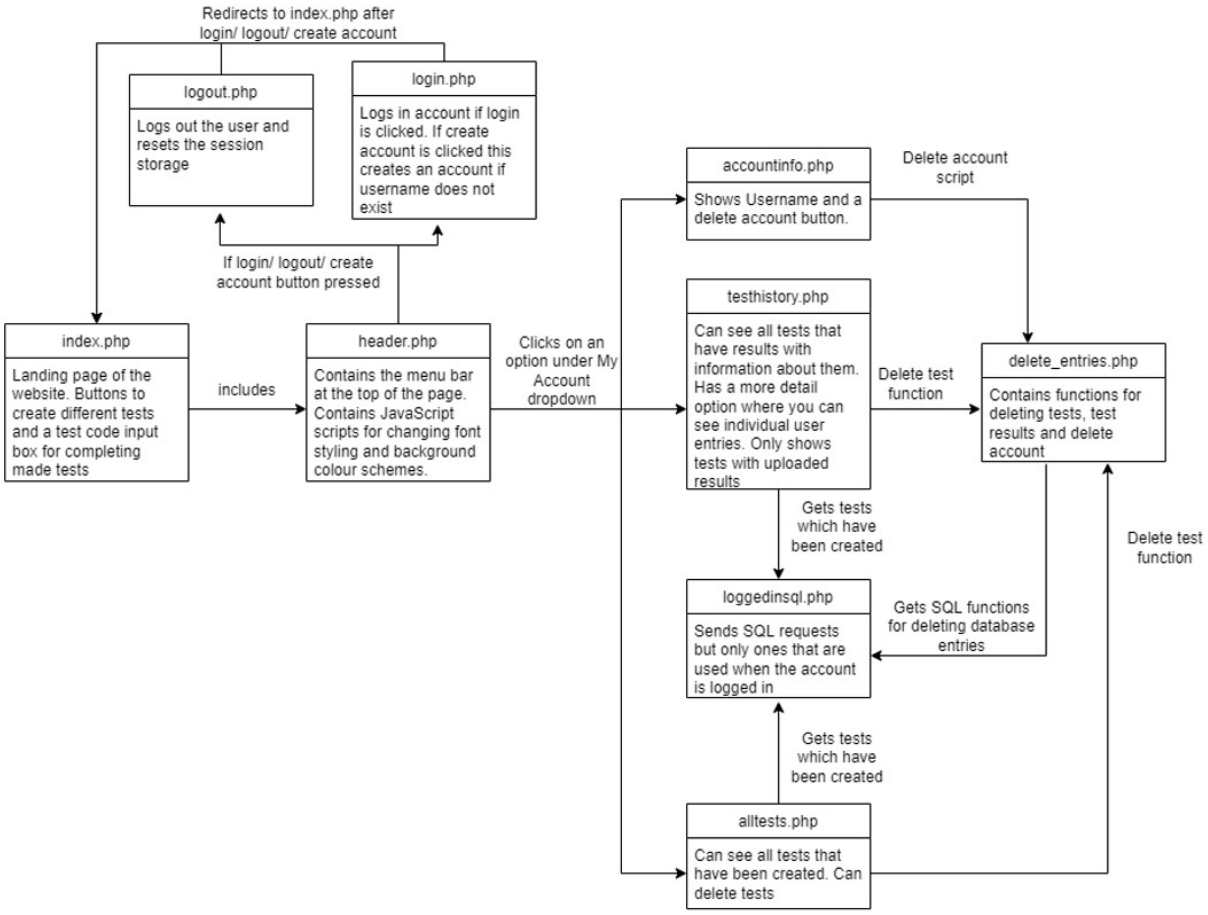
Results_page.php will display the results of the test in a few different formats based on the input results options selected when the test was created. More detail on result options and what they are is under results and evaluation.

All tests feed into spelling_test.php and then end on results_page.php.

Revision mode has changed from what my original idea was for it. In my initial plan my idea was to have a continuous revision mode where you enter in a word and the system reads it then stores the results. Using these results, it would then suggest words from the most frequently incorrect categories of words. The revision mode I have made is a big step that needed to happen before I could code the continuous revision mode. I decided to stop at revision mode due to it taking around a hundred hours of coding only the revision mode to get to the point it is currently at where it is stable and works consistently. I will go into more detail in future work about the continuous revision mode.

Menu/ Header Options and Structure

(**Figure 12:** Structure of the menu at the top of the screen and the different pages it brings you to)



Starting from the landing page index.php there are various menu items in the header. The header is concurrent across all pages on this website to keep the website consistent. The login form has an option for creating an account, currently the only information needed to create an account is a unique username and a password. If the username exists then it will display an error to the user telling them to enter a different username. When you click either “login” or “create account” it will redirect you to login.php where it completes the checks and if an account is able to be created it will create the account. If the account has been created it will redirect to index.php where a message saying the account has successfully made will be displayed. If the user is logging in login.php will retrieve the relevant account information and store it in session storage for later use. When you are logged in a dropdown label “my account” will appear in place of the login form. When hovering on the dropdown menu a few options appear with one being logout. When the user clicks on logout you are redirected to logout.php where all session storage is reset. After it has reset the session storage it will redirect to index.php.

Under the dropdown “my account” there are three more options. The first option is “Test History”, when the user has clicked on this it will go to testhistory.php where a history of the completed tests will be displayed. Only tests that have results submitted will display on this page. There is some information about the tests displayed including the test name, test code, times completed and the average score in percent of the results submitted. There are also two buttons, one to delete the test and one to see more details about the test. If you click on more details it will reload testhistory.php but instead of showing each of the different tests it will show you all the results of the selected test. At the top of the page it displays the test code and a back button. If the user has an account all their scores will be displayed in one chart. If the user does not have an account they will have separate

entries displaying their ‘name’ entered when completing the test. If the user has completed the test multiple times with an account it will display all of their results on one bar chart with the score for each attempt in chronological order. It will also display the username next to the results if the results are from an account. The total amount of times that the test has been attempted and the total number of words in the test is displayed.

The next item in the dropdown is “All Tests”. When this button is clicked it will display all tests created by the user in a list. This list contains the test code, test name, word list and a delete test button. I decided to include this page as a quick reference for the user to see all their selected tests and have the ability to delete them. This also allows users to look at all their test codes if they are sharing the test.

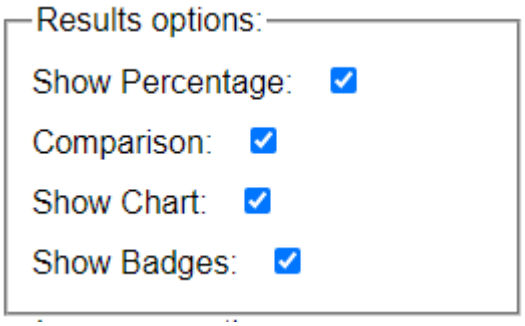
The final option in the dropdown is called “Settings”. This option takes you to accountinfo.php. On this page you will see your account name and an option to delete your account. Although this page does not have much on it I thought that having a way to delete account through the website was a necessary feature.

Database implementation

The database was implemented as designed. The only bit that was different to my initial plan for the database is that “wordlist” in the table “user” has not been used due to the approach to use case 7 being different to what was the initial plan. I have kept in the column “wordlist” due to plans for future work that would expand on use case 7, more of an explanation will be in future work.

Results options in more detail

(**Figure 13:** Screenshot of the results options)



The results options decided on in the end were as shown above. Show percentage shows the percentage score of the test rounded to one decimal place. The comparison displays what the word is according to when it is created alongside the answer that was submitted by the user completing the test. Under the users entered word it will either display “correct” or “incorrect” in bold to allow the user to easily identify their answers.

The third option, “Show Chart”, allows for a bar chart to be displayed on the results page with the words on the bottom and the bars showing whether the answer was correct or incorrect. The results are fed into the same function that created the charts in “Test History” to reduce duplicated code.

The final option “Show Badges” allows for a ‘badge’ to be given. Currently there are no images for the badges only the names of the badge. This was implemented as an alternative to showing results and scores. The badges are based off golf scoring as I thought they would be recognisable but not inappropriate for primary school children. The badges and what score they display at are as below:

(**Figure 14:** Screenshot of the different badges and the scores needed to attain them)

```
'Bogey Badge'; # <50%
'On Par'; # <60%
'Birdie Badge'; # <70%
'The Eagle'; # <80%
'Albatross'; # <= 100%
```

The main reason for having badges is that as a child completing a test and being given a score instantly could demoralize them if they get a low score. Although “Bogey Badge” might not sound the best I thought that as a child it would be more fun than upsetting. For this reason I also made it so any score under 50% would show the “Bogey Badge”.

Results and Evaluation

Integration testing

For integration testing I will carry out tests based on the use cases that were chosen. This will test all use case requirements created through user stories. The test will test out the various functions that you can use the system for. These will be outlined in test case 1-7 representing the use case 1-7.

These tests will be run with multiple spelling tests and different accounts to make sure the testing is well rounded. Images will be attached as evidence for each step showing the outcome of each step.

Results

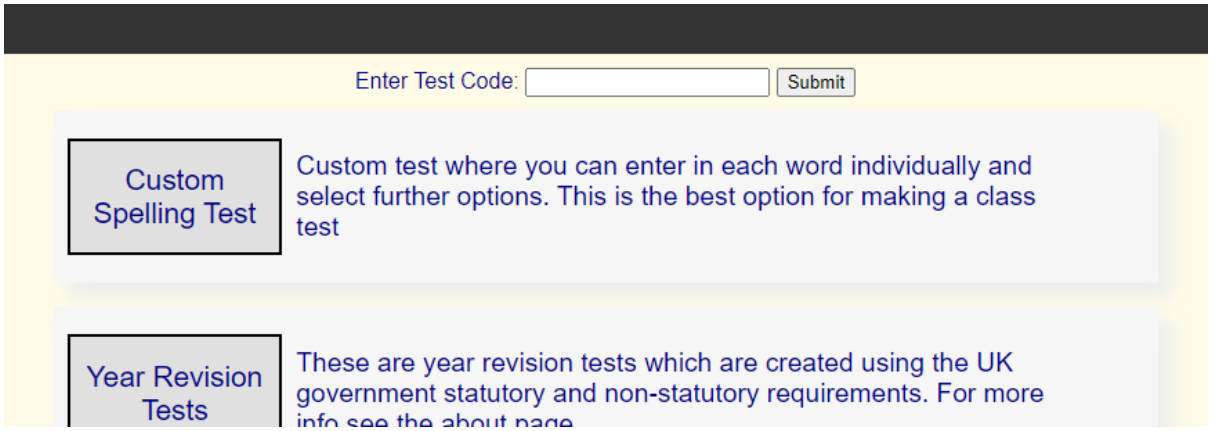
(**Table 2:** Test case 1)

Test Case ID: 1		**Test Purpose:** Demonstrate a **s**pelling test creation	
Preconditions: the user has clicked on custom spelling test on the home page while logged in			
Test Case Steps:			
Step No	**Procedure**	**Response**	**Pass/Fail**
1	The user clicks on “Custom Spelling Test”	Taken to a new screen with the options for test name, length of test, word definitions, results options and language options	Pass
2	The user fills out the test options and clicks create test	The user is taken to a screen where they have a form to enter in words for a test	Pass
3	The user fills out the words in the form and clicks on submit	The user is taken to a test screen where they can see the test code for later use. Test is uploaded to the database	Pass
4	The user can demo the test	Test results from demo not uploaded	Pass
Date: 11/05/2022			

Evidence

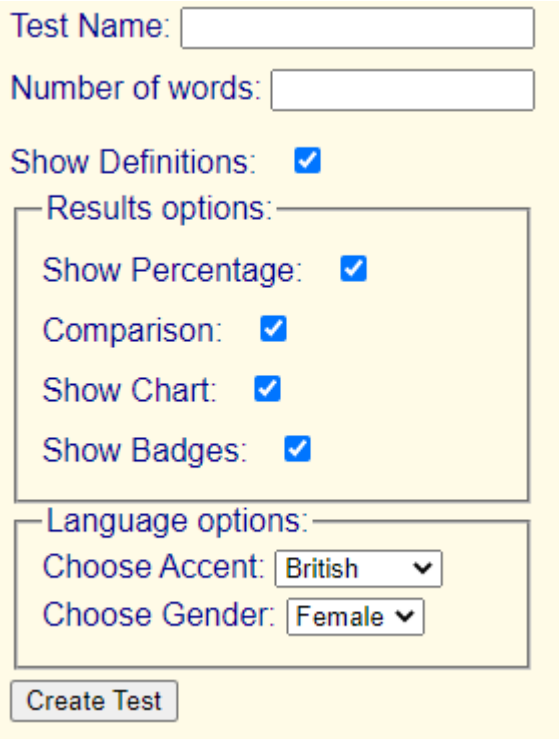
Step 1.

(**Figure 15:** Step 1 for test case 1)



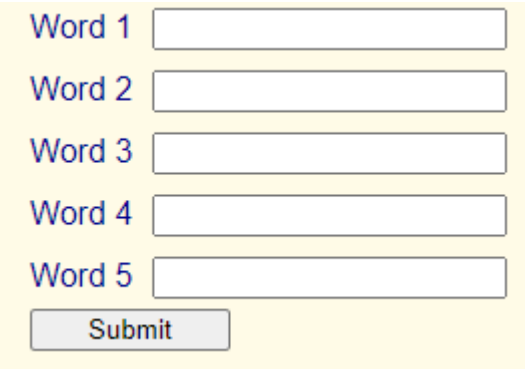
Step 2:

(**Figure 16:** Step 2 for test case 1)



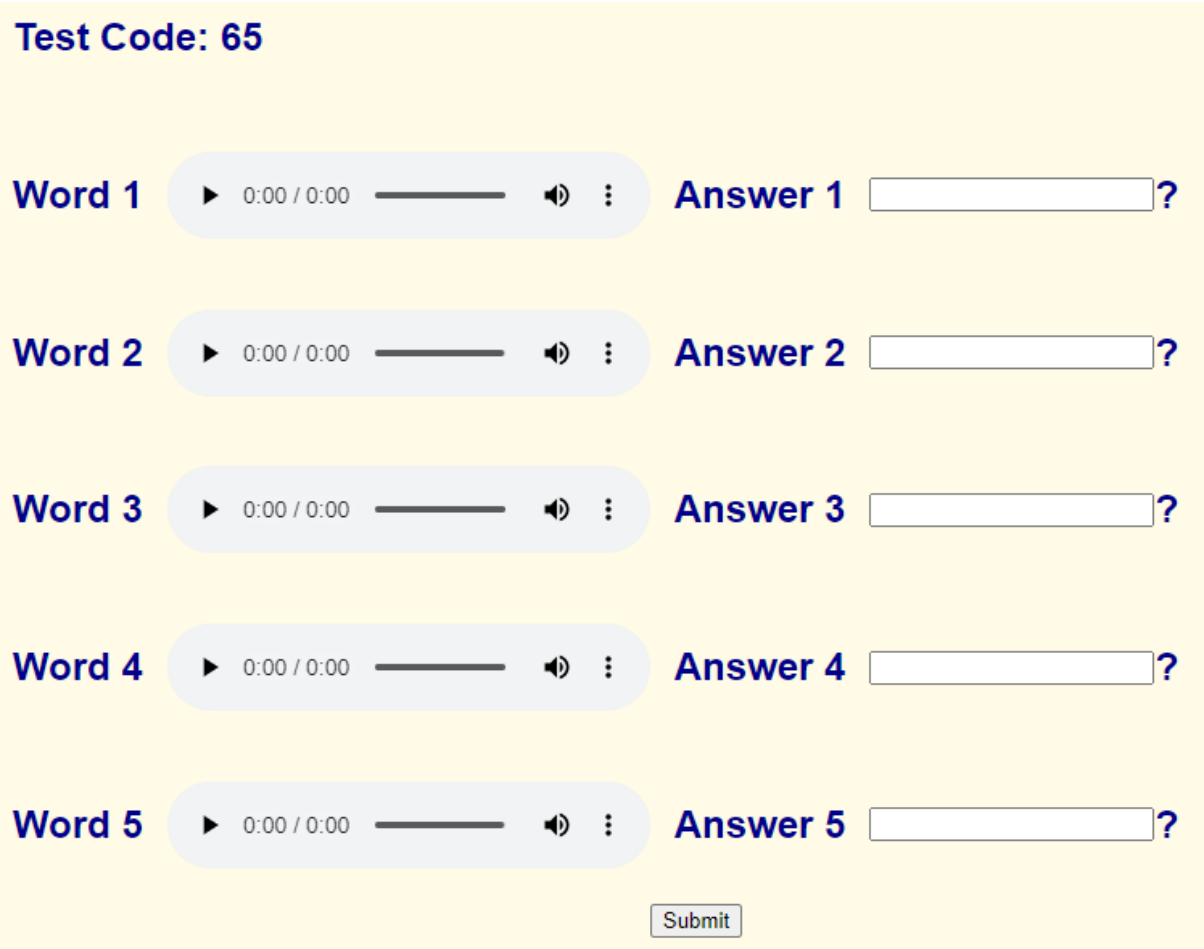
Step 3:

(**Figure 17:** Step 3 for test case 1)



Step 4:

(**Figure 18:** Step 4 for test case 1)



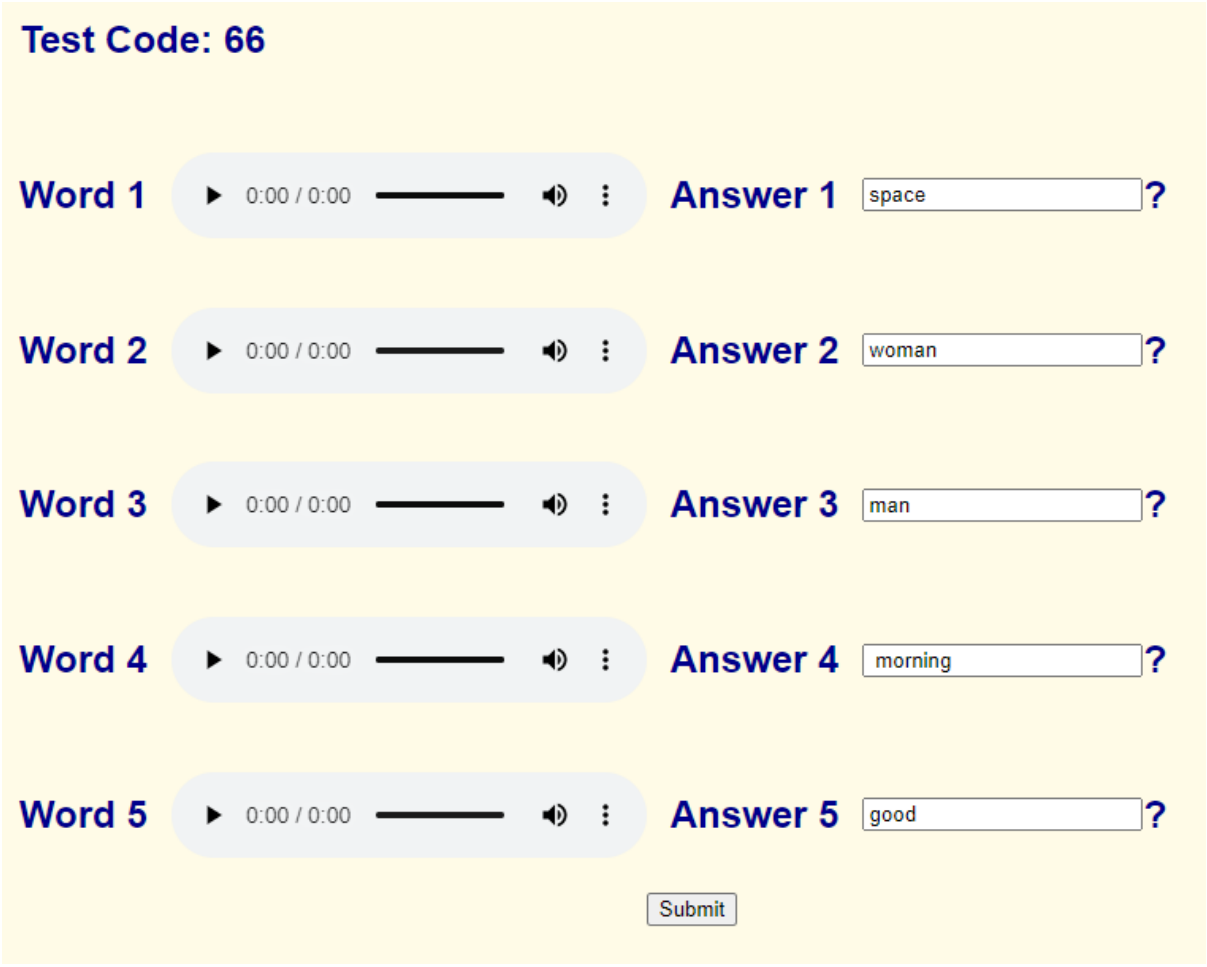
(**Table 3:** Test case 2)

Test Case ID: 2		**Test Purpose:** Show the results page that appears when you complete a test and uploads the result of able to	
Preconditions: The user is able to complete a test			
Test Case Steps:			
Step No	**Procedure**	**Response**	**Pass/Fail**
1	The user has completed a spelling test and clicks on submit	The user is taken to the results page	Pass
2	The user can see the different selected results options	The user can click on the back button to go to the home screen. If the test was created by a user with an account upload the result of the test	Pass
Date: 11/05/2022			

Evidence

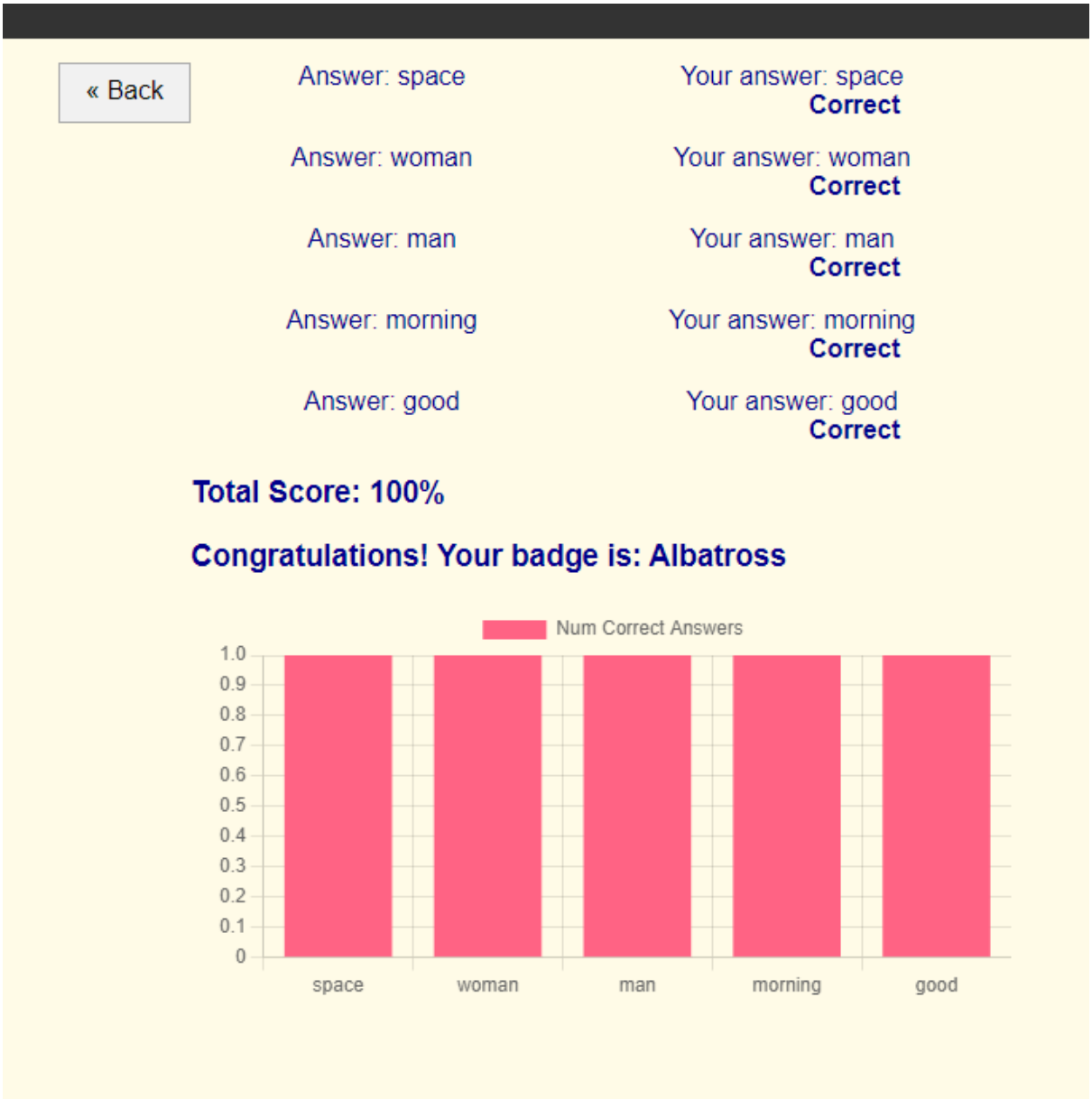
Step 1:

(**Figure 19:** Step 1 for test case 2)

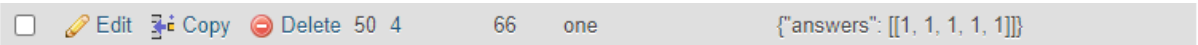


Step 2:

(**Figure 20:** Step 2 for test case 2)



(**Figure 21:** Step 2 for test case 2, uploaded result)



(**Table 4:** Test case 3)

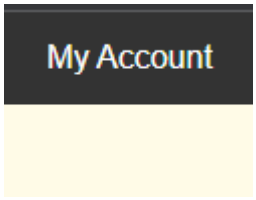
Test Case ID: 3		**Test Purpose:** See the test history of the users tests	
Preconditions: The user has created tests before and has results for those tests			
Test Case Steps:			
Step No	**Procedure**	**Response**	**Pass/Fail**
1	User is logged in	Top menu shows “My Account”	Pass
2	User hovers over “My Account”	Dropdown appears with the option “Test History”	Pass

3	User clicks on “Test History”	User is taken to the test history page showing all tests which previously have tests results uploaded	Pass
4	The user clicks on see more details	The user is taken to a page where a breakdown of the individual submissions for the test are shown. Results with multiple entries are shown in chronological order	Pass
Alternative Flow: User wants to see all tests they have created, continues from previous step 2			
3	The user clicks on “All Tests”	The user is able to see a screen with all their created tests on one screen	Pass
Date: 11/05/2022			

Evidence

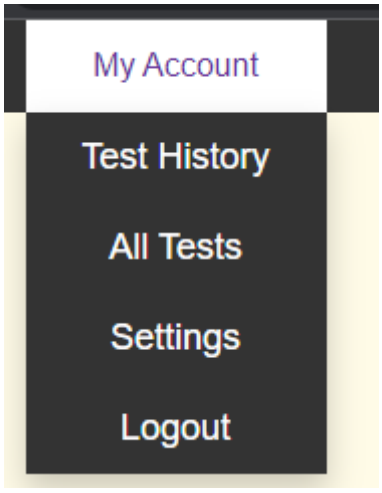
Step 1:

(**Figure 22:** Step 1 for test case 3)



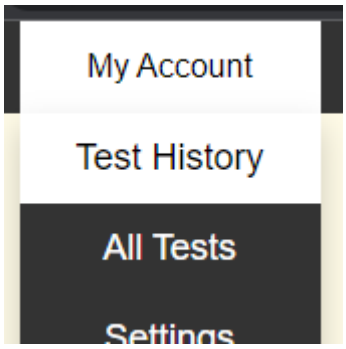
Step 2:

(**Figure 23:** Step 2 for test case 3)

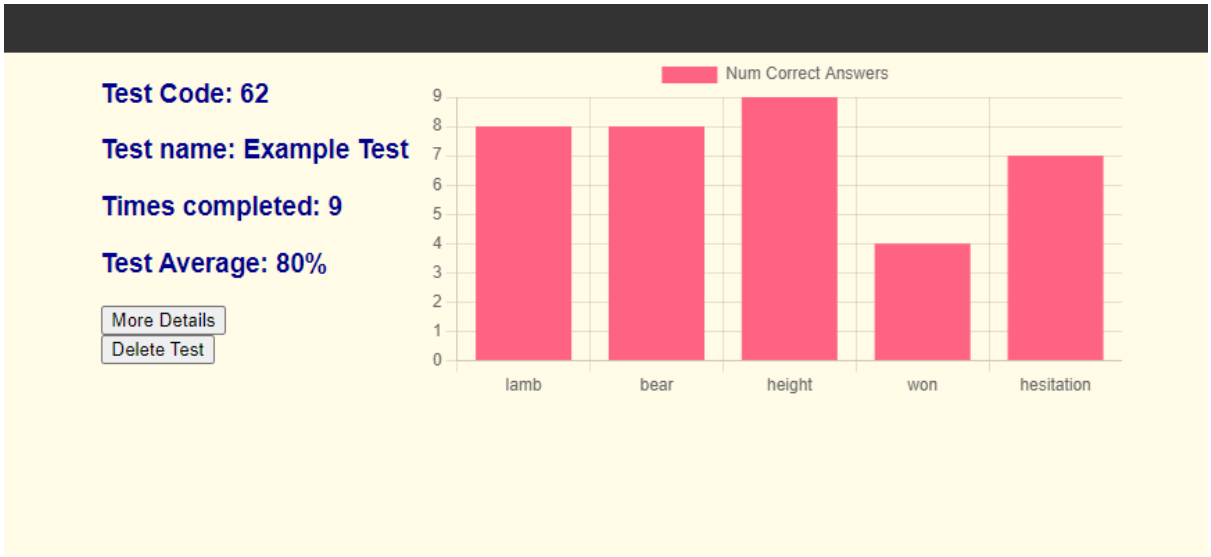


Step 3:

(**Figure 24:** Step 3 for test case 3)

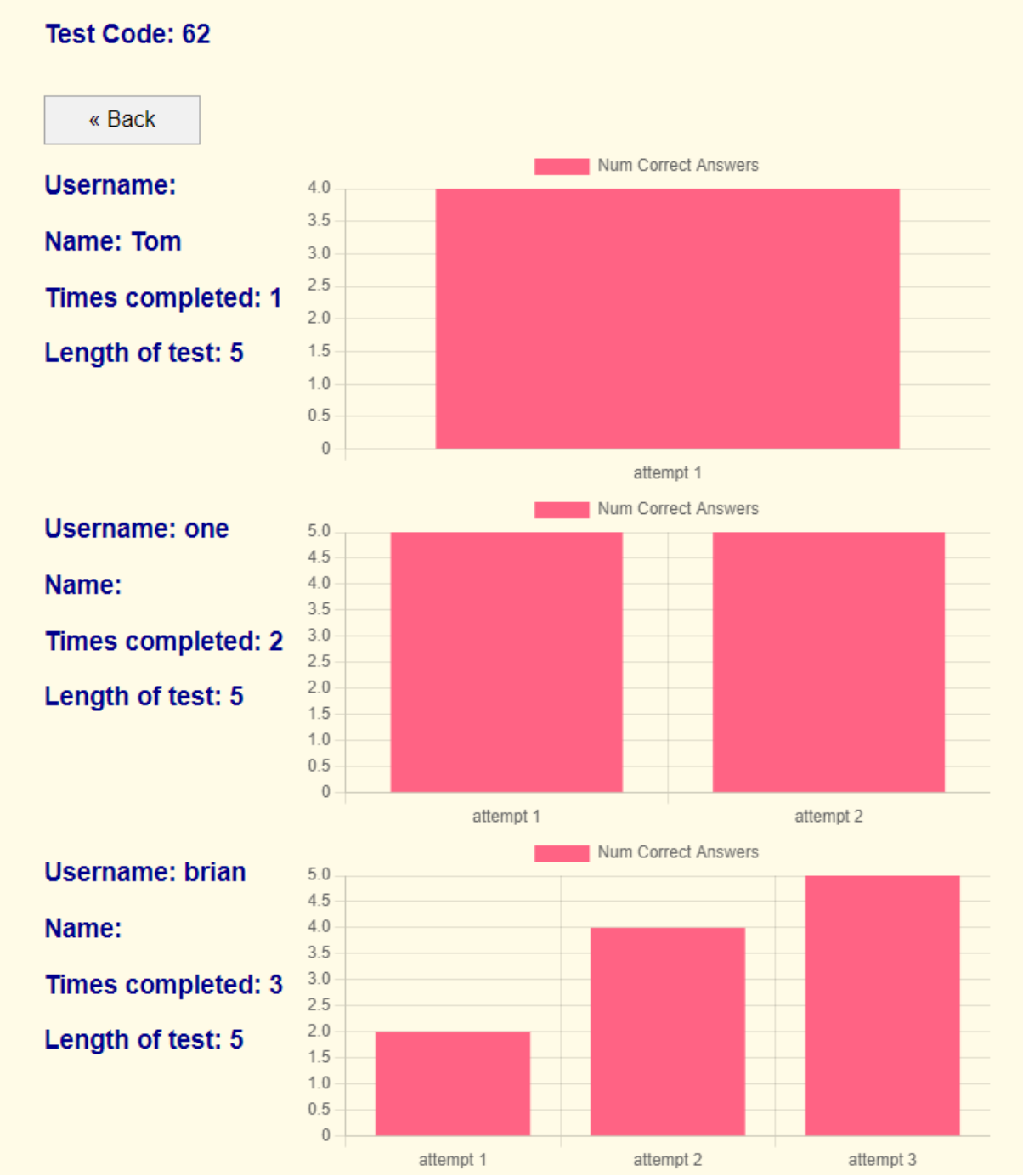


(**Figure 25:** Step 3 for test case 3)



Step 4:

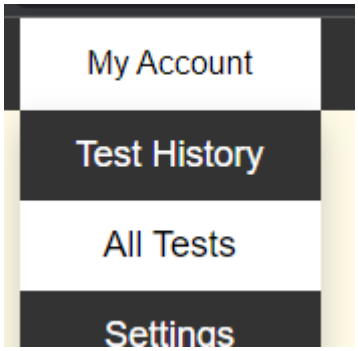
(**Figure 26:** Step 4 for test case 3)



Evidence for Alternative flow

Step 3:

(**Figure 27:** Step 3 for test case 3 alternative flow)



(**Figure 28:** Step 3 for test case 3 alternative flow)

(**Table 5:** Test case 4)

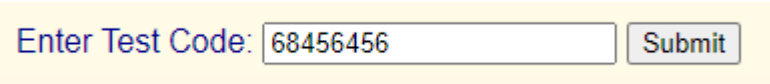
Test Case ID: 4		**Test Purpose:** Complete a spelling test	
Preconditions: A spelling test must have been created beforehand and the correct code entered into the test code from on the home page.			
Test Case Steps:			
Step No	**Procedure**	**Response**	**Pass/Fail**
1	The user enters in a test code on the home page	1. If test code not found tell user 2. Test code is found and test loaded	Pass
2	Test is loaded and if user is not logged in they can enter in a 'name'. If logged in username is taken for identification. User	The user is taken to the results page	Pass

	then fills in the test form and hits submit		
3	The user can see the results based on the selected result settings	The system uploads the results from the user completing the test to the database	Pass
4	User is logged in and it is not the first time completing the test	They system retrieves their previous result and appends the new results to the end to keep a record of results	Pass
Date: 11/05/2022			

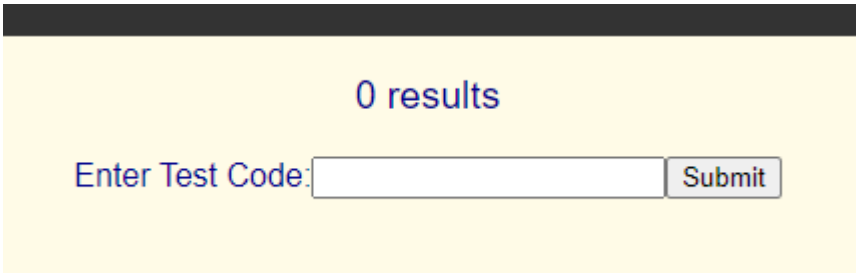
Evidence

Step 1:

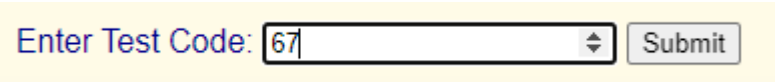
(**Figure 28:** Step 1 for test case 4)



(**Figure 29:** Step 1 for test case 4, no results found)

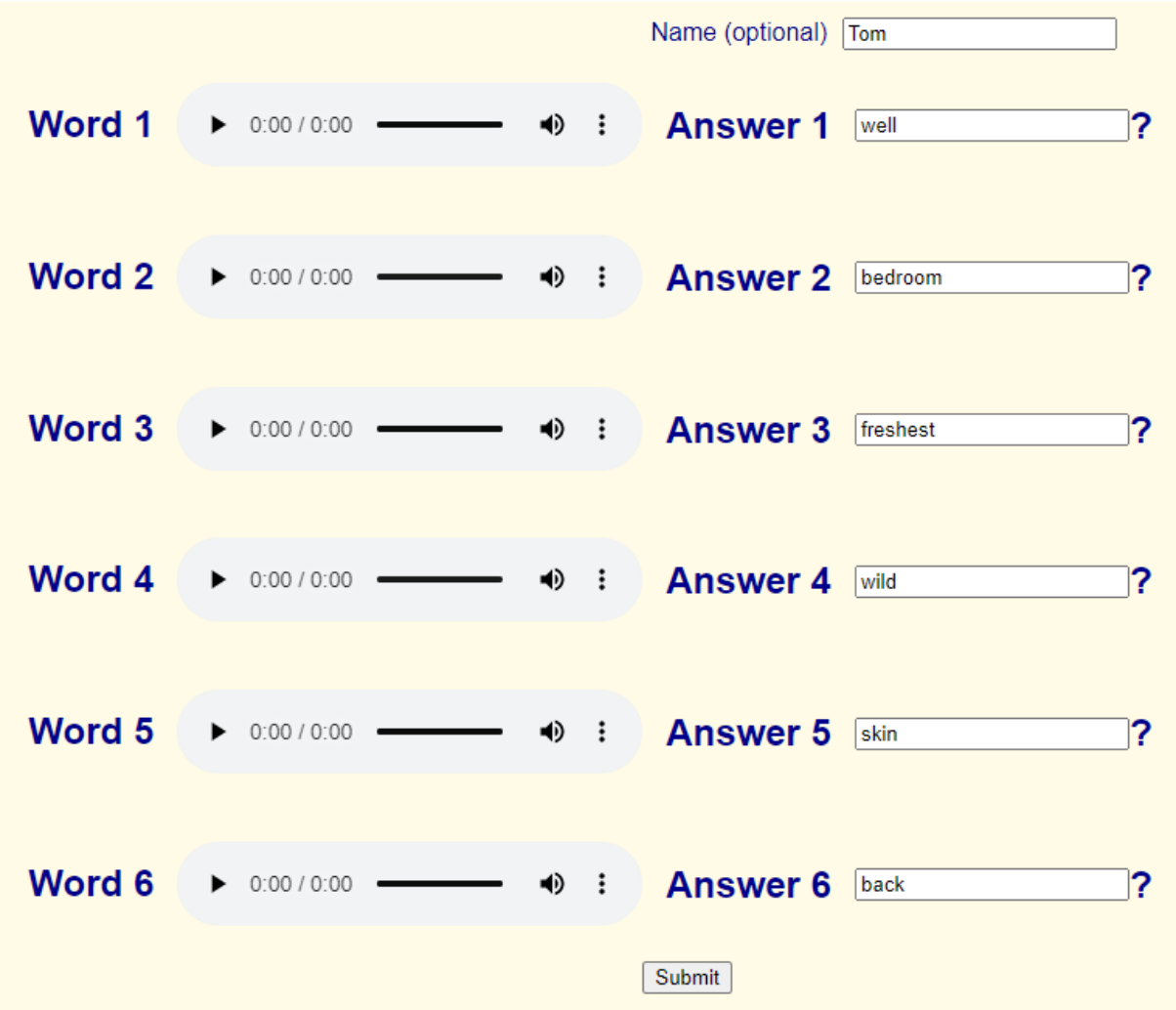


(**Figure 30:** Step 1 for test case 4)



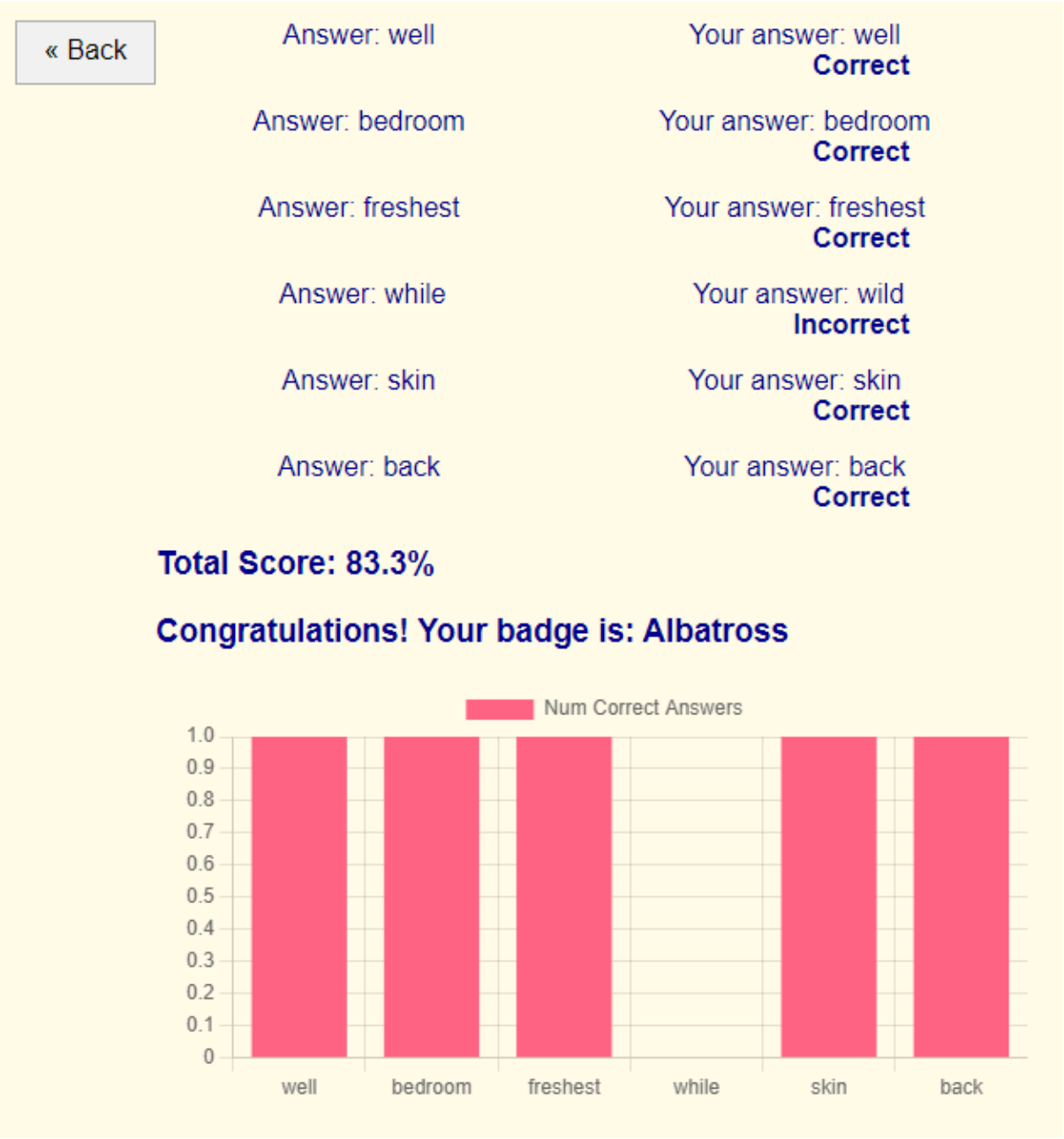
Step 2:

(**Figure 31:** Step 1/2 for test case 4, result found)



Step 3:

(**Figure 32:** Step 3 for test case 4)

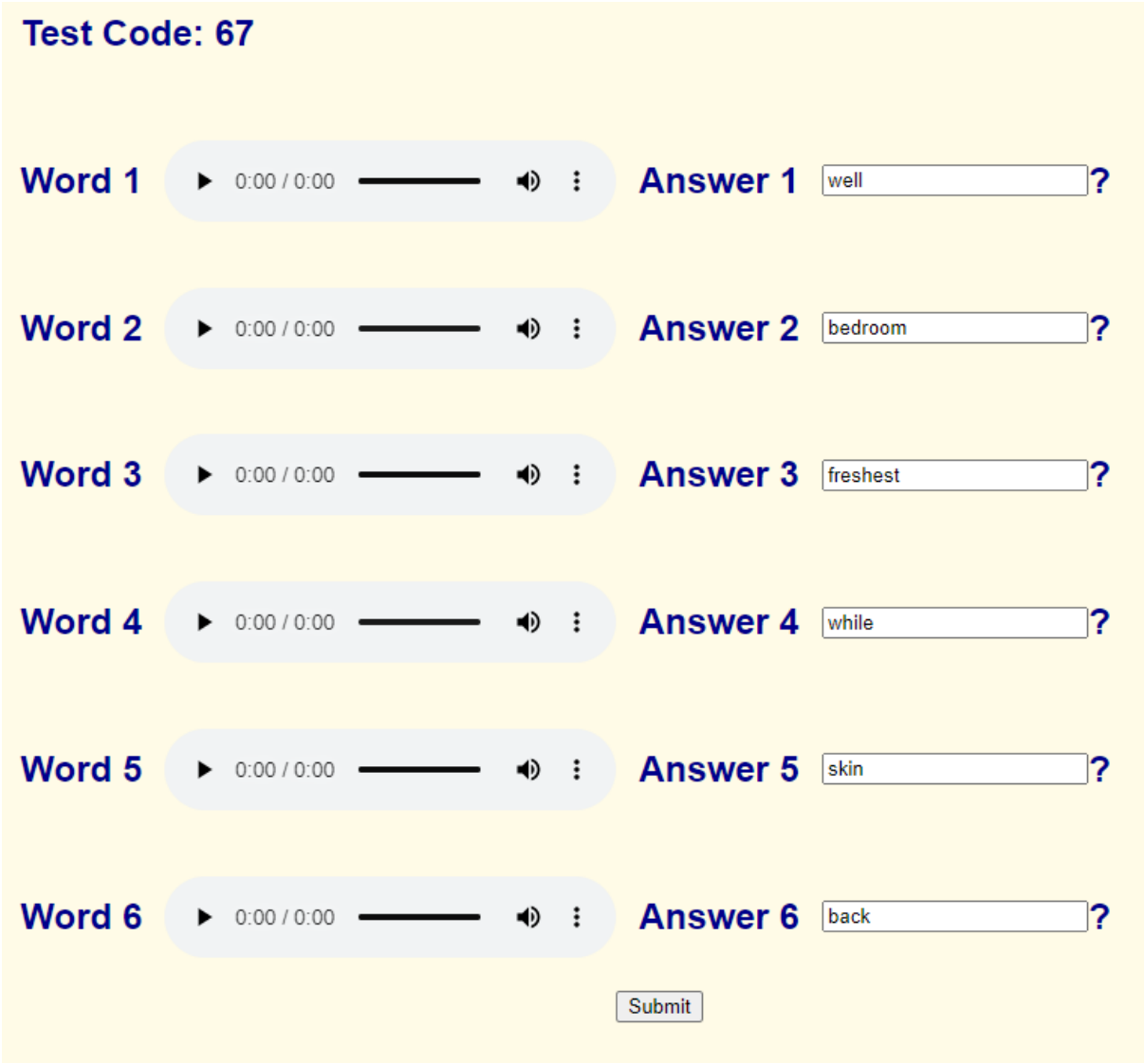


(**Figure 33:** Step 3 for test case 4, evidence of result taken with 'name' stored)

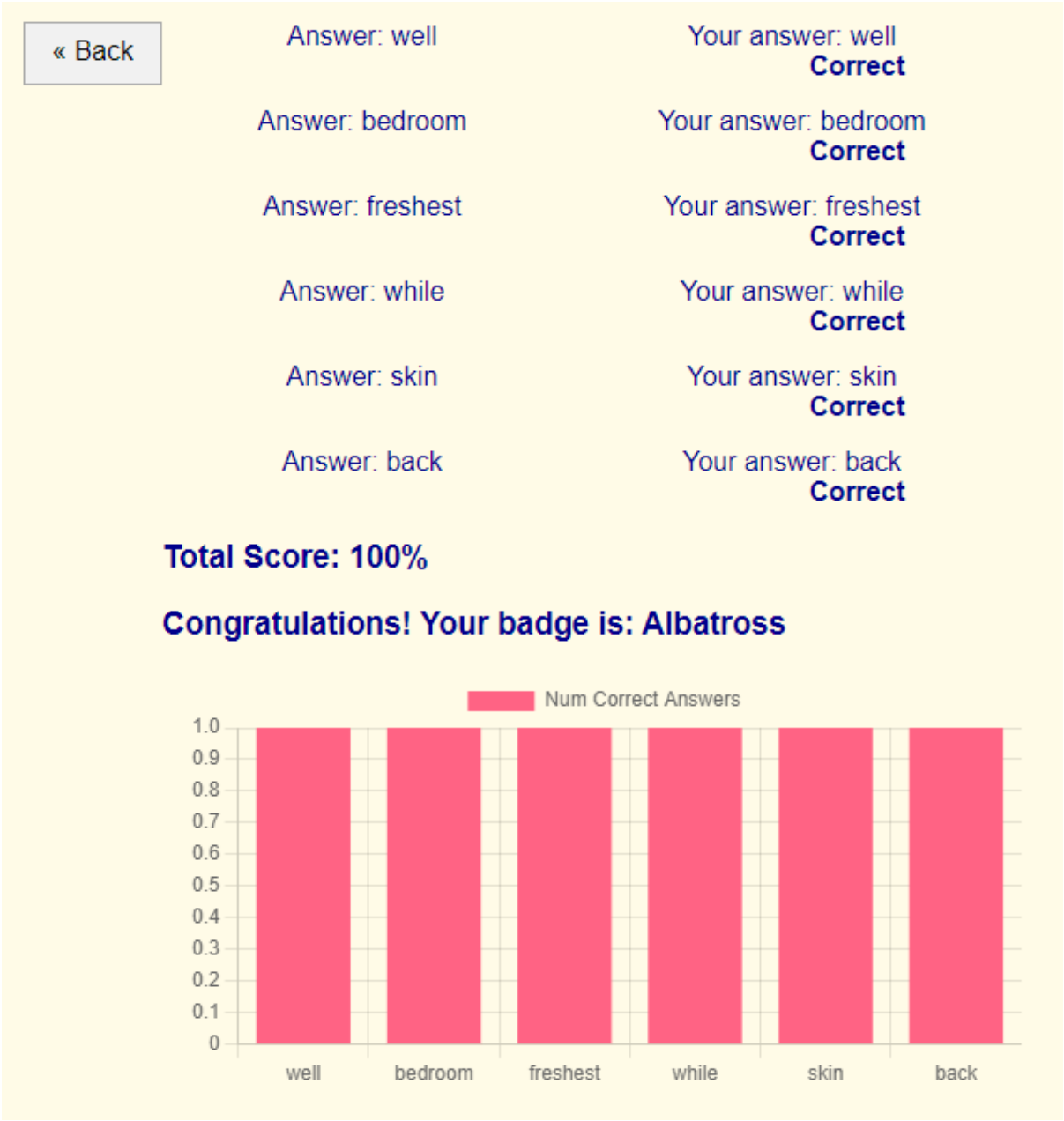
ete	51	5	67	Tom	{"answers": [[1, 1, 1, 0, 1, 1]]}

Step 4:

(**Figure 34:** Step 4 for test case 4)



(**Figure 35:** Step 4 for test case 4)



(**Figure 36:** Step 4 for test case 4, evidence of multiple results for one user)

(**Table 6:** Test case 5)

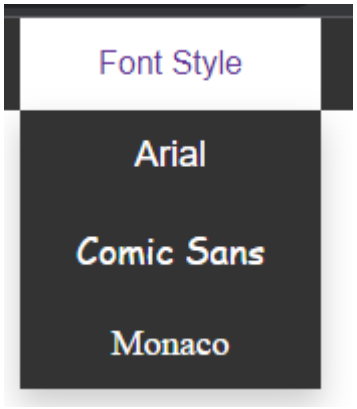
Test Case ID: 5		**Test Purpose:** Change font style, size and background colour scheme	
Preconditions: The user is on the website			
Test Case Steps:			
Step No	**Procedure**	**Response**	**Pass/Fail**
1	User selects hovers over “Font Style”	Dropdown appears with options “Arial”, “Comic Sans”, “Monaco”	Pass

1.1	User clicks on “Arial”	Font changes style to “Arial”	Pass
1.2	User clicks on “Comic sans”	Font style changes to “Comic sans”	Pass
1.3	User clicks on “Monaco”	Font style changed to “Monaco”	Pass
2	User hovers over “Font Size”	Dropdown appears with the options “Default”, “Large”, “X-Large”	Pass
2.1	User clicks on “Default”	The font size changes to the default size of 1rem	Pass
2.2	User clicks on “Large”	The font size changes to the medium size of 1.2rem	Pass
2.3	User clicks on “X-Large	The font size changes to the default size of 1.5rem	Pass
3	User hovers over “Colour Scheme”	Dropdown with the different colour scheme options appears. Options are “White”, “Cream”, “Orange”, “Dark”	Pass
3.1	User clicks on the colour scheme “White”	Colour scheme changes to “white” with black text colour	Pass
3.2	User clicks on the colour scheme “Cream”	Colour scheme changes to “cream” with dark blue text	Pass
3.3	User clicks on the colour scheme “Orange”	Colour scheme changes to “orange” with very dark blue text	Pass
3.4	User clicks on the colour scheme “Dark”	Colour scheme changes to “dark” with very light grey text	Pass
Date: 11/05/2022			

Evidence

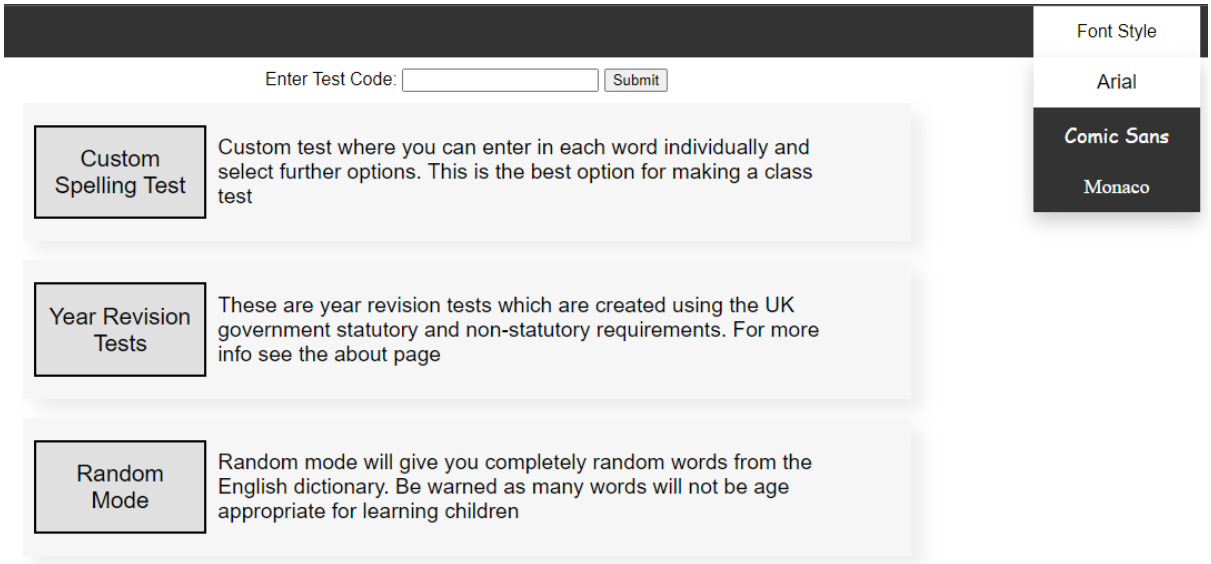
Step 1:

(**Figure 37:** Step 1 for test case 5)



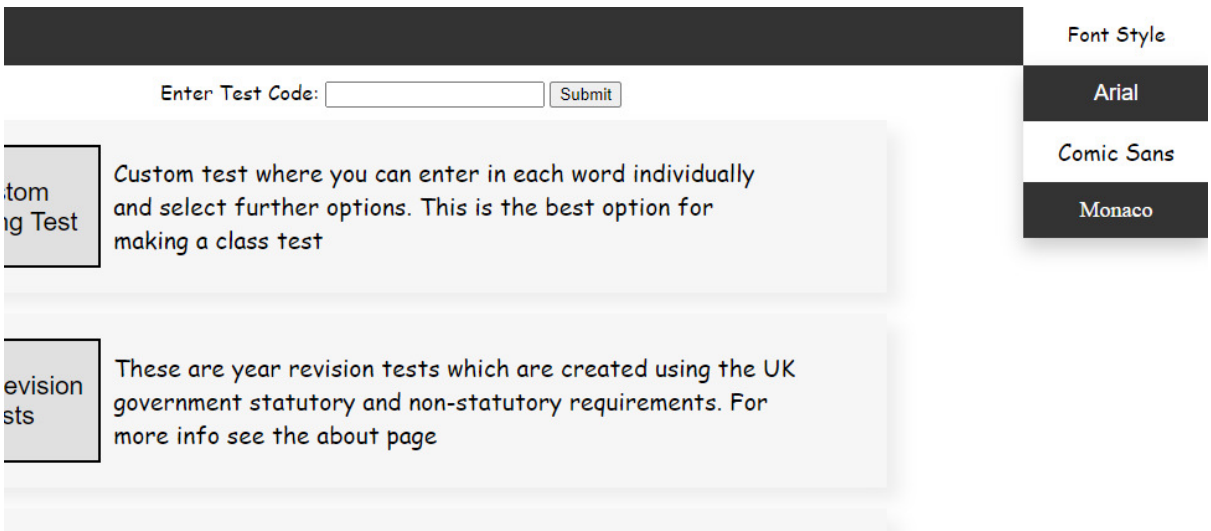
Step 1.1:

(**Figure 38:** Step 1.1 for test case 5)



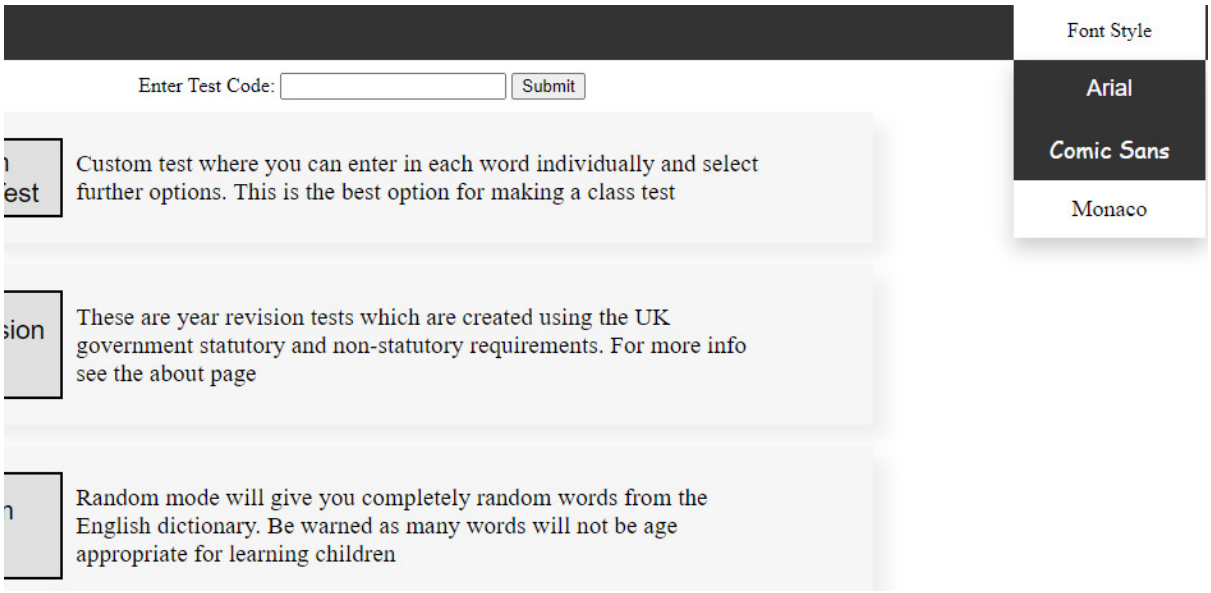
Step 1.2:

(**Figure 39:** Step 1.2 for test case 5)



Step 1.3:

(**Figure 40:** Step 1.3 for test case 5)

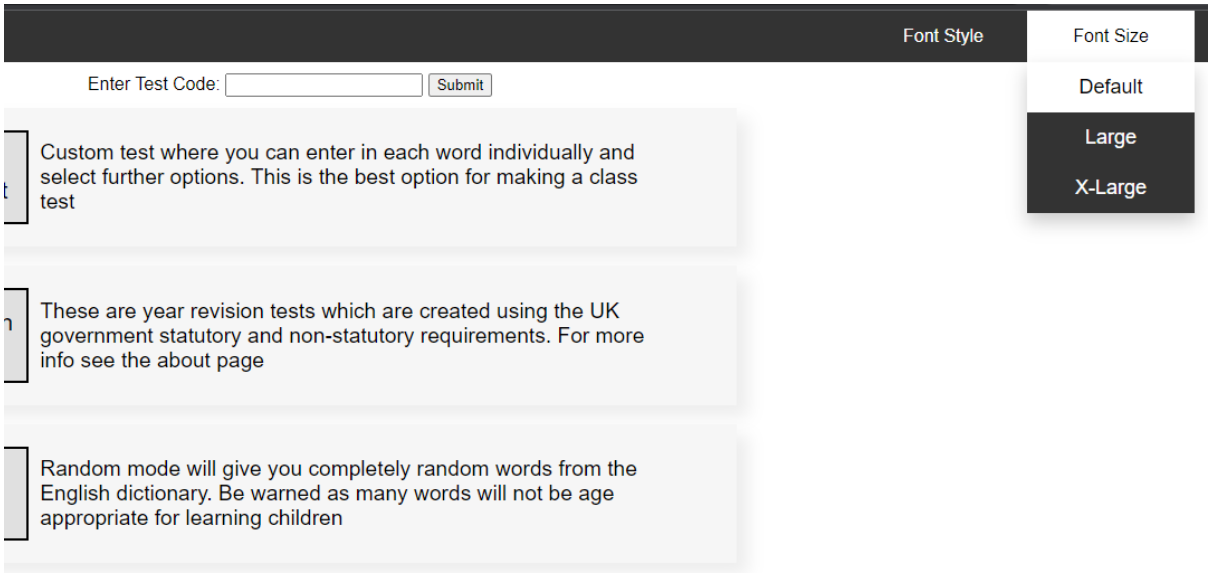


Step 2:

(**Figure 41:** Step 2 for test case 5)

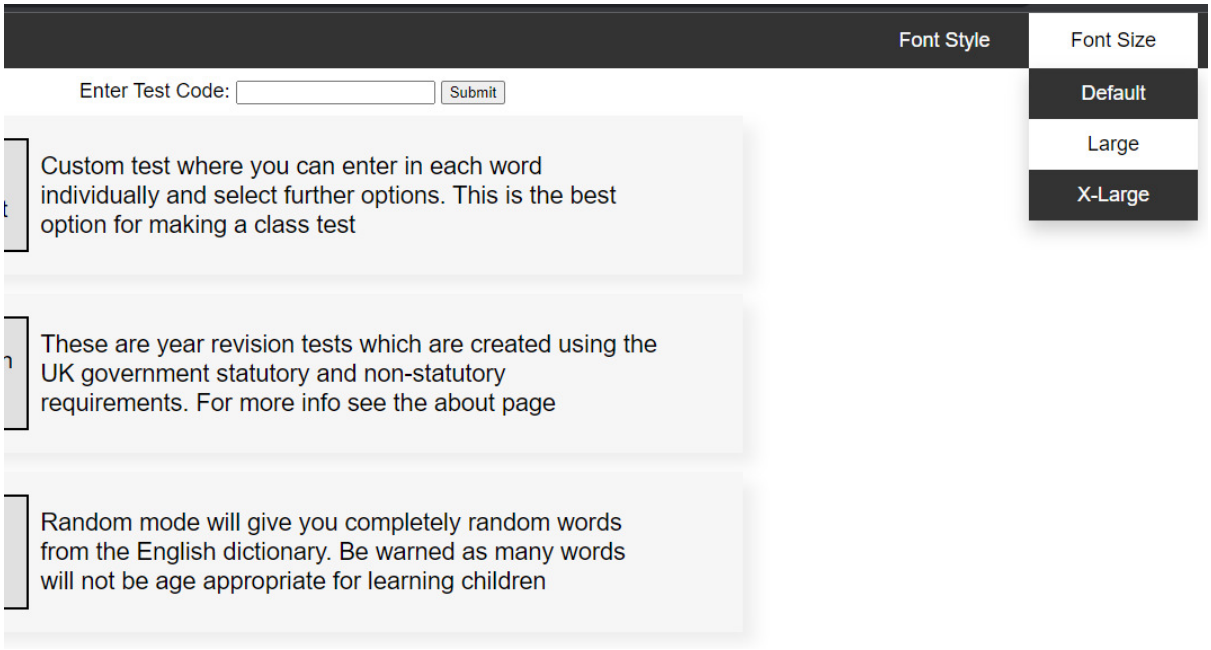
Step 2.1:

(**Figure 42:** Step 2.1 for test case 5)



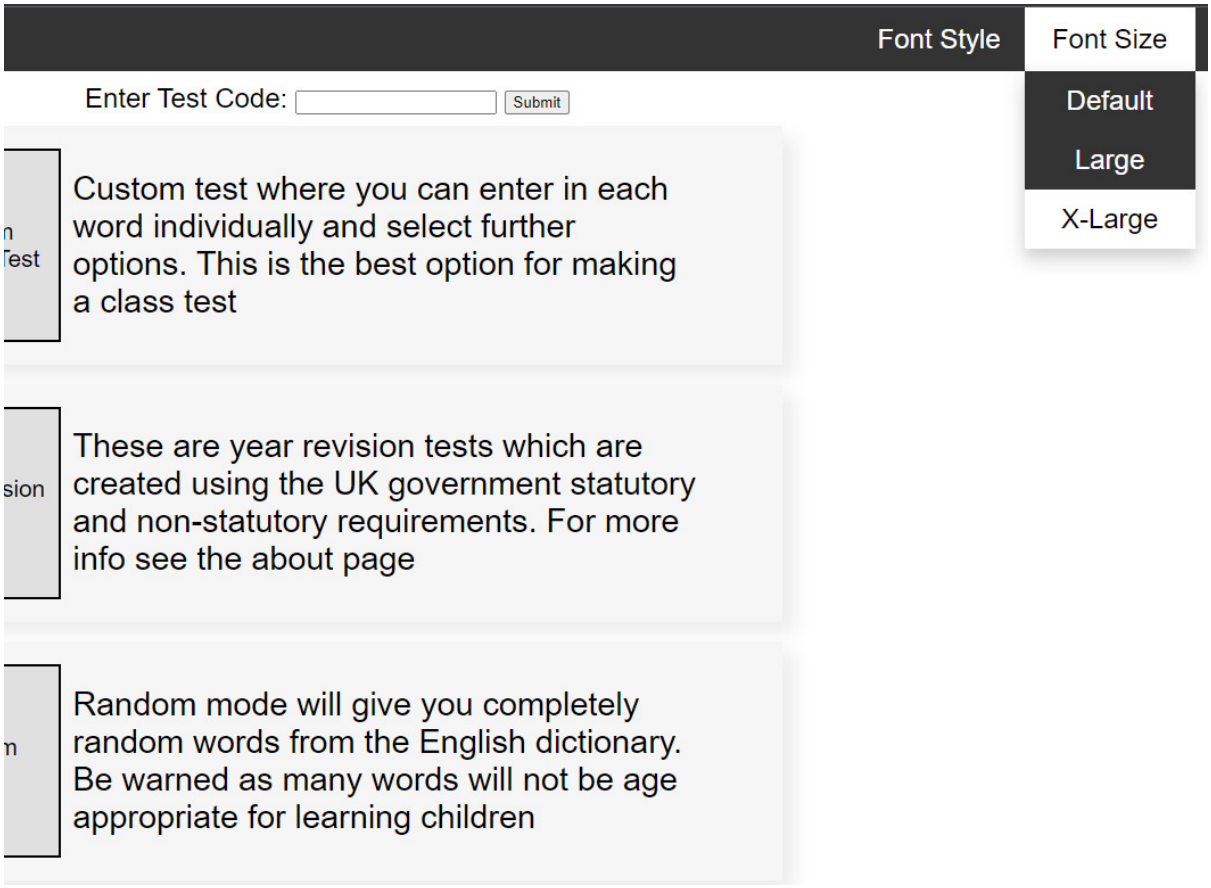
Step 2.2:

(**Figure 43:** Step 2.2 for test case 5)



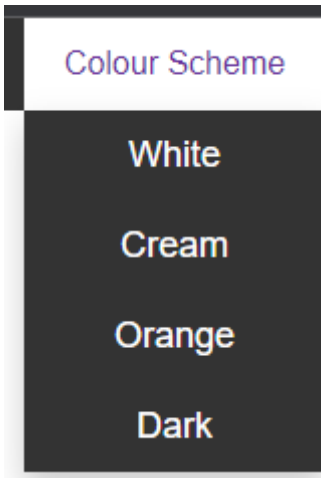
Step 2.3:

(**Figure 44:** Step 2.3 for test case 5)



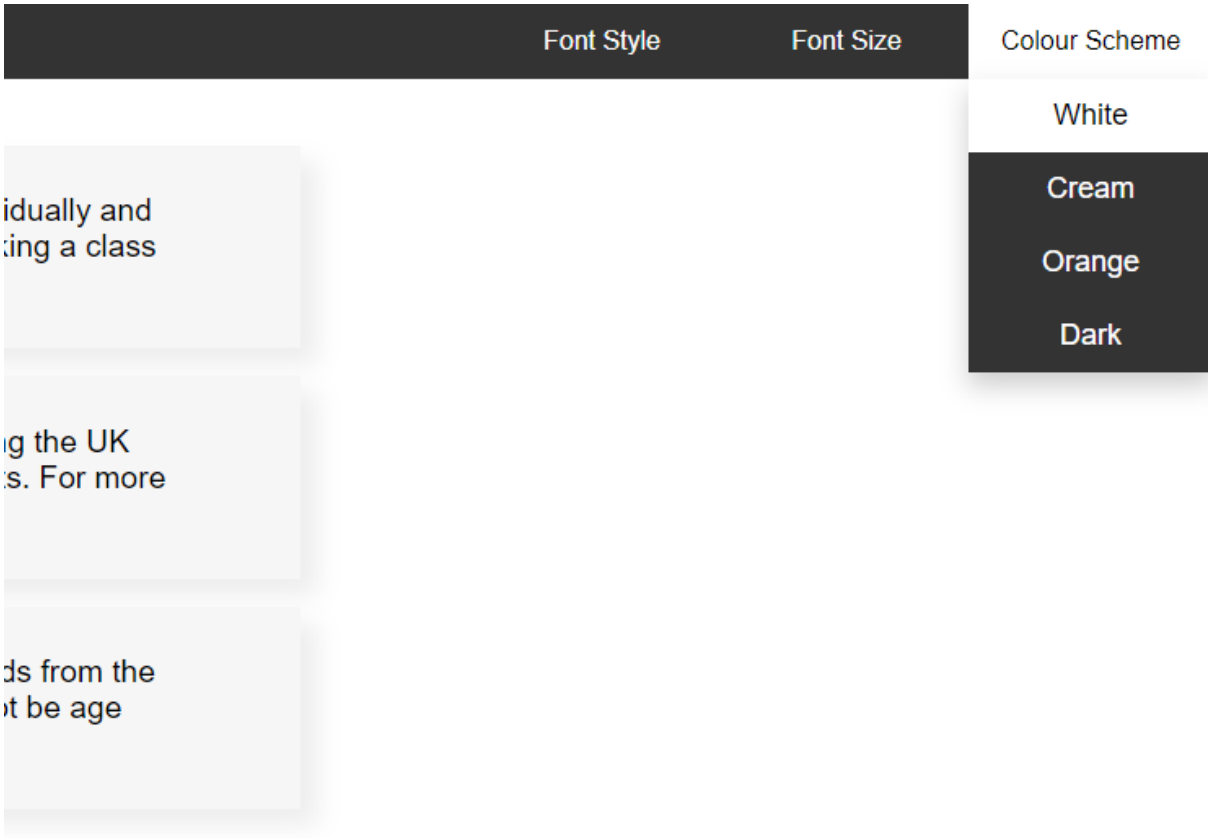
Step 3:

(**Figure 45:** Step 3 for test case 5)



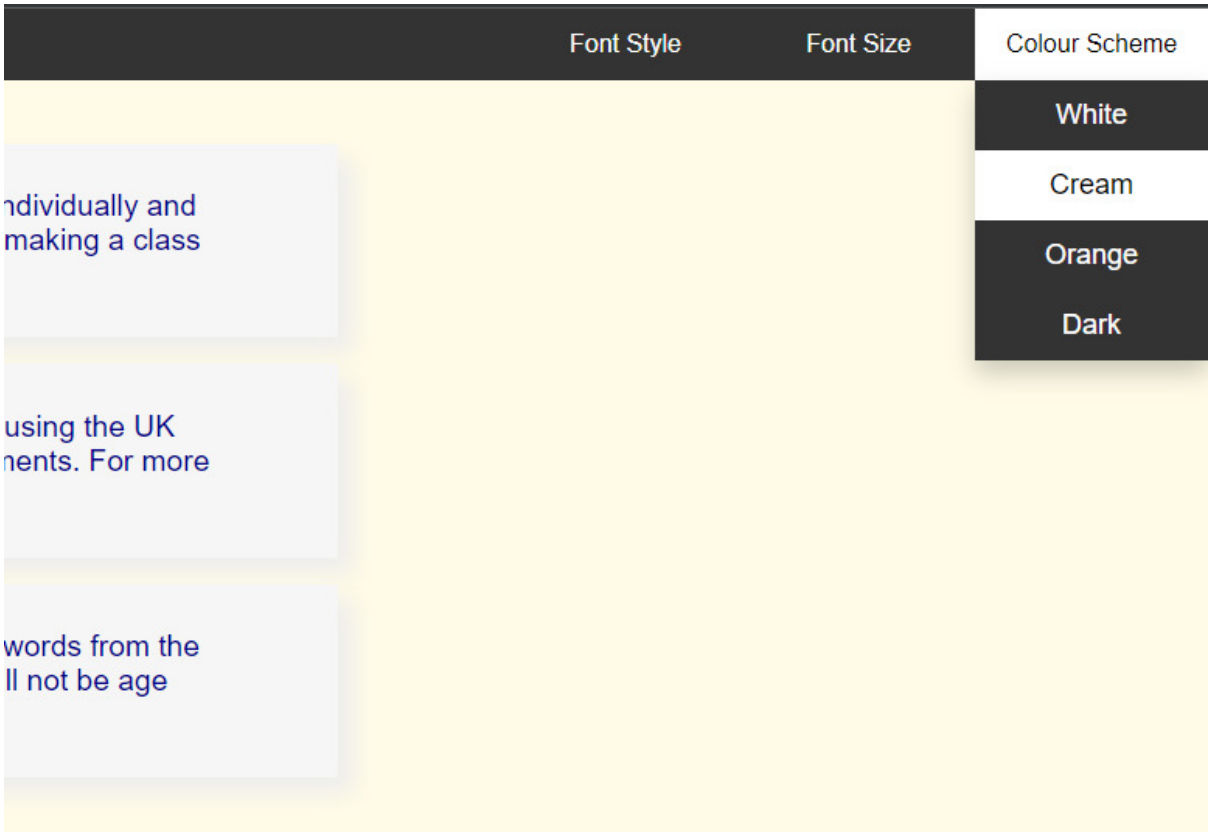
Step 3.1:

(**Figure 46:** Step 3.1 for test case 5)



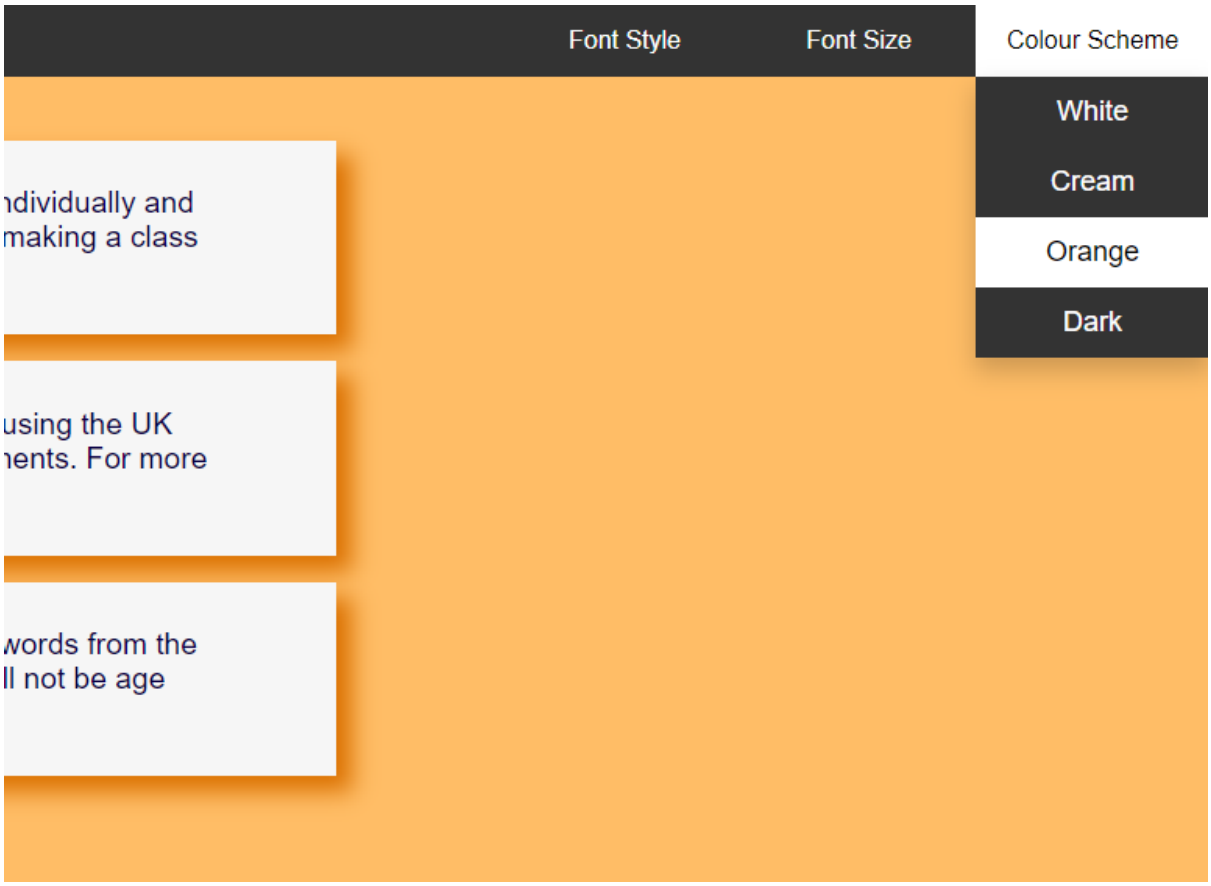
Step 3.2:

(**Figure 47:** Step 3.2 for test case 5)



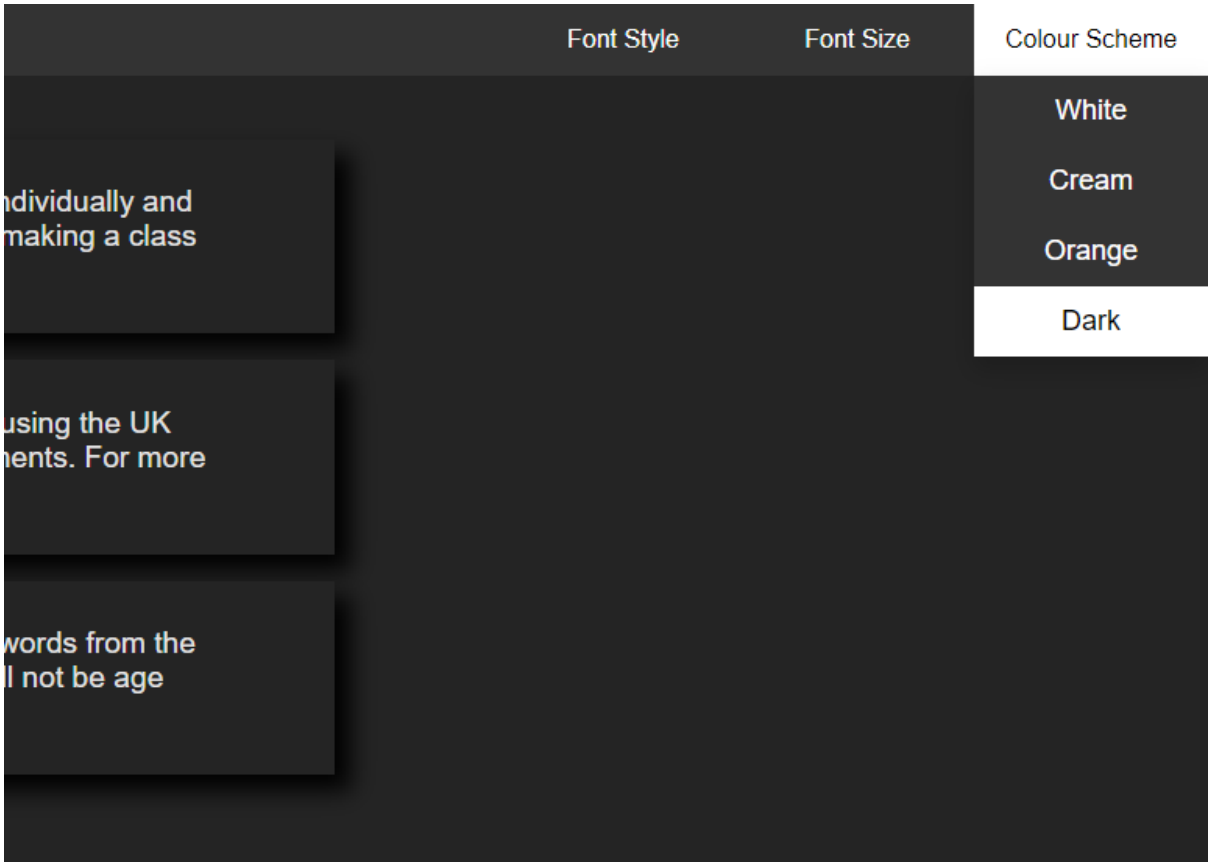
Step 3.3:

(**Figure 48:** Step 3.3 for test case 5)



Step 3.4:

(**Figure 49:** Step 3.4 for test case 5)



(**Table 7:** Test case 6)

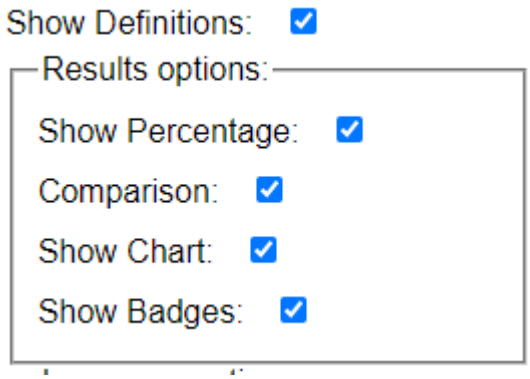
Test Case ID: 6		**Test Purpose:** Demonstrate that varying levels of feedback can be given	
Preconditions: The user creates a spelling test and decides the feedback options.			
Test Case Steps: As a note I will include the language options but cannot demonstrate the varying accents in text so I will just show the options			
Step No	**Procedure**	**Response**	**Pass/Fail**
1	User clicks on a button on the home screen to start the creation of a test	Taken to the first form for creating a test	Pass
2	User selects an option in the “results options” section	Results options are changed. By default all are true. The different options are “Show Percentage”, “Comparison”, “Show Chart” and Show Badges”	Pass
2.1	All results options are selected	All results options are shown	Pass
2.2	Only percentage is selected	Only the percentage is shown	Pass
2.3	Only comparison is selected	Only the comparison is shown	Pass

2.4	Only show chart is selected	Only the bar chart is shown	Pass
2.5	Only show badges	Only badges are shown	Pass
2.6	Percentage and badges are selected	The percentage and badge are shown for the test	Pass
2.7	Comparison and show chart are selected	The comparison and bar chart are shown on the results page	Pass
3.	Language options are selected	Change the accent of the text-to-speech audio file output	Pass
Date: 11/05/2022			

Evidence

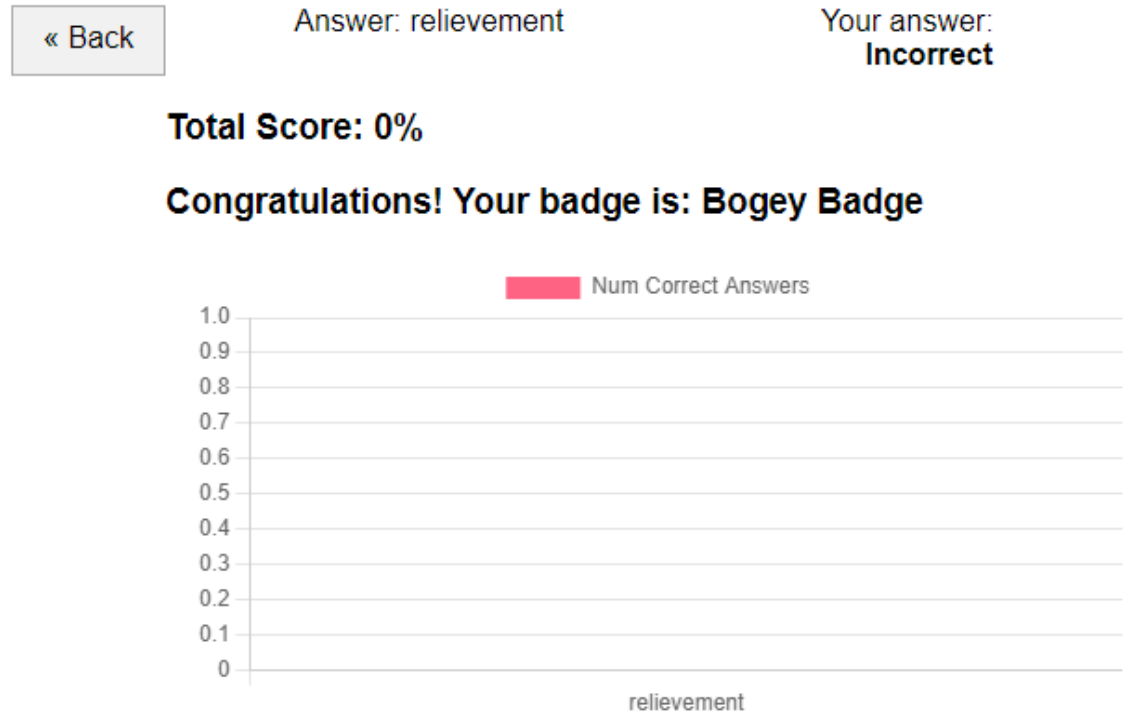
Step 1, 2 & 2.1:

(**Figure 50:** Step 1,2 & 2.1 for test case 6)



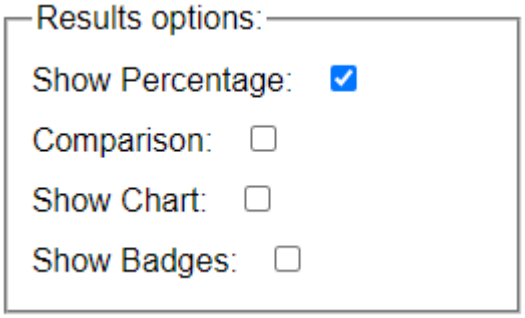
Step 2.1:

(**Figure 51:** Step 2.1 for test case 6)

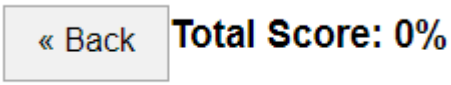


Step 2.2:

(**Figure 52:** Step 2.2 for test case 6)

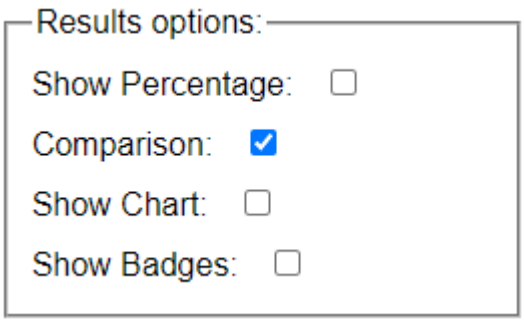


(**Figure 53:** Step 2.2 for test case 6)

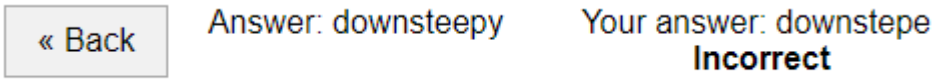


Step 2.3:

(**Figure 54:** Step 2.3 for test case 6)

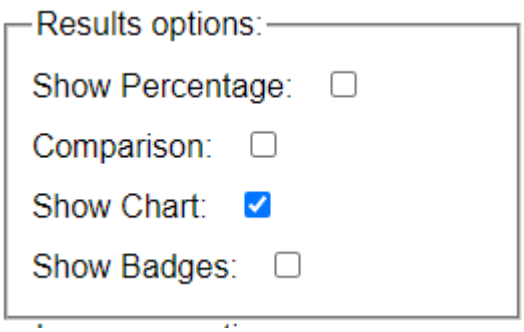


(**Figure 55:** Step 2.3 for test case 6)

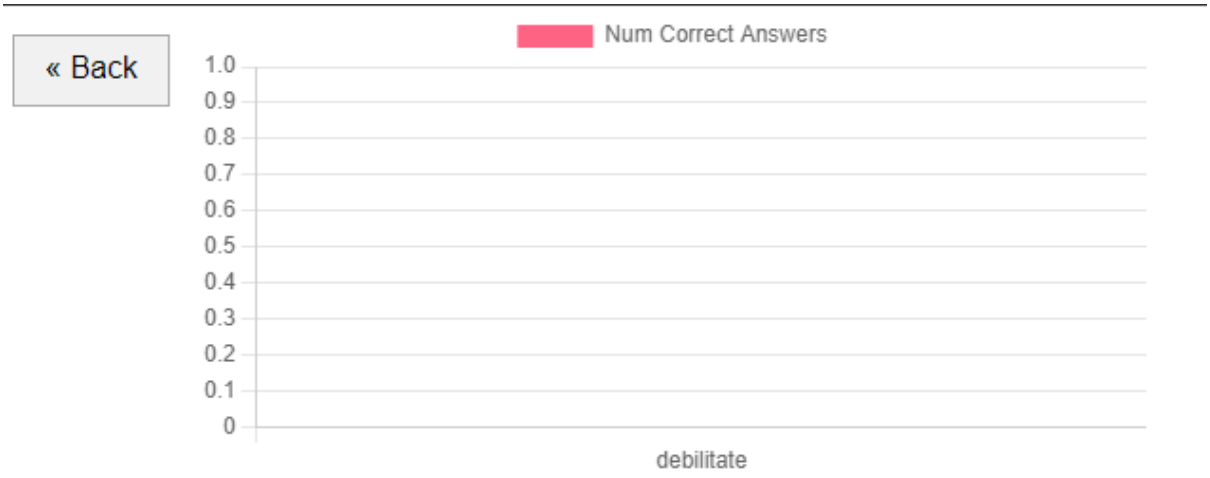


Step 2.4:

(**Figure 56:** Step 2.4 for test case 6)

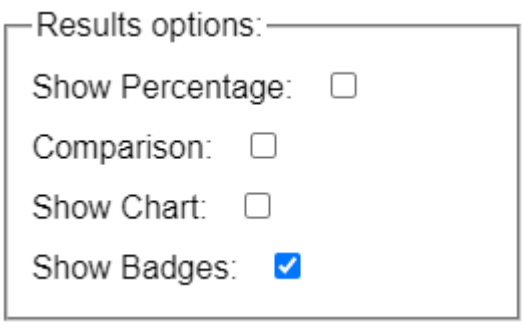


(**Figure 57:** Step 2.4 for test case 6)

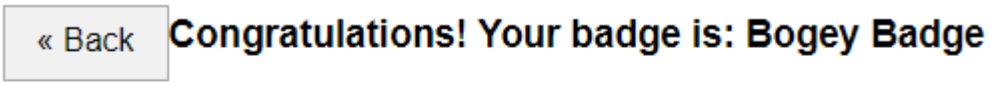


Step 2.5:

(**Figure 58:** Step 2.5 for test case 6)

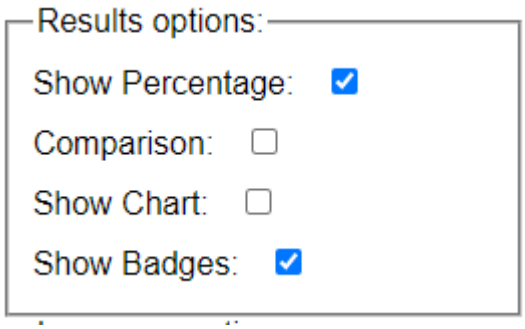


(**Figure 59:** Step 2.5 for test case 6)

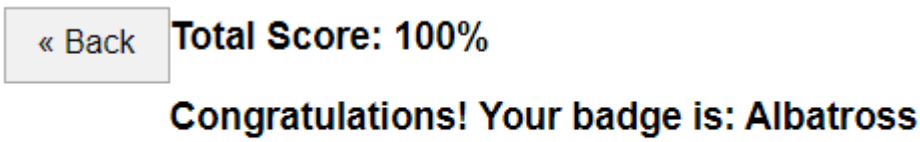


Step 2.6:

(**Figure 60:** Step 2.6 for test case 6)

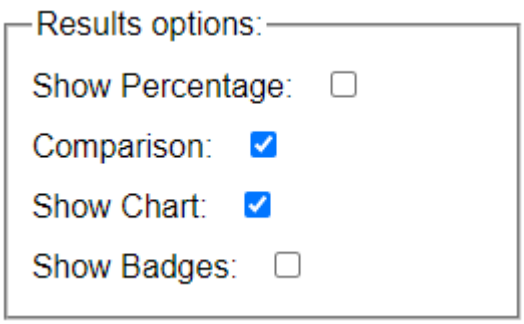


(**Figure 61:** Step 2.6 for test case 6)

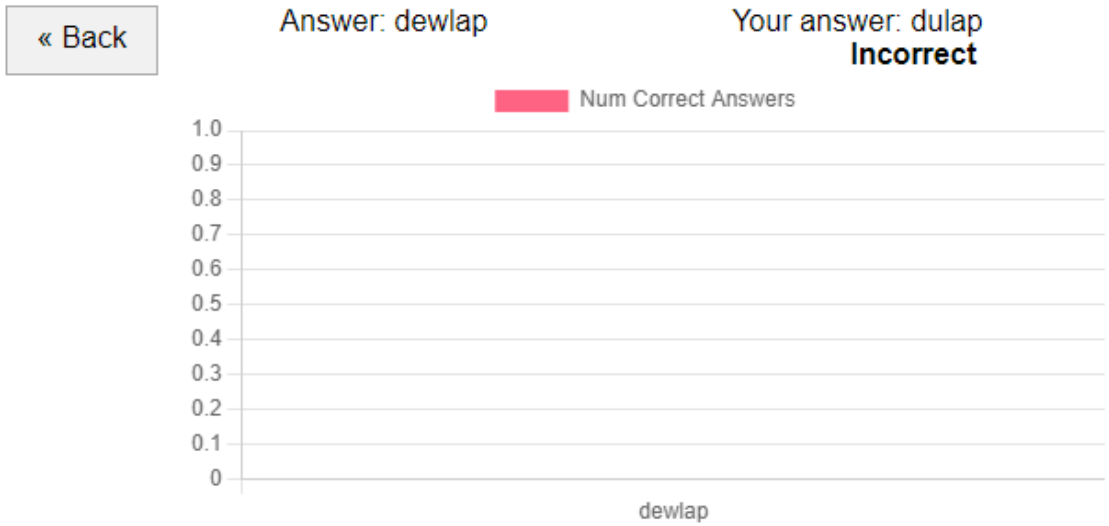


Step 2.7:

(**Figure 62:** Step 2.7 for test case 6)

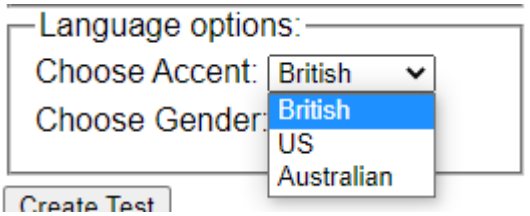


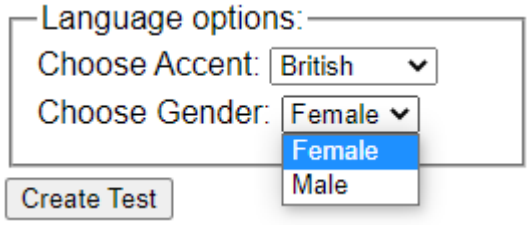
(**Figure 63:** Step 2.7 for test case 6)



Step 3:

(**Figure 64:** Step 3 for test case 6)





(**Table 8:** Test case 7)

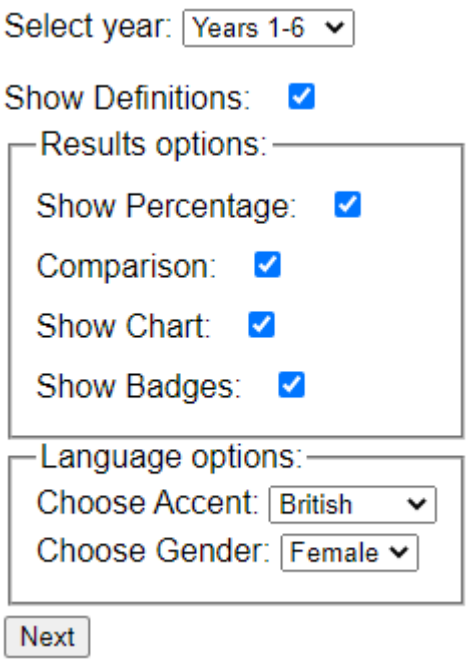
Test Case ID: 7		**Test Purpose:** Demonstrate the spelling tests modes not including 'custom' tests	
Preconditions: The user is on the come page and has clicked an option that is not "custom test"			
Test Case Steps:			
Step No	**Procedure**	**Response**	**Pass/Fail**
1	The user is on the home page and selects "Year Revision Tests"	Taken to the first form for revision tests.	Pass
2	The user selects what year they want to have the test based on	The year changes according to the selection	Pass
3	The user selects the test options/ settings	The options/ settings are changes	Pass
4	If all years are selected	The user is taken to another form where they enter in the number of words. If the user is logged in they can name the test	Pass
5	The user clicks on "Random Words"	A test is created and if they are logged in the test is uploaded and the test code is displayed	Pass
5.1	The user clicks on "Custom"	The user is taken to a word input screen with word suggestions in the text boxes	Pass
5.2	The user makes or does not make any changes to the suggestions and clicks on submit	The user is taken to the test demo screen	Pass
Alternative Flow 1: Continuing from step 2			
2	The user selects a year that is not all years	The year changes according to the selection	Pass
3	The user selects the test options/ settings	The options/ settings are changes	Pass
4	The user clicks next after selecting the settings and year	The user is taken to a new page where a dropdown is built containing the different statutory requirements for the selected year. If the	Pass

		user is logged in they can name the test	
5	The user selects a category from the dropdown and fills in the rest of the form then clicks “Random Words”	The user is taken to the test demo page where they can see what the test would look like. The words are taken from the selected category and randomised if possible	Pass
5.1	The user selects a category from the dropdown and fills in the rest of the form then clicks “Custom”	The user is taken to a screen where they have a form with suggested words for them to use based on the selected category	Pass
5.2	The user submits the words for the test	The user is taken to the test demo screen where they can se what the test would look like	Pass
Alternative Flow 2: The user creates a test using “Random Mode”			
1	The user clicks on “Random Mode” on the home screen	The user is taken to the test creation form	Pass
2	The user names the test if logged in, selects the test length and then selects the rest of the test options	The users options are changed to what they want to change them to	Pass
3	The user clicks on “Create Test”	Test is created and the user is taken to the test screen. The test is uploaded if the user is logged in.	Pass
Date: 11/05/2022			

Evidence

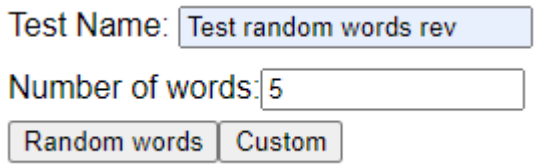
Step 1, 2 & 3:

(**Figure 65:** Step 1, 2 & 3 for test case 7)



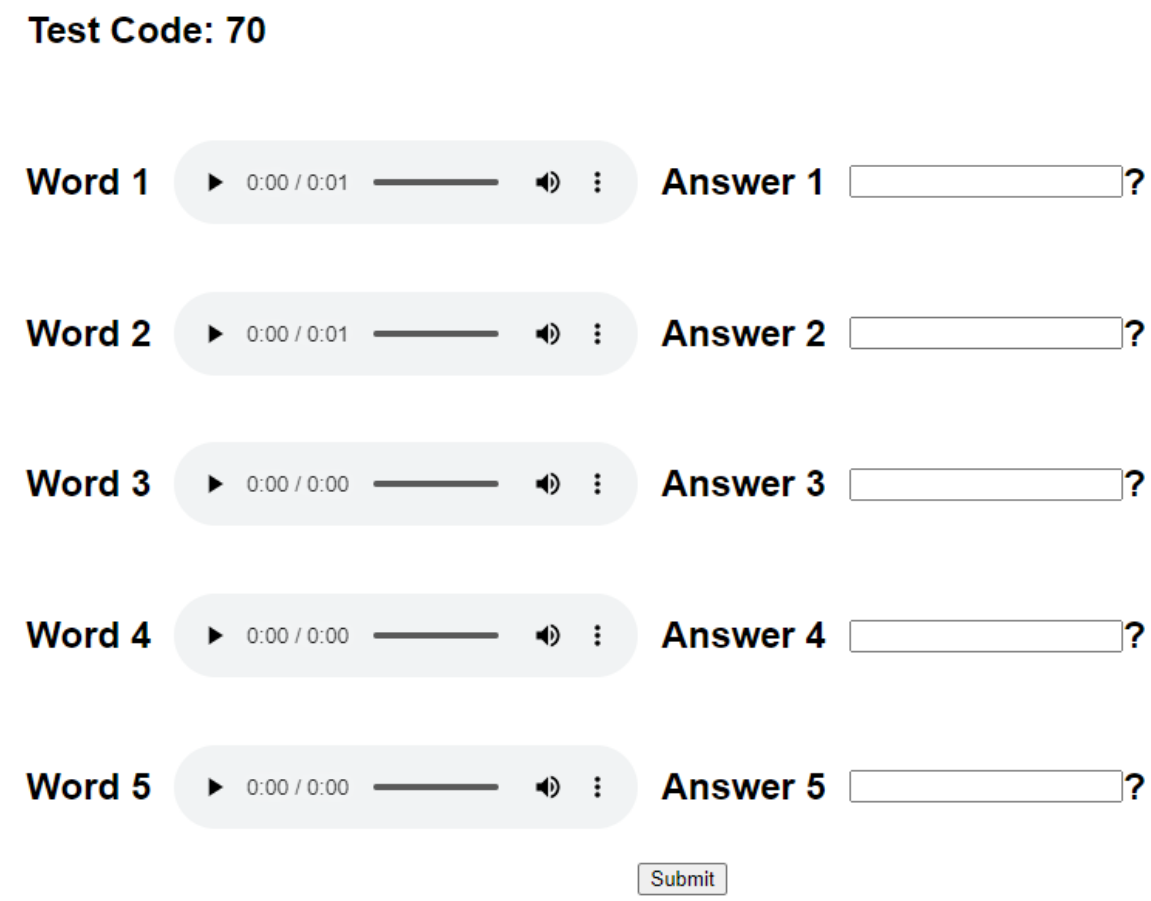
Step 4:

(**Figure 66:** Step 4 for test case 7)



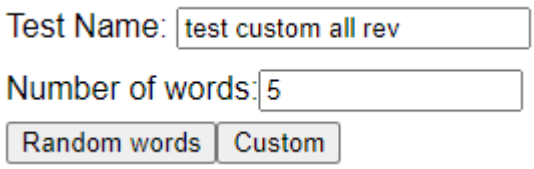
Step 5:

(**Figure 67:** Step 5 for test case 7)

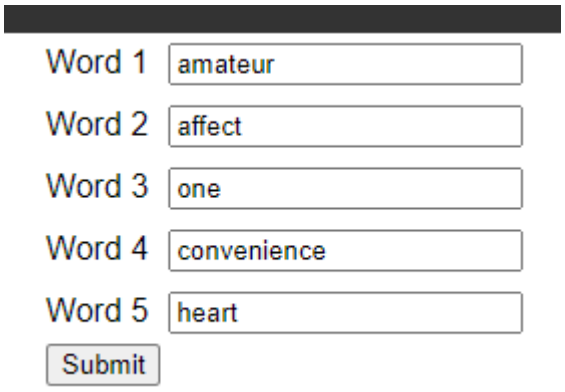


Step 5.1:

(**Figure 68:** Step 5.1 for test case 7)

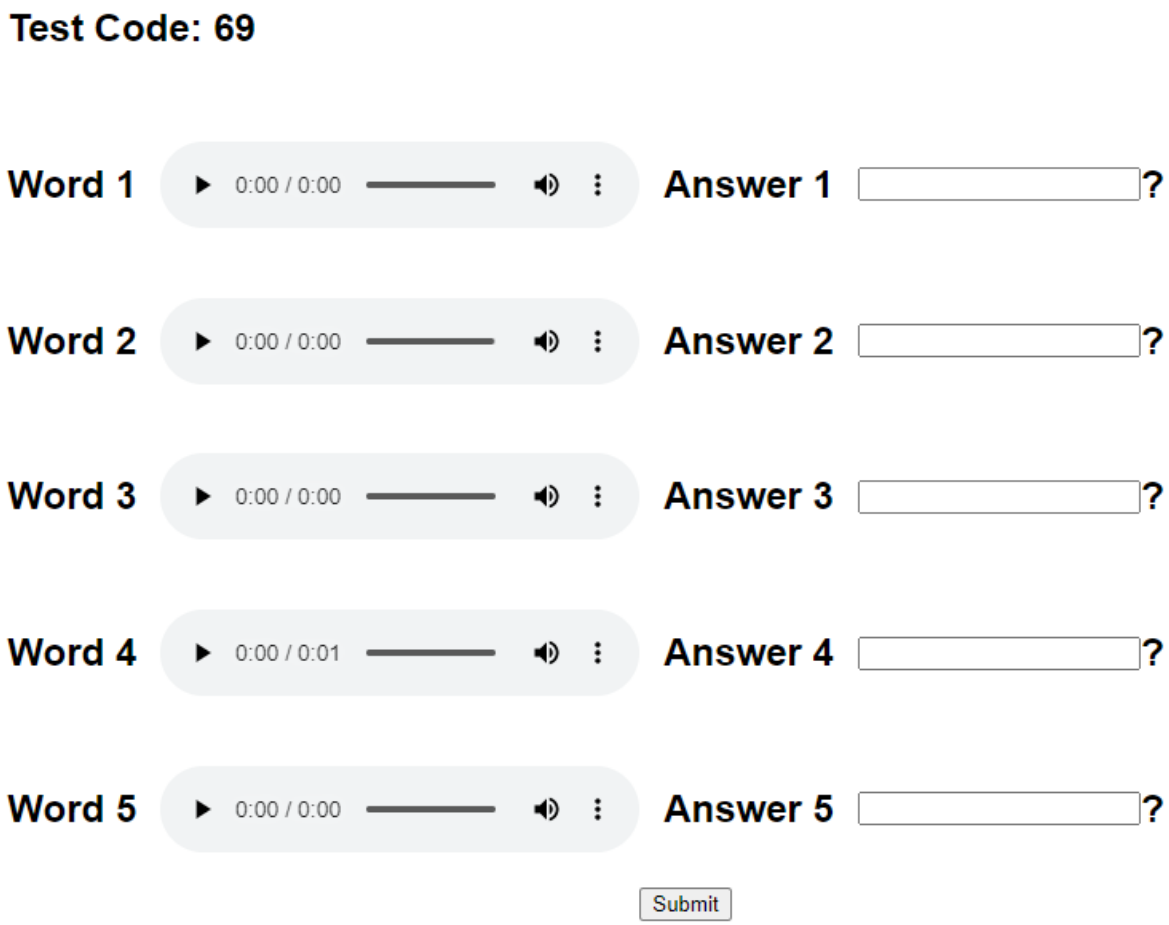


(**Figure 69:** Step 5.1 for test case 7)



Step 5.2:

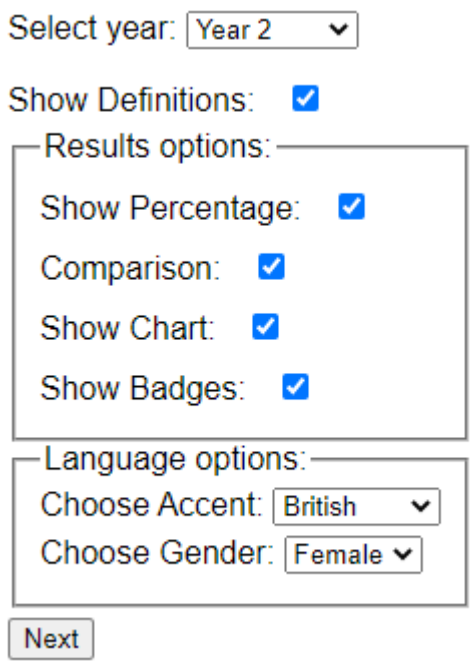
(**Figure 70:** Step 5.2 for test case 7)



Alternative flow 1

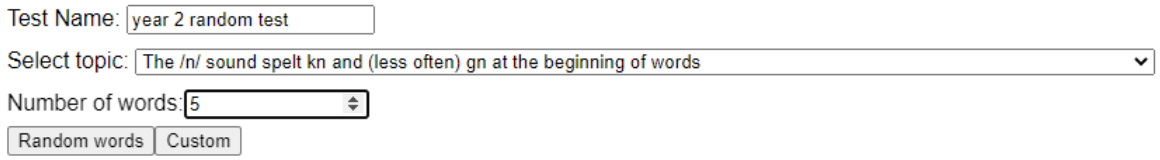
Step 2 & 3:

(**Figure 71:** Step 2 & 3 for test case 7 alterative flow 1)



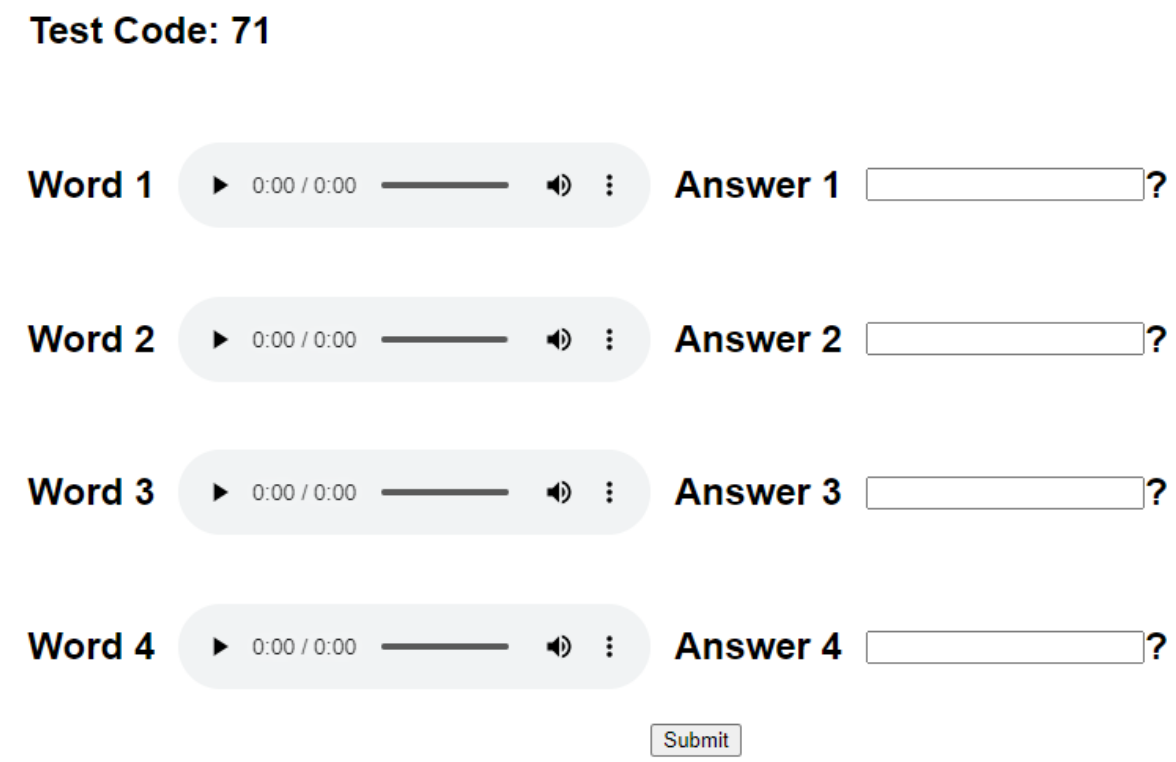
Step 4:

(**Figure 72:** Step 4 for test case 7 alterative flow 1)



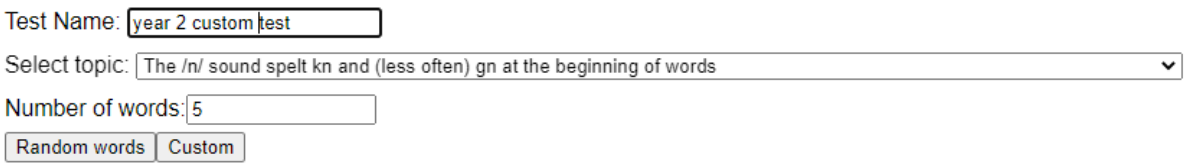
Step 5:

(**Figure 73:** Step 5 for test case 7 alterative flow 1)

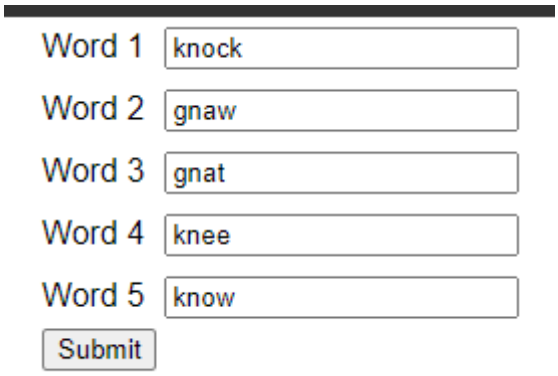


Step 5.1:

(**Figure 74:** Step 5.1 for test case 7 alterative flow 1)

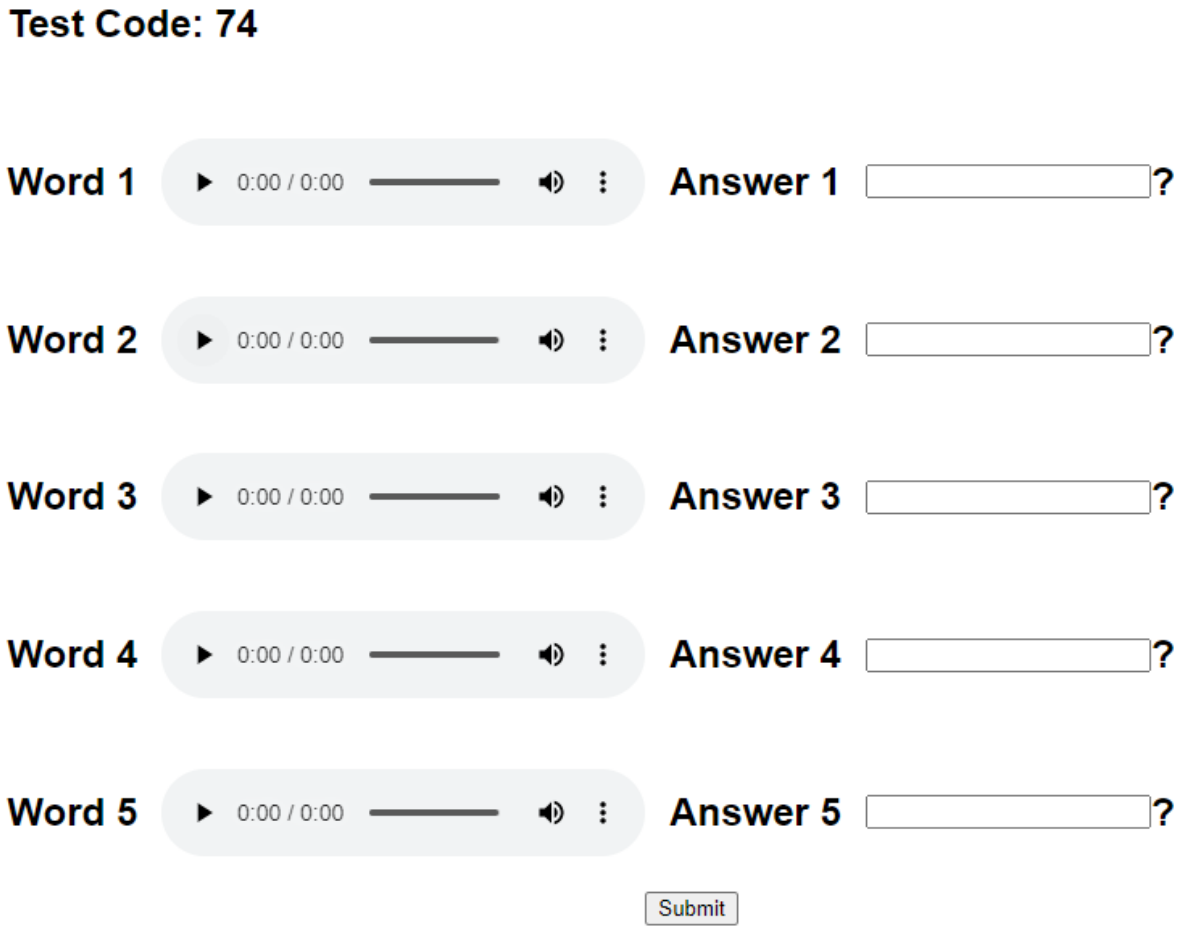


(**Figure 75:** Step 5.1 for test case 7 alterative flow 1)



Step 5.2:

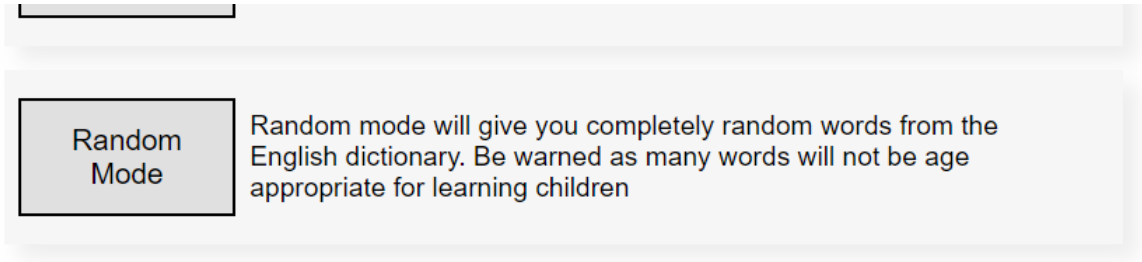
(**Figure 76:** Step 5.2 for test case 7 alterative flow 1)



Alterative Flow 2

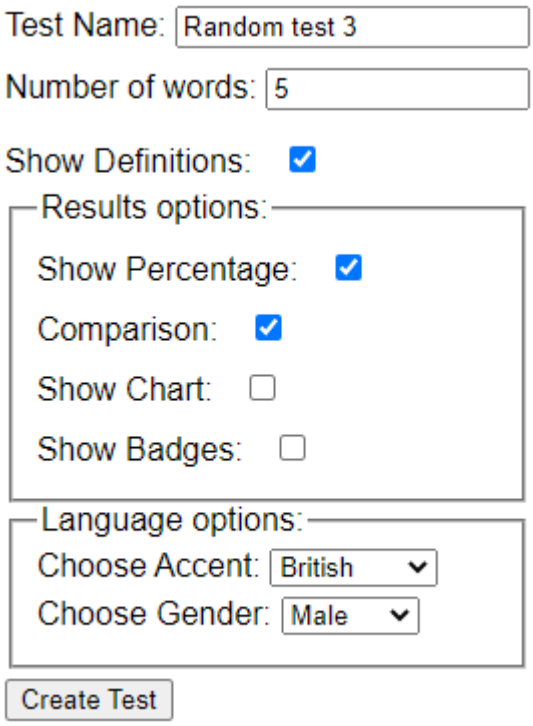
Step 1:

(**Figure 77:** Step 1 for test case 7 alterative flow 2)



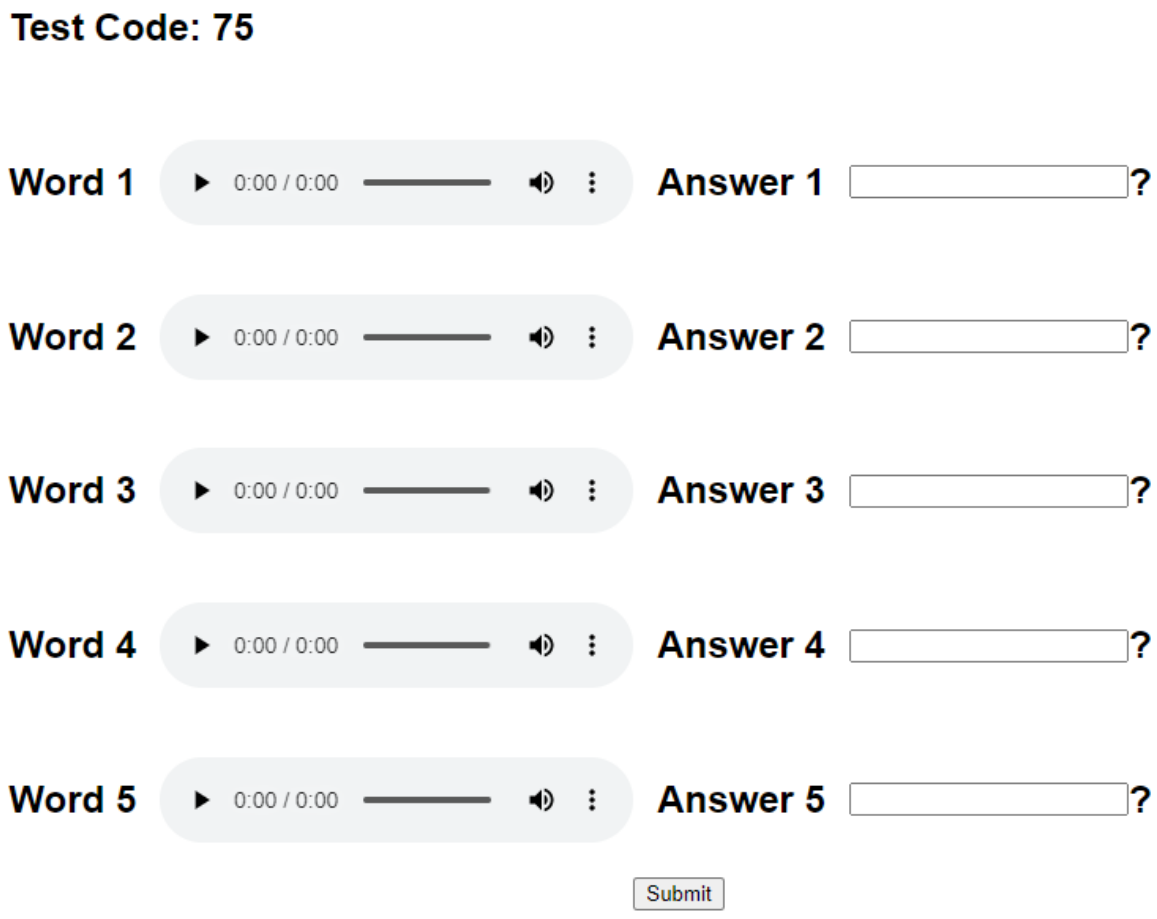
Step 2:

(**Figure 78:** Step 2 for test case 7 alterative flow 2)



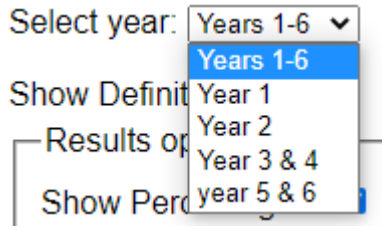
Step 3:

(**Figure 79:** Step 3 for test case 7 alterative flow 2)

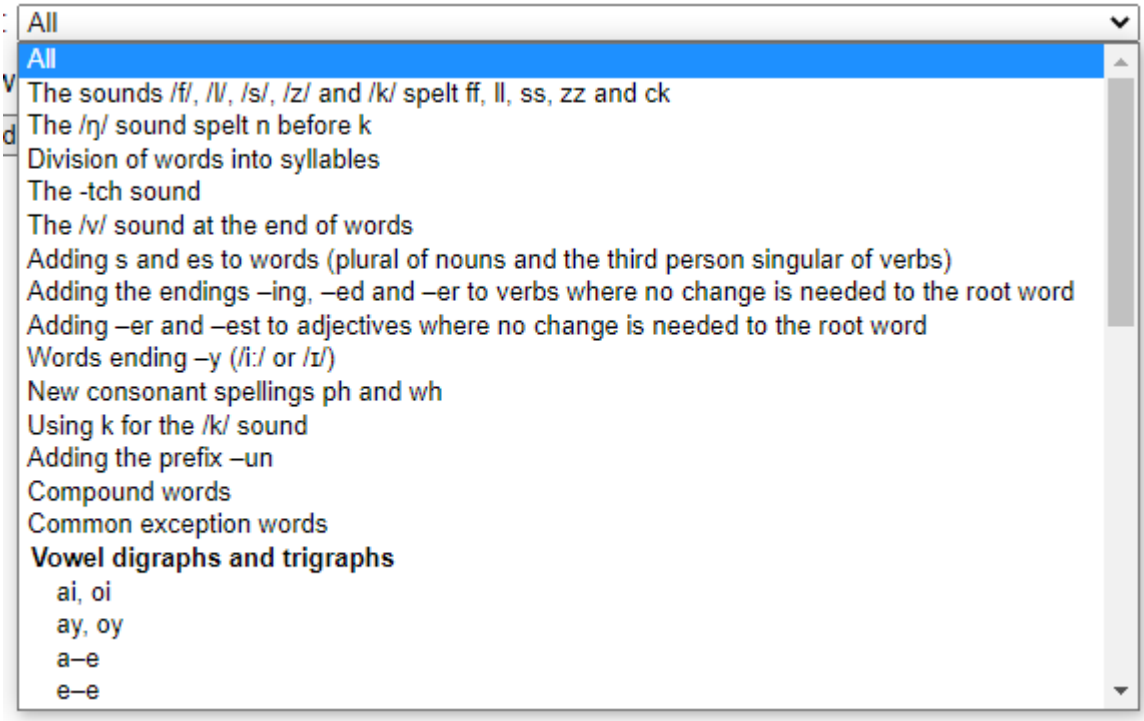


Extra figures for test case 7

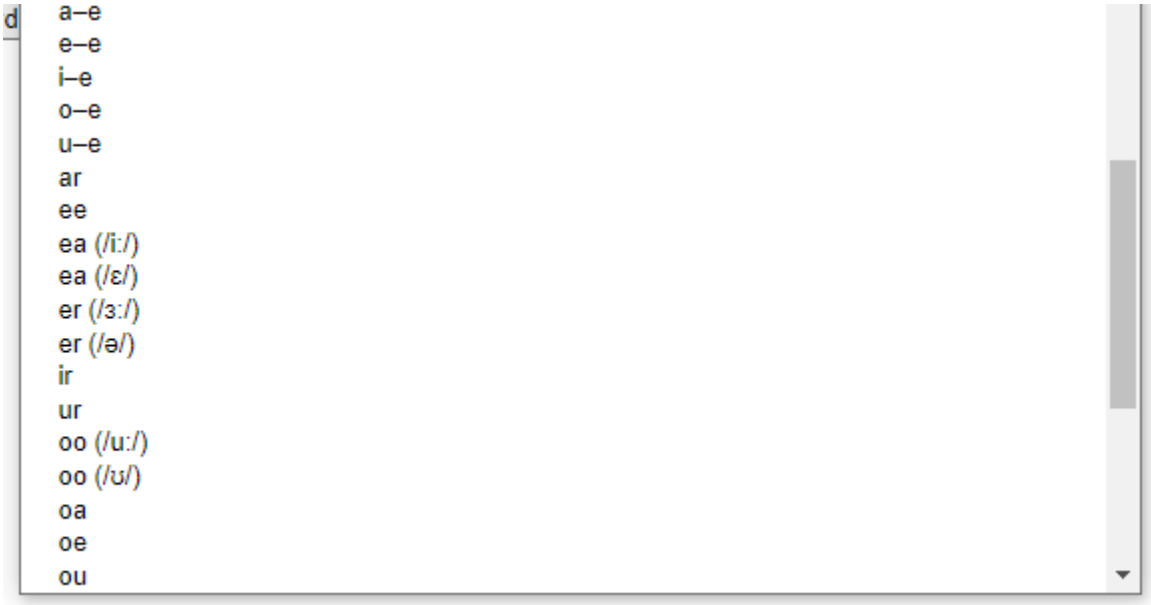
(**Figure 80:** Different years dropdown)



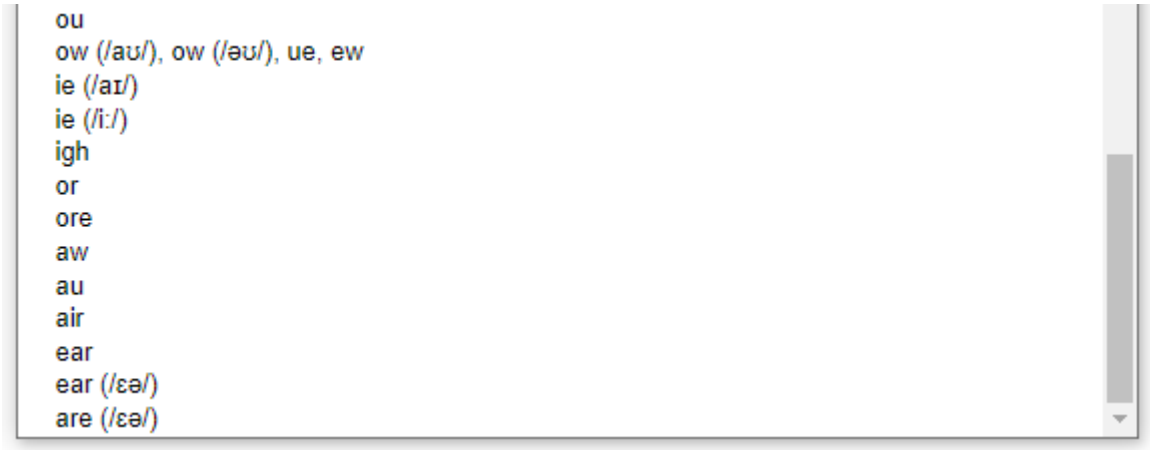
(**Figure 81:** year 1 dropdown part 1)



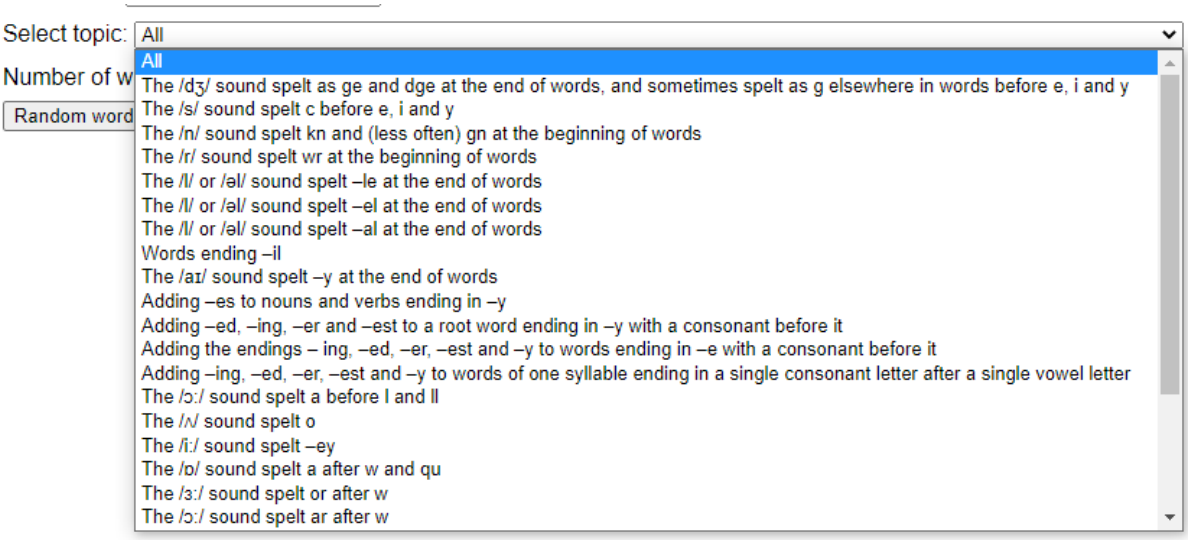
(**Figure 82:** year 1 dropdown part 2)



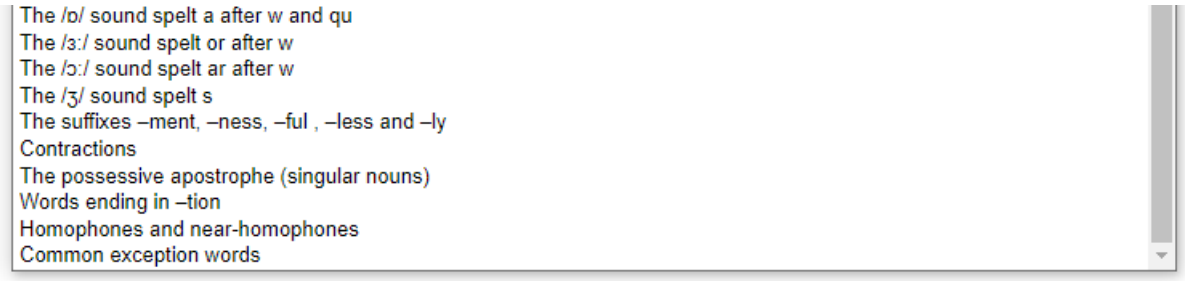
(**Figure 83:** year 1 dropdown part 3)



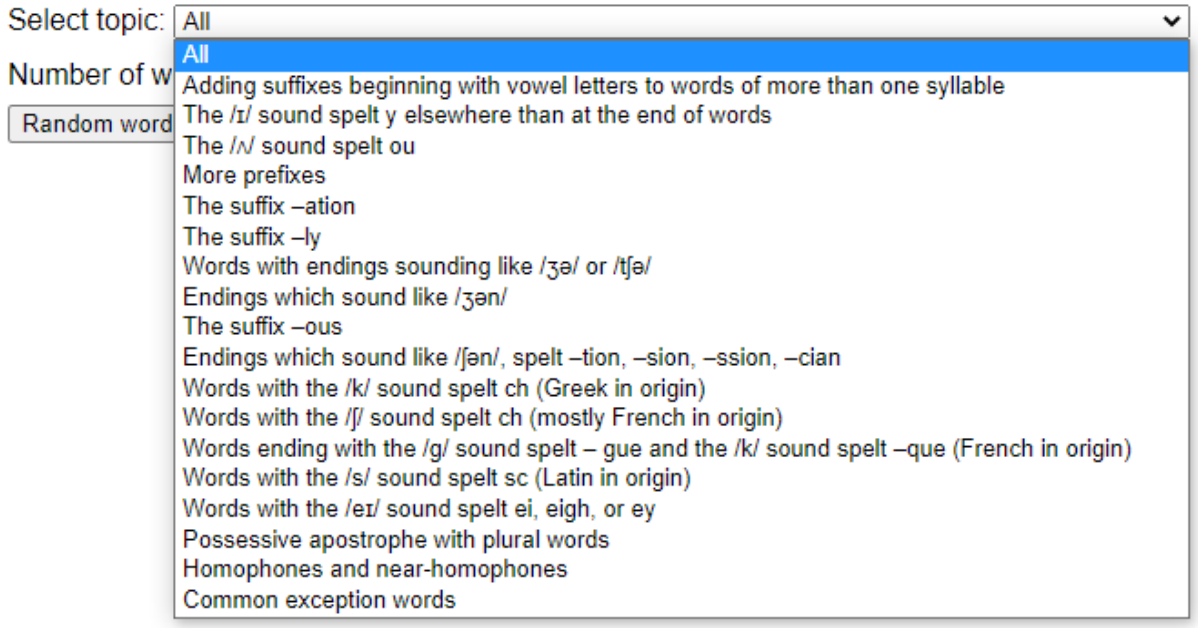
(**Figure 84:** year 2 dropdown part 1)



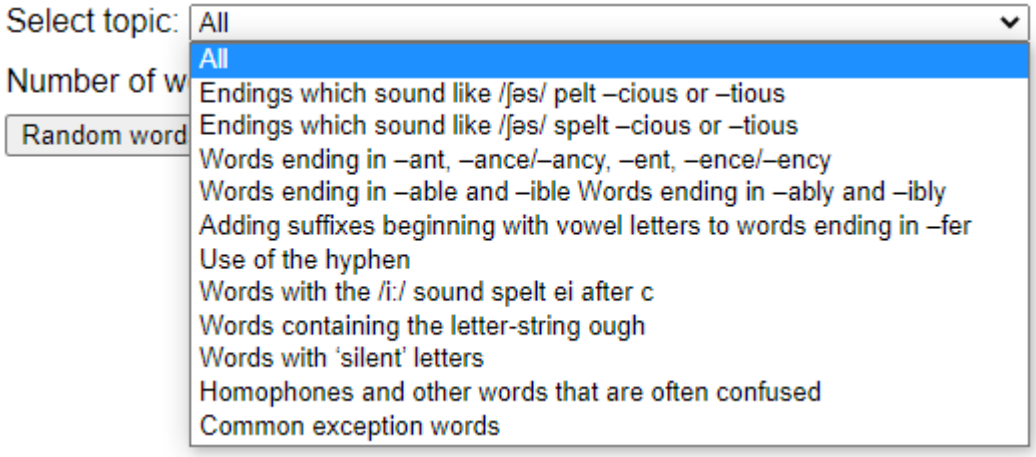
(**Figure 85:** year 2 dropdown part 2)



(**Figure 86:** year 3 & 4 dropdown)



(**Figure 87:** year 5 & 6 dropdown)



External Testing

Once I had my website in a finished state, and before I did integration testing, I did some external testing with a Parent and her two children. The parent is a medical professional with two children, one in year 2 and one in year 5. The children are native English speakers who have spent most of

their time within the UK school system, with the exception of two years in the Australian schooling system.

This testing was mostly for evaluation purposes and to get concrete feedback from children of what they like and do not like about the website. The children were asked to complete a few tests each using the different test options mostly focusing on the revision tests due to them knowing these words best.

The parent was asked to evaluate the website from the perspective of a parent using it for their children. They did not complete tests but created tests using the revision test builder and looking at the test history charts to see how useful it would be to them.

Positives

The UI was very clear and easy to understand. The children commented that when filling out a spelling test they liked how they could see all the word audio play buttons and the input boxes next to each other because it made it very clear where to enter which word and allowed them to choose which order they completed the test in. They also liked that the audio played gave them the option to change the volume. Both were features that were not present in the spelling test website they used for school.

A positive from the parent about the UI is that they liked the overall simplicity of the website. It made it easy for them to understand what to do and the forms were ordered well and clearly with clear reasons for why each form was different.

The parent also thought that the revision mode was good making it easy for their children to practice words from the lists of words they should know. They thought that having the words suggestions in revision mode when custom is clicked was also useful allowing them to tailor a test if they want to or just submit the created the test.

The parent and the children both like the options to change font style, size, and colour scheme. For the parent it was so that they could read the site without their glasses on as for the children they liked having the option to decide on the look of the website.

On the results page the children liked the ideas of the badges and thought the names for the badges were fun. The parent thought that the badge names were appropriate for children to see.

Negatives

When completing a form there is no way to go back to the previous form without losing the data entered into the previous form. This was the parent that suggested this as when trying out creating a test they realised them made a mistake on the previous form and went back but had to fill out the form again.

I noticed when they were navigating the website the parent and children would click on the dropdowns that are a part of the menu at the top of the page. Although they work by hovering over them the mouse changes to the hand symbolising that you can click it making them think they need to click it to get the menu to stay open. The quick fix for this would be to remove any of the pointer changes when hovering.

From the children's perspective the results page when you complete a test was a bit boring. They would like the correct words in the comparison to have a "big green tick" next to the correct ones because it makes them feel good.

When having an account there is no option to have a profile picture. The parent did not care too much but the children would like to have a few options of premade profile pictures that they could choose from when creating an account.

Suggestions

There were a few suggestions given that have not been covered in the negatives. One was that the charts showing the detailed information of the created spelling tests was originally the same as the overarching chart for the test where if a user has multiple test results entered in it would give an average of all tests and total up the correct answers for each individual word. The suggestion was that it would be better to show these results over time when looking at an individuals results. I decided to implement this after they did the testing as the code for it was mostly there I just had to reorganise the information to output it into the chart chronologically for each test entry. See **Figure 26** for the updated charts.

The options for changing the colour schemes could be more. As a child they like colourful things and suggested “rainbows”, “unicorns” and “dinosaurs”. They would also like more control over how the font colours change and would like to be able to set the colour of the font to something of their own choosing such as “pink”. The children also would like to have more sound effects added to the website. They said that on the results screen having a celebration sound would be great and having a “boing” sound when they are typing in words on a test would be a fun addition. From the parent’s perspective they do not mind sounds being played but also would like to have an option to turn it off.

Future Work

Graphics and charts

One of the plans I would do when continuing this project would be to add in more graphics. This would be the “rainbows”, “unicorns” and “dinosaurs” suggested by the children as well as more. I also would like to add graphics for the badges that are displayed to make them more entertaining for children. To avoid any copywrite infringement I would have to get these graphics commissioned by a graphics designer for the website. This way I could also have custom made to fit the page properly. The downside of this is that it would detract from the purpose of this being aimed at children with dyslexia. In the research done on choosing dyslexia friendly colour schemes they said having images or graphics as a background could make it more difficult to read the whole page and to try and avoid it. One potential way around it I have thought of would be to have the graphics display in side bars. The main content of the page is currently contained in a flexbox so adding in two more flexboxes to create colourful side bars will not be too difficult once the graphics have been obtained.

I would also like to add in more chart options for displaying data such as showing where the student lies compared to the rest of the students who have completed the spelling test. This would be based on standard deviations and would end up being a bell curve to display all results of a test.

Sounds and accents

As the children suggested adding sounds is something I would like as an additional feature. Although this is not key to the functionality I thought it would be a good step towards gamification and engage children more when they are completing a spelling test. I would add in a setting when you are on the test page to enable or disable the sounds. I would also add in an option when creating the test which would either allow or disallow the option for sounds during the test.

Accents would be something I would change a bit. I would like to add in a demo for the different voice types so the test creator can decide which is best before they create the rest of the test. An interesting paid feature of the Google text-to-speech API was that you could record your own voice and it would create a text-to-speech translation based off the voice data it receives. I thought that this could be a good way for teachers to have a voice that is familiar to their students and therefore easier to understand to them that the default accent options.

Adding in more accent options is also something that can be easily done. With the Google text-to-speech API they have over 100 different accents to choose from. For the purpose of this project being a proof of concept I only included a few options just to show that it is possible to do this but I would like to add in more in the future.

Continuous Revision Mode

This was something I originally planned to make but the reason I did not was because the step before I could start on creating a continuous mode took a lot more time and thought than I believed it would. The step before was the revision mode based on the UK government requirements [5].

My plan for continuous revision mode would be to use the column "wordlist" in the database to store the words that have been submitted with a result by the user. The user would have to have an account for this to work. There were two ways I thought of doing this. The first way was for each user to have a copy of the full word list from the government requirements saved into "wordlist". When a user completes a test with a word from the full word list in it the result would be recorded in their account data. Using this data, continuous mode would look at what you get wrong most often and then suggest to you words similar to the most incorrect words based on what is in the same category from the government requirements. The similar words would only be given to the user every 5-10 words entered in through continuous mode with the other words being random but below the school year of the child. To stop the child from having one word that ends up with so many incorrect answers that the system thinks they are not improving I would only store the previous 5 results for each word with the oldest results being removed whenever the new 5th answer is submitted. However, one of the problems with this system is that to get enough entries for the system to start to suggest their weaker spellings and words consistently would require the user to submit a lot of words. The more this mode is used the better it would be. The benefit of having the full word list stored would mean that when the student is using continuous mode, and does not have many word results recorded, I would not need to parse through the JSON file containing the UK government requirements but could randomly choose words from the full word list. It would only be every few words that I would need to parse through the JSON file when I am getting words similar to their most incorrect words.

The other way of doing this is very similar but instead of saving the full word list to the account data I would slowly increase the length of "wordlist" as they complete tests and continuous mode. The downside of this is that I would need to parse the JSON file and get a new word each time the user submits a word in continuous mode. This would increase processing times but the benefit is that the total amount of stored data would be less.

(**Figure 88:** Example of what "wordlist" column would look like in the table "users")

```
wordlist
{'space': [0,1,1,1,1]},
{'hello' : [0,0,1,1,1]},
{'door' : [0,0,0,0,1]},
```

Conclusions

I believe that I have completed the original main goals of this project. In the brief there was a lot that was up to interpretation. The brief wanted me to create a spelling test platform that could be used for symmetric and asymmetric learning, aimed at primary school children with dyslexia and with options for varying forms of feedback. Just looking at this part of the brief I have completed a website that enables you to create a spelling test and have others complete it regardless of whether they are logged in or not. You can change the font size, style, and background colour scheme where the available options are based on research into what is easy to read for people with dyslexia. When creating a spelling test you can give varying levels of feedback by selecting the specific feedback options that you wish to display to the user.

The main features of the website are the different types of tests you can take with the revision mode being the most complicated of them. I was pleased with how the revision mode turned out and felt that it added in an integral part of the website that makes it stand out from other spelling test platforms. The way that I have built the tests allows myself to easily add in more modes in the future with functions allowing me to reuse code easily. Initially when I was coding I did not use any functions but about halfway through went through a large refactoring process where I tried to reduce duplicated code as much as possible. This opened up more possibilities and caused myself to continue coding in the same way to allow me to separate my new code out into functions and integrate them into the existing program flow.

For changing the font style, size and background colour schemes I was especially pleased with how I could get the selected options to save across sessions and across pages consistently. Storing the settings within the local session storage allowed me to do this. Having never used session storage before I had tried to stay away from it but for something like selecting the schemes of the website using session storage is perfect for it.

I was also pleased with the database that I designed. I had never used JSON with SQL before but using this has allowed me to improve my data management skills. When designing the database I wanted to future proof it and using JSON to store information and settings individually has allowed me to progress through the project without having to remake the database. Although there was the one column that was not used in the end it will allow me to expand on the website later.

Reflection on Learning

While writing this report and coding there were a few things that I learned as I went along. I would not change the database design and in future projects I will most likely continue to use SQL as the materials out there are abundant and extensive. In comparison, I wanted to try and use MondoDB [14] initially however there was very little material on how to get it to work with PHP with the instructions eventually looping around to the start again if you had the error that I had.

I had never used JSON as extensively before as I did in this project. Through the course of developing the code I came to realise that it is an immensely powerful way to store and organise data. The read write times of JSON is extremely fast with almost no delay when parsing through even though there

would be over 1000 loops in certain circumstances. In the future I will design my programs to work with JSON.

The test code was originally going to be a large and unique number. In the end I decided that the best and easiest way to set a test code would be to simply use the ID of the test in the database table “tests” as I knew that this would be unique. I would then avoid having to do any checks to make sure the test code is unique meaning less database requests and no processing when assigning a code is be needed.

Although PHP is a powerful platform to build dynamic websites on in retrospect creating the website using JavaScript would have been better. PHP has a lot of good features and due to it being dynamic the processing is done server side reducing the strain on systems if it was to become a publicised website. However, JavaScript has more functions in it that are built in whereas in PHP I would have to find workarounds or alternatives to these functions. This caused a lot of time to be put into finding the correct workarounds for the different features I wanted to include. JavaScript is also able to easy parse through JSON as an object but in PHP it cannot and it needs to be converted to an array with loops being needed to access certain locations of data.

The external user testing was incredibly valuable to myself. I had never done external testing before but getting unbiased opinions from people helps to really emphasise the good and bad points of the system. The feedback obtained was fairly close to what I was expecting which helped because I was never sure if my opinions were too biased or not. One of the things I had been worried about was that the forms for creating spelling tests might be too confusing. When I was told they were clear and easy it was very nice because I had spent quite a while thinking and designing the forms in a way that I thought was easy to understand.

If I worked for a company I would most likely complete this project in a similar fashion. The user testing would of course be more extensive with more external testing being done. In this project I met with my supervisor daily and the client three times. If I was in a company I would have shorter ‘sprints’ of about 2 weeks where I would then see the client every two weeks to get feedback on the changes. The problem I had is that as I was doing this alone everything took longer to code. If I was I a company I would assume that I would be a part of a team meaning in two weeks we would have made a lot of progress and there would be large changes that I could show the client. With how I was coding it, a lot of it was backend and processing of data and information so there is not a lot of big changes that I could show the client towards the end.

References

[1] Merriam-Webster. 2022. *Spell It*. [online] Available at: <https://www.merriam-webster.com/word-games/spell-it> [Accessed 7 April 2022].

[2] Englishclub.com. 2022. *Commonly Misspelled Words: Quiz 1 | ESL Quizzes | EnglishClub*. [online] Available at: <https://www.englishclub.com/esl-quizzes/spelling/common-misspellings-quiz-1.htm> [Accessed 7 April 2022].

[3] Dailywritingtips.com. 2022. *Spelling Test 1*. [online] Available at: <https://www.dailywritingtips.com/spelling-test-1/> [Accessed 7 April 2022].

[4] Spellingshed.com. 2022. *Spelling Shed*. [online] Available at: <https://www.spellingshed.com/en-gb/> [Accessed 4 May 2022].

[5] GOV.UK. 2013. *National curriculum in England: English programmes of study*. [online] Available at: <https://www.gov.uk/government/publications/national-curriculum-in-england-english-programmes-of-study> [Accessed 4 February 2022].

[6] Association, B., 2022. *British Dyslexia Association*. [online] British Dyslexia Association. Available at: <https://www.bdadyslexia.org.uk/> [Accessed 10 February 2022].

[7] Association, B., 2018. *Dyslexia friendly style guide - British Dyslexia Association*. [online] British Dyslexia Association. Available at: <https://www.bdadyslexia.org.uk/advice/employers/creating-a-dyslexia-friendly-workplace/dyslexia-friendly-style-guide> [Accessed 10 February 2022].

[8] Chapman, C., 2018. *Exploring the Gestalt Principles of Design*. [online] Toptal Design Blog. Available at: <https://www.toptal.com/designers/ui/gestalt-principles-of-design> [Accessed 30 February 2022].

[9] js.foundation, J., 2022. *Download jQuery | jQuery*. [online] Jquery.com. Available at: <https://jquery.com/download/> [Accessed 8 April 2022].

[10] Getcomposer.org. 2022. *Composer*. [online] Available at: <https://getcomposer.org/> [Accessed 10 February 2022].

[11] Google Cloud. 2022. *Text-to-Speech client libraries | Cloud Text-to-Speech Documentation | Google Cloud*. [online] Available at: <https://cloud.google.com/text-to-speech/docs/libraries#client-libraries-install-php> [Accessed 10 February 2022].

[12] GitHub. 2019. *PHP Fatal error: Uncaught GuzzleHttp\Exception\RequestException: cURL error 60: SSL certificate problem: unable to get local issuer certificate · Issue #1821 · googleapis/google-cloud-php*. [online] Available at: <https://github.com/googleapis/google-cloud-php/issues/1821> [Accessed 11 February 2022].

[13] Chartjs.org. 2022. *Chart.js | Open source HTML5 Charts for your website*. [online] Available at: <https://www.chartjs.org/> [Accessed 20 April 2022].

[14] Pecl.php.net. 2020. *PECL :: Package :: mongodb 1.12.1 for Windows*. [online] Available at: <https://pecl.php.net/package/mongodb/1.12.1/windows> [Accessed 15 February 2022].