**Software Engineering Group Project -**

**Group 6 Design Specification**

Authors: Jacob Back & Jaime Sanchez

Config Ref: DesignGroup6

Date: 02/04/2020

Version: 1.0

Status: Release

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Copyright © Aberystwyth University 2020

[1- INTRODUCTION 3](#_Toc39625062)

[1.1- Purpose of the document 3](#_Toc39625063)

[1.2- Scope 3](#_Toc39625064)

[2- DECOMPOSITION DESCRIPTION 4](#_Toc39625065)

[2.1- Programs in System 4](#_Toc39625066)

[2.2- Significant Classes 4](#_Toc39625067)

[3- DEPENDENCY DESCRIPTION 5](#_Toc39625068)

[4- INTERFACE DESCRIPTION 9](#_Toc39625069)

[4.1.1- Word: public, abstract, implements Comparable<Word>. 9](#_Toc39625070)

[4.1.2- Other: public, extends Word 9](#_Toc39625071)

[4.1.3- Verb: public, extends Word. 10](#_Toc39625072)

[4.1.4- NounF: public, extends Word. 10](#_Toc39625073)

[4.1.5 - NounM: public, extends Word. 10](#_Toc39625074)

[4.1.6- Dictionary: public. 11](#_Toc39625075)

[4.1.7- Main: public. 11](#_Toc39625076)

[4.2.1- Main 11](#_Toc39625077)

[4.2.2- HomeController. 11](#_Toc39625078)

[4.2.3- HelpSceneController. 12](#_Toc39625079)

[4.2.4- AddWordsController. 12](#_Toc39625080)

[4.2.5- RevisionController. 12](#_Toc39625081)

[4.2.6- FlashcardsController 12](#_Toc39625082)

[4.2.7- MatchTheWordsController 13](#_Toc39625083)

[4.2.8- PickTheCorrectWordController 13](#_Toc39625084)

[4.2.9- TestController 13](#_Toc39625085)

[4.3.1 – AddWordsController 13](#_Toc39625086)

[4.3.2 – MatchTheWordsController 13](#_Toc39625087)

[4.3.3 – PickTheWordController 13](#_Toc39625088)

[5- DETAILED DESIGN: 14](#_Toc39625089)

[6- REFERENCES: 14](#_Toc39625090)

# 1- INTRODUCTION

## 1.1- Purpose of the document

The purpose of this document is to detail all aspects of the design for the architecture portion of the welsh language learning program and how the program we’ve designed will meet all system requirements set out in SE.QA.05[1]. This will be achieved by providing detailed descriptions of how the methods in each class will work.

## 1.2- Scope

This document should cover everything you need to know about the architecture aspect of the program. No other document needs to be consulted before reading this paper. Included in this paper are sections that describe how each class in the program will function and what techniques are being used to make sure they work correctly.

Also included are diagrams that will layout the classes and their methods clearly so the inner workings of the program can be better understood. We will also, by class, talk about each method and explain how they will work on a fairly granular level.

## 1.3 - Objectives

The primary objective of this paper is to make sure that whoever reads it is fully able to understand all aspects of the program’s design. And if the reader understands the architectural design behind the program after reading this document then the objective of this document has been met.

# 2- DECOMPOSITION DESCRIPTION

## 2.1- Programs in System

For the system that is being built it has been decided that there will be only one program. The program will be used to store the database of words, which will be made up of English words with the Welsh translations and the word type. The program will allow the user to develop their language skills by allowing them to view all of the words that are stored in the dictionary, add words to a ‘Practice List’ to save words that they wish to revise and the user will be able to test themselves on words that they have selected. This has been designed to allow the user to expand their knowledge in ways that suit them best.

## 2.2- Significant Classes

**Word:** The Word class is meant to represent the words in the database as objects and hold all the information needed from the words. This class will be set as an abstract class.

**Other:** The Other class is for words that are not nouns or verbs. This class will inherit from the Word class.

**Verb:** The Verb class is meant to represent the verbs in the database as objects and hold all the information we will need from the verbs. This class will inherit from the Word class.

**Noun:** The Noun class is meant to represent the nouns in the database as objects and hold all the information we will need from the nouns. This class will also inherit from the Word class.

**Dictionary:** The Dictionary class is meant to store and manage the two databases allowing the user to choose the default language to display. The class will also be responsible for storing the users favourite words, adding new words, and the flashcards system.

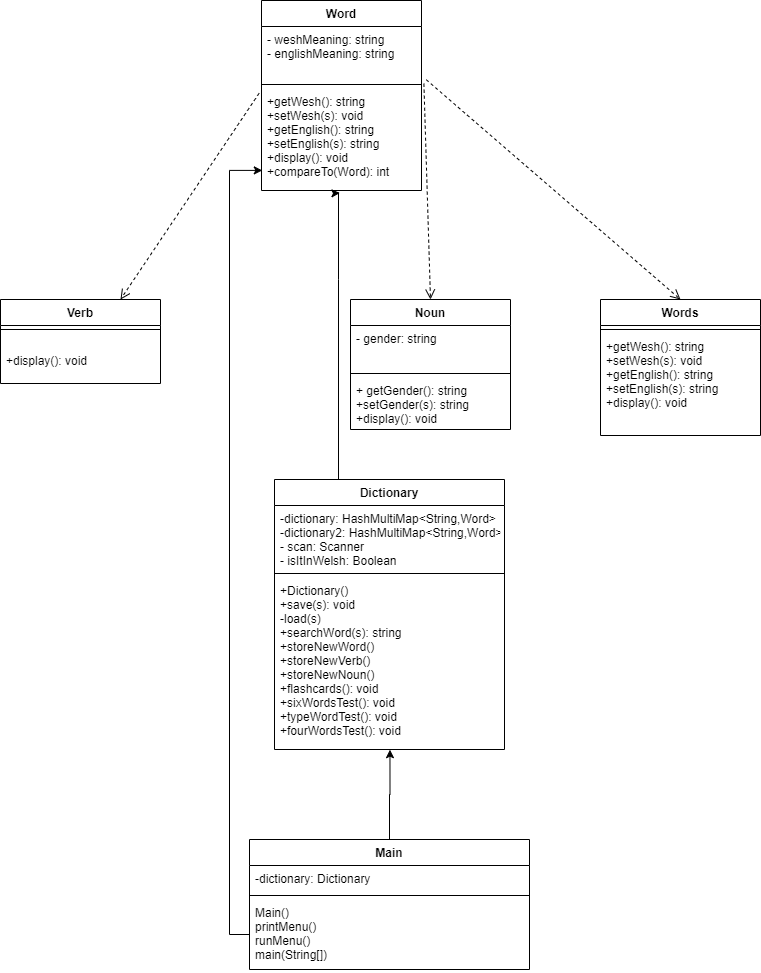
**Main:** The Main class is meant to run the program when called.

2.3- Table mapping requirement onto classes:

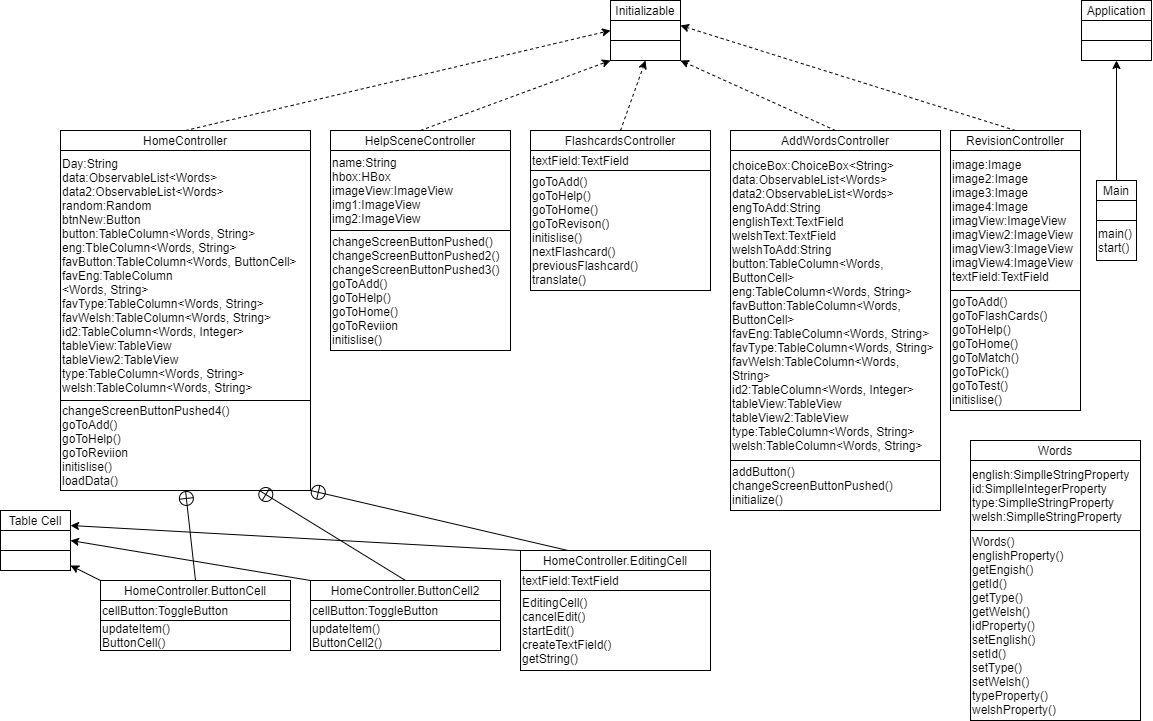
|  |  |
| --- | --- |
| Requirement | Classes providing requirement |
| FR1 | Main, Dictionary, Word, Words, Verb, Noun |
| FR2 | Main, Dictionary |
| FR3 | Main, Dictionary |
| FR4 | Main, Dictionary |
| FR5 | Main, Dictionary, Word, Words, Verb, Noun |
| FR6 | Main, Dictionary, Word, Words, Verb, Noun |
| FR7 | Main, Dictionary |
| FR8 | Main, Dictionary, Word, Words, Verb, Noun |
| FR9 | Main, Dictionary, Word, Words, Verb, Noun |

# 3- DEPENDENCY DESCRIPTION

Backend:



Frontend:



**Backend dependencies**

**Dictionary**

|  |  |  |
| --- | --- | --- |
| **storeNewWord**  **-**setEnglish (Word)  -setWelsh (Word)  -getEnglish (Word) | **storeNewVerb**  -setEnglish (Verb)  -setWelsh (Verb)  -getEnglish (Verb) | **storeNewNoun**  -setEnglish (NounF)  -setWelsh (NounF)  -setGender (NounF)  -getEnglish (NounF) |
| **load**  -setEnglish (NounF)  -setWelsh (NounF)  -setGender (NounF)  -setEnglish (NounM)  -setWelsh (NounM)  -setGender (NounM)  -setEnglish (Verb)  -setWelsh (Verb)  -setEnglish (Word)  -setWelsh (Word) | **loadPrac**  -setEnglish (NounF)  -setWelsh (NounF)  -setGender (NounF)  -getEnglish (NounF)  -setEnglish (NounM)  -setWelsh (NounM)  -setGender (NounM)  -setEnglish (NounM)  -setEnglish (Word)  -setWelsh (Word)  -getEnglish (Word)  -setEnglish (Verb)  -setWelsh (Verb)  -getEnglish (Verb) | **findWordInEnglish**  -display (Word) |
| **findWordInWelsh**  -getWelsh (Word)  -display (Word) |  |  |

**Main**

|  |  |  |
| --- | --- | --- |
| **runMenu**  -load (Dictionary)  -storeNewWord (Dictionary)  -storeNewVerb (Dictionary)  -storeNewNoun (Dictionary)  -findWordInWelsh (Dictionary)  -findWordInEnglish (Dictionary) | **NounF**  getWelsh  -getWelsh (Word)  **setWelsh**  -setWelsh (Word) | **getEnglish**  -getEnglish (Word)  **setEnglish**  -setEnglish (Word)  **display**  -getEnglish (Word)  -getWelsh (Word) |
| **NounM**  getWelsh  -getWelsh (Word)  **setWelsh**  -setWelsh (Word)  **getEnglish**  -getEnglish (Word) | **setEnglish**  -setEnglish (Word)  **display**  -getEnglish (Word)  -getWelsh (Word)  **Verb**  getWelsh  -getWelsh (Word) | **setWelsh**  -setWelsh (Word)  **getEnglish**  -getEnglish (Word)  **setEnglish**  -setEnglish (Word) |
| **display**  -getEnglish (Word)  -getWelsh (Word)  **Word**  -NO DEPENDENCIES  **Others**  getType  -getType (Word) | **getWelsh**  -getWelsh (Word)  **setWelsh**  -setWelsh (Word)  **getEnglish**  -getEnglish (Word) | **setEnglish**  -setEnglish (Word)  **display**  -getEnglish (Word)  -getWelsh (Word) |

**Frontend dependencies**

**AddWordsDependencies**

|  |  |  |
| --- | --- | --- |
| **changeScreenToHelp**  -showHelp (Main)  **changeScreenToHome**  -showHome (Main)  **changeScreenToRevision**  -showRevision | **(updateTable)\_\_\_**  Which constructor was used and how many times it was used  -Word x2  Method calls:  -getType (Word)  -getEnglish (Word) x3  -getWelsh (Word) x3 |  |

**FlashCardsController**

|  |  |  |
| --- | --- | --- |
| **nextFlashcard**  -getEnglish (Word)  **previousFlashcard**  -getEnglish (Word)  **translate**  -getWelsh (Word) | **changeScreenToHelp**  -showHelp (Main)  **changeScreenToHome**  -showHome (Main)  **changeScreenToRevision**  -showRevision (Main) | **changeScreenToAddWord**  -showAdd (Main)  **updateTable**-getType (Word)  -Word (Word) x2  -getEnglish (Word) x2  -getWelsh (Word) x2  **initialise**  -getEnglish x2 |

**HelpSceneController**

|  |  |  |
| --- | --- | --- |
| **changeScreenToHelp**  -showHelp (Main)  **changeScreenToHome**  -showHome (Main) | **changeScreenToRevision**  -showRevision (Main) | **changeScreenToAddWord**  -showAdd (Main) |

**HomeController**

|  |  |  |
| --- | --- | --- |
| **changeScreenToHome**  -showHome (Main)  **changeScreenToRevision**  -showRevision (Main)  **changeScreenToAddWord**  -showAdd (Main)  **updateTable** Which constructor were used and how many times it was used  -Word  -NounF  -NounM  -Verb  Method calls:  -getEnglish (Word) x5  -getWelsh (Word) x5  -getType (Word) | **removingButtons** -> updateItem  -getEnglish (Word)  **addButtons** -> updateItem  Which constructor were used and how many times it was used:  -Word x2  Method Calls:  -getEnglish (Word) x5  -getType (Word) x2  -getWelsh (Word) | **updateTable2** Which constructor were used and how many times it was used  -Word x3  Method calls:  -getEnglish x3  -getWelsh x3  Main  -NO DEPENDENCIES |
|  |  |  |

**MatchTheWordsController**

|  |  |  |
| --- | --- | --- |
| **newSet**  -getEnglish (Word)  -getWelsh (Word)  **changeScreenToHelp**  -showHelp (Main)  **changeScreenToHome**  -showHome (Main) | **changeScreenToRevision**  -showRevision (Main)  **changeScreenToAddWord**  -showAdd (Main)  **updateTable**  -getType (Word)  -getEnglish (Word) x2  -getWelsh (Word) x2 | **assignAnswers**  -getEnglish (Word) x6  **initialize**  -getEnglish (Word)  -getWelsh (Word) |

**RevisionController**

|  |  |  |
| --- | --- | --- |
| **changeScreenToHelp**  -showHelp (Main)  **changeScreenToHome**  -showHome (Main)  **changeScreenToRevision**  -showRevision (Main) | **changeScreenToAddWord**  -showAdd (Main)  **changeScreenToMatch**  -showMatch (Main)  **changeScreenToPick**  -showPick (Main) | **changeScreenToTest**  -showTest (Main)  **changeScreenToFlashcards**  -showFlashcards (Main) |

# 4- INTERFACE DESCRIPTION

Java Classes:

### 4.1.1- Word: public, abstract, implements Comparable<Word>.

The Word class will represent all the words in the database as an object, the object will contain the English word, the Welsh word and the word type. This class was made an abstract a class so the ADT could store all of the different word types.

+ getWesh(): String

+ setWesh(String welsh): void

+ getEnglish(): String

+ setEnglish(String english): String

+ Word(String welshMeaning, String englishMeaning)

+ display(): void - This method detects the user's input asking for the details of the word object and inserting it in the verb constructor to then include it in the dictionary.

### 4.1.2- Other: public, extends Word

The Other class is being used for all words that are not noun or verbs, this class inherits from the word class.

+ getWelshMeaning(): String

+ setWelshMeaning(String welshMeaning): void getEnglishMeaning(): String

+ setEnglishMeaning(String englishMeaning): void

+ Words(String welshMeaning, String englishMeaning)

+ display(): void - This method detects the user's input asking for the details of the word object and inserting it in the verb constructor to then include it in the dictionary.

### 4.1.3- Verb: public, extends Word.

The Verb class is being used for all words that verbs, this class will inherit from the Word class.

+ getWelsh(): String

+ setWelsh(String welsh): void

+ getEnglish(): String

+ setEnglish(String english): String

+ Verb(String welshmeaning, String englishMeaning)

+ display(): void - Detect the user's input asking for the details of the verb object and inserting it in the verb constructor to then include it in the dictionary.

### 4.1.4- NounF: public, extends Word.

The NounF class is being used for all of the feminine nouns, this class will inherit from the Word class.

+ getWelsh(): String

+ setWelsh(String welsh): Void

+ getEnglish(): String

+ setEnglish(String english): String

+getGender(): String

+setGender(): void

+ getType(): String

### 4.1.5 - NounM: public, extends Word.

The NounF class is being used for all of the Masculine nouns, this class will inherit from the Word class.

+ getWelsh(): String

+ setWelsh(String welsh): Void

+ getEnglish(): String

+ setEnglish(String english): String

+getGender(): String

+setGender(): Void

+ getType(): String

### 4.1.6- Dictionary: public.

The dictionary class stores and manages the two databases which allows the user to choose the default language for how the contents of the dictionary are ordered. This class will also be used to store the users practice words, words that the user has added through the ‘add new words’ function and it will be used to implement the flashcards system.

+ Dictionary() -

+ storeNewWord(): void -

+ storeNewVerb(): void - This method detects the user's input asking for the details of the verb and inserting it in the verb constructor to then include it in the dictionary.

+storeNewNoun(): void - This method detects the user's input asking for the details of the verb and inserting it in the verb constructor to then include it in the dictionary.

+searchWords(): string- Search for the word, suggesting results close to what the user is typing.

+ save (String datafile, String favfile): void- Save all data in the database to external files

+ load (String datafile, String favfile): void- Load all word data into the database from external json files

+searchWord(): void - This method checks the characters in the search bar and compares the characters to the words in the database and offers the matching options to the user. If there are not any different matches, offer the user the three words that match the most characters.

+flashcards(): void - This method allows the user to test their knowledge through the flash card system.

+sixWordsTest(): void - This method runs a test based on the user’s favourite words.

+typeWordTest(): void - This method runs a test based on the user’s favourite words.

+fourWordsTest(): void - This method runs a test based on the user’s favourite words.

### 4.1.7- Main: public.

This class will run the program when called.

+ Main()

- printMenu(): void

- runMenu(): void

+ main(): static, void

JavaFX Classes

### 4.2.1- Main

+start(Stage Primary Stage): void - Initialises the app, sets the Stage as the primaryStage and sets the tile to “Welsh Learner”.

+main() - Responsible for running the program

### 4.2.2- HomeController.

The HomeController class is responsible for implementing everything you see on the home page. The largest process the class has to deal with is displaying the dictionary on the page and the buttons that control the adding and removing of words from the dictionary. The class is also responsible for implementing the search bar for the user to look up specific words from the dictionary, it attempts to autocomplete the words depending on what you type in.

* goToAdd() - attached to "Add Words" menu option button changes scenes to AddWordsScene
* goToHelp() - attached to "Help" menu option button changes scenes to HelpScene
* goToRevision() - attached to "Revision" menu option button changes scenes to RevisionScene

### 4.2.3- HelpSceneController.

The HelpSceneController class is responsible for implementing everything you see on the help page. This is the simplest controller. Its purpose is to display buttons that, when clicked, should take the user to a page that has useful information about the other pages in the application.

* goToHome() - attached to "Home" menu option button changes scenes to HomeScene
* goToAdd() - attached to "Add Words" menu option button changes scenes to AddWordsScene
* goToHelp() - attached to "Help" menu option button changes scenes to HelpScene
* goToRevision() - attached to "Revision" menu option button changes scenes to RevisionScene
* changesScreenButtonPushed(1,2,3) - responsible for changing image placed in the middle according to desired help information

### 4.2.4- AddWordsController.

The AddWordsController class is responsible for implementing everything you see on the add words page. This controller is identical to the home page except, the practice section is replaced by the add words section. The add words section has 2 places to input the English and Welsh word, a dropbox to select what the word type is, and a submit button.

* goToHome() - attached to "Home" menu option button changes scenes to HomeScene
* goToAdd() - attached to "Add Words" menu option button changes scenes to AddWordsScene
* goToHelp() - attached to "Help" menu option button changes scenes to HelpScene
* goToRevision() - attached to "Revision" menu option button changes scenes to RevisionScene
* addButton() - is attached to "Submit" button which takes the word (user input) and passes it to dictionary ADT

### 4.2.5- RevisionController.

The RevisionController class is responsible for implementing everything you see on the revision page. The revision page displays four buttons. These buttons lead to the flashcards, match the words, pick the correct words, and the test functions of the application.

* goToHome() - attached to "Home" menu option button changes scenes to HomeScene
* goToAdd() - attached to "Add Words" menu option button changes scenes to AddWordsScene
* goToHelp() - attached to "Help" menu option button changes scenes to HelpScene
* goToMatch() - changes scenes to MatchTheWordsScene
* goToPick() - changes scenes to PickTheCorrectWordScene
* goToTest() - changes scenes to TestScene

### 4.2.6- FlashcardsController

The FlashcardsController class is responsible for implementing everything you see on the flashcards page. The flashcards page displays randomly selected words from the user’s practice list in the middle of the page. Underneath this are three buttons, the button to translate the word, a next button, and a previous button. When the user reaches the last word the next button is replaced with a menu button that returns them back to the revision page.

* goToHome() - attached to "Home" menu option button changes scenes to HomeScene
* goToAdd() - attached to "Add Words" menu option button changes scenes to AddWordsScene
* goToHelp() - attached to "Help" menu option button changes scenes to HelpScene
* goToRevision() - attached to "Revision" menu option button changes scenes to RevisionScene
* nextFlashcard(), previousFlashcard() - responsible for specific buttons and according to that displays the flashcard
* translate() - attached to "translate" button on the middle, "flips" the flashcard revealing the translation of the word in the other language
* getFlashcard() - method that randomly takes a word from the ADT assigning welsh and english meaning to the variables, on which the flashcard works. It gets called everytime user presses the "next flashcard" button.

## 4.2.7- MatchTheWordsController

The MatchTheWordsController class is responsible for implementing everything you see on the match the words page. On the left side, eight buttons are displaying four English words and four Welsh words. When a button is pressed the button changes colour and the next button pressed changes to that same colour. When the user clicks submit the score is displayed on the top right corner.

## 4.2.8- PickTheCorrectWordController

The PickTheWordController class is responsible for implementing everything you see on the pick the words page. The page displays a randomly selected word from the users practice list in the center of the page with six options possible translations beneath it. When the user’s answer is selected all the answers change colour. The incorrect answers change to red and the correct answers turn green. If the user correctly matches the words then the word turns green and the score is updated. When the user is finished they must press the next button to continue. The page also comes with an option to change what language you’re translating.

## 4.2.9- TestController

The TestController class is responsible for implementing everything you see on the test page. The test page displays a randomly selected word from the practice list and a text box underneath it for the user to translate the word. When the user clicks the submit and the word is translated correctly the score at the top left is updated and if the word is not translated correctly the correct answer is displayed above the text box. The next button at the top right is used to move onto the next word.

Testing Classes

## 4.3.1 – AddWordsController

This class prevents you from adding a word if a word type has not been selected from the word type dropbox. It also prevents the user from adding English words longer than 48 and adding Welsh words longer than 28 because there aren’t any words longer than that.

## 4.3.2 – MatchTheWordsController

This class prevents the user from playing the game if the user has fewer than 8 words in their practice list. It also prevents the user from submitting their answers if the user has not assigned any of the words.

## 4.3.3 – PickTheWordController

This class prevents the user from playing the game if the user has fewer than 6 words in their practice list.

# 5- DETAILED DESIGN:

searchWord(): Look through the MultiMap that stores words, get the words in English or welsh and compare them to what the user is typing. If four or more letters match, offer the word as a suggestion.

storeNewWord(): Add word will ask the user to specify if the user wants to add a noun,verb or another type of word. By scanning the user's input and using an if statement, it will call storeNewNoun(), storeNewVerb() if it’s told that it is a noun or verb. If not it will ask for the details of the word and add it to the dictionary.

# 6- REFERENCES:

[1] **-** SE.QA.05: Software Engineering Group Project - Design Specifications Standards

7- VERSION CONTROL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | CCF No. | Date | Changes Made | Changes By |
| 1.0 | N/A | 02/04/2020 | N/A – original version | jas117, jab167 |