**Software Engineering Group Projects –**

**Design Specification Standards**

|  |  |
| --- | --- |
| Author: | Rik7, jub27, bas17 & bas21 |
| Config Ref: | SE.QA.05 (for CS22120) |
| Date: | 25th January 2020 |
| Version: | 1.4 |
| Status: | Release |

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Copyright © Aberystwyth University 2016

**CONTENTS**

[1 INTRODUCTION 3](#_Toc35443008)

[1.1 Purpose of this Document 3](#_Toc35443009)

[1.2 Scope 3](#_Toc35443010)

[1.3 Objectives 3](#_Toc35443011)

[2 DECOMPOSITION DESCRIPTION 3](#_Toc35443012)

[2.1 Programs in system 3](#_Toc35443013)

[2.2 Significant classes in each program 4](#_Toc35443014)

[2.3 Modules shared between programs – N/A 5](#_Toc35443015)

[2.3 Mapping from requirements to classes 5](#_Toc35443016)

[3 DEPENDENCY DESCRIPTION 5](#_Toc35443017)

[3.1 Component Diagrams 5](#_Toc35443018)

[4 INTERFACE DESCRIPTION 6](#_Toc35443019)

[4.1 Main 6](#_Toc35443020)

[4.2 ConfigClass 7](#_Toc35443021)

[4.3 Word 7](#_Toc35443022)

[4.4 Question 7](#_Toc35443023)

[4.5 MyWords Controller 8](#_Toc35443024)

[4.6 LearnController 8](#_Toc35443025)

[4.7 Controller 8](#_Toc35443026)

[5 DETAILED DESIGN 8](#_Toc35443027)

[5.1 Sequence diagram 9](#_Toc35443028)

[5.2 Significant algorithms 13](#_Toc35443029)

[5.3 Significant Data Structures 14](#_Toc35443030)

# INTRODUCTION

## Purpose of this Document

The purpose of this document is to describe the implementation of the software Welsh Learning App. This document tracks the necessary information required to effectively define architecture and system design of the Welsh Learning App.

## Scope

This document describes the functions of the software “Welsh Learning App” and the implementation of classes, interfaces, controllers and relationship used in the back end. It also provides an explanation of the algorithms used in the implementation of the software’s logic and examples of the algorithms.

## Objectives

The main objective is to provide overall structure of the Welsh Learning App, in terms of classes, interfaces and controllers. Explaining how each feature of the application has been implemented, and how these features match the requirement specification.

# DECOMPOSITION DESCRIPTION

## Programs in system

There is only one program build using Java. The packages have been separated according to their use and role within the application.

Json package, as the name suggests, is responsible for holding all the text inputs and outputs. This package deals with loading data into the program and saving data into multiple files.

Mains package, the most crucial package, is responsible for running the user interface and enables functionality of the game.

Assets package is responsible for all the labels, buttons and text readers.

## Significant classes in each program

There is only one program present in the project, therefore all the classes are associated to the Welsh Learning App. These will be the main 4 classes in the Welsh Learning App.

### 2.2.1 Main

|  |  |
| --- | --- |
| Class | Main |
| Purpose | The purpose of this class is to primarily connect the backend and frontend together. |

### 2.2.2 Config

|  |  |
| --- | --- |
| Class | Config |
| Purpose | Config is responsible for initiating all the communication between the classes. |

### 2.2.3 Word

|  |  |
| --- | --- |
| Class | Word |
| Purpose | Word class is responsible for loading and saving data. |

### 2.2.4 Question

|  |  |
| --- | --- |
| Class | Question |
| Purpose | This class creates generates a multiple-choice question for the user to answer. |

## 2.3 Modules shared between programs – N/A

## Mapping from requirements to classes

|  |  |
| --- | --- |
| *Requirement* | *Classes providing requirement* |
| FR1 | Config, Word |
| FR2 | Main, DictionaryController |
| FR3 | Main, DictionaryController |
| FR4 | Config |
| FR5 | Config, Word |
| FR6 | Config, Word |
| FR7 | Config, Question, Word |
| FR8 | Config, Word |
| FR9 | Config, Word, Question |
| FR10 | Config, Word, Question |

# DEPENDENCY DESCRIPTION

# 

## Component Diagrams

A description of each element mentioned in the outline structure is presented in the following sections.

# INTERFACE DESCRIPTION

## Main

This class is the main class that is used to run the program.

public void start(Stage primaryStage) - A method to initialize the app, sets Stage as primaryStage, and sets its Title to "Welsh Learning App". Also runs showMenu() method

Public void showMenu() - A method to load Menu.fxml file as a mainLayouts. Creates a new Scene with mainLayout and sets primaryStage scene as mainLayout and shows it.

Public static void showLearn() - A method to load Learn.fxml file sets it as a center part of mainLayout.

Public static void showDictionary() - A method to load Learn.fxml file sets it as a center part of mainLayout.

Public static void showPractice() - A method to load Practice.fxml file in the background.

Public static void backToMenu() - A method to load Menu.fxml file in the background.

Public static void showOptions() - A method to load Options.fxml file in the background.

Public static void showMyWords() - A method to load MyWords.fxml file in the background.

Public static void showFlashCards() – A method to load Flashcards.fxml file in the background.

Public static void showGuessWord() - A method to load GuessWord.fxml file in the background.

## Config

This class is used to store lists of words and also sort and search these lists.

Public Void run() – this method runs the program and calls all the methods based on the users input.

Private String extractEngWord(String term) – this method searches the treemap and outputs all the words that fit the search condition in alphabetical order and translate it to Welsh.

Private String extractWelWord(String term) – this method searches the treemap and outputs all the words that fit the search condition in alphabetical order and translate it to English.

Public Void addWordtoDictionary () – this method is used to add a word to the treemap in the sorted treemap.

Public Void removeWord(String word) – this method is used to remove a word in the sorted treemap.

Public Void save(String file) – this method is used to put words from the ArrayList in the object into a json file.

Public Void load(String file) – this method is used to load words from an json file into the ArrayList in the object.

Private Void dictionaryMenu() – Shows all words to the user in alphabetical order.

Private Void addToFavourite(String word) – this method is used to add a word from the main WordList to the favourite WordList.

Private Void removeFromFavourite(String word) – this method removes a word from the favourite WordList.

Public void practiceGame() – this method runs the practice quiz and depending on the input, it either uses the favourite list of words or the main tree map . It then displays the results and all the words the user got incorrect.

Public Void flashCard(Word word) – this method allows the user to look at the word in a selected language and lets the user then check if it was the word they thought it was. (apparently this is in the specification).

Private Void learnMenu() – Randomly display words in sets of 4 to this user.

## Word

This class is used for storing information for each word.

Word(String English, String welsh, String type) – this creates a word object.

Package-private String getEnglish() – this method gets the English translation of the word.

Package-private String getWelsh() – this method gets the Welsh translation of the word.

Package-private String getType() – this method gets the word type of the word.

Public void load(Iterator<JSONObject> iterator) – loads the words from json file.

Public void save(JSONObject object, String welsh, String wordType, String english) – saves word to json file.

## Question

This class is used to create and display multiple choice questions

Question(String wordToAnswer, String option1, String option2, String option3, String option4, String answer) – this is a constructor that creates a question with the input of a set of strings.

Package-private String getWordToAnswer() – this method returns the question

Package-private String getOption1() – this method returns option 1 of the multiple choice

Package-private String getOption2() – this method returns option 2 of the multiple choice

Package-private String getOption3() – this method returns option 3 of the multiple choice

Package-private String getOption4() – this method returns option 4 of the multiple choice

Package-private String getAnswer – this method returns the correct answer.

## PracticeController

This class displays all the options used for learning activities like guess the word and flash cards.

Public void goFlashcards() – This method is used to change the screen to flashCards.

Public void goGuessWord() - This method is used to change the screen to guessWord.

Public void goMenu() – Used to go to main menu.

Public void Exit() - A method that terminates the program.

Public void goBack() – Changes screen to the previous screen.

## FlashcardsController

Public void goMenu() – This method changes screen to main menu.

Public void Exit() - A method that terminates the program.

Public void goBack() – Changes screen to the previous screen.

## GuessWordController

Public void goMenu() – This method changes screen to main menu.

Public void Exit() - A method that terminates the program.

Public void goBack() – Changes screen to the previous screen.

## MyWords Controller

This class is used to display all the words in the program onto the screen.

Public void goMenu() - A method that changes scene to Menu.fxml using Main method backToMenu().

Public void Exit() - A method that terminates the program.

Private ObservableList<Word> getWords() - A method that returns ObservableList of Words.

Public void goPractice() - A method that changes the scene to Practice.fxml using Main method showPractice().

Private void addButtons() - A method that adds a button to each cell in a Table Column.

Private void searchForWords() - A method that searches content of table columns to find a matching sequence of Characters to user input in Text Field.

## LearnController

Public void goMenu() - A method that changes scene to Menu.fxml using Main method backToMenu().

Public void Exit() - A method that terminates the program.

Private ObservableList<Word> getWords() - A method that returns ObservableList of Words that consists of all Words added by a user to that List.

Private void addButtons() - A method that adds a button to each cell in a Table Column.

## DictionaryController

Public void goMenu() – Takes user back to main menu.

Private void addWordsWindow() – shows window which is used to add words.

Public void Exit() - A method that terminates the program.

Private ObservableList<Word> getWords() - A method that returns ObservableList of Words that consists of all Words added by a user to that List.

Private void addButtons() - A method that adds a button to each cell in a Table Column.

## AddWordController

Public void newWordButtonPushed – pushes words into the dictionary fxml.

## Controller

Public void goToLearn() - A method that changes the scene to Learn.fxml using Main method showLearn().

Public void goMenu() - A method that changes the scene to Menu.fxml using Main method backToMenu().

Public void goToDictionary() - A method that changes the scene to Dictionary.fxml using Main method showDictionary().

Public void goToOptions() - A method that changes the scene to Options.fxml using Main method showOptions().

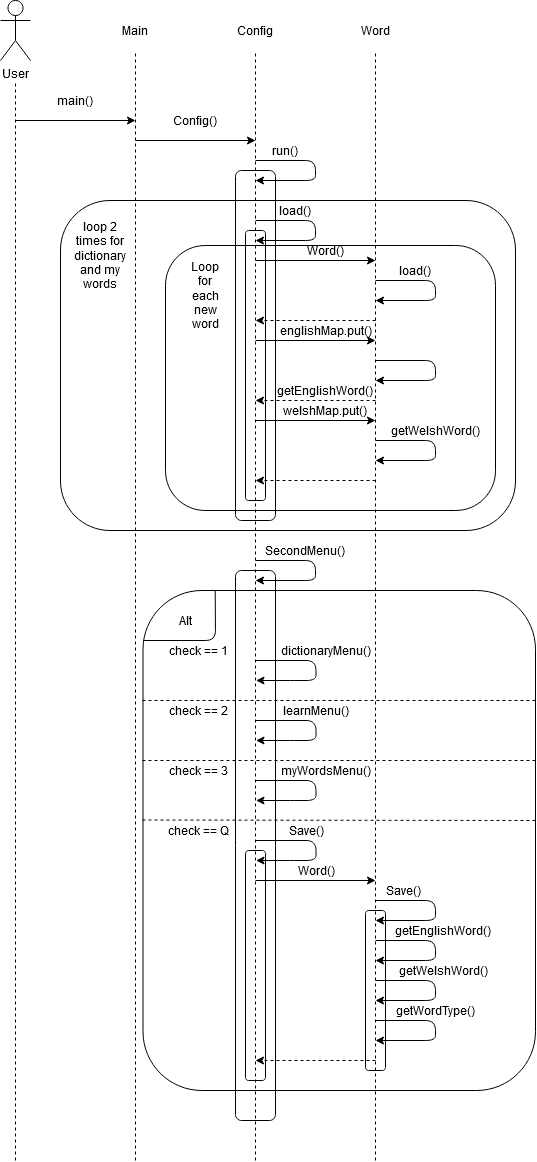
Public void goToMyWords() - A method that changes the scene to MyWords.fxml using Main method showMyWords().

Public void Exit() - A method that terminates the program.

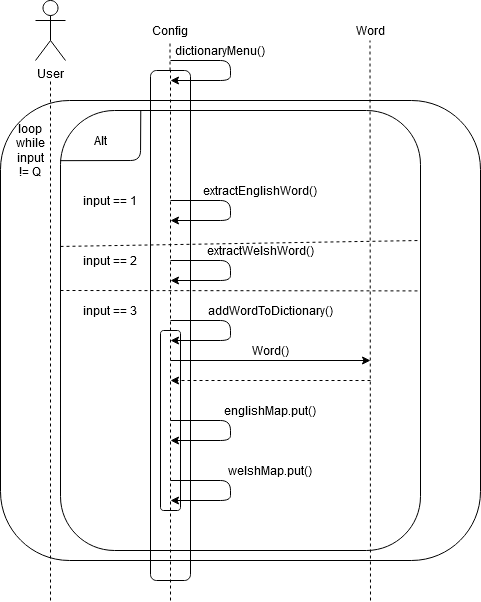
# DETAILED DESIGN

## Sequence diagram

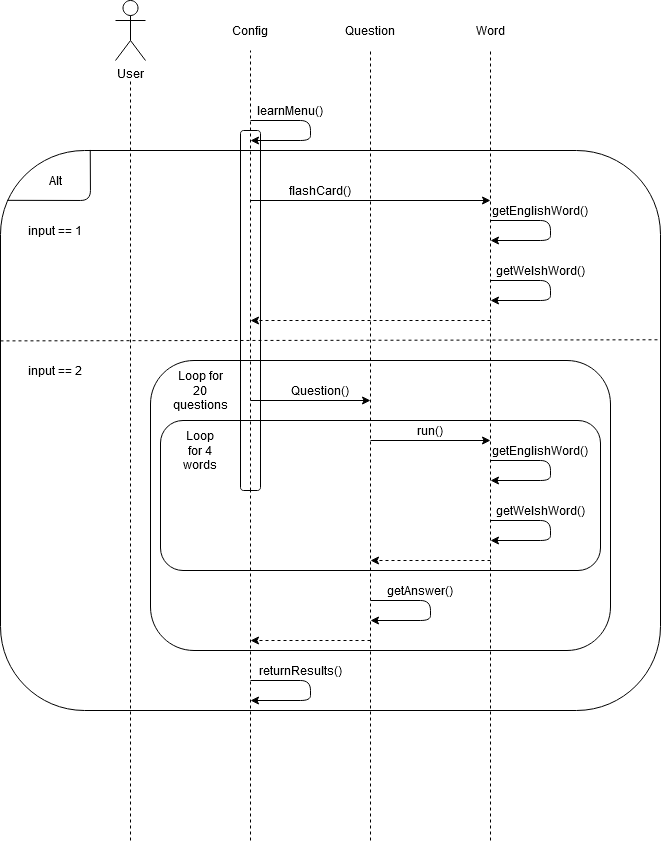
### Start, load, and save



### dictionary



### Flash Cards and Multiple Choice



## Significant algorithms

### LOADING

For loading of the dictionary, the application will take a JSON file as input. The default for this is the dictionary.json

provided by the client, however, when the user adds a new word to the dictionary a new JSON file that includes the added word will be created and used when loading the dictionary. This is to ensure that the user is able to save new words and that they appear in the dictionary when the app is launched again. The default file will remain unchanged and not overwritten so that if the new file is corrupted the application will be able to still load a dictionary.

In order to read JSON files, the application will use JSON-simple-1.1.jar.

The JSONParser will be used to parse the file information into a JSONArray which contains the dictionary.

Then the array will be iterated over and while there is a word in the array the iterator will loop over it.

For each of the word, a new Word object will be created which contains the Welsh and English translation and the word type. This Word object will be put into two maps, one English map and one Welsh map. The type of maps used is TreeMap, the reason for using the Tree version of the Map is that it is a sorted map which allows for the words to be sorted alphabetically. There is a map for both English and Welsh due to the fact that the alphabetical order of the words will be different in the two languages and this makes it simpler to do the ordering.

### Searching

When searching the user will choose to either search by Welsh or English, which will also select which map to use for

searching the dictionary. After input from the user, the system will check if the selected map contains the key which will be

either the English or Welsh word. If the map contains the specified key it will return the word with all its attributes and if it does not contain the key then no word will appear.

The overall runtime for .containskey method for SortedMap is O(1) - O(logn).

### GuessTheWord

The game where the user has to pick the correct translation of a word from their pratice list will chose a random word

from the users list of word and will display it. This is done by using the Random utility class to generate random numbers which

will correspon the the index of the words in the pratice list. Then the wortd with that index is chosen to be translated by

the user. Four options will apear in either Welsh or English depending on what user choses. These will include the correct

translation of the word and three randomly selected translations from the dictionary, which will be selected in a similar

way to that of the word from the practice list. A correct selection will add a point to the score and an incorrect will not.

The points are stored in an int and saved to enable them to see their progress.

## Significant Data Structures

**REFERENCES**

1. QA Document SE.QA.01 - Quality Assurance Plan.

1. QA Document SE.QA.03 - Project Management Standards.

1. QA Document SE.QA.08 - Operating Procedures and Configuration Management Standards.

1. QA Document SE.QA.02 - General Documentation Standards.

1. QA Document SE.QA.09 - Java Coding Standards.

**DOCUMENT HISTORY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Version* | *CCF No.* | *Date* | *Changes made to document* | *Changed by* |
| 1.0 | N/A | 20/02/20 | Document Created | Rik7 |
| 1.1 | N/A | 06/03/20 | Updated Interface Description | Jub27 |
| 1.2 | N/A | 17/03/20 | Added Sequence Diagrams | Jub27 |
| 1.3 | N/A | 18/03/2020 | Updated Interface Description | Rik7 |