Software Engineering Group Projects – Design Specification Standards

Author:	Rik7, jub27, bas17 & bas21
Config Ref:	SE.QA.05 (for CS22120)
Date:	25th February 2020
Version:	1.0
Status:	Draft

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB

CONTENTS

1 INTRODUCTION

1.1 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.

1.2 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.

1.3 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.

2 DECOMPOSITION DESCRIPTION

2.1 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.

2.2 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 5 classes in the Welsh Learning App.

1. 2.2.1 Main

Class	**Main**
Purpose	Main is the class in which the program is launched and you will find that this class is used primarily to connect the backend and frontend together.

2. 2.2.2 Config

Class	**Config**
Purpose	Config is the main class that handles all backend communications, including searching, adding, saving and loading words from json file. This class should also handle how any core backend features interacts with other backend features.

3. 2.2.3 Word

Class	**Word**
Purpose	Word class is responsible for holding all the data for each word that will be stored in memory, it contains English, Welsh and their Word type.

4. 2.2.4 Question

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

5. 2.2.4 UI

Class	**UI**
Purpose	This class controls all the FXML files. Opening, closing and changing scenes are possible from this class.

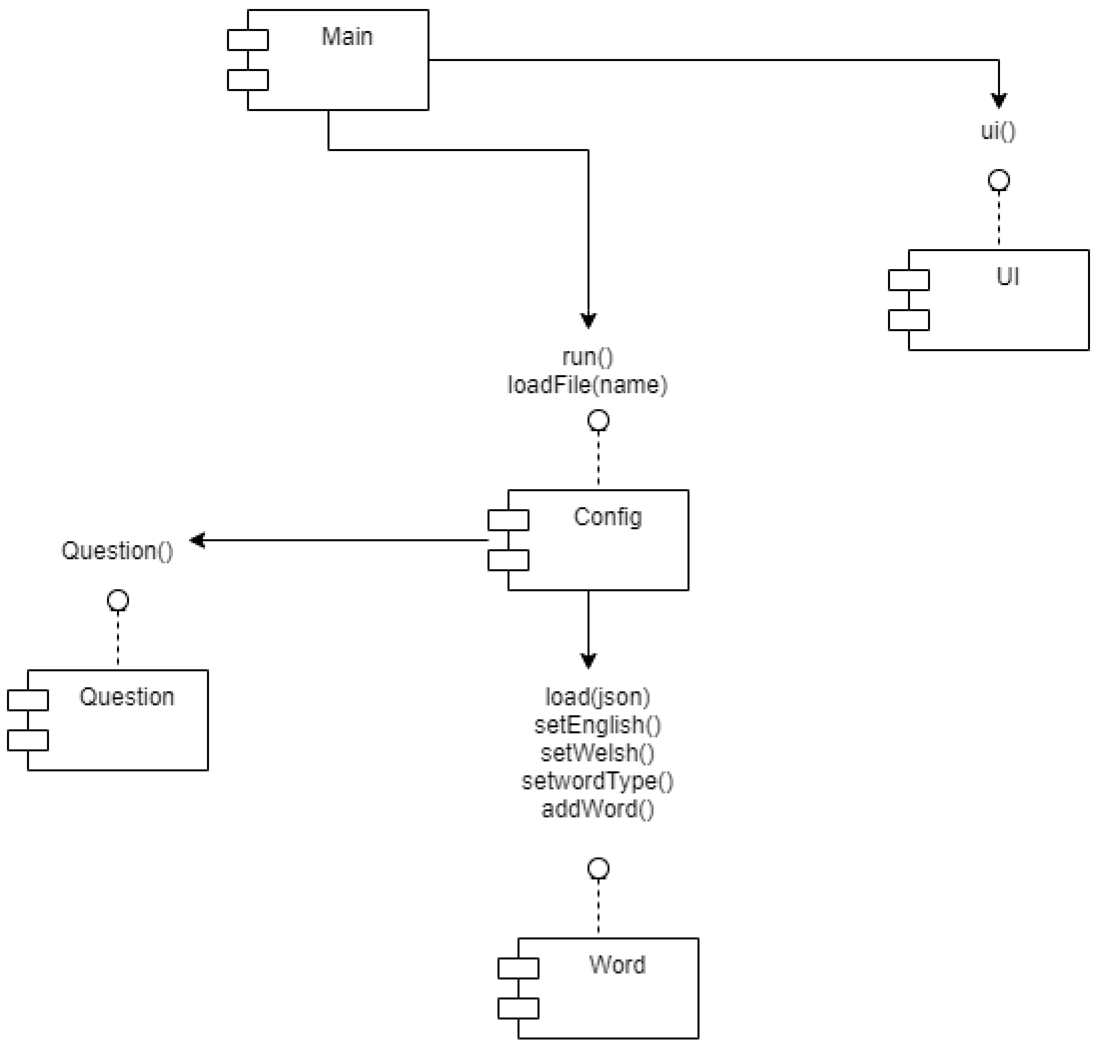
2.3 Modules shared between programs – N/A

2.3 Mapping from requirements to classes

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

3 DEPENDENCY DESCRIPTION

3.1 Component Diagrams



4 INTERFACE DESCRIPTION

4.1 Main

This class is the main class that is used to run the program. Main class is responsible for running backend and frontend.

- Public static void main(String[] args) – A method to start operations on the backend as well as frontend.

4.2 UI

This class deals with all the frontend of the application. The UI class extends the imported JavaFX Application class, allowing us to provide the user an interface based on JavaFX.

- 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.

4.3 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.

4.4 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.

4.5 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.

4.6 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.

4.7 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.

4.8 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.

4.9 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.

4.10 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.

4.11 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.

4.12 AddWordController

- Public void newWordButtonPushed – pushes words into the dictionary fxml.

4.13 Controller

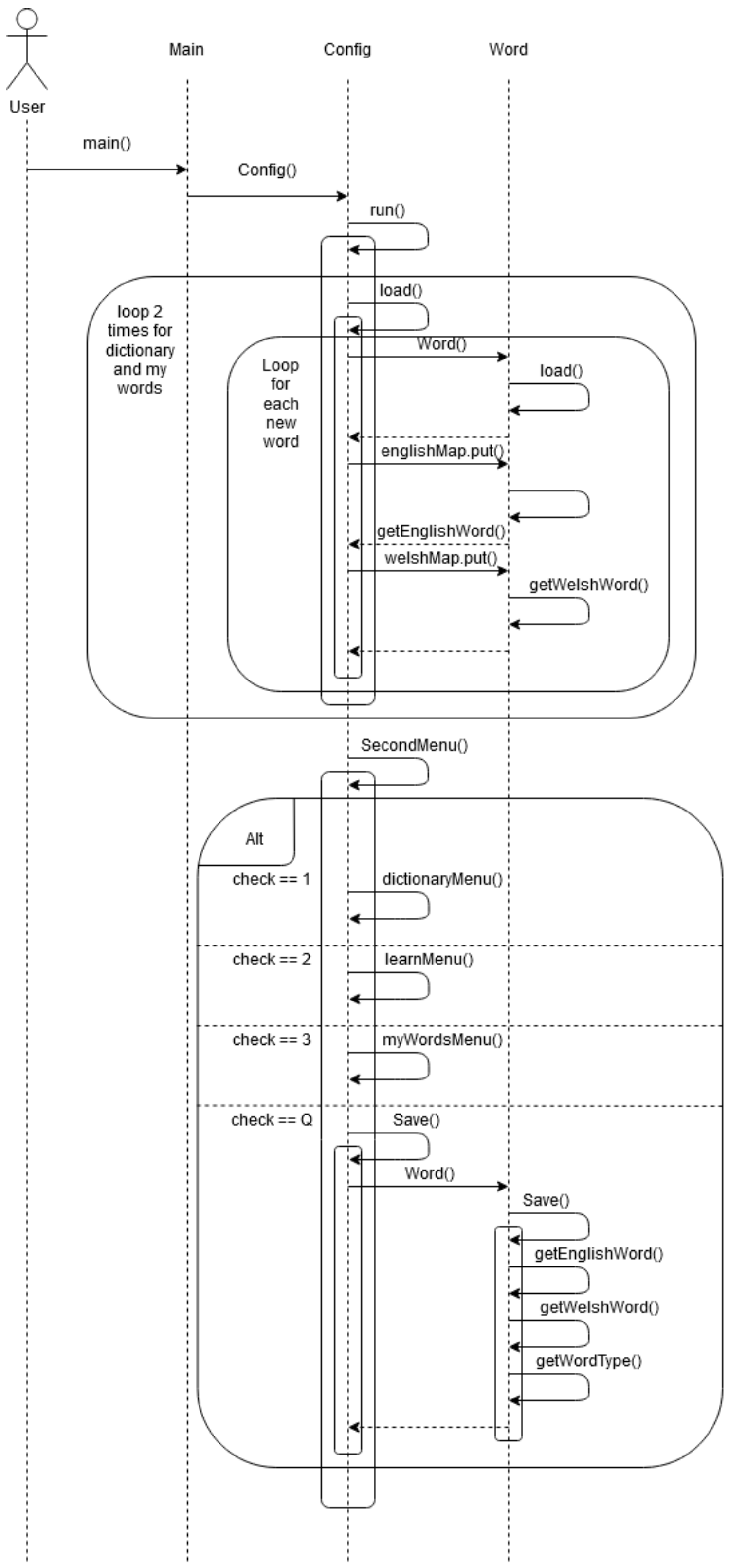
- Public void goToLearn() - A method that changes the scene to Learn.fxml using Main method showLearn().
- Public void goMenu() - A method that changes the scene to Menu.fxml using Main method backToMenu().
- Public void goToDictionary() - A method that changes the scene to Dictionary.fxml using Main 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.

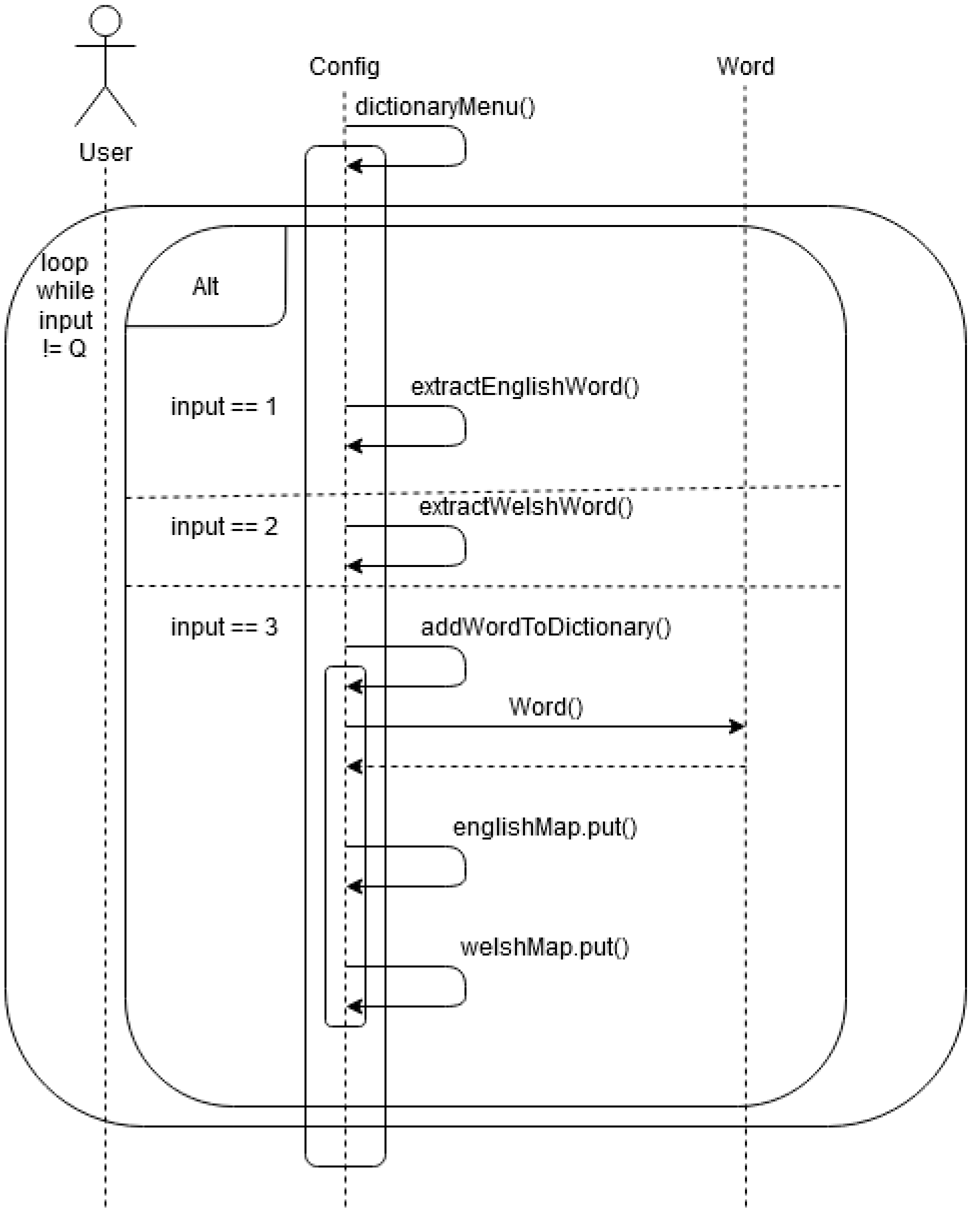
5 DETAILED DESIGN

5.1 Sequence diagram

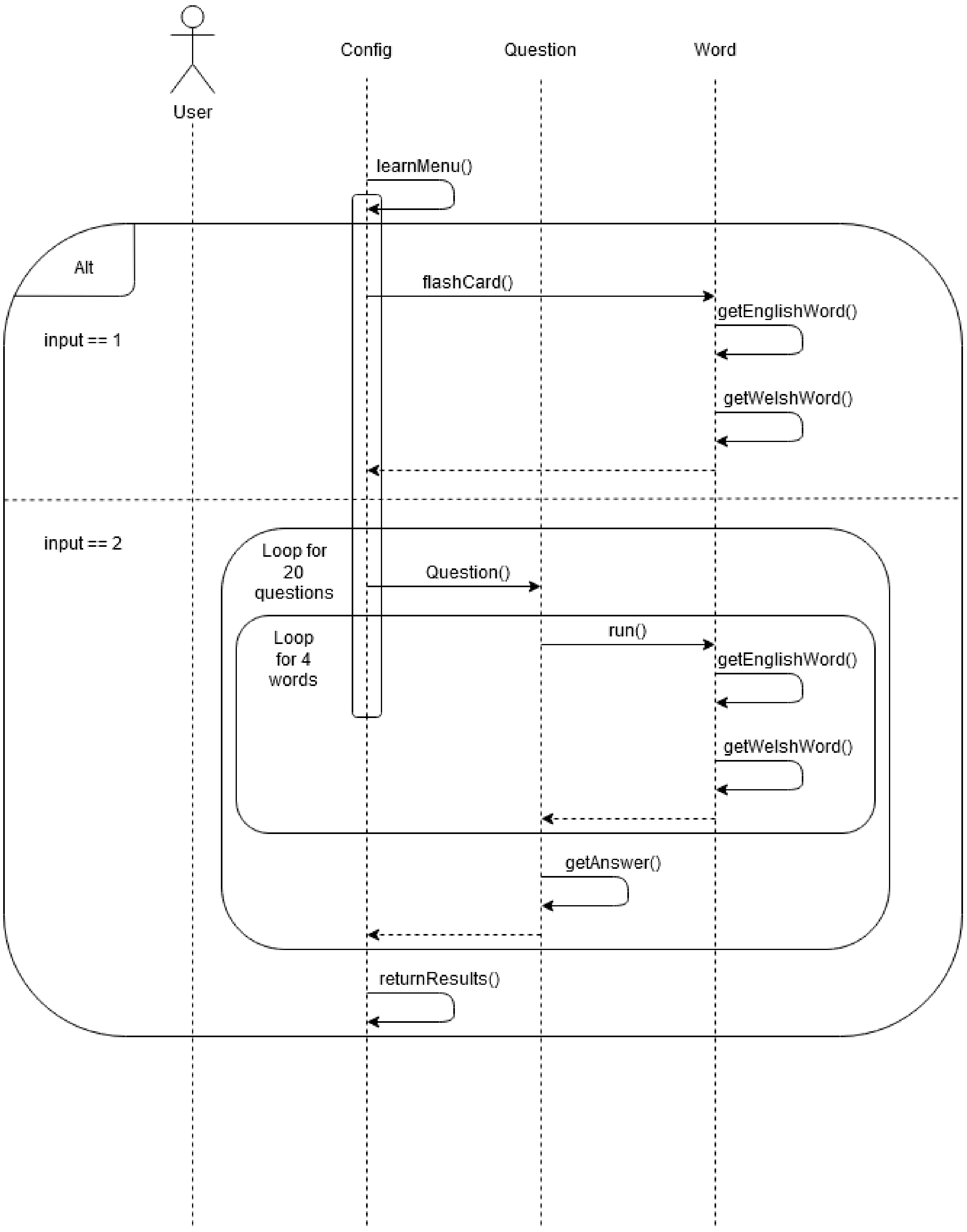
1. Start, load, and save



2. Dictionary



3. Flash Cards and Multiple Choice



5.2 Significant algorithms

1. 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, the word gets appended into the JSON file.

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 since the alphabetical order of the words will be different in the two languages and this makes it simpler to do the ordering.

2. Searching

When searching the user will choose to either search by Welsh or English, which will also select which Treemap 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).

3. GuessTheWord

The game where the user has to pick the correct translation of a word from their practice list will choose a random word from the user's list of word and will display it. This is done by using the Random utility class to generate random numbers which will compare the index of the words in the practice list. Then the word with that index is chosen to be translated by the user.

Four options will appear in either Welsh or English depending on what the user chooses. 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.

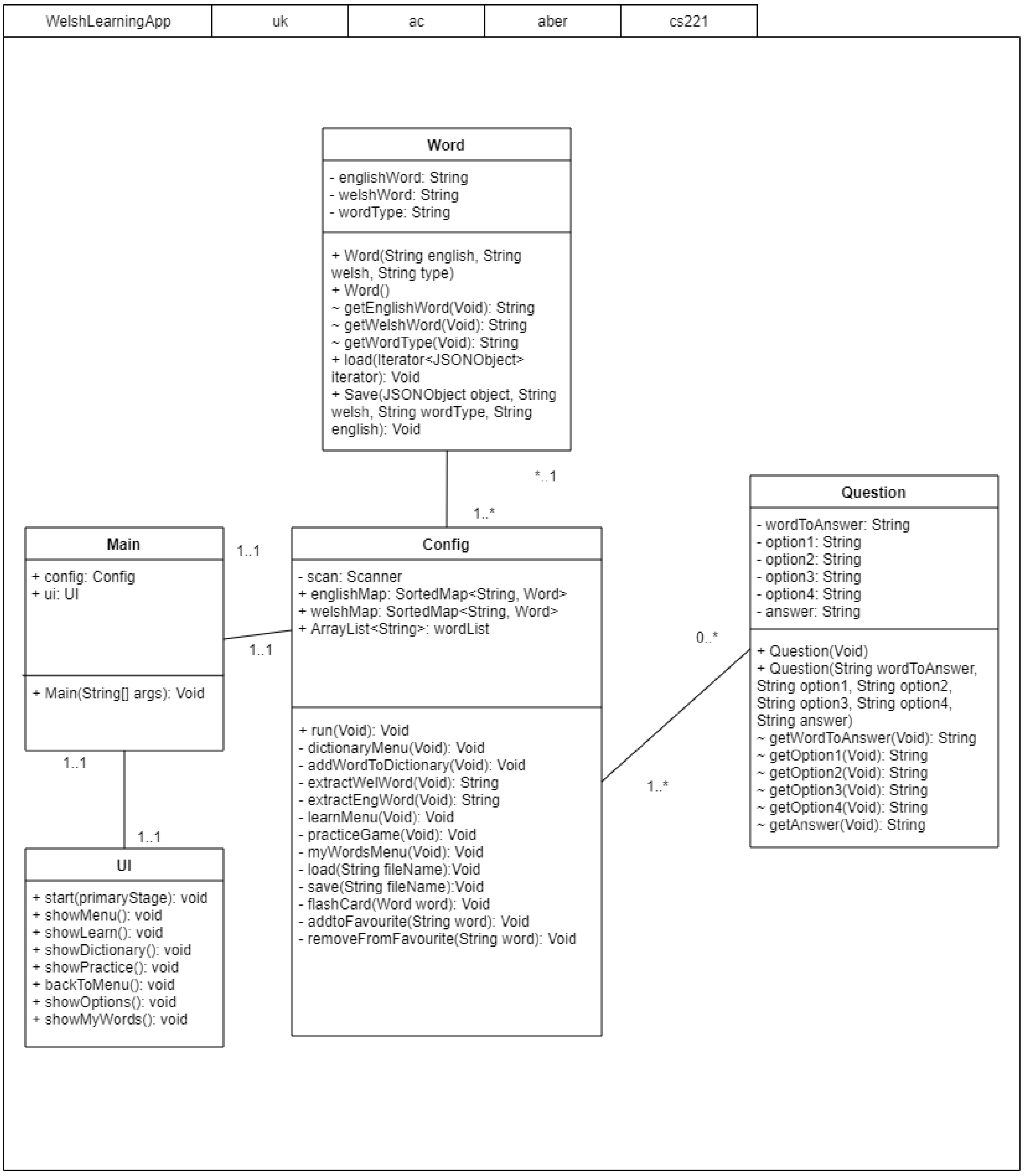
4. Flashcards

When the user uses the flashcard practice tool. They will select if they want English or Welsh. It will then output a random word in the selected language.

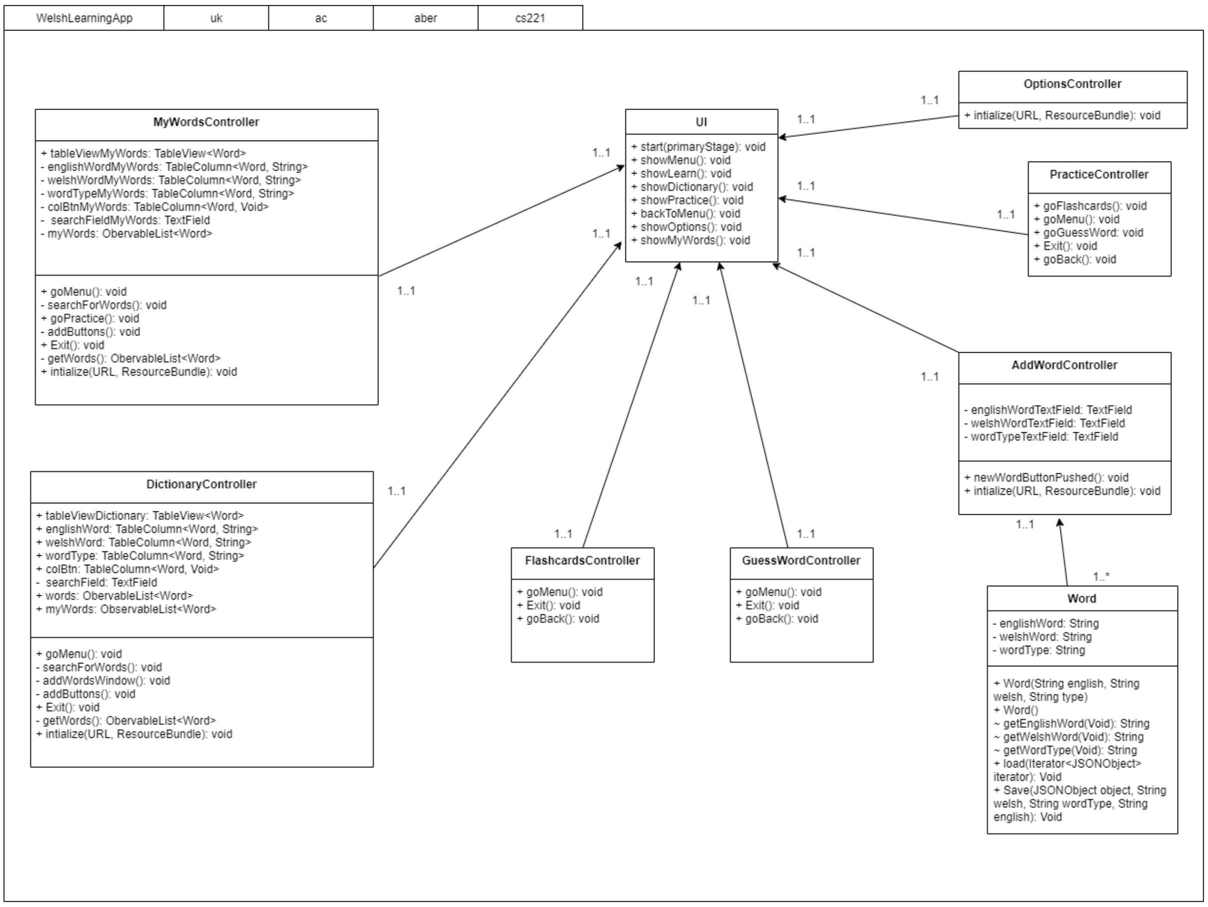
They will then click the button, the button text which is the word in the selected language will be changed to the translated word. This will simulate a flash card in real life.

5.3 Significant Data Structures

1. Application Logic


WelshLearningApp
uk
ac
aber
cs221
Word
- englishWord: String
- welshWord: String
- wordType: String
+ Word(String english, String welsh, String type)
+ Word()
~ getEnglishWord(Void): String
~ getWelshWord(Void): String
~ getWordType(Void): String
+ load(Iterator<JSONObject> iterator): Void
+ Save(JSONObject object, String welsh, String wordType, String english): Void
*..1
1..*
Main
+ config: Config
+ ui: UI
+ Main(String[] args): Void
1..1
1..1
Config
- scan: Scanner
+ englishMap: SortedMap<String, Word>
+ welshMap: SortedMap<String, Word>
+ ArrayList<String>: wordList
+ run(Void): Void
- dictionaryMenu(Void): Void
- addWordToDictionary(Void): Void
- extractWelWord(Void): String
- extractEngWord(Void): String
- learnMenu(Void): Void
- practiceGame(Void): Void
- myWordsMenu(Void): Void
- load(String fileName):Void
- save(String fileName):Void
- flashCard(Word word): Void
- addtoFavourite(String word): Void
- removeFromFavourite(String word): Void
0..*
1..*
Question
- wordToAnswer: String
- option1: String
- option2: String
- option3: String
- option4: String
- answer: String
+ Question(Void)
+ Question(String wordToAnswer, String option1, String option2, String option3, String option4, String answer)
~ getWordToAnswer(Void): String
~ getOption1(Void): String
~ getOption2(Void): String
~ getOption3(Void): String
~ getOption4(Void): String
~ getAnswer(Void): String
1..1
1..1
UI
+ start(primaryStage): void
+ showMenu(): void
+ showLearn(): void
+ showDictionary(): void
+ showPractice(): void
+ backToMenu(): void
+ showOptions(): void
+ showMyWords(): void

2. Application Interface


WelshLearningApp
uk
ac
aber
cs221
MyWordsController
+ tableViewMyWords: TableView<Word>
- englishWordMyWords: TableColumn<Word, String>
- welshWordMyWords: TableColumn<Word, String>
- wordTypeMyWords: TableColumn<Word, String>
- colBtnMyWords: TableColumn<Word, Void>
- searchFieldMyWords: TextField
- myWords: ObervableList<Word>
+ goMenu(): void
- searchForWords(): void
+ goPractice(): void
- addButtons(): void
+ Exit(): void
- getWords(): ObervableList<Word>
+ intialize(URL, ResourceBundle): void
UI
+ start(primaryStage): void
+ showMenu(): void
+ showLearn(): void
+ showDictionary(): void
+ showPractice(): void
+ backToMenu(): void
+ showOptions(): void
+ showMyWords(): void
OptionsController
+ intialize(URL, ResourceBundle): void
PracticeController
+ goFlashcards(): void
+ goMenu(): void
+ goGuessWord: void
+ Exit(): void
+ goBack(): void
AddWordController
- englishWordTextField: TextField
- welshWordTextField: TextField
- wordTypeTextField: TextField
+ newWordButtonPushed(): void
+ intialize(URL, ResourceBundle): void
DictionaryController
+ tableViewDictionary: TableView<Word>
+ englishWord: TableColumn<Word, String>
+ welshWord: TableColumn<Word, String>
+ wordType: TableColumn<Word, String>
+ colBtn: TableColumn<Word, Void>
- searchField: TextField
+ words: ObervableList<Word>
+ myWords: ObservableList<Word>
+ goMenu(): void
- searchForWords(): void
- addWordsWindow(): void
- addButtons(): void
+ Exit(): void
- getWords(): ObervableList<Word>
+ intialize(URL, ResourceBundle): void
FlashcardsController
+ goMenu(): void
+ Exit(): void
+ goBack(): void
GuessWordController
+ goMenu(): void
+ Exit(): void
+ goBack(): void
Word
- englishWord: String
- welshWord: String
- wordType: String
+ Word(String english, String welsh, String type)
+ Word()
~ getEnglishWord(Void): String
~ getWelshWord(Void): String
~ getWordType(Void): String
+ load(Iterator<JSONObject> iterator): Void
+ Save(JSONObject object, String welsh, String wordType, String english): Void
1..1
1..*

REFERENCES

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

[2] QA Document SE.QA.03 - Project Management Standards.

[3] QA Document SE.QA.02 - General Documentation Standards.

[4] QA Document SE.QA.09 - Java Coding Standards.

DOCUMENT HISTORY

Version	*CCF No.*	*Date*	*Changes made to document*	*Changed by*
1.0	N/A	25/02/20	Document Created	Rik7