Software Engineering Group Projects – Design Specification Standards

Author:	Rik7, jub27, bas17 & bas21
Config Ref:	SE_G17_DESIGN_DOC
Date:	25th February 2020
Version:	1.0
Status:	Release

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 8 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 executes all backend communications. This class should also handle how any core backend features interacts with other backend features.

3. 2.2.3 Dictionary

Class	**Dictionary**
Purpose	This class is responsible for adding, saving and loading words from json file.

4. 2.2.4 PracticeGames

Class	**PracticeGames**
Purpose	PracticeGames class executes all the playable games.

5. 2.2.5 MyWords

Class	**MyWords**
Purpose	This classes handles all the saved words by the user. This class inherits methods from the dictionary class.

6. 2.2.6 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.

7. 2.2.7 Question

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

8. 2.2.8 UI

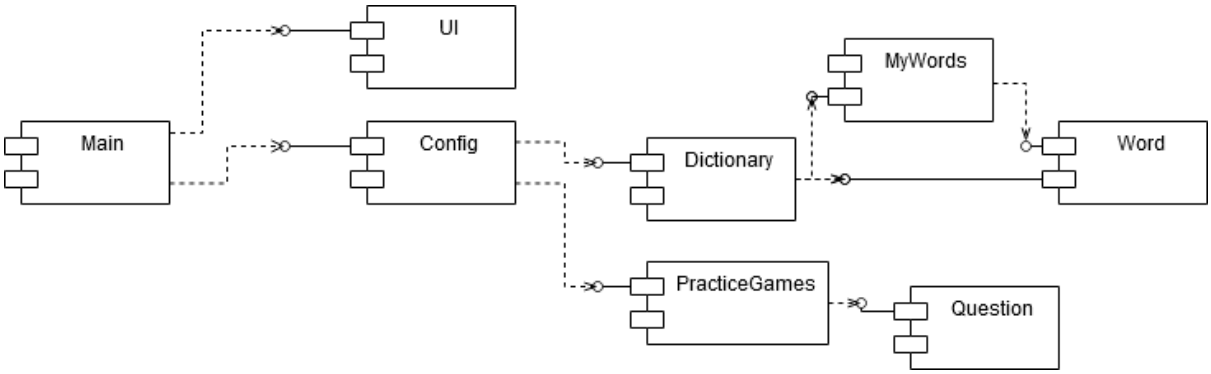
Class	**UI**
Purpose	This class controls all the FXML files. Opening, closing and changing scenes are possible from this class.

2.3 Mapping from requirements to classes

Requirement	*Classes providing requirement*
FR1	Main, UI, Config, Dictionary, Word
FR2	UI, Config, DictionaryController
FR3	UI, Config, DictionaryController
FR4	UI, Config, MyWordsController
FR5	UI, DictionaryController, AddWordController
FR6	UI, Config, DictionaryController, Word
FR7	Config, UI, Dictionary, Word
FR8	Config, UI, PracticeGames
FR9	Config, UI, PracticeGames, FlashCardController, MatchWordController, TranslateWordController, GuessWordController, Question
FR10	Config, UI, PracticeGames,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.

- Public static void main(String[] args) – This method runs the program and calls UI to launch the application.

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. This class also launches the backend.

- 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 method from config class to load word into the application.
- Public static void showDictionary() - A method to load Dictionary.fxml file sets it as a centre part of mainLayout.
- Public static void showPractice() - A method to load Practice.fxml file in the background.
- Public static void backToDictionary() - A method to load Dictionary.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.
- Public static void showMatchWord() – A method to load MatchWords.fxml file in the background.
- Public static void showTranslateWord() - A method to load TranslateWord.fxml file in the background.
- Public void loadWordsToDict() – loads dictionary words from dictionary to UI.
- Public void loadWordsToPrac() – loads practice words from myWords to UI.
- Public static void resetDict(Dictionary d) – this method resets dictionary after a word is added to the dictionary.
- Public static void resetPracList(MyWords myWords) – This method resets myWord list after an addition or deletion of a word from the list.
- Public void save(ObservableList<Word> dictList, ObservableList<Word> pracList) – This method calls save method from dictionary class using configuration.dictionary to save the words to file.

4.3 Config

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

- Public Void runApp() – this method loads words from the json file into the system.

4.4 Dictionary

- Protected Void saveList(String fil String fileName, ObservableList<Word> dictWords) – this method is used to put words from the ObservableList to a json file.
- Protected Void loadDictList(String file) – this method is used to call another method which loads the json file.
- Protected Void loadList(String filename, SortedMap<String, Word> englishMap) - This method loads words from an json file into the sorted map.
- Protected ArrayList<Word> displayWords(SortedMap<String, Word> map) – this method loads words from sortedMap to arrayList and returns the following arrayList.
- Public SortedMap<String, Word> getDictEnglishMap() – Returns the words of dictionary as a sortedMap.

4.5 PracticeGames

- Public void resetDict(Dictionary d) – Resets dictionary every time a new word is added.
- Protected Word getWord() – Randomly returns an index of the word.
- Protected ArrayList<Word> getWords(int amount) – loops a word for till certain amount reached.
- Protected ArrayList<Word> jumbleWordsShuffleWords(ArrayList<Word> words, int amount) – method used to shuffle words and return them in random order.
- Protected String jumbleWordsMark(ArrayList<Word> words, Word word1, Word word2, Word word3, Word word4)– this method calculates the score and returns it.

4.6 Word

This class is used for storing information for each word.

- Word(String English, String welsh, String type) – this creates a word object.
- Public String getEnglish() – this method gets the English translation of the word.
- Public String getWelsh() – this method gets the Welsh translation of the word.
- Public 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.

4.7 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.

4.8 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 goDictionary() – This method is used to go to Dictionary screen.
- Public void Exit() - A method that terminates the program.
- Public void goBack() – Changes screen to the previous screen.
- Public void goMatchWord() – Changes screen to the MatchWord.
- Public void goTranslateWord() – Changes screen to the TranslateWord.

4.9 FlashcardsController

- Public void goMenu() – This method changes screen to dictionary screen.
- Public void Exit() - A method that terminates the program.
- Public void goBack() – Changes screen to the previous screen.
- Public void newFlashCard() – this method puts a new word on the screen.
- Public void clickFlashCard() – this method changes the language of the word on the flash card.
- Public void setNextWord() – this method gets a new word for the flash card.
- Public void updateProgressBar() – this method updates the progress bar

4.10 GuessWordController

- Public void goMenu() – This method changes screen to dictionary screen.
- Public void Exit() - A method that terminates the program.
- Public void goBack() – Changes screen to the previous screen.
- Public void correctAnswer() – this method outputs the users result and displays the correct answer.
- Public void newQuestion() – this method generates a new question and puts it on screen.
- Public void button1Pressed() – this method calls the mark function with the value in the pressed button.
- Public void button2Pressed() – this method calls the mark function with the value in the pressed button.
- Public void button3Pressed() – this method calls the mark function with the value in the pressed button.
- Public void button4Pressed() – this method calls the mark function with the value in the pressed button.
- Public void button5Pressed() – this method calls the mark function with the value in the pressed button.

- Public void button6Pressed() – this method calls the mark function with the value in the pressed button.
- Public void mark(String answer) – checks if the answer input is correct.

4.11 MyWords Controller

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

- Public void goDictionary - A method that changes scene to Dictionary.fxml.
- 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.
- Private void addButtons() - A method that adds a "Remove" button to each cell in a table column.

4.12 DictionaryController

- Public void goMyWords() – Takes user to the practice list.
- 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 "Save" button to each cell in a table column.
- Private void searchForWords() - A method that searches, sorts each character and returns the word in the search list.

4.13 AddWordController

- Public void newWordButtonPushed – adds word into the dictionary.

4.14 TranslateWordController

- Public void goMenu() – This method changes screen to dictionary screen.
- Public void Exit() - A method that terminates the program.
- Public void goBack() – Changes screen to the previous screen.
- Public void pass() – this method gets the word from the dictionary.
- Public void mark() – this method checks the answer of the user.
- Public void updateProgressBar() – this method updates the score based on the answers.

4.15 MatchWordController

There are 8 methods which are color coded. This is done to help calculate the answer.

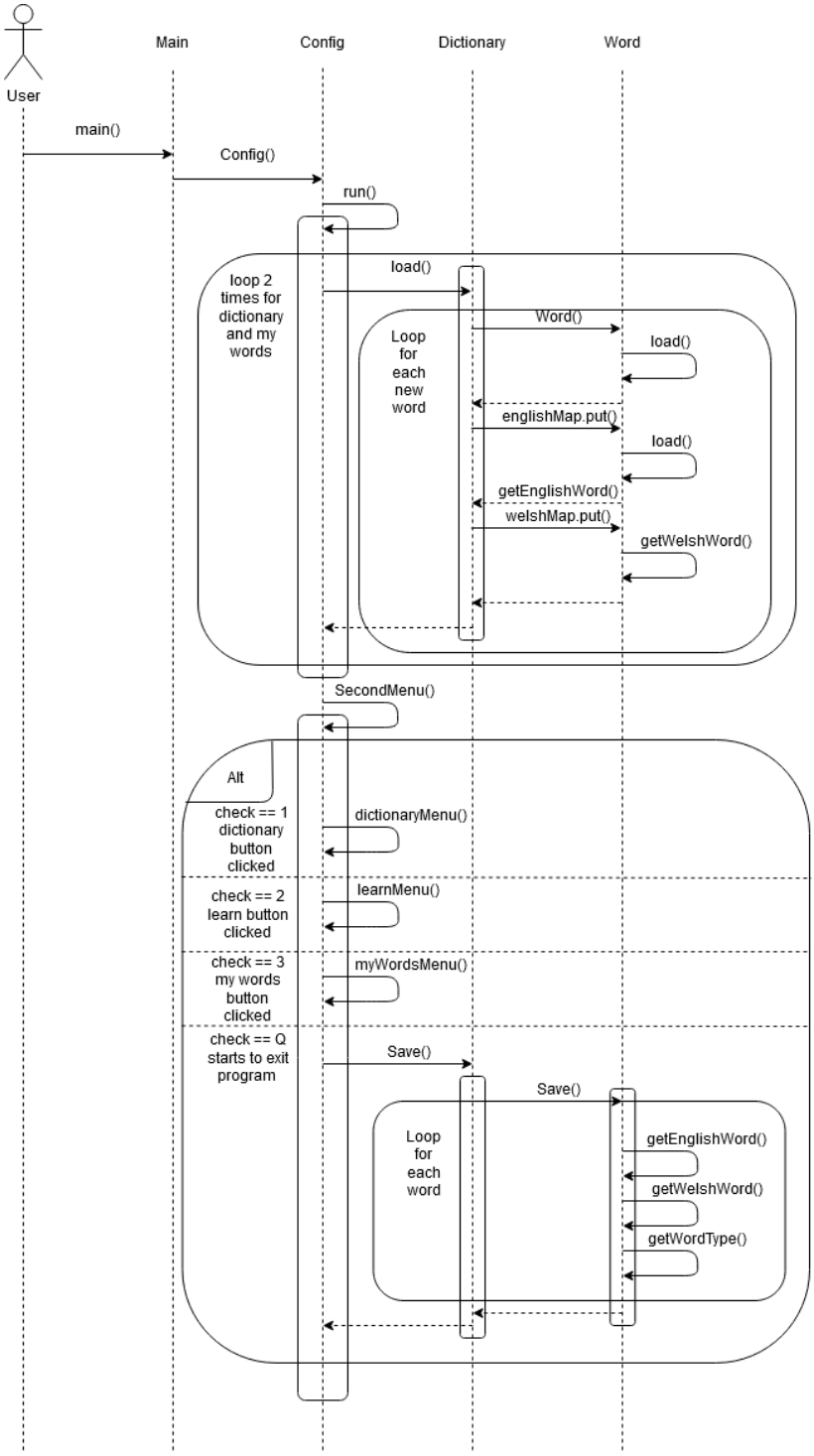
- Public void clickEnglishButton1() – sets color of 1 English word to Blue.
- Public void clickEnglishButton2() – sets color 1 English word to Red.
- Public void clickEnglishButton3() – sets color1 English word to Green.
- Public void clickEnglishButton4() – sets color 1 English word to Orange.
- Public void clickWelshButton1() – sets color 1 Welsh to Blue.
- Public void clickWelshButton2() – sets color 1 Welsh to Red.
- Public void clickWelshButton3() – sets color 1 Welsh to Green.
- Public void clickWelshButton4() – sets color 1 Welsh to Orange.
- Public void goMenu() – This method changes screen to dictionary screen.
- Public void Exit() - A method that terminates the program.
- Public void goBack() – Changes screen to the previous screen.
- Public void mark() – this method checks the answer.

-

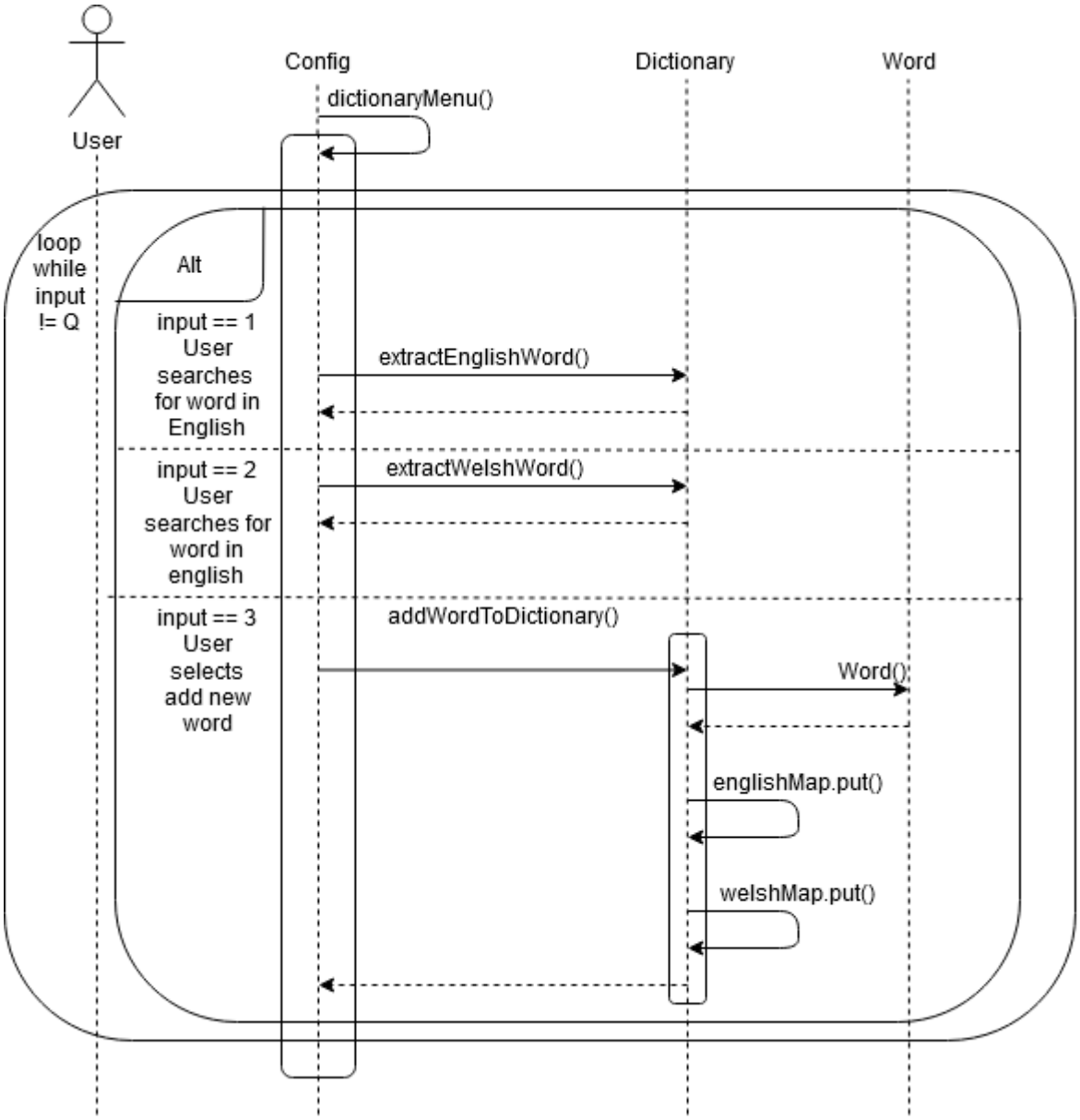
5 DETAILED DESIGN

5.1 Sequence diagram

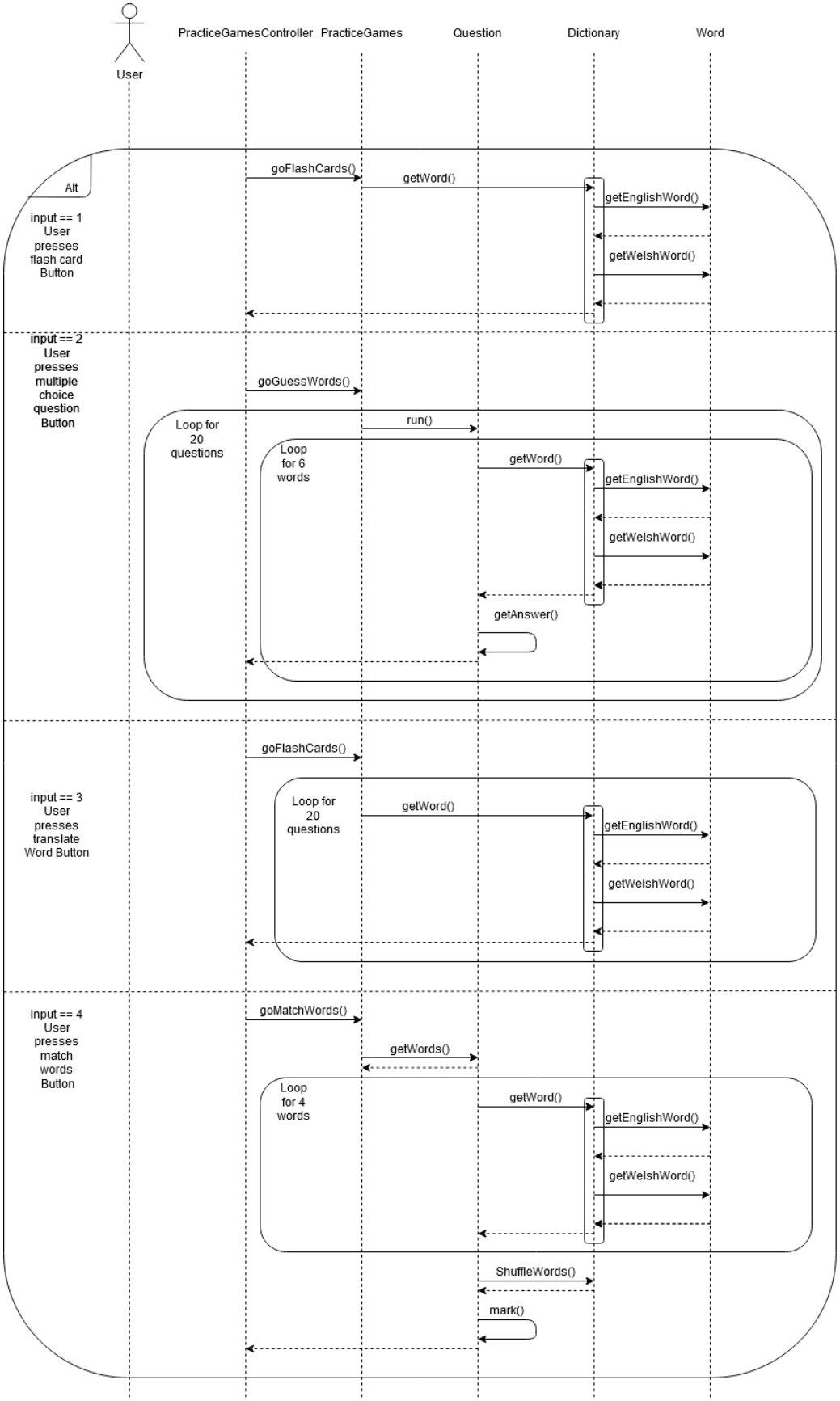
1. Start, load, and save



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. Translate

When the user uses the translate practice tool, they will be given a word. They will then be expected to write the translation of the word into a text field.

This goes on for 20 words after which, the program will tell them how many they translated correctly.

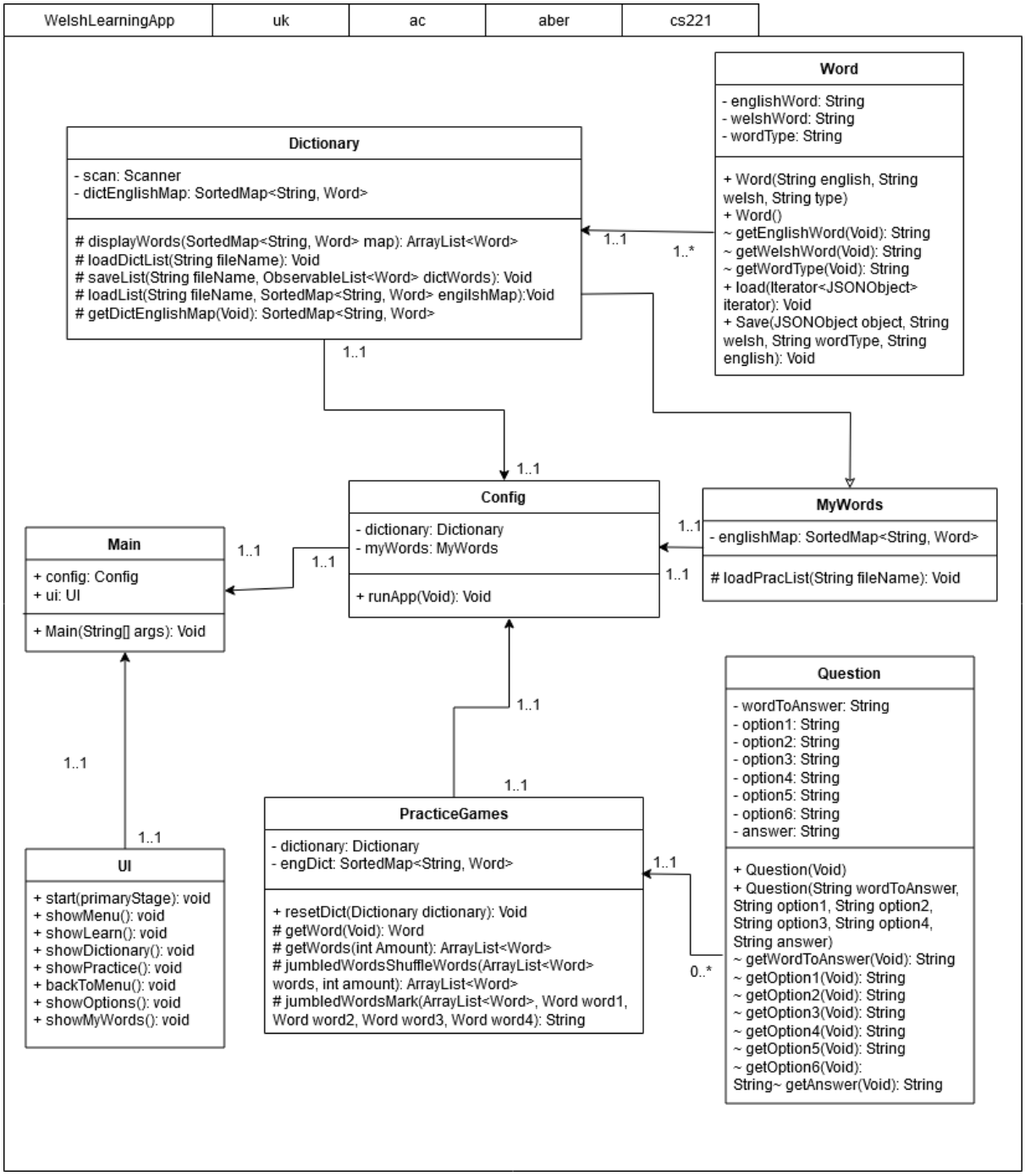
6. MatchWords

When the user uses the match words practice tool, they will be given 8 words. 4 will be in English and the other 4 will be the words translated into welsh. The user is then expected to match the words to their translations.

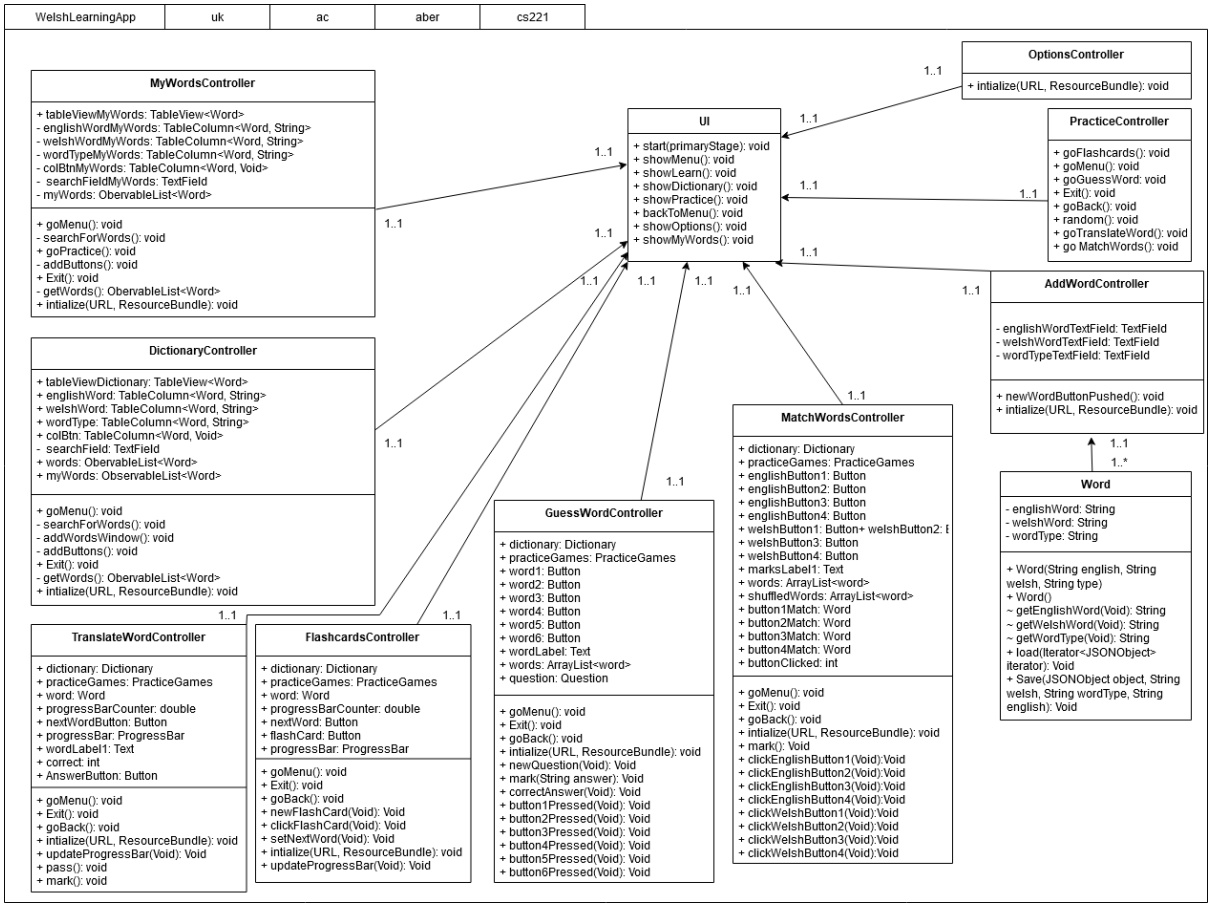
They will then click the submit answer button. This will show them the correct matchings as well as tell them how many they got correct

5.3 Significant Data Structures

1. Application Logic



2. Application Interface



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
1.1	6	05/05/20	Updated diagrams and methods for classes	Rik7, Jub27