|  |
| --- |
| department of computer science |
| RFA: Final Report |
| Capstone Project |
| prepared by |
| **Lauren Antrobus**  *antlau001@myuct.ac.za*  **Merada Richter**  *rchmer002@myuct.ac.za*  for  **Sonia Berman**  **(p.p. help2read)**  on |
| **18 September 2013** |

|  |
| --- |
|  |

Table of Contents

[2. Executive Summary 2](#_Toc367267033)

[3. Introduction 2](#_Toc367267034)

[4. Requirements Captured 2](#_Toc367267035)

[5. Design Overview 2](#_Toc367267036)

[6. Implementation 2](#_Toc367267037)

[7. Program Validation and Verification 2](#_Toc367267038)

[8. Conclusion 2](#_Toc367267039)

[9. User Manual 2](#_Toc367267040)

# Executive Summary

The software developed by this project is a tool directed at young children whose English reading and comprehension skills are not strong. *Flua*, the name of the software, aims to encourage and improve children’s reading fluency through the use of a gamified environment. Two games are available to users (designed to be played by a child paired with a volunteer) as well as a dictionary function to aid in broadening vocabulary.

**[not sure if this should be longer? But didn’t want to go into too much detail and end up repeating things later]**

# Introduction

This software was designed specifically with the needs of the Help2Read organisation in mind. Learners are mostly primary school children from disadvantaged backgrounds with little reading experience, and often for whom English is not a first language.

Because the users are typically disadvantaged children, it is likely that they are not experienced with using a computer. For this reason, it was highly important that the user interface is intuitive and easy to use.

Another important aspect that we focused on is user experience; the software needed to be fun to use, so that the children’s interest is kept. For this reason we chose to gamify the activities, as well as make use of a fun and attention-grabbing graphical user interface (GUI).

Currently, Help2Read uses a paired reading program that teams each child with an adult volunteer. The children learn through reading books, playing games and doing comprehension activities under the guidance of the volunteer. In keeping with this model, we designed our games to be played by a child and volunteer team – however the interaction is focused on the child and the volunteer’s role is largely supervisory.

In our approach to designing our software, we used a mixture between traditional and agile software engineering methods. Our planning and design processes followed traditional methods as the whole system was planned and designed before implementation started and we made use of an evolutionary prototype. During implementation, however, we used the agile notion of iterative development, as we developed the system in increments and adapted to difficulties by modifying the design.

# Requirements Captured

**[not so sure about this section – see what you think]**

In terms of Help2Read’s existing setup – the organisation has a few basic computers available to the children. These computers are not connected to the Internet, nor are they even networked with each other. Thus, we needed to design standalone software that would be able to run on each individual computer and was not dependent on external resources.

Non-functional requirements included that the children would enjoy using the software, so that they would be encouraged to continue using it (thereby developing their reading fluency).

**(that the activities develop reading capabilities?)**

Usability requirements were an important consideration during the design process, as it was important that the children were able to use the software in spite of possibly being unfamiliar with computers. It was therefore vital to the success of the project that the final program is intuitive to use and easy to pick up.

**Use cases (wasn’t sure whether these had changed, so I didn’t copy them)**

**Overall description of classes**

# Design Overview

Design class diagram

Overall architecture of the system, including architecture diagram

Subsections: algorithms & data organisation used, why they are best

# Implementation

**(I’ve filled in this section for the dictionary, but I’m not 100% sure on how your code was implemented … Happy to fix it up though, if you can just explain it.)**

Dictionary:

We had initially hoped to make use of a MySQL database for the dictionary data storage, but were unable to find one suited to our needs. Considering the context of Flua’s users and their weak literacy skills, the existing databases that we found were unnecessarily extensive and contained overly complex words and definitions.

Our next option was making use of dictionary data from a file, which also proved to be difficult. Many of the sources we found contained only words (without definitions) or too contained words and definitions of an inappropriate level for the children’s reading abilities.

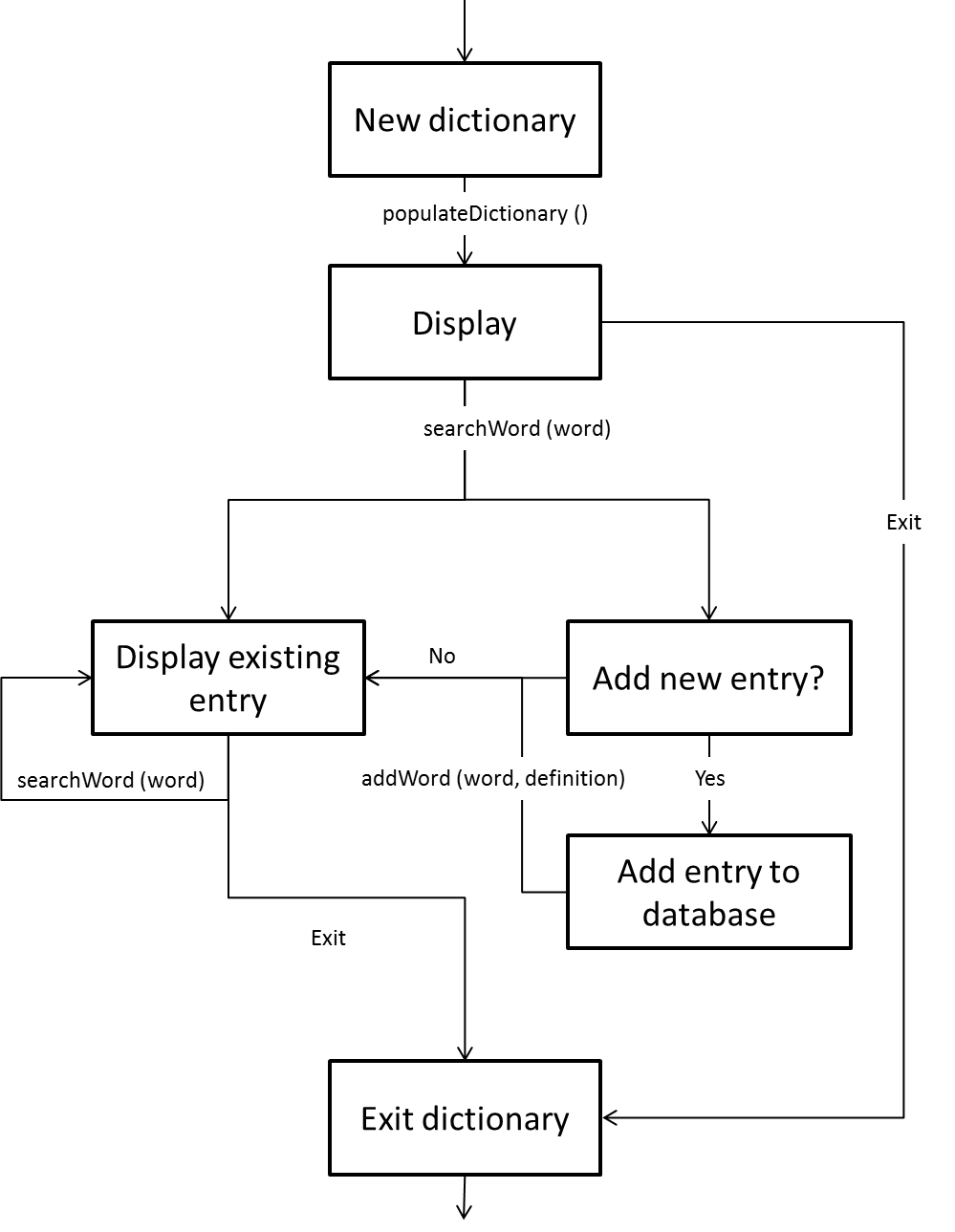
Ultimately, we created our own text file for the dictionary data – building on a suggested vocabulary list from ReadingRockets.org **(reference at bottom)** and filling their definitions from the dictionary definition generator EasyDefine.com **(reference at bottom)**. Once we had data to load, we used a hash table to store the dictionary elements for a few reasons. By choosing this data structure, searching for words becomes easy and efficient. It also lent itself well to a dictionary application, as words can be used for the key and their definitions as the corresponding value.

* Describe how UI was implemented

Flua’s user interface was implemented using a graphic user interface (GUI) in order to make it easy and intuitive to use. This allowed us to simplify user input by making use of GUI widgets such as buttons, text fields and frames. It also meant we could make use of colours and images, making Flua appealing to the children using it.

* Discuss function of most significant methods in each class (flowcharts, sequence diagrams)
* Special programming techniques or libs used

**(I made this flow chart for the dictionary … I think it’s pretty intuitive, so not sure if we should include it, but just in case)**



# Program Validation and Verification

Tell us how you tested the system and why you believe it works. Describe all the steps taken to validate the correctness of the program.

If you had user tests then say what you did and what the results were. Describe why these test data were chosen (what test conditions the data was testing). Table 1 provides an example of the sorts of results we are looking for. The full detail of the test runs should be appended to the report.

*Table 1: Tests performed to validate FLUA*

|  |  |  |  |
| --- | --- | --- | --- |
| Data Set and reason for its choice | Test Cases | | |
| Normal Functioning | Extreme boundary cases | Invalid Data (program should not crash) |
| Preliminary test (see Appendix 3) | Passed | n/a | Fell over |
|  |  |  |  |
|  |  |  |  |

|  |  |
| --- | --- |
| **Test case** | **Inputs, expected behaviour and outcomes** |
| *User input test cases* | |
| Fill in a word:  Input a text in textbox | Users should be able to complete the sentence by inputting text into textboxes via the keyboard. The entered text should appear on the screen, in the selected textbox. |
| Create a comprehension: Input question into textbox | Users should be able to enter their own comprehension questions into textboxes via the keyboard. The entered text should appear on the screen, in the selected textbox. |
| Create a comprehension:  Reading a story | Stories will be displayed in a “book” on the screen. Users should be able to click on arrows to turn pages forwards to backwards. |
| *Database test cases* | |
| Loading story and/or images | The appropriate story and/or images should be loaded from the database, depending on which activity and which story has been selected. These should then be correctly displayed to the screen. |
| User log in | User log in details should be checked against those stored in the database – resulting in a successful or unsuccessful log in. If the user does not exist in the database, a new record should be created. |
| Dictionary lookup | Users should be able to search for a chosen word. The system should search for the specified word. If the search is successful, the corresponding word details should be displayed on the screen. If it is unsuccessful, the user should be informed of this. |
| *GUI test cases* | |
| Frame sequencing | Different GUI frames should link to each other as specified in planning. When leaving one GUI frame, it should dispose of itself as well as display the following GUI frame correctly. |
| Button response | Buttons should respond to user stimulation intuitively and as planned. The outcomes of pressing a button will depend on the button – but may include actions such as updating the database, displaying another GUI frame, checking log in details etc. |

# Conclusion

What worked, what didn’t work

Recommendations for future use

# User Manual

How to use the system

# References

Graham, S., Harris, K.R. and Loynachan, C. 1993. *The Basic Spelling Vocabulary List*. Available from: http://www.readingrockets.org/article/22366/. [2013, September 20]

Jain, P. & Choi, C. 2009. *EasyDefine*. Available from: http://www.easydefine.com/ [2013, September 20]