|  |
| --- |
| 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 |
| **22 September 2013** |

|  |
| --- |
|  |

# Table of Contents

[1. Table of Contents 1](#_Toc367636201)

[2. Executive Summary 2](#_Toc367636202)

[3. Introduction 2](#_Toc367636203)

[3.1. Statement of Scope 2](#_Toc367636204)

[4. Requirements Captured 3](#_Toc367636205)

[5. Design Overview 5](#_Toc367636206)

[6. Implementation 6](#_Toc367636207)

[7. Program Validation and Verification 7](#_Toc367636208)

[8. Conclusion 8](#_Toc367636209)

[9. User Manual 9](#_Toc367636210)

[10. References 9](#_Toc367636211)

# 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 (implemented using Java). 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.

# Introduction

Flua[[1]](#footnote-1) 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. English is often not a first language for these children.

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.

Because the users of Flua 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).

In our approach to designing our software, we used a mixture of 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.

Statement of Scope

This section summarises the scope of the project, according to the agreement between the client and the development team. Please see ***Table 1*** for details.

*Table 1: Statement of Scope*

|  |  |
| --- | --- |
| Functions | Fill-a-Word game:   * Users are presented with an image of a busy scene, and sentences describing something happening in the picture. These sentences will be missing words, which should be filled in by the user.   Create-a-comprehension game:   * Pupils at Help2read often struggle with words such as “who”, “what” or “how”, which are difficult to sound out and explain. This activity aims to make the pupils more familiar with these kinds of words. Users will be presented with a story, which they can read in a “book” on the GUI. They will then have to create their own comprehension questions about the story. Suggestive words (such as “which” or “who”) will be provided in a word tray on the side of the screen.   Dictionary:   * Users can look up a word they might not understand, and can add a word if it does not form part of the database. This will hopefully help to ensure users’ understanding and develop their vocabulary. |
| Inputs | **Keyboard inputs:**   * Words for sentence completion in the Fill-in-the-word game * Questions for the Create-your-own-comprehension game   **Mouse inputs:**   * Start a game * Select menu/control options * Select words from the word tray * Activate input tray |
| Outputs | **Monitor outputs:**   * Display game resources * Echo user input in input tray * Show dictionary lookup results |
| Constraints | * Due to lack of internet availability, the program must contain all necessary resources as it will be run locally. * The user interface is required to be simple, intuitive and user friendly, as end users are disadvantaged children who are not highly proficient in reading and are unlikely to be very familiar with computers. |

# Requirements Captured

This section details the requirements for the Flua program, as determined by the client. Each requirement is briefly explained before being described in terms of its ***Use cases*** and ***Use case*** ***narratives***.

### Functional Requirements

##### Main Menu

The menu is used to interact with the program via the following options:

* Start a Fill-a-Word game
* Start a Comprehension game
* Exit

It is the main entry point into the graphical user interface.

##### Dictionary

The dictionary function allows learners to look up words they do not understand, and volunteers to add definitions for words which are not in the current database.

##### Resources

Resources for the various games and dictionary functionality are loaded and displayed in the user interface. These resources include:

* Images
* Text (for stories, questions and help)
* Dictionary files

##### User input

Users interact with the interface via keyboard and mouse.

##### Words for word tray

The word tray contains word hints for learners to assist with vocabulary development.

##### Local operation

The computers on which the program will run do not have Internet access and are not networked. This requires Flua to be implemented as standalone software complete with all necessary resources prior shipping.

### Non-Functional Requirements

1. Maintainability

Since there will be little on-site help once the program is released, Flua should not crash or become unusable due to code errors. Should a fault occur in the program, Flua should return to a previously operational state, such as the main menu.

1. Documentation

All aspects of the program should be well recorded and this documentation included with the release to allow developers to quickly comprehend the system should it be extended (please see the next point).

1. Extensibility

There is great potential for Flua and many more functions can be added to the program to make it a better teaching tool. It should be easy to expand the program and add additional functionality without having to rewrite major sections of code.

### Usability Requirements

1. Easy to learn

Since the users of this system are not guaranteed to be familiar with computer programs, the interface should be intuitive and simple to learn. Complex user interactions should be avoided.

1. Enjoyable to use

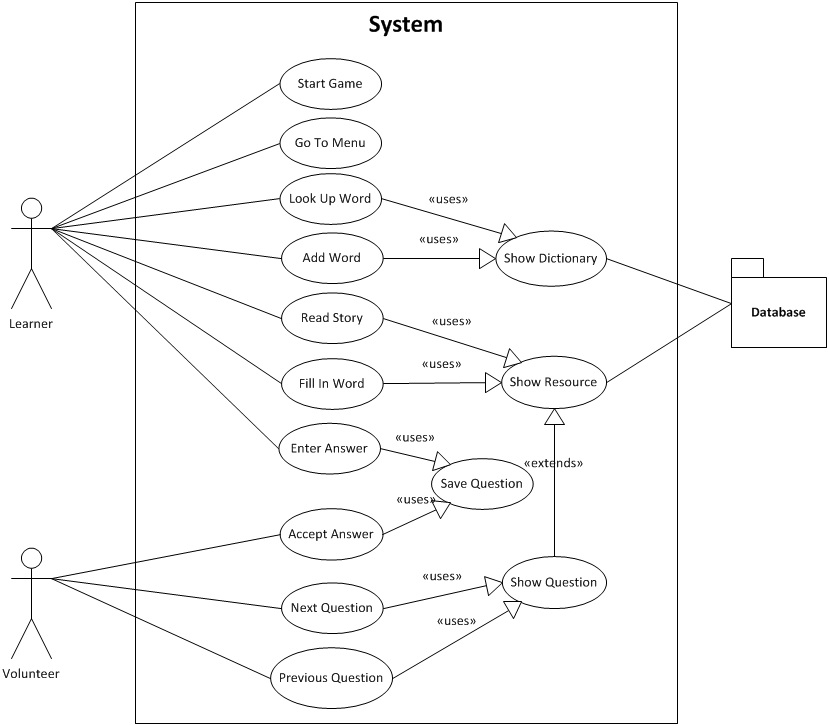
Users should enjoy using Flua so that they are encouraged to continue using it, thereby further developing their reading fluency.

1. Interesting and colourful interface

Bright, bold colours are appealing to young children, and are likely to retain interest for longer. The interface should reflect these traits in order to increase user creativity and receptiveness to learning.

### Use cases

***Figure 1*** details the use cases of the Flua program.



*Figure 1: Use case diagram of Flua*

For the final class implementation of the program, please see ***Figure 2***.

### Use case narratives

1. Start Game
2. Got to Menu
3. Show Dictionary
4. Look Up Word
5. Add Word
6. Show Resource
7. Read Story
8. Fill In Word
9. Enter Answer
10. Accept Answer
11. Save Question
12. Show Question

**Initialize a Session**

***Actors:*** Learner, Volunteer

The Learner and Volunteer enter their usernames. The System looks up previous users. If no match is found, a new user is created. If a match is found, that user’s progress is loaded for the Session.

***Alternative:*** If the Learner is working alone, no Volunteer username is required.

The Learner selects a game (either Fill-in-the-word or Create-your-own-comprehension), and a game Session is started.

**2.2. Create/Delete a User**

***Actor:*** User (either Learner or Volunteer)

[Create] The User enters his/her username. The System searches for the username, and if it does not exist, creates a new User with the specified username and role.

[Delete] The User enters his/her username. The System searches for the username, and removes it.

***Alternative:*** The User can cancel the request to create/delete a User account at any time before the System state is saved.

**2.3. Read a Story**

***Actor:*** Learner, Volunteer

***Precondition:*** Create-your-own-comprehension game chosen

The System displays a story the Learner has not yet read in the resource section. The Learner reads the story, using the mouse to turn pages. The Volunteer assists the learner in reading and sounding out words.

At the end of the story, the Learner is requested to enter questions for the comprehension.

**2.4. Enter a question**

***Actor:*** Learner

***Precondition:*** Create-your-own-comprehension game chosen

The Learner types a question. The Volunteer checks the question. An accepted question is saved. A rejected question remains displayed and the Learner is given the chance to redo the question, using the rejected question as reference.

The Learner’s progress is saved. 4

Once the story is complete, the Learner can either choose another story, or switch to another game.

**2.5. Fill in a word**

***Actor:*** Learner

***Precondition:*** Fill-in-the-word game chosen

The System displays a picture in the resource tray, and an unfinished paragraph in the input tray. The Learner selects an empty word bracket and types in a word for completion.

The Volunteer checks the answer. And accepted answer is saved in black. A rejected answer is displayed in green text and the Learner is given another chance to answer.

The Learner’s progress is saved.

Once the paragraph is complete, the Learner can either choose another story, or switch to another game.

**2.6. Look up a word**

***Actors:*** Learner, Volunteer

***Service:*** Dictionary

The Learner clicks on a word. The System looks up the word in the Dictionary. The word’s definition is displayed in the word tray.

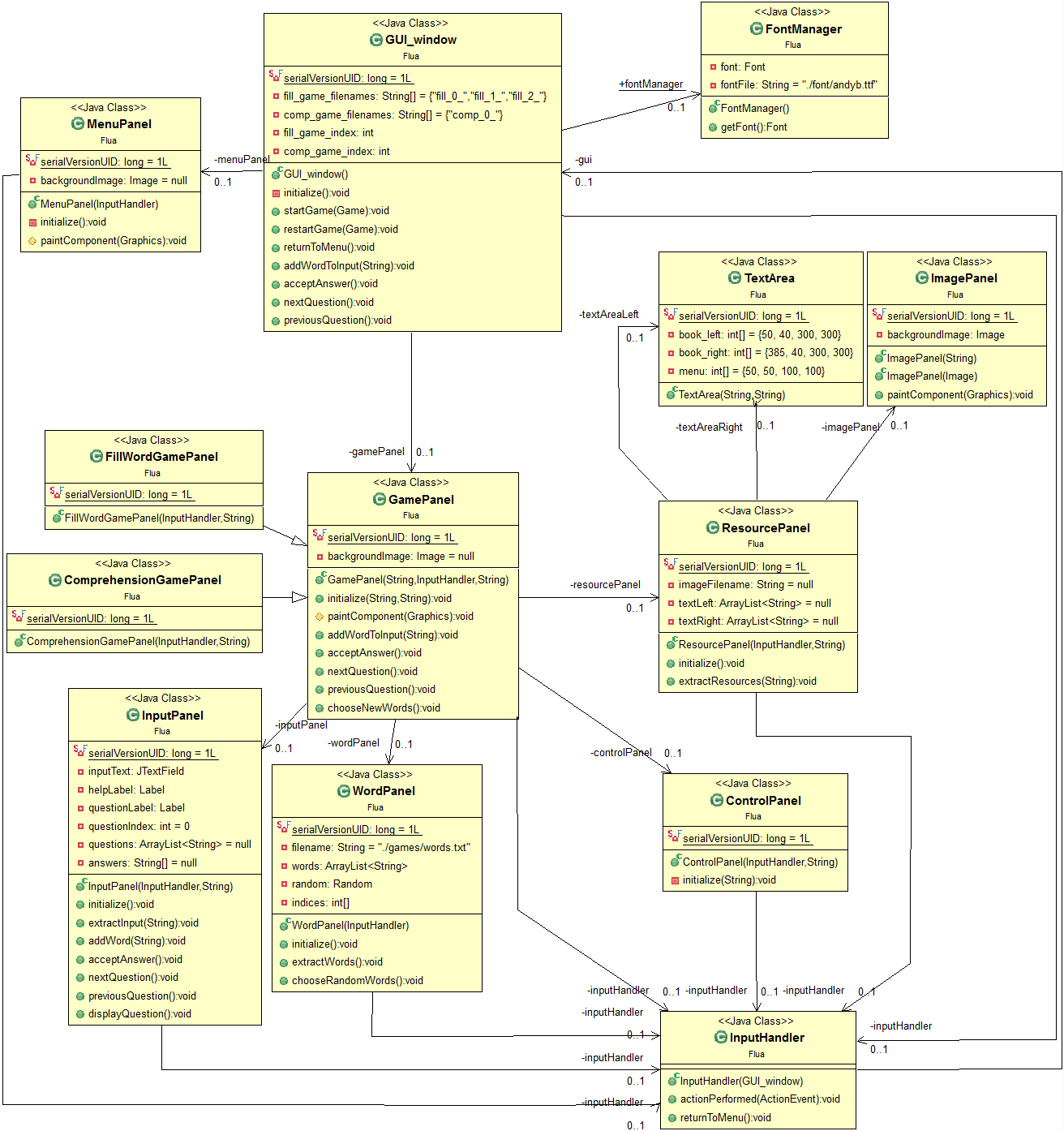
***Alternative:*** If the word is not found, the Volunteer is requested to define it.

# Design Overview

Architecture diagram

[INSERT]

Design class diagram for gui



*Figure 2: Design class diagram for Flua's user interface*

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.

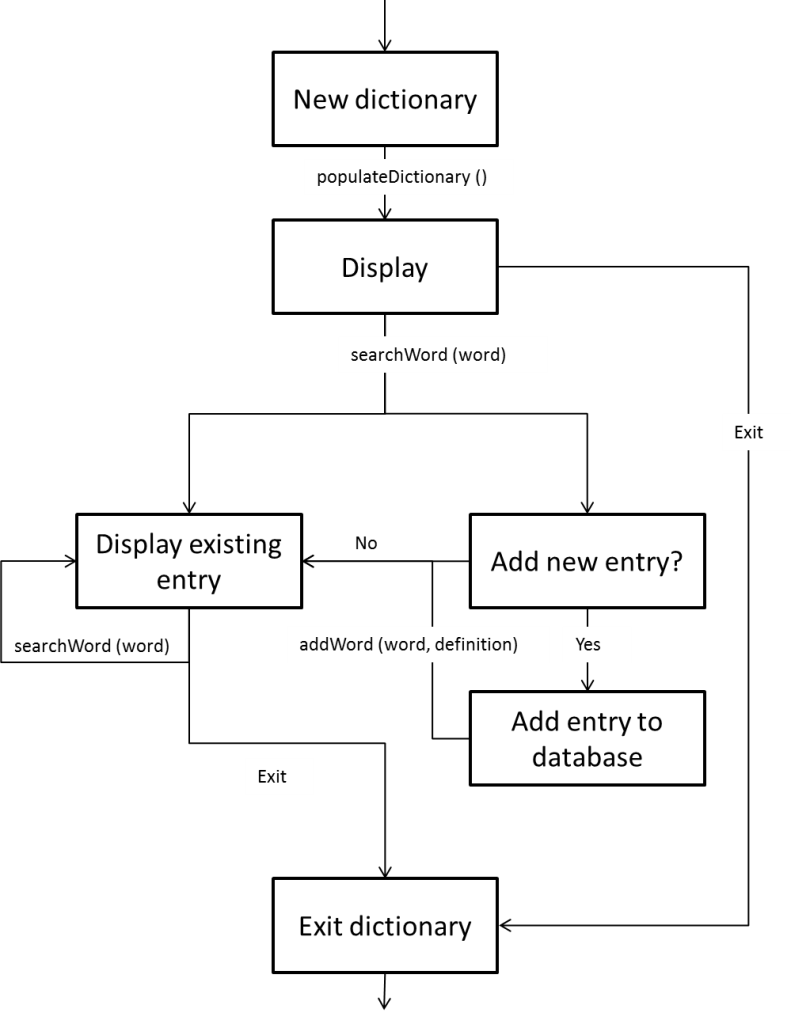
Ultimately, we created our own text file for the dictionary data – building on a suggested vocabulary list from ReadingRockets.org **(reference at bottom)** and filling their definitions from the dictionary definition generator EasyDefine.com **(reference at bottom)**. Once we had data to load, we used 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 2: 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

Very easy to extend to allow the following functionalities:

User log in

Additional games

Save user progress (all questions and answers)

# 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]

1. **FLU**ency **A**id [↑](#footnote-ref-1)