Department of Computer Science & Engineering
The University of Texas at Arlington

Architectural Design Specification
CSE 4316: Senior Design I
Fall 2020

ALS Eye Tracking Group
ALS Eye Tracking APP

Anthony Vardaro
Michael Kosta
Zixiu Su
Thanhthao Le

REVISION HISTORY

Revision	Date	Author(s)	Description
1.0	11.15.2015	AV, MK, KL, ZS	document creation and finish

Contents

List of Figures

List of Tables

1 Introduction

Our product aims to simplify computer operability for physically impaired users. By empowering physically impaired users with the ability to maneuver a computer with characteristic eye movements, we can help them to be able to communicate where it would normally be impossible or difficult. We accomplish this by affixing an eye tracker camera to the users computer, then they will draw their attention to the screen where it will detect input through eye movement. The system must be capable of interpreting gaze data and recording user input. It must also provide a user interface as an overlay containing widgets and buttons that the user interacts with via the eye tracker. The buttons need to adjust in size to accommodate for the word being typed. The core functionality of the product, and primary deliverable, is a system that enables users to emulate keyboard strokes using an eye tracker, allowing them to overcome challenges presented by physical impairment.

2 System Overview

Our product will have three layers to the system, the Eye-Tracking layer, the Menu layer, and the Keyboard layer. Each of these layers will make up the basic processes for our application. The Eye-Tracking layer will be the main layer and the layer that everything communicates through. This layer will have the Eye-Tracker and will take input from the users gaze. The next two layers will all be connected to the Eye-Tracking layer, sending and receiving data through this layer. Finally, the Menu layer will be the drop down bar and extra buttons that are not connected to the keyboard. While this is not entirely necessary, it will help the user navigate and edit the text that appears on the screen. The Keyboard layer is the layer that runs with the keyboard. It will be using different algorithms to help make it easier for the user to navigate the keyboard.

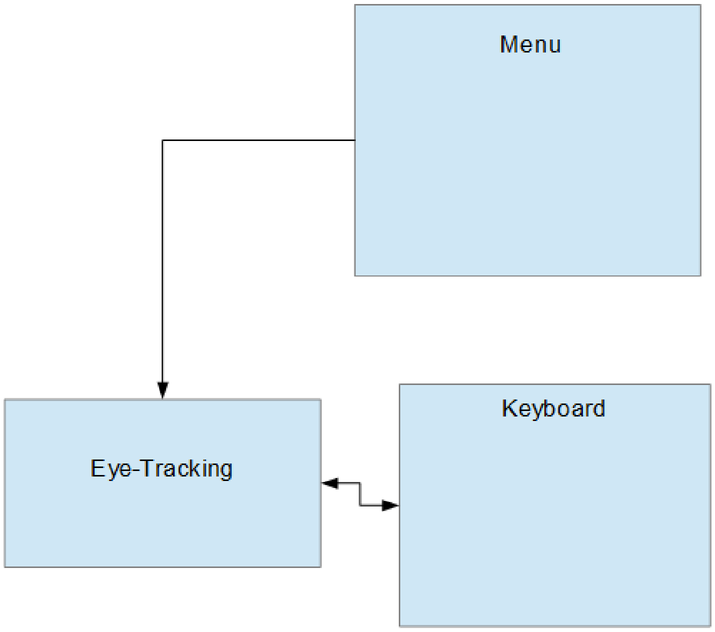

Figure 1: Eye-Tracking architectural layer diagram

2.1 Eye-Tracking Layer Description

The Eye-Tracking Layer will be the main layer of this application. It will be the layer that communicates to all the other layers. This layer will be where the Eye-Tracker comes into play. The Eye-Tracker will be connected to the Personal Computer of the user, and it will be run using the software that comes with the Eye-Tracker. The Eye-Tracker will pick up visual data from the users eyes, the data will be used to decide what letters are being typed. Basically, the user will connect the Eye-Tracker to their PC, they will open our application and login, or they will already be logged in if they have already made an account, they will then be able to look at the screen and type letters. The application will also know to recalibrate after a certain amount of time passes and the Eye-Tracker has not gone over a button. The amount of time will be changeable.

2.2 Keyboard Layer Description

The Keyboard Layer will be the layer that contains the keyboard and all the keyboard functions. When the user logs in to the application, they will see a keyboard set up like the one on regular computers.

The keyboard will be designed so that the user can select a letter by either staring for a period of time at a letter, or by blinking. When a letter is selected it will be displayed in the text box above the keyboard. Also, when a letter is selected, certain keys on the keyboard will either get bigger or smaller, depending on what the next predicted letters would be. The predictive text algorithm will be running in the background, to determine what letters sizes should change. There will also be a history that will display a list of words that have been previously built by the user.

2.3 Menu Layer Description

The Menu Layer will be the layer that has all the drop bar and top of screen, menu options. These options will be for the users convenience. The basic options are, first selecting your user account info. This is useful in case you need to change accounts or change account information. Second, we have a save function, that allows the user to save a word, sentence, or phrase that they commonly use. Third, we have a load function, that allows the user to load a saved word, sentence, or phrase. Finally, we have a close function, that can close the user out of the application or close the application itself.

3 Subsystem Definitions & Data Flow

For the Eye-Tracker Layer, the two subsystems are Calibration and Interface. The Interface is the eye-tracking interface, allowing the user to move the cursor with their eyes and click on things. The Interface is connected to Calibration and Text. When the cursor is idle for a time, then the interface leads to calibration and the eye-tracker recalibrates. For the Menu Layer, there are five subsystems, Save, Load, User, Close, and Interface. Save lets you save the text box, load lets you load a saved text box, user lets you edit user info and change users, and close closes the user or the application. These all lead to the interface, which leads to the Eye-Tracking Layers Interface. The Keyboard Layer has four subsystems, History, Predictive Test, Letters, and Text. History has a history of all the words and words typed by the user, Predictive Text is an algorithm that runs during letter typing, and Letters are on the keyboard and are the buttons that are pressed to build the word. These all lead to Text, which displays the text in a text box and leads to the Eye-Tracking Layers Interface.

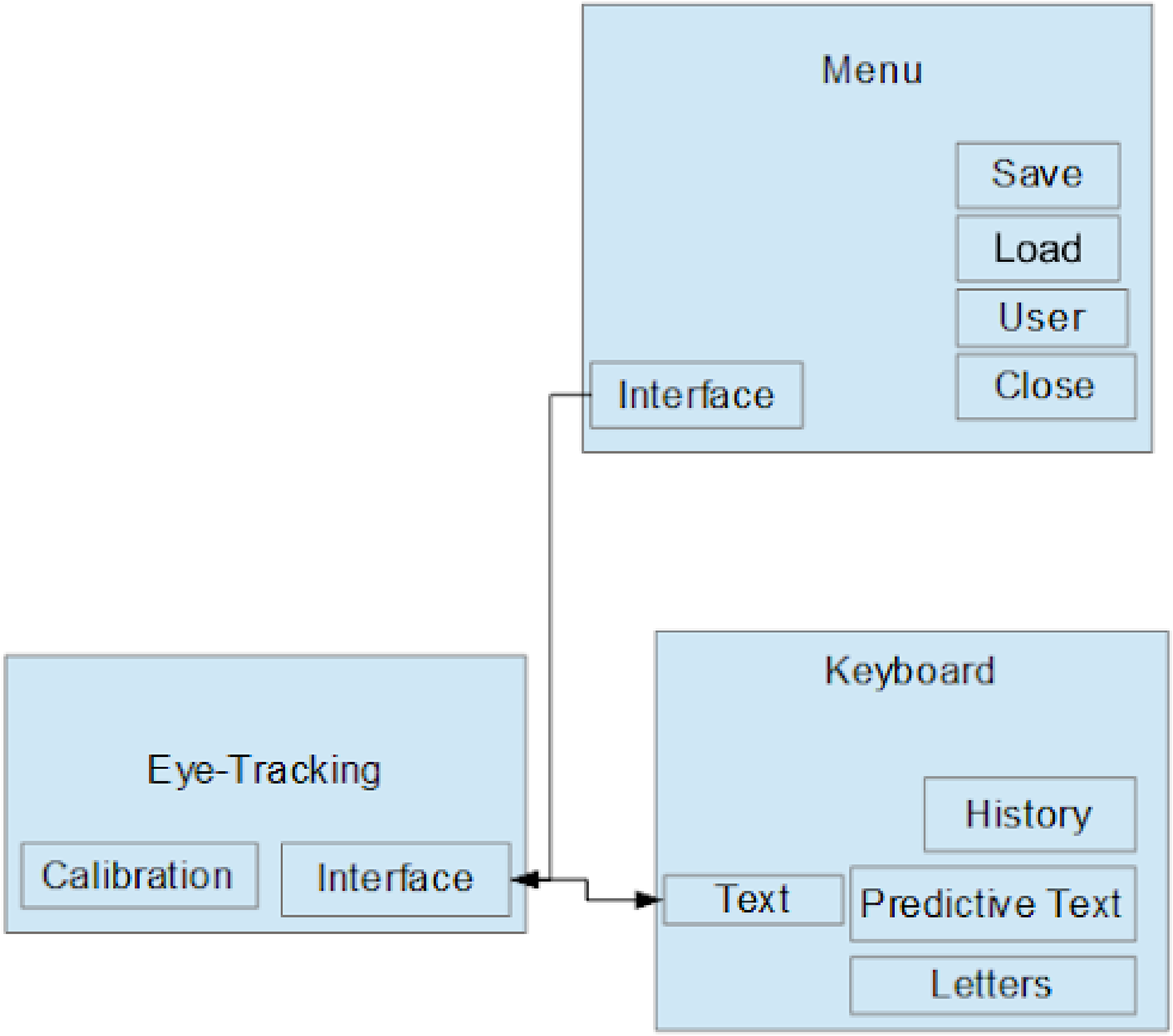

Figure 2: Eye-Tracking data flow diagram

4 Eye-Tracking Subsystems

At a high level, The eye tracking subsystem is composed of two core components: the interface and calibration modules. The interface subsystem is an outward-facing API that other modules within the app can subscribe to to collect stream data in real-time. This provides a clean gateway through which the rest of the app can listen, parse, and interact with the users gaze input. The eye tracker is also responsible for autonomously recalibrating itself to ensure high quality input is captured when the user is using the device. The Eye Tracking subsystems is tightly packaged as an independent module built solely for interacting with Tobii Eye Trackers, and is imported to the rest of the app for usage.

4.1 Interface Subsystem

The Interface subsystem is the main function of the Eye-Tracker, it is where the gaze data is collected. The Eye-Tracker connects to the users PC and is set up, then the interface starts to work, even without our application. Taking in the data from the users eye and showing the cursor where the user is looking.

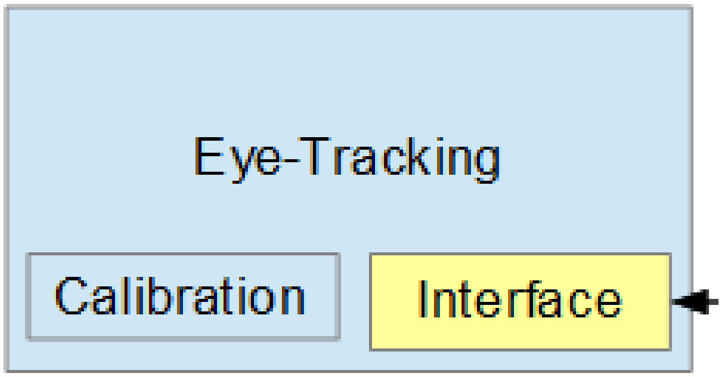

Figure 3: Eye-Tracking Interface subsystem description diagram

4.1.1 Assumptions

The interface provides a clean API that can be used by other systems. Additionally, the interface must be able to extract gaze data quickly and stream it to listeners accurately, which is contingent on the quality of the Eye Tracker device itself.

4.1.2 Responsibilities

The interface helps the user interact with the application and its components, such as creating texts and interacting with the users profile. The interface must be independent and isolated from the rest of the app. This scopes the responsibility of handling gaze input to a single module, and mitigates risk of modules crossing over and making the system overly complex.

4.1.3 Subsystem Interfaces

Table 2: Eye-Tracking Interface Subsystem interfaces

ID	Description	Inputs	Outputs
Predictive Text Subsystem	Eye tracker emits data to listener	A callback function from the user of the Eye Tracker module to execute when new gaze data is captured	Stream of gaze information
Menu Subsystem	Eye tracker emits data to listener	A callback function from the user of the Eye Tracker module to execute when new gaze data is captured	Stream of gaze information

4.2 Calibration Subsystem

This subsystem is used for calibrating the Eye-Tracker when the user gets desynced. When the user is desynced or their eyes are too tired, and they are no longer looking at the buttons on the screen, the system will recalibrate the eye-tracking.

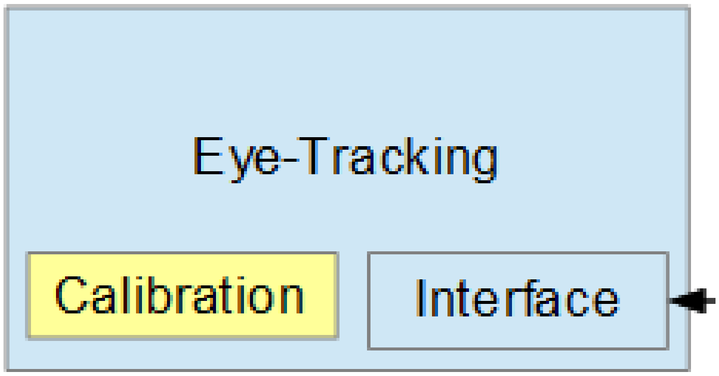

Figure 4: Calibration subsystem description diagram

4.2.1 Assumptions

If the user is not looking at any button or the textbox, or if the user is not looking at the screen, the system is going to start a time at the length of time the user sets. Something like 5-10 seconds. After the time goes off, the application will prompt something to ask the user to resync themselves to the eye-tracker.

4.2.2 Responsibilities

The Calibration helps the eye-tracker connect to the program whenever the eye tracker is not connected to the software or connectivity issues have occurred.

4.2.3 Subsystem Interfaces

Table 3: Calibration Subsystem interfaces

ID	Description	Inputs	Outputs
Eye-Tracking Interface	The Calibration emits data to the application interface for recalibration	A variable for the time saying that it reached zero	A new screen window asking the user to recalibrate their eye-tracker, and will remain until recalibrated

5 Keyboard Subsystems

The keyboard subsystems consist of letters or symbols displayed for the user to interact in order to create words or phrases. The words can appear by selecting letters from the keyboard, predictive texts, where the words appear as common use words, or the users past used words.

5.1 Predictive Text Subsystem

Predictive Text is a feature that helps the user create words that were commonly used by other users. This feature uses a database that collects commonly used words from text.

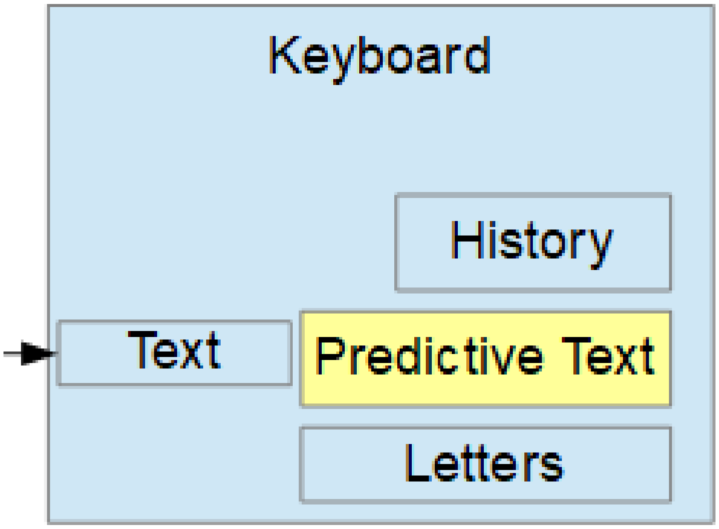

Figure 5: Predictive Text subsystem description diagram

5.1.1 Assumptions

Words are collected from a database that is commonly used from different users. The words are later displayed next to the text box to provide suggestions for the user to select the word instead of letting the user select each individual letter.

5.1.2 Responsibilities

The predictive text creates a completed word based on the current users selection of letters. This word is created based on the common word used or how frequently the word is used by the user.

5.1.3 Subsystem Interfaces

Table 4: Predictive Text Subsystem interfaces

ID	Description	Inputs	Outputs
Predictive Text	Displays Words that is commonly typed based on the sequence of letters by the user	Users eyes location from the eye tracker	Words Displayed

5.2 Letters Subsystem

The Letters on the keyboard. They change sizes depending on what letters should come next in the word.

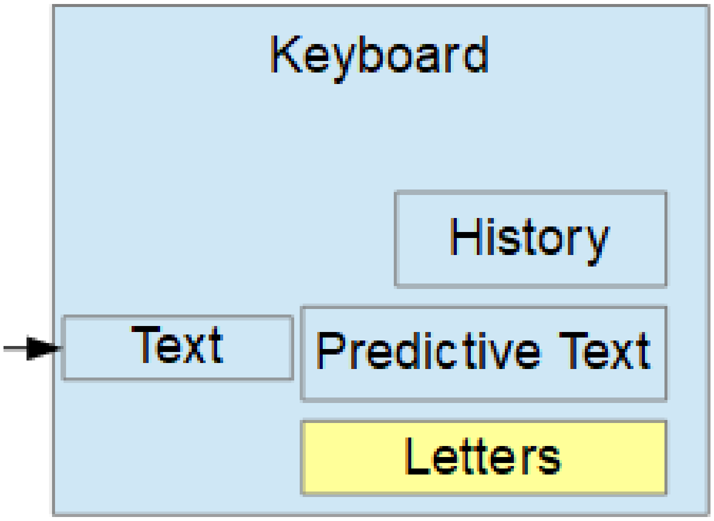

Figure 6: Letters subsystem description diagram

5.2.1 Assumptions

There are letters displayed in a regular keyboard format, QWERTY keyboard. This format makes the sequence familiar for the user to recognize.

5.2.2 Responsibilities

The letters are displayed for the user to interact with the application. The letter is selected by the location of where the user is looking at. This feature helps the user create words.

5.2.3 Subsystem Interfaces

Table 5: Letters Subsystem interfaces

ID	Description	Inputs	Outputs
Letters	Displays letters selected by the user using the keyboard interface	Users eye location based on the eye tracker	Letters Displayed in Text Box

5.3 History Subsystem

Subsystem that stores the history of the words that the user has typed into the text box.

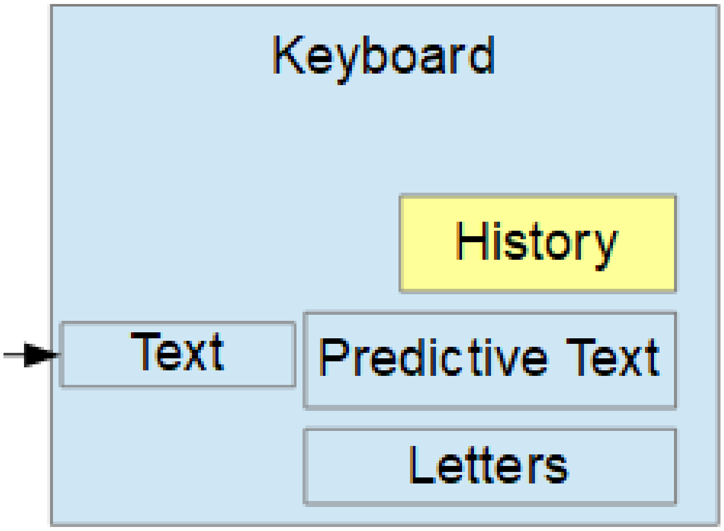

Figure 7: History subsystem description diagram

5.3.1 Assumptions

The users activity has been stored as a file in the machine, which contains a list of words that the user created.

5.3.2 Responsibilities

The history function displays a list of words that were previously created by the user. The list of words can be selected by the user, so that the user would not consume as much time as creating a new word in the text box.

5.3.3 Subsystem Interfaces

Table 6: History Subsystem interfaces

ID	Description	Inputs	Outputs
History	Displays a list of words that were previously created by the user	Eye-Tracker pointed at the History box	List of Words next to the Text box

5.4 Text Subsystem

The Text Subsystem has a text box that displays letters or words by the user. The user may choose what to do with the word in the text box.

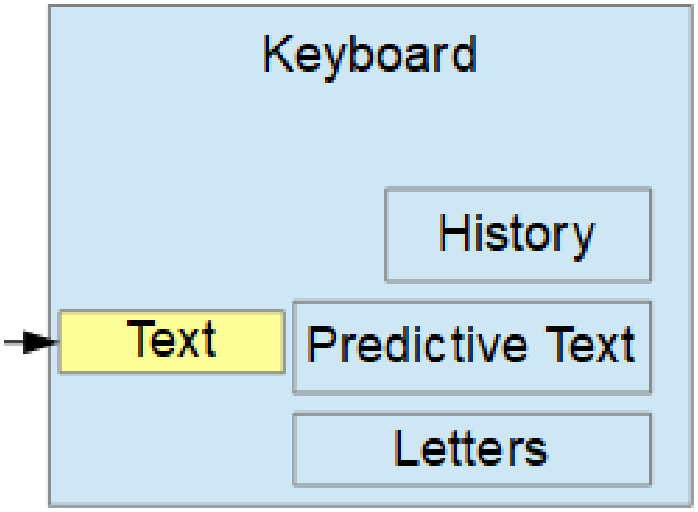

Figure 8: Text subsystem description diagram

5.4.1 Assumptions

The user is able to interact with the keyboard to help create words in the text box. The text box displays the words that are created by the user.

5.4.2 Responsibilities

The text box displays letters or words that are created by the user. The user can interact with the text box within the application, such as save the word or clear the text box.

5.4.3 Text Subsystem Interfaces

Table 7: Subsystem interfaces

ID	Description	Inputs	Outputs
Text Box	Displays words that are created by the user	The users eye location by the eyetracker from using the keyboard function	Displays words in the text box

6 Menu Subsystems

The menu section lets the user interact with the application.

6.1 Save Subsystem

This feature saves the progress of the users application by creating/modifying a file within the machine.

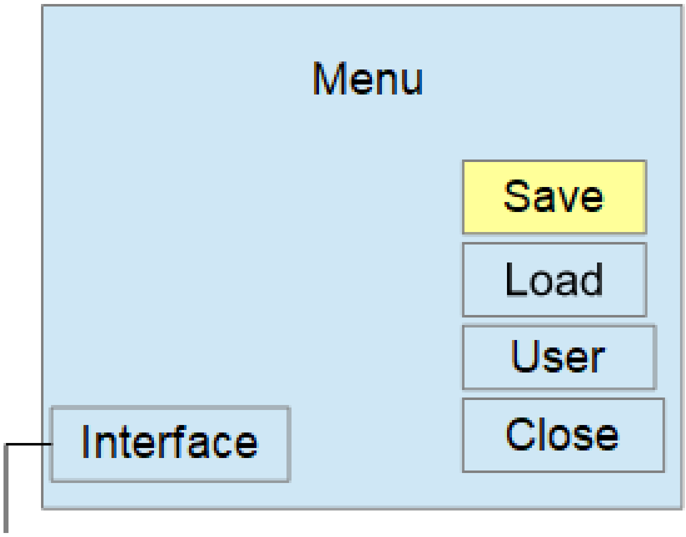

Figure 9: Save subsystem description diagram

6.1.1 Assumptions

The save subsystem saves the users activities, such as word creations and how many times the words were created by the user.

6.1.2 Responsibilities

This function saves the users profile and their activity within the session by creating a file and stored in the users machine. The file can be modified if there is an existing file stored in the machine.

6.1.3 Subsystem Interfaces

Table 8: Save Subsystem interfaces

ID	Description	Inputs	Outputs
Save	Creates/Modifies a file that stores the users activity	N/A	File Created/Modified

6.2 Load Subsystem

This feature loads the file into the application, so the user can continue their progression or have a list of words that were previously created to display.

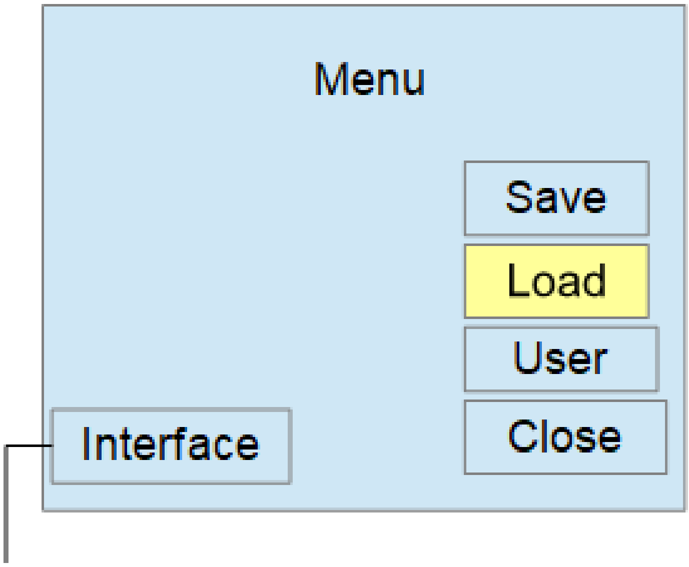

Figure 10: Load subsystem description diagram

6.2.1 Assumptions

There is an existing file stored in the users machine, that can be loaded into the application. The application loads the file and keeps track of the users progression based on their activity, such as the number of times the user has used certain words.

6.2.2 Responsibilities

An existing file is stored into the application. The application loads up the users past activities within the application.

6.2.3 Subsystem Interfaces

Table 9: Load Subsystem interfaces

ID	Description	Inputs	Outputs
Load	Loads the file into the application	File used for the application	Loads users activities in the application

6.3 User Subsystem

The user has a profile in the application, which consists of the users activities.

6.3.1 Assumptions

The user is able to create and modify their information within the application.

6.3.2 Responsibilities

The user system displays the users information and their activities, such as their name and their selection of words. The user is able to modify their information.

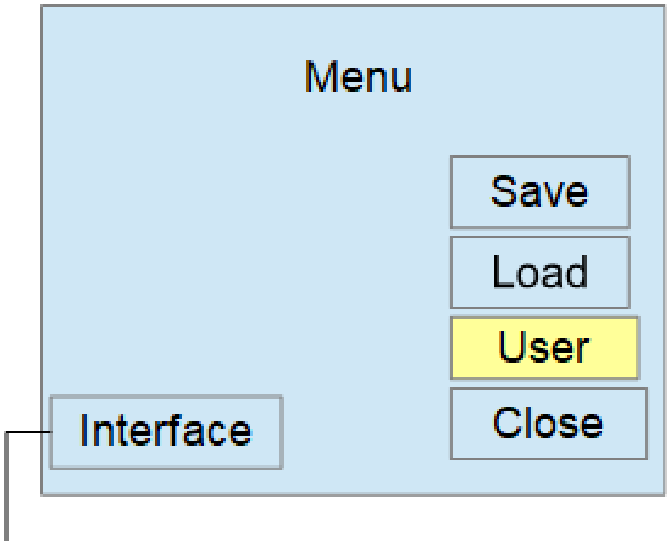

Figure 11: User subsystem description diagram

6.3.3 Subsystem Interfaces

Table 10: User Subsystem interfaces

ID	Description	Inputs	Outputs
Users Profile	Create/Modify/View the users profile	Users Selection on the profile selection	Displays users information based on which section is selected

6.4 Close Subsystem

This function closes out the application.

6.4.1 Assumptions

The progress has been saved by the user. If not saved, the application will prompt the user if they would want to save their progress. If saved, the application will create/modify the file and store in the users machine. The application will close in the users machine.

6.4.2 Responsibilities

The function will prompt the user if the user wants to save their progress. The application will close once the user has selected their decision.

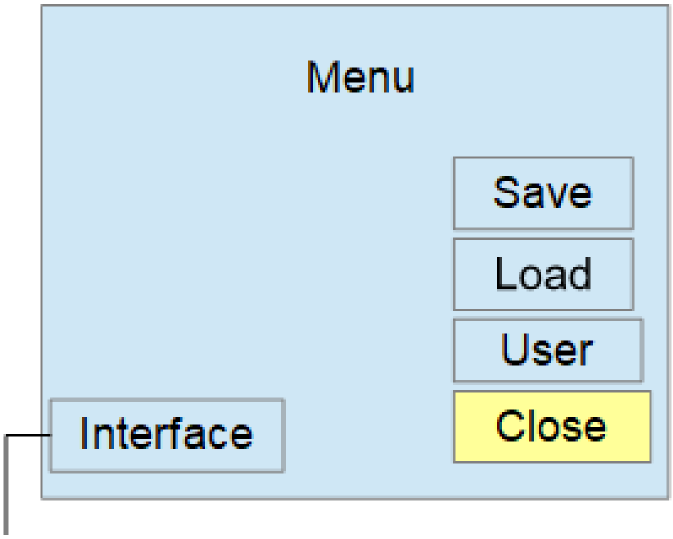

Figure 12: Close subsystem description diagram

6.4.3 Subsystem Interfaces

Table 11: Close Subsystem interfaces

ID	Description	Inputs	Outputs
Close	Closes out the application	N/A	N/A

6.5 Interface Subsystem

This feature lets the user interact with the application.

6.5.1 Assumptions

The interface lets the user interact with the application using the eye-tracker.

6.5.2 Responsibilities

The interface feature allows the user to interact with the menu functions.

6.5.3 Subsystem Interfaces

Table 12: Menu Interface Subsystem interfaces

ID	Description	Inputs	Outputs
Interface	Allows the user to interact the menu screen	Eye-Tracker	Menu Selection Opens

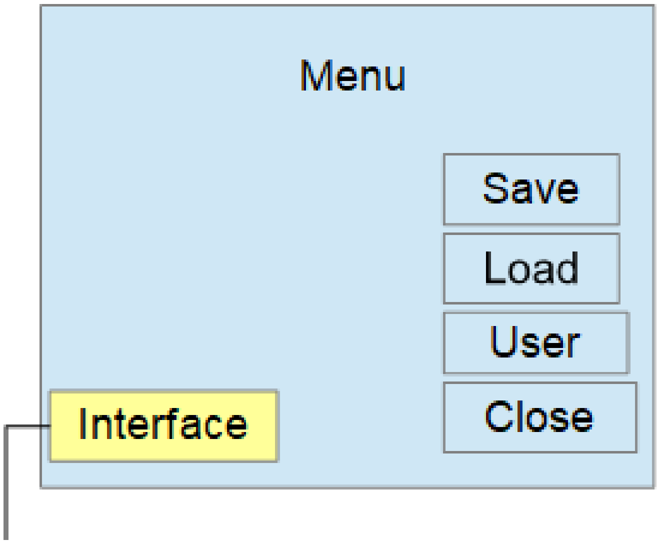

Figure 13: Menu Interface subsystem description diagram

REFERENCES