Our prototype consisted of two different aspects combined into one technology. The first part was

the eye tracking technology and the second one was the automatic adjustment system. The eye

tracking technology was developed by using an open source common platform called webgazer.js.

Although we used this open source platform, we made some intensive modifications to develop

our prototype. Our main objective was to understand the usage of eye tracking technology to

distinguish between people with dyslexia from those without dyslexia. Besides this goal, we also

needed to focus on some parameters to achieve this objective such as the application had to be

simple in design to be accessible for everyone, it had to be non-invasive in nature, it needed to be

secure and safe to use and the data that was collected must also be stored securely.

The prototype was based upon the findings from the previous studies which suggested that the eye

movement patterns of people with dyslexia were very different from those without dyslexia. These

studies suggested that the reading patterns varied between dyslexic and non-dyslexic individuals.

The same was also true for non-reading tasks as well. These differences of eye movement patterns

were displayed by various factors such as regressions, fixations durations, number of fixations and

saccades. Although there were some studies conducted prior to our study regarding the effects of

these factors on eye movement patterns of dyslexic readers, there was a research gap when it came

to technologies which could automatically detect these changes in patterns of eye movements.

**Features of the prototype:**

Our prototype is an eye tracking technology designed specifically to detect dyslexic eye movement patterns.

It is different from other technologies in terms of the innovative concept of detection of eye movements using simple webcams located in a normal laptop. The prototype application is independent of hardware or software platforms. Most of the previous applications or eye detection technologies which we came across in our research were dependent on hardware platforms or software technologies. They also had further issues related to portability and interchange of data due to various data formats. The technologies which have been developed for detection of dyslexia are quite new and they still have a long way to go to become completely reliable and accurate as well.

Our prototype however, does not have any barriers related to the hardware or software platforms. It is written in simple JavaScript which can be easily used in almost all web platforms. It is easy to use as well as it has been proven to be quite reliable and accurate. It has been developed on the webgazer.js platform, which is an eye tracking library which makes it possible to use simple webcams for the purpose of tracking eyes. A combination of three libraries were used for the prototype development which was included in the webgazer.js , they were:

1. Clmtrackr
2. Js\_objectdetect
3. Tracking.js

For the specific need of our prototype, we have modified libraries as well as written our own prototype specific code. For example: Although there were four different regression modules available by default within the webgazer.js, we have only used “ridge” regression module as it was suitable to detect the map the eyes to the exact locations on the screen.

**Structure of our prototype:**

Our prototype consists of three major parts:

1. Input and storage
2. Processing
3. Output
4. Input and storage: - This part of the prototype is responsible for detecting the eyes and the various movement patterns within the eyes. The “ridge” regression module along with “clmtrackr” were used for this purpose. There are different tracking and regression models available in webgazer.js however, we have opted to use the above mentioned models as they are suitable for our project.The eye movement information in terms of number of fixations and the duration of reading time are stored in the local-storage. The underlying modules were used from the webgazer.js however, they were modified to store specific eye movement patterns and the reading time.
5. Processing: The data stored in the local-storage was then analyzed. From various literature and previous experiments a logic was derived for the prototype in terms of two particular logic statements for number of fixations and the reading time. In order to obtain better accuracy the two values were interlinked and if both conditions matched then the result would be positive otherwise it would be negative.
6. Output: Once the processing is completed, the output is shown which consists of two things. The first one is the detection of dyslexia which can be positive or negative. The second thing is the automatic customization of textual contents which only occurs if the user is found to have dyslexia otherwise no changes occur. The automatic customizations are set according to the standards set forth by various previous literature, researches and dyslexia organizations.

**Apparatus:**

Passages: In total there were 5 passages. The passages consist of 77 words and each passage is only shown once. Each passage is shown in a consecutive manner one after the other for all five passages.

Webcam: A webcam which is in-built in a laptop or a dedicated webcam can be used.

**Algorithm for the prototype:**

**High level pattern of algorithm:**

**Start: Initiate prototype-**

1. **Input**
2. Load the window/canvas
3. Clear the local-storage
4. Repeat and show passages
5. Start the timer and save in milliseconds
6. Activate regression module and tracker module (ridge and clmtrackr) [Note: This is done within the webgazer.js platform but it is modified for the purpose of this project]
7. Start the data collection process [Save data]
8. Declare variables for coordinates
9. Initialize variables
10. Declare arrays to store all coordinates and points
11. if coordinates are not equal to 0

add new item into the array containing all points

1. The reading time taken by the participant to read the first passage is stored along with the number of fixations which occur while reading the passage
2. **Processing / Logic**
3. The data thus stored in the local storage is analyzed
4. **If** number of fixation > threshold **AND** reading time > threshold
5. The user is diagnosed as dyslexic and the text content is customized
6. **Else** the reader is not diagnosed as dyslexic then the customizations do not take place
7. **Output**
8. The text content is changed/customized in terms of the font-size, font-color, background color and font-style.

**End:**

**Functional Algorithm:**

window.onload = function() // Load window/canvas

// Clear the local-storage

//Setting time

//Starts timer and saves in milliseconds

webgazer.setRegression('ridge') //activate ridge regression model

.setTracker('clmtrackr') // activate clmtrackr tracking model

.setGazeListener(function(data, clock)

/\* data is an object containing an x and y key which are the x and y prediction coordinates (no bounds limiting) \*/

/\* elapsed time in milliseconds since webgazer.begin() was called \*/

.begin() //start the process of data collection

//Initiate Canvas and show passage

//declare Xvalues and YValues

//Initialize variables:

1. XValues <- localStorage(X)
2. YValues <- localStorage(Y)

//Declare variables xArr, yArr, pointArr

//Save XValues and YValues in a file for download

//Declare pointArr array

//Repeat until the length of array xArr

// **If** xArr, yArr values are not equal to zero

//add new item to pointArr array

//object created for the X and Y values

/\* Create an array to store the count for pointArr array \*/

// Declare and initialize finalCountArray

// Declare and initialize counts variable

//Declare and intialize threshold value

//Repeat comparison of pointArr array elements

//Compare pointArr array element X,Y to pointArr array element X+1, Y+1 until the end of array

//If comparison value is positive increment the counter

//else do not increment

// If counter value is greater than threshold for number of fixations and the time taken to read

// Initiate customization of text

//Remove duplicate values

//When continue button is clicked and confirmed

//Change the text passage

// **If** user is Dyslexic show the changes

//**Else** do not show changes

//store the data

//Clear the old datavalues

**Steps to follow for using the prototype:**

These steps are to be followed by the conductor and participants of the prototype testing:

1. The instructor shall ask the participant to sit down in front of a laptop with a built-in camera or a desktop computer with a webcam.

2. The instructor will ask the user to directly look straight towards the webcam and remain focused without sudden tilting of head or other sudden gestures.

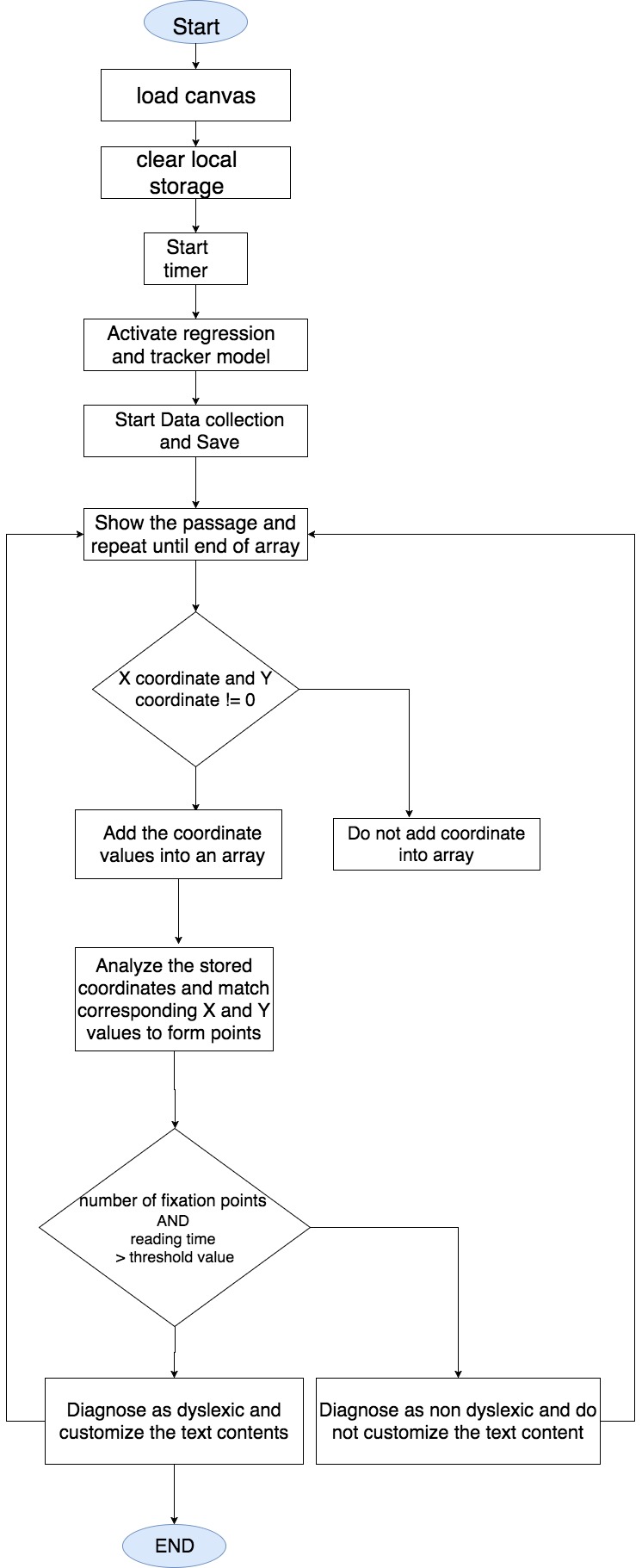
3. The instructor shall run the code from any normal code editor.

4. Once the code is live, an alert box will be displayed asking for the participant to agree that the webcam is going to be activated. This alert is placed in terms of universal design standards as well as following the data security protocols.

5. Once the participant agrees, the webcam is activated. The instructor shall ask the participant to read the displayed text. The text paragraphs shall be displayed for a total of 5 times and each time the text will be different. Also, at a given time only one passage will be visible and the rest of the text will remain invisible. (Principle 3: Understandable Guideline 3.3: Input Assistance – Help users avoid and correct mistakes)

6. The participant shall read the text paragraphs and once completed the data is automatically recorded within our prototype. Once the participants signal that he/she has finished reading the text, the instructor shall stop the process.

7. Once the participant clicks on the button, the data that is stored within our system is automatically calculated and the user will be prompted with an alert box on whether or not they want the system to make the adjustments to make the text more readable. (Principle 3: Understandable Guideline 3.3: Input Assistance – Help users avoid and correct mistakes) If they agree, then the text will be changed for better readability.

****