

**DOCUMENT READING SYSTEM FOR BLIND PEOPLE
("READING EYE")**

19-20-J 17

P.S.N.Kularathne

IT16165762

Bachelor of Science (Hons) Degree in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

May 2020

DOCUMENT READING SYSTEM FOR BLIND PEOPLE ("READING EYE")

19-20-J 17

P.S.N.Kularathne

IT16165762

Project Final Report

(Dissertation submitted in partial fulfillment of the requirements for the B.Sc. (Honors)
degree in Information Technology Specialization in Information Technology)

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

May 2020

DECLARATION

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to Sri Lanka Institute of Information Technology, the nonexclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature
(P.S.N.Kularathne)

Date

The above candidate has carried out research for the bachelor's degree Dissertation under my supervision.

Signature of the supervisor
(Ms. Suranjini De Silva)

Date

Signature of the Co-supervisor
(Dr. Anuradha Jayakody)

Date

ABSTRACT

There are 285 million visually impaired people in the world according to the World Health Organization (WHO). And out of that 39 million are legally blind. According to WHO, the visually impaired people as a ratio is higher in developing countries [1]. Visually impaired communities in countries like Sri Lanka are often ignored and not well looked after by the government bodies. Since Sri Lanka is a developing country, most people who are blind or visually impaired are also not that much wealthy and cannot help themselves too. Therefore, this project is proposed as an action towards helping the visually impaired and as a start for raising awareness about our responsibility towards the visually impaired people.

As mentioned before there are many problems that the visually impaired people encounter while reading printed documents. To read and write they normally use Braille coding system which was introduced a very long-time age. It was the best method for them to achieve literacy.

Although many methods were introduced, Braille was the most practical method to use. Because in the Braille method, person has to touch the raised dots in a document in order to read. People have to memorize the specific code for each character or symbol. Since the primary method is touch, it worked well for them so far. While it is a practical and convenient method for reading, it has some certain limitations too which cannot be solved using Braille alone. For example, the average reading speed of a person who uses Braille is about 125 words per minute, but greater speed can be up to 200 words per minute. The average reading speed of most adults (sighted) is around 200 to 250 words per minute. Clearly, there is a significant difference between the two average reading speeds for both types. Not only slow reading time but inaccessibility of the Braille materials is also a major problem. Though there are many software applications and hardware tools available in the market nowadays, they do not solve all the problems these users have. Besides most of these apps and devices are expensive and many people cannot afford them.

The proposed system will be an all-in-one solution to these problems blind users have when reading. They can use the system read any kind of document they find when completing their daily tasks.

This document comprises the full depth description of the CEBCRD component which will guide the VI user to read and get the idea about the mathematical equations and the charts in the documents. Furthermore, the document illustrates all the tools and technologies, applied concepts for implementation, the flow of the system through use case diagrams, use case scenarios and other related Unified Modeling Language diagrams, libraries and external interfaces allied to the CEBCRD component.

The developed component utilizes Machine learning (ML) techniques and Image processing techniques. Moreover, the Optical context of character recognition (OCR) to identify and separate the charts based-contents and Equations based contents. Using a android mobile application outcome will present to the VI end-user through using azure text to speech API.

Therefore, this document will guide the proposed methodology to accomplish the target objectives of the developed CEBCRD component.

Keywords – Visually Impaired, Assistive Technology, Machine learning, Image processing Optical context of character recognition, API

ACKNOWLEDGEMENT

I would sincerely like to express my gratitude to our supervisor Ms. Suranjini Silva as well as our co-supervisor Dr. Anuradha Jayakody for their immense support, motivation, knowledge and enthusiasm rendered to us while carrying out this research.

Moreover, I also grateful to thank my parents for the unceasing encouragement, support and attention as well as my research group members for moral and emotional support given whenever necessary.

TABLE OF CONTENTS

DECLARATION	iii
ABSTRACT	iv
ACKNOWLEDGEMENT	v
TABLE OF CONTENTS	vi
LIST OF FIGURES	vii
LIST OF TABLES	viii
LIST OF DEFINITIONS, ACRONYMS AND ABBREVIATIONS	ix
1. INTRODUCTION	1
1.1 Background & Literature Survey	1
1.2 Research Gap	3
1.3 Research Problem	4
1.4 Research Objectives	5
2. METHADODOLOGY	6
2.1 System Overview	6
2.2 Charts and Equations based Contents Reading and Detection (CEBCRD)	7
2.2.1 Mathematical Equations based Content Identification (MEBCI)	8
2.2.2 Chart based Content Identification (CBCI)	8
2.3 Commercialization Aspects of the Product	12
2.4 Testing and Implementation	12
2.4.1 Testing	12
2.4.2 Implementation	13
2.4.3 Tools and Technologies	22
3. RESULTS & DISCUSSION	23
3.1 Results	23
3.2 Research Findings	27
3.3 Discussion	27
3.4 Summary of Student Contribution	28
4. CONCLUSION AND FUTURE WORKS	29
5. REFERENCES	30
6. APPENDICES	32
Appendix A - Use Case Diagram of CEBCRD Component	32
Appendix B – Use Case Scenarios of CEBCRD Component	32
Appendix C - Class Diagram of CEBCRD Component	35
Appendix D - Works Breakdown Structure (WBS) of Reading-Eye System	36

LIST OF FIGURES

Figure 1: High Level Architecture Diagram of Reading - Eye System	6
Figure 2: High Level Architecture Diagram of CEBCRD Component	7
Figure 3: Pie Chart Segmentation and Percentages with Original Image	9
Figure 4: Otsu Binarization Applied Bar Chart Image for Thresholding	10
Figure 5: Horizontal and Vertical Lines Detected, Weighted Parameters Applied Image	11
Figure 6: Bounding Box Identified Final Bar Chart Image	11
Figure 7: Source Code of Mathematical Equations Based Content Identification	14
Figure 8: Source Code of Pie Chart Based Content Identification Part 1	14
Figure 9: Source Code of Pie Chart Based Content Identification Part 2	15
Figure 10: Source Code of Pie Chart Based Content Identification Part 3	16
Figure 11: Source Code of Pie Chart Based Content Identification Part 4	17
Figure 12: Source Code of Bar Chart Based Content Identification Part 1	18
Figure 13: Source Code of Bar Chart Based Content Identification Part 2	19
Figure 14: Source Code of Bar Chart Based Content Identification Part 3	20
Figure 15: Box Detection Algorithm Applied Source Code of Bar Chart	20
Figure 16: Source Code of Implementation of REST API Part 1	21
Figure 17: Source Code of Implementation of REST API Part 2	21
Figure 18: Sample Result of MEBCI	24
Figure 19: Sample Result of Bar Chart Identification	25
Figure 20: Sample Result of Pie Chart Identification	25
Figure 21: API Request to Get the Sample Result of MEBCI	26
Figure 22: API Request to Get the Sample Result of Bar Chart	26
Figure 23: API Request to Get the Sample Result of Pie Chart	27
Figure 24: Use Case Diagram of CEBCRD Component	32
Figure 25: Class Diagram of CEBCRD Component	35
Figure 26: Work Breakdown Structure Diagram	36

LIST OF TABLES

Table 1: Definition for the Terms	ix
Table 2: Glossary of Acronyms	ix
Table 3: Comparing Reading- Eye System with Existing Systems	4
Table 4: Accuracy of Mathematical Equation Based Content Identification	23
Table 5: Summary of Student Contribution	28
Table 6: Use case Scenario 1	32
Table 7: Use case Scenario 2	33
Table 8: Use case Scenario 3	33
Table 9: Use case Scenario 4	34
Table 10: Use case Scenario 5	34
Table 11: Use case Scenario 6	35

LIST OF DEFINITIONS, ACRONYMS AND ABBREVIATIONS

Table 1: Definition for the terms

Term	Definition
Reading-Eye	The name of the proposed system
Machine Learning	A subset of artificial intelligence which is a study of computer algorithms that improves automatically through experience.
Image Processing	A technique which use for digital computer to process digital images through an algorithm.
Postman	A collaboration platform for API development.
Tesseract OCR	Open source optical character recognition engine
OpenCV	The library which used for Image Processing
SciPy package	A free and open-source Python library used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing
Scikit-image	An open-source image processing library for the Python programming language. It includes algorithms for segmentation, geometric transformations, color space manipulation, analysis, filtering, morphology, feature detection, and more.

Table 2: Glossary of Acronyms

Acronym/Abbreviation	Definition
CEBCRD	Charts and Equations based contents Reading and Detection.
MEBCI	Mathematical Equations based Content Identification
CBCI	Chart based Content Identification
ML	Machine Learning
OCR	Optical Context of character Recognition
VI	Visually Impaired

1. INTRODUCTION

1.1 Background & Literature Survey

Assistive technology assists people with disabilities to achieve their ordinary life tasks and helps to improve their tasks in working, transportation, and majorly in academic activities at the same time, it guides them to accomplish better independence and makes the life more comfortable. According to the World Health Organization (WHO) official statistics, there are 285 million visually impaired people (partially or completely blind) in the world in which 39 million of them are blind and 87% of them are from developing countries [1]. At the same time, globally, it is estimated that approximately 1.3 billion people live with some form of vision impairment [2].

Assistive technology area is broken down into low, mid, and high tech categories. Low tech encompasses equipment that is often low cost and does not include batteries or requires charging. Examples include adapted paper and pencil grips for writing or masks and color overlays for reading. Mid tech supports used in the school setting include the use of handheld spelling dictionaries and portable word processors used to keyboard writing. High tech supports involve the use of tablet devices and computers with accompanying software. Software supports for writing include the use of auditory feedback while keyboarding, word prediction for spelling, and speech to text. Supports for reading include the use of text to speech (TTS) software and font modification via access to digital text.[3]

According to the major the development of the technology, there are some developed products to avoid the difficulties of the reading documents which help visually impaired people to improve their reading ability. [4],[5].

However, focusing on the improvement of reading ability and gain knowledge for academic purposes for the visually impaired people there is a major impediment because they have limited resources for access to most of the specific types of documents.

Limited supports are available for math instruction and mostly consist of grid based software to allow younger students to keyboard equations and auditory feedback of more complex equations using MathML and Daisy. [3]

Reading and detecting charts based-contents and Equations based contents.

The Most software applications developed to detect and analyze text contents but most of them cannot detect chart-based content and equations-based content [8],[9],[10],[11].

Detecting equation regions from scanned books has received attention in the document image research community in the past few years. Compared with regular text blocks, equation regions have more complicated layouts so we cannot simply use text lines to model them. On the other hand, these regions consist of text symbols that can be reflowed, therefore that the OCR engines should parse them instead of rasterizing those like image regions.

A Simple Equation Region Detector for Printed Document Images in Tesseract is one of the researches that have been conducted for one of the equation detector, which use to detect equations using Tesseract OCR engine. The equation detector is built into the layout analysis stage. Specifically, it works after the text line detection and before the paragraph/block

partition, so that the detected regions won't be mistakenly grouped into a text paragraph. In addition, we skip these regions during the recognition stage, because no special parser has been built yet in the Tesseract engine [6]. This will basically classify the text symbols then identify the text regions and expand the text regions to read and detect the equations-based contents.

Another research called Schmoozer mainly focuses to detect mathematical equations based on the separately recognize characters and symbols in a mathematical equation that are printed on documents and convert them into digitized text. The first step was designing of mathematical symbols and numbers using Photoshop in bitmap format. Bitmap files are easy to create and pixel data stored in a bitmap file could be accomplished by using a set of coordinates that allow the data to be conceptualized as a grid. Therefore, all the major mathematical characters were created in bitmap format and stored inside a folder called "letters_numbers". Then created a MATLAB template and match each character /mathematical symbol/number into created bitmap images [6].

Moreover, according to the Schmoozer use to identify the equations by obtained input images from the graphical regions identification function Read image, Convert to grayscale and converted into a binary value and finally match every character of the input image with the binary value obtained from the created template and regenerate the equation as digitized text. Function's output is a set of single digitized values for a single character/symbol in the equation and converts to DDL file for use to web service [6].

When considering the literature survey for chart-based content detection, we were unable to find comparatively numerous researches since this feature compromise of new methodologies and techniques.

One of the major challenges in chart image classification is the variability of the structure and visual appearance of each chart type found from and conducted research called Convolutional Neural Network Based Chart Image Class [12].

However, the different approaches have been proposed in the literature to deal with the problem of chart image classification, the variation in chart style, size, color, resolution, and the content are not yet resolved.

1.2 Research Gap

The research gap could demonstrate why should we need to implement the proposed system and this will indicate what are the various drawbacks and needed improvements of the previously developed systems as a comparison with proved system and the previously developed systems by referencing the gathered information in the literature survey. This could show basically, how much the proposed system deviates from previously conducted researches.

As discussed in the literature review previously it seems that many visually impaired people also can improve their reading ability and, they can exposure to new areas by gain knowledge through listening to audio document reading systems.

However, after completion of the background and the literature survey, there were certain identified limitations that laid the foundation for our research. The proposed solution is based on to reduce those limitations and perform an improved better solution for visually impaired people and at the same allow the visually impaired people to reach the same level of reading level as a normally sighted person without a barrier by providing the descriptive idea and best understanding of texts, digital images, charts, tables and mathematical equations.

Most of the solutions the ones had been carried out in advance had a similarly faced when compared to the proposed system solution, furthermore developed features and integrated functionalities were minimal when compared with the proposed solution.

When reviewing the research gap, the technological advancements that have taken place over the restrictions and the limitations in the technologies at past times of development should not be disregarded, as the client-server data security and cloud storage advancements present as per today, did not exist back then.

The use of advanced machine learning techniques and image processing libraries, conventional neural networks, Deep Learning Frameworks and techniques, cloud hosting with data security are the sum of technologies that improve the usability and the widening factors which are the most trending technologies in the present that our proposed system wish to implement. With the use of new technologies, the result of the proposed system will become an accurate, more efficient, effective and optimal solution for the targeted assistive technology area for visually impaired people.

Below table represents the comparison of the Reading – Eye system features and the existing systems features identified by the literature review. The table is basically highlights the Charts and Equations based Contents Reading and Detection (CEBCRD) component.

Table 3: Comparing Reading- Eye System with Existing Systems

Existing Product Features	Amazon Kindle	BARD Mobile	Capti Voice	KNFB Reader	Schmoozer	Reading-Eye (Proposed System)
Equations based contents Identification and Reading	✗	✗	✗	✗	✓	✓
Charts based contents Identification and Reading	✗	✗	✗	✗	✗	✓
Images' size reduction for efficient Communication	✗	✗	✗	✗	✗	✓
Voice output	✓	✗	✓	✓	✓	✓

1.3 Research Problem

Reading is not a difficult task for a person who has the ability to see. However, it is a comparatively much difficult task for a visually impaired person. Especially the documents with huge contents are not accessible in the braille system. Moreover, there is a limitation of identification and to get proper knowledge of complex mathematical equations and chart based contents for them because most documents are not written in the braille system the complex contents. Therefore in such a situation, anyone will prefer to have a guide with a proper document reading system that acts promptly.

Due to the major development of modern technology, there are many existing document reading apps are available for visually impaired people. However, the majority of software applications developed to detect and analyze text contents but most of them cannot detect chart-based contents and equations-based contents [8],[9],[10],[11].

Moreover, most of the mathematical equation detecting researches had found impediments as follows in document reading systems [13].

- The difficulty of recognizing the structure of the mathematical expression.
- The difficulty or an error on the recognition of the numbers, variables, and the operators (symbols) in the equation.
- The difficulty of identifying the region of the equation in the complicated layout.

When considering the chart based content detection, there were the least amount of results and unable to find comparatively numerous researches since this feature compromises new methodologies and techniques but still it not resolved with a proper solution.

The major challenges in chart image classification as found as follows in research called Convolutional Neural Network Based Chart Image Class [14].

- The difficulty to detect the variability in the structure of the chart.
- The difficulty of identifying the size, region, content and the color of the chart.
- The difficulty of identifying the visual appearance of each chart type.

Therefore, to reduce the impediments occur for visually impaired readers, the developed Reading-Eye system solution compromises to guide visually impaired users to get a detailed description of the mathematical equations and the chart based contents when reading those contents in the document through listening to audio. Moreover, the identified problems would be minimized and the solution will give better outcome for visually impaired readers to gain knowledge by listening to the audio descriptions of complex documents.

1.4 Research Objectives

The charts and equations based contents reading and detection (CEBCRD) is a main component of the Reading-Eye System. This component will mainly analyze and detect the equations based contents and the charts based contents in the documents.

In order to achieve this task, the implementation of the system is done using Machine Learning methodologies and Image Processing techniques which supported to mobile and cloud platforms. The existing data in the cloud storage is used by the component to get real-time updates.

1.4.1 Main Objective

The main goal of the proposed system is to facilitate visually impaired people to read printed documents that are not written using the braille system and help them to improve their reading capability as normally sighted people.

1.4.2 Specific Objectives

The specific goals for the CEBCRD component are as follows.

- Identify and detect the chart based contents separately by chart type, size, and the resolution of the chart and make a detailed description according to the identified content with containing the chart labels and values.
- Identify and detect the mathematical equations based contents separately by the numbers, variables, and operators contained in the equations and make a detailed description according to the identified content.

2. METHADODOLOGY

This section describes the techniques and mechanism which that used to build the “Reading-Eye” system.

The explanation contains implementation of the project components basically highlights the Charts and Equations based Contents Reading and Detection (CEBCRD) component, materials and data to be used, how to gather the data and the realistic timeline and schedule that are needful in achieving the project objectives. Furthermore, it specifies the research areas that we have identified to accomplish the research and the tools which used for identify the research component contents.

2.1 System Overview

According to the system overview of the proposed Reading-Eye application, it has the capability to auto capture a printed document and uploads it to a server automatically, and it provides better solution for detecting text, natural images, equations, charts and table based contents in a document. Moreover as the outcome, this application generates the audio of the detailed description of the identified content for the user to listen. Furthermore, the application is equipped with a secure data connection.

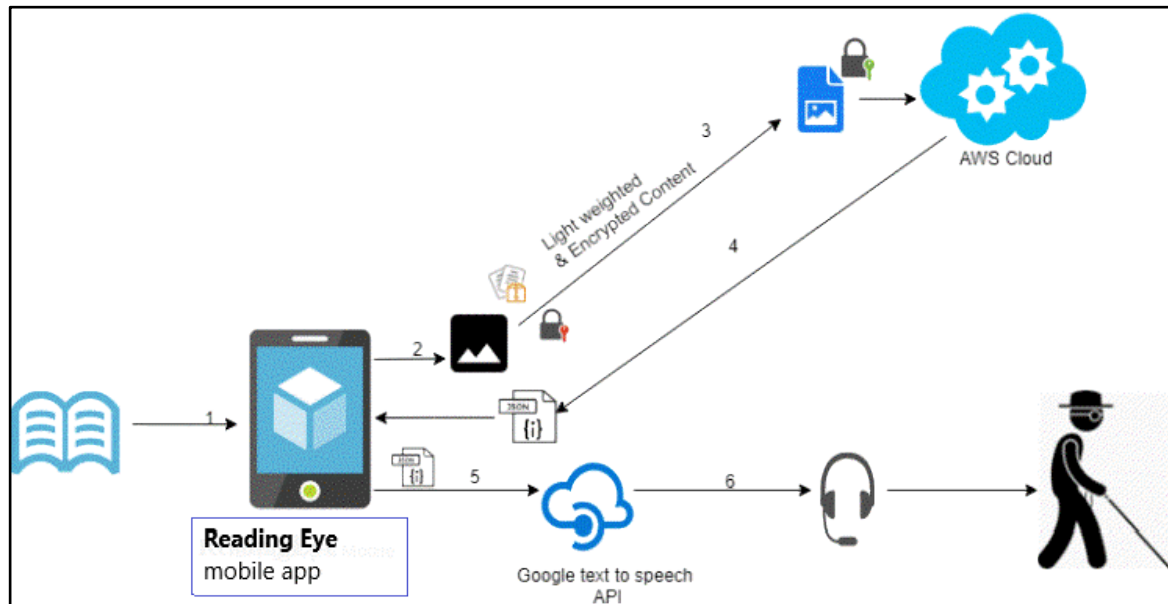


Figure 1: High Level Architecture Diagram of Reading - Eye System

2.2 Charts and Equations based Contents Reading and Detection (CEBCRD)

The CEBCRD is a component of the proposed Reading-Eye application. By analyzing the captured photograph of the document by the user system will encrypt the data and store it in the cloud storage. The captured image will separately divide into the relevant sub-modules as equations based contents and chart based contents etc. Moreover after that stage graphical image of the equations based contents and chart based contents identified separately.

The identification of the chart based contents is done by separating the chart size, type, color, resolution etc. Moreover, the equations also separate from the variables, operators, numbers, etc.

After that initial process, separated identification of the charts and equations based contents of the Reading-Eye system will generate detailed descriptive text content which contained the contents of the charts and equations. As the final output of the component, a JSON file is generated for the user to listen.

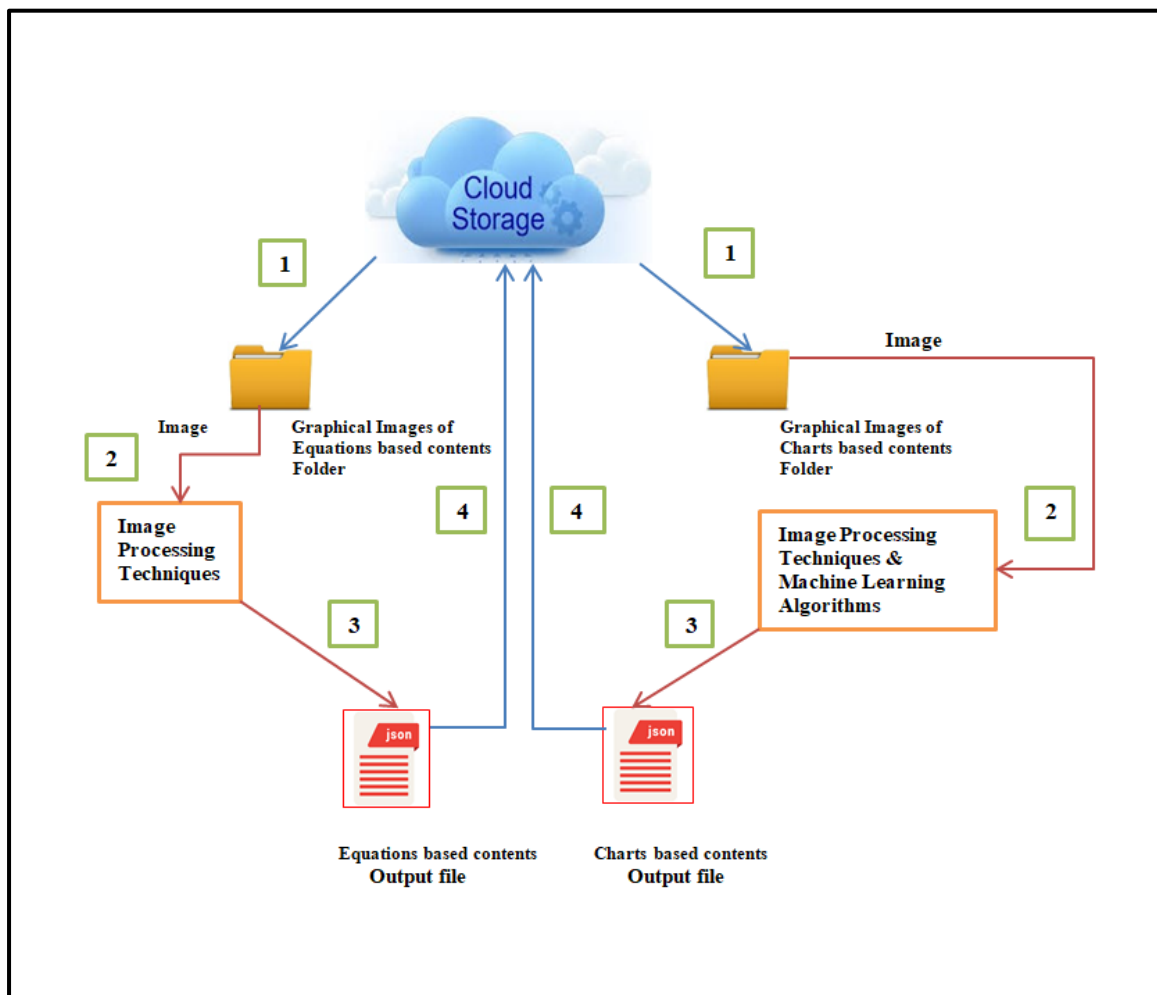


Figure 2: High Level Architecture Diagram of CEBCRD Component

2.2.1 Mathematical Equations based Content Identification (MEBCI)

The main purpose of the Mathematical equations-based content identification function is to identify mathematical equations separately from the printed documents and recognize the numerical constants, characters and mathematical operators contained in the equation. Then the function will generate a description about the equation and save in a JSON file.

The first step of the function is identifying the mathematical equation using the JPEG file. This file format is used since it is a common image file format that digital cameras use to save their images. Furthermore, JPEG files apply the lossy compression method which helps to reduce the file size significantly without compromising the quality.

The Mathematical equations-based content identification function obtains input images from the graphical region identification function. The segmented image with equations will go through this function and will be preprocessed prior to the reading by Optical Character Recognition (OCR) tool.

The data inside the image of the equation is read using Tesseract OCR engine. After that, image will be converted to the grayscale using thresholding [15] for dimension reduction and model complexity reduction. Furthermore, image noise will be removed by applying the erosion and dilation (morphological operations) [16] using OpenCV library. After preprocessing, text of the preprocessed image will be read by the Tesseract engine.

Finally, mathematical equations-based content identification function will generate a description about the equation and saves it in a JSON file with separately identified numerical constants, characters and mathematical operators. Just like previous functions, output of this function will also be used to concatenate with other outputs to make the audio file later.

2.2.2 Chart based Content Identification (CBCI)

The Charts-based content identification function mainly focuses on identifying pie charts and bar charts from the obtained JPEG files which created by the graphical region identification function. As the final output of the function it will generate a description using the contents of both pie charts and bar charts.

Initially, the obtained JPEG image files will be preprocessed before proceeding to the main identification stage. The data inside image files will be read using in OCR engine. For the dimension reduction and model complexity reduction the read image will be converted to the grayscale using thresholding technique.

Furthermore, the morphological operators [16] of openCV such as, erosion and dilation will be applied to remove the noise of the image because result in pixel values that do not reflect the true intensities of the real scene with the noise of image.

Subsequently, the Charts – based content identification function proceeds two main identification stages as below.

▪ Pie Chart Identification

The significance of the function was based on the assumption that pie charts do not contain 3D effects, each wedge has a distinct color, and the corresponding labels are presented adjacent to the wedge.

As the first step the loaded preprocessed image file should use to detect the edges. The edge detection is performing through the Hough transform pattern recognition algorithm [17]. When detecting the edges of pie charts Hough transform pattern recognition algorithm use to detect circle positions and match their edges.

Given in a black circle on a white background, first we should guess its radius or a range of radiuses to construct a new circle. This circle is applied on each black pixel of the original picture and the coordinates of this circle are voting in an accumulator. From this geometrical construction, the original circle center position receives the highest score and most prominent circles.

Furthermore, by applying scikit-image open-source image processing library [18] which includes segmentation algorithms for geometric transformations, the image crop as the masked data and segmented. Moreover, founded segments count and segments values with the percentage of the pie chart will be identified through the SciPy package of key algorithms and functions [19] which act as the core to Python's scientific computing capabilities.

Finally, by creating the subplots and unpack the created array and generates a description of the chart. It contains the identified pie chart segments with the percentage values.

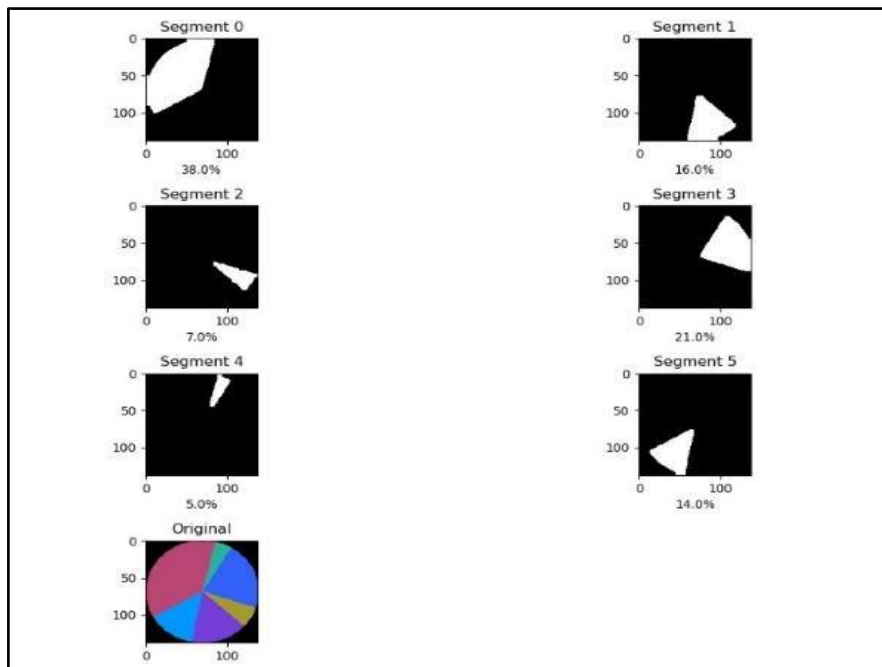


Figure 3: Pie Chart Segmentation and Percentages with Original Image

▪ Bar Chart Identification

The significance of the function was based on the assumption of bar chart orientation should be in the horizontal or vertical and we assume that bar charts do not contain any 3D effects. In addition, we assumed there are no texts inside the stacked bars and each stack has a distinct color.

As the first step the loaded preprocessed image file of the bar chart with correct orientation. Otsu Binarization is applied for threshold [15] with using various filters to smooth the image. It is done by applying the Tesseract OCR and OpenCV image processing algorithms. After that the image will be inverted.

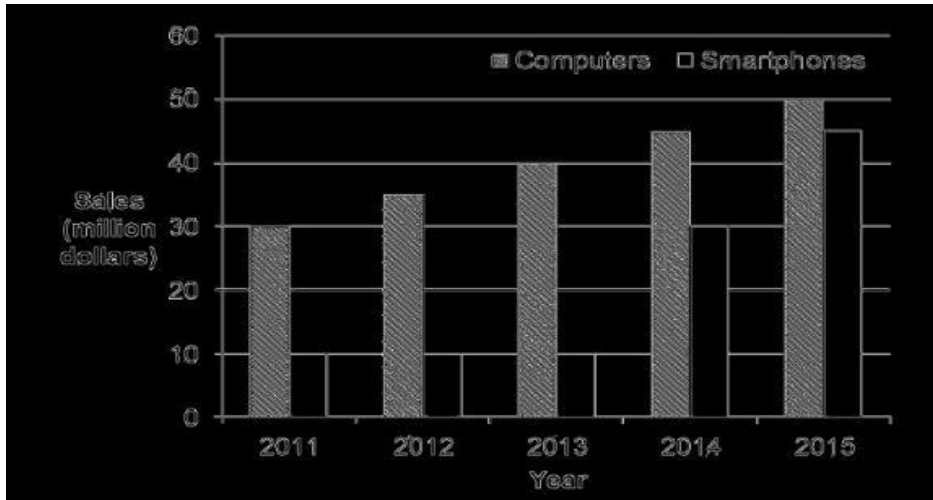


Figure 4: Otsu Binarization Applied Bar Chart Image for Thresholding

Perfectly preprocessed image file should confine our search space to the bars extracted from the connected component analysis. For that purpose, local search algorithm is used as follows. It will generate the X and Y scale dimension of the bars.

- As the beginning, the midpoint of the length of two edges of the rectangle should find along with its length. X-axis is used for the horizontally and Y-axis is used for vertically for the lengthwise in the bar chart.
- With the identification of start and end points we move along with the line made by them and detect sudden changes in the pixel values between the two adjacent points. The border or the adjacent point that separates two adjacent stacks is taken as the center of the pixel points.
- The process should continue until the finding the all the stacks inside all the bars.

Hough transform pattern recognition algorithm [17] and box detection algorithm [20] are used to detect rectangular bars separately in the bar chart. The process of applying the above algorithms and detecting the bar chart from the preprocessed image as follows.

- Defining two kernels as to detect horizontal lines and vertical lines. It will define rectangular kernel with the length based on the width of the preprocessed image.

- After defining the kernels, morphological operations of the image processing are applied for detect horizontal and vertical lines.
- Getting rid of grid lines accurately to detect the bars and then no noise will occur for false bar extraction.
- Adding waiting parameters as $\alpha = 0.5$ and $\beta = 1 - 0.5$. Then it will make the quantity of an image to be added to make new image and this function helps to add two images with specific weight parameter to get a third image as summation of two image.

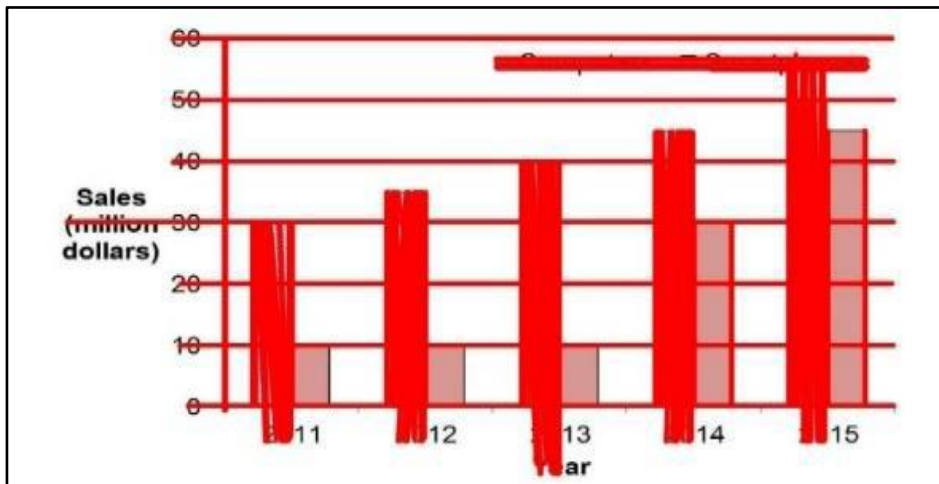


Figure 5: Horizontal and Vertical Lines Detected, Weighted Parameters Applied Image

As the next step, contours for image, this will detect all the bars and sort them from top to bottom by sorting counters using Python and OpenCV.

Following figure 6 depicts the bounding box as a yellow color line in the identified bar chart.

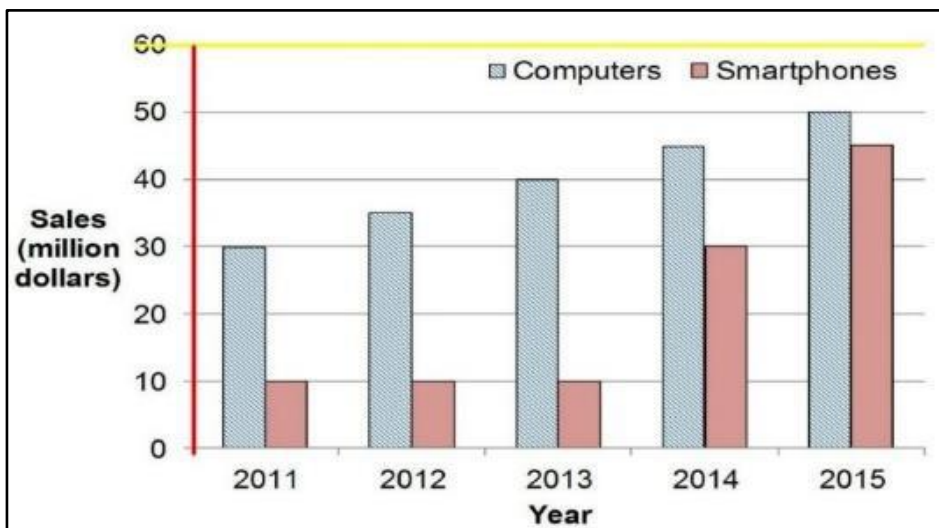


Figure 6: Bounding Box Identified Final Bar Chart Image

Furthermore, loop over all the counters to find the all the boxes and crop the part which has a rectangle and save those into a new folder. The folder contains the exact detected bars of images.

Finally, generates a description using the identified bounding boxes of the bar chart. It contains the identified bar charts with maximum and minimum scaled values of the X and Y axis's. The description will be merged with outputs of other function as necessarily to generate the final audio file.

2.3 Commercialization Aspects of the Product

The “Reading-Eye” application is mainly focuses on the vision impaired community. Moreover it is mainly targeted for the vision impaired students who are willing to read the complex document contents like mathematical equations, charts, tables, natural images and text based contents through an audio.

The application is developed through the high security in cloud platform and our main goal is to reduce the impediments of the reading capability of complex documents contents for vision impaired readers. Moreover to make their reading capability and improve their knowledge through reading as a normal sighted people.

2.4 Testing and Implementation

2.4.1 Testing

This section of the document will be discussing about the testing phases carried out to validate the integrity of the work done to check whether the actual results match the expected results and to ensure that the software system is defect free and also to involves execution of a software component or system component to evaluate one or more properties of interest. Moreover, to identify errors, gaps or missing requirements in contrary to the actual requirements.

As per the ongoing standard procedures, testing was carried out in five different phases which are discussed below.

- **Unit Testing**

Unit testing is a type of software testing where individual units or components of software are tested. The purpose is to validate that each unit of the software code performs as expected. In Reading-Eye application basically region identification, text based content identification, chart based content identification, mathematical equations based content identification, images based content identification and table based content identification test as unit wise in the initial stage.

- **Module Testing**

Module testing is used to check whether the individual modules are work properly. In Reading-Eye application we checked the charts and equations based content identification module, tables and image identification module, region and text identification module and security module separately in the module testing.

- **Integration Testing**

Upon the completion of unit and module testing, the modules are to be integrated which gives raise to integration testing. Therefore, applying the integration testing to the Reading-Eye system that verifies the functional, performance, and reliability between the modules that are integrated.

- **System Testing**

System Testing is a level of software testing where complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements. Before the application is launch to the end user system testing is applied to the Reading-Eye system.

- **User Acceptance Testing**

This was the phase where the system went through testing done by the end user, which gave feedback on the status on whether the expectations are fulfilled.

2.4.2 Implementation

The implementation of the Reading-eye system Charts and Equations based Contents Reading and Detection (CEBCRD) done by applying the machine learning and image processing techniques. The voice output which use as the final output for the end user was generated though the implementation of a REST API.

- **Implementation of Mathematical Equations based Content Identification (MEBCI)**

The implementation of the MEBCI component of the Reading-Eye system was done using importing the all necessary OpenCV libraries and the use of Tesseract is an optical character recognition. The below figure displays the implementation of MEBCI component for mathematical equations identification.

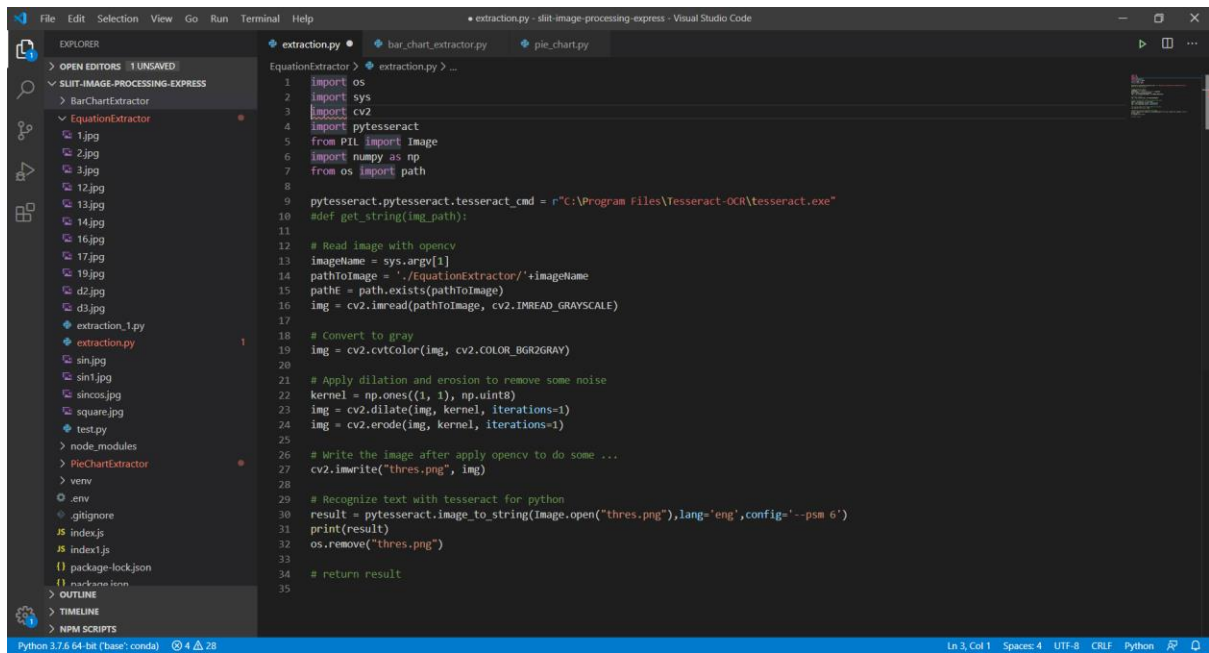


Figure 7: Source Code of Mathematical Equations Based Content Identification

■ Implementation of Chart based Content Identification (CBCI)

The implementation of the CBCI component of the Reading-Eye system was done using importing the all necessary OpenCV libraries and the use of Tesseract is an optical character recognition. Moreover, for separate identification of bar charts and pie charts scikit-image open-source libraries and functions and SciPy package of key algorithms and functions were applied.

The implementation of Pie chart identification as shown in the following figures.

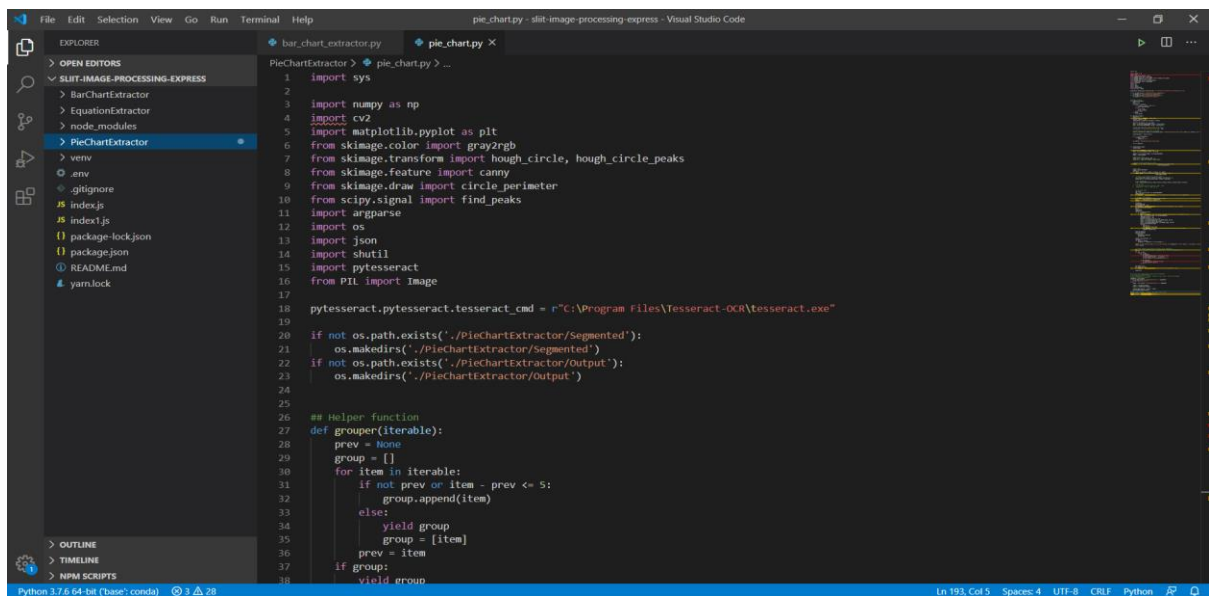


Figure 8: Source Code of Pie Chart Based Content Identification Part 1

```

24
25
26 ## Helper function
27 def grouper(iterable):
28     prev = None
29     group = []
30     for item in iterable:
31         if not prev or item - prev <= 5:
32             group.append(item)
33         else:
34             yield group
35             group = [item]
36         prev = item
37     if group:
38         yield group
39
40 def getLabels(image):
41     labelList = ''
42     height, width, channels = image.shape
43     startX = height - 100
44     legendImage = image[startX:height, 0:width]
45
46     kernel = np.ones((1, 1), np.uint8)
47     img = cv2.dilate(legendImage, kernel, iterations=1)
48     img = cv2.erode(legendImage, kernel, iterations=1)
49
50     # Write the image after apply opencv to do some ...
51     cv2.imwrite("./PieChartExtractor/thres.png", img)
52
53     # Recognize text with tesseract for python
54     result = pytesseract.image_to_string(Image.open("./PieChartExtractor/thres.png"), lang='eng', config='--psm 6')
55
56     splittedLabels = result.split()
57
58     for x in splittedLabels:
59         if (len(x) > 4):
60             labelList += x + " "
61

```

```

55
56 splittedLabels = result.split()
57
58 for x in splittedLabels:
59     if (len(x) > 4):
60         labelList += x + " "
61
62 os.remove("./PieChartExtractor/thres.png")
63
64 return labelList
65
66
67 def readPieChart(image):
68     output=image
69
70     height,width,channel = image.shape
71     mask = np.zeros((height,width), np.uint8)
72
73     image2 = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
74     edges = canny(image2)
75
76     # Detect two radii
77     hough_radii = np.arange(50, 500, 1)
78     hough_res = hough_circle(edges, hough_radii)
79
80
81     # Select the most prominent 3 circles
82     accums, cx, cy, radii = hough_circle_peaks(hough_res, hough_radii,
83                                             total_num_peaks=2)
84
85
86     # Draw them
87     image = gray2rgb(image)
88     counter=1
89     for center_y, center_x, radius in zip(cy, cx, radii):
90         circy, circx = circle_perimeter(center_y, center_x, radius,
91                                         shape=image.shape)
92

```

Figure 9: Source Code of Pie Chart Based Content Identification Part 2


```

154
155
156 # Creates subplots and unpacks the output array immediately
157 fig, axes = plt.subplots(nrows=int(segcount/2)+1, ncols=2, figsize=(12, 8))
158 it = 0
159 for row in axes:
160     for ax in row:
161         if it<segcount:
162             ax.imshow(segments[it], cmap=plt.cm.gray)
163             ax.set_title("Segment " + str(it))
164             ax.set_xlabel(str(np.round(per[it]))+"%")
165
166         if it==segcount:
167             ax.imshow(crop, cmap=plt.cm.gray)
168             ax.set_title("Original")
169         it += 1
170
171 plt.axis('off')
172 plt.tight_layout()
173 plt.savefig("./PieChartExtractor/output/output"+str(counter)+".jpg")
174
175 counter+=1
176
177
178 # construct the argument parser and parse the arguments
179 # ap = argparse.ArgumentParser()
180 # ap.add_argument("-i", "--image", required = True, help = "Path to the image")
181 # args = vars(ap.parse_args())
182 imageName = sys.argv[1]
183 if not os.path.exists("./PieChartExtractor/" + imageName):
184     print("Image not exist")
185 else:
186     image = cv2.imread("./PieChartExtractor/" + imageName)
187
188     labels = getLabels(image)
189     values = readPieChart(image)
190
191     pieChartDescription = values + labels

```

```

191     pieChartDescription = values + labels
192     print(pieChartDescription)
193
194 #Remove output folders.. if want u can remove below lines
195 shutil.rmtree("./PieChartExtractor/output")
196 shutil.rmtree("./PieChartExtractor/Segmented")
197

```

Figure 11: Source Code of Pie Chart Based Content Identification Part 4

■ Implementation of Bar chart Identification

The implementation of bar chart identification as shown in the following figures.

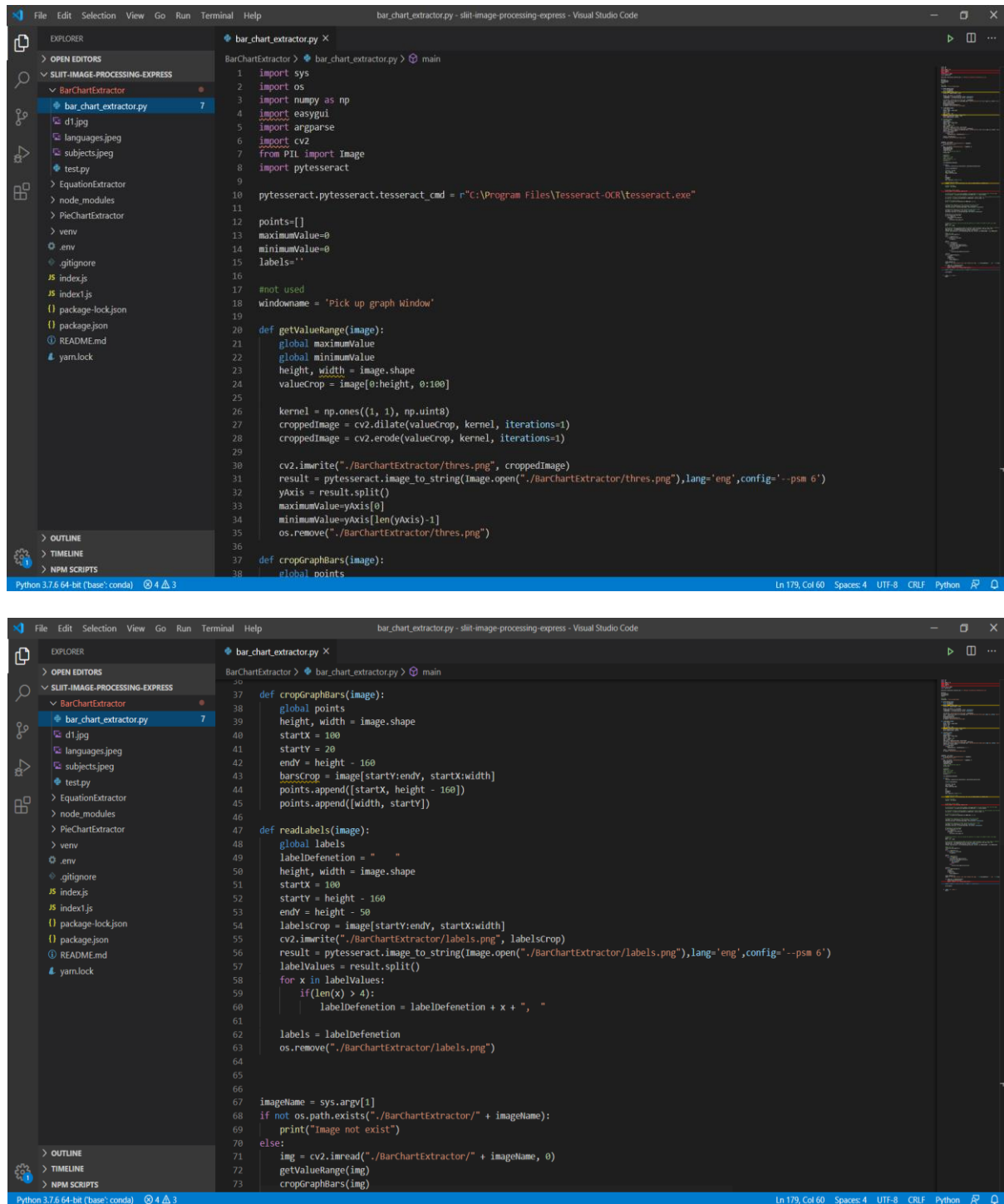


Figure 12: Source Code of Bar Chart Based Content Identification Part 1

```

66
67 imageName = sys.argv[1]
68 if not os.path.exists("./BarChartExtractor/" + imageName):
69     print("Image not exist")
70 else:
71     img = cv2.imread("./BarChartExtractor/" + imageName, 0)
72     getValueRange(img)
73     cropGraphBars(img)
74     readLabels(img)
75     # img = cv2.imread("bar1.jpeg",0)
76     cp=img.copy()
77
78     # fontScale
79     fontScale = 1
80     # Blue color in BGR
81     color = (0, 0, 0)
82     # Line thickness of 2 px
83     thickness = 2
84
85     cv2.namedWindow(windowname)
86
87 def main():
88     crop_img = cp[points[1][1]:points[0][1], points[0][0]:points[1][0]]
89
90     scale=int(maximumValue)
91
92     orig_img = crop_img
93     img = orig_img
94     height,width=img.shape
95
96     y=0
97     x=0
98     h=height-y
99     w=width-x
100     img = img[y:y+h, x:x+w] ## Crop
101
102     # Thresholding the image
103     (thresh, img_bin) = cv2.threshold(img, 150, 255, cv2.THRESH_BINARY, cv2.THRESH_OTSU)

```

```

104
105     # Thresholding the image
106     (thresh, img_bin) = cv2.threshold(img, 150, 255, cv2.THRESH_BINARY, cv2.THRESH_OTSU)
107
108     # Invert the image
109     img_bin = 255-img_bin
110
111     # Defining a kernel length
112     kernel_length = np.array(img).shape[1]/80
113
114     # A verticle kernel of (1 X kernel length), which will detect all the verticle lines from the image.
115     verticle_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (1, kernel_length))
116
117     # A horizontal kernel of (kernel length X 1), which will help to detect all the horizontal line from the image.
118     hori_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (kernel_length, 1))
119
120     # A kernel of (3 X 3) ones.
121     kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
122
123     # Morphological operation to detect vertical lines from an image
124     img_temp1 = cv2.erode(img_bin, verticle_kernel, iterations=3)
125     verticle_lines_img = cv2.dilate(img_temp1, verticle_kernel, iterations=3)
126
127     # Morphological operation to detect horizontal lines from an image
128     img_temp2 = cv2.erode(img_bin, hori_kernel, iterations=3)
129     horizontal_lines_img = cv2.dilate(img_temp2, hori_kernel, iterations=3)
130
131     ## Getting rid of grid lines ##
132     h,w=horizontal_lines_img.shape
133     for y in range(h-1):
134         r=horizontal_lines_img[y][0]
135         if r==255:
136             horizontal_lines_img[y]=0
137

```

Figure 13: Source Code of Bar Chart Based Content Identification Part 2

```

137
138 # Weighting parameters, this will decide the quantity of an image to be added to make a new image.
139 alpha = 0.5
140 beta = 1.0 - alpha
141
142 # This function helps to add two image with specific weight parameter to get a third image as summation of two image.
143 img_final_bin = cv2.addWeighted(verticle_lines_img, alpha, horizontal_lines_img, beta, 0.0)
144 img_final_bin = cv2.erode(~img_final_bin, kernel, iterations=2)
145 (thresh, img_final_bin) = cv2.threshold(img_final_bin, 128,255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
146
147 ###get the count
148 r=verticle_lines_img[h-5][:]
149 ll=[]
150 for i in range(len(r)-1):
151     if r[i]==0 and r[i+1]==255:
152         ll.append(i)
153
154 sum1=[]
155 for j in range(len(ll)):
156     for i in range(h-1):
157         r=verticle_lines_img[i][ll[j]+1]
158         r1=verticle_lines_img[i+1][ll[j]+1]
159         if r==0 and r1==255:
160             sum1.append(i)
161             break
162         else:
163             verticle_lines_img[i][ll[j]+1]-127
164
165 values=[]
166 for i in range(len(sum1)-1):
167     r=sum1[i]
168     r1=sum1[i+1]
169     if abs(r1-r)>2:
170         values.append(h-r)
171
172 values.append(h-r1)
173 speech = ' The image contains a bar chart between the range ' + str(minimumValue) + ' and ' + str(maximumValue) + ' with ' +
174

```

Figure 14: Source Code of Bar Chart Based Content Identification Part 3

```

152 ll.append(i)
153
154 sum1=[]
155 for j in range(len(ll)):
156     for i in range(h-1):
157         r=verticle_lines_img[i][ll[j]+1]
158         r1=verticle_lines_img[i+1][ll[j]+1]
159         if r==0 and r1==255:
160             sum1.append(i)
161             break
162         else:
163             verticle_lines_img[i][ll[j]+1]-127
164
165 values=[]
166 for i in range(len(sum1)-1):
167     r=sum1[i]
168     r1=sum1[i+1]
169     if abs(r1-r)>2:
170         values.append(h-r)
171
172 values.append(h-r1)
173 speech = ' The image contains a bar chart between the range ' + str(minimumValue) + ' and ' + str(maximumValue) + ' with ' +
174
175 for index,val in enumerate(values):
176     actual_round((scale/h)*val)
177     speech = speech + "{1}".format(index,actual) + ', '
178
179 speech = speech + " values for each " + labels + " respectively."
180
181 print(speech)
182
183
184 if __name__ == "__main__":
185     main()
186
187
188

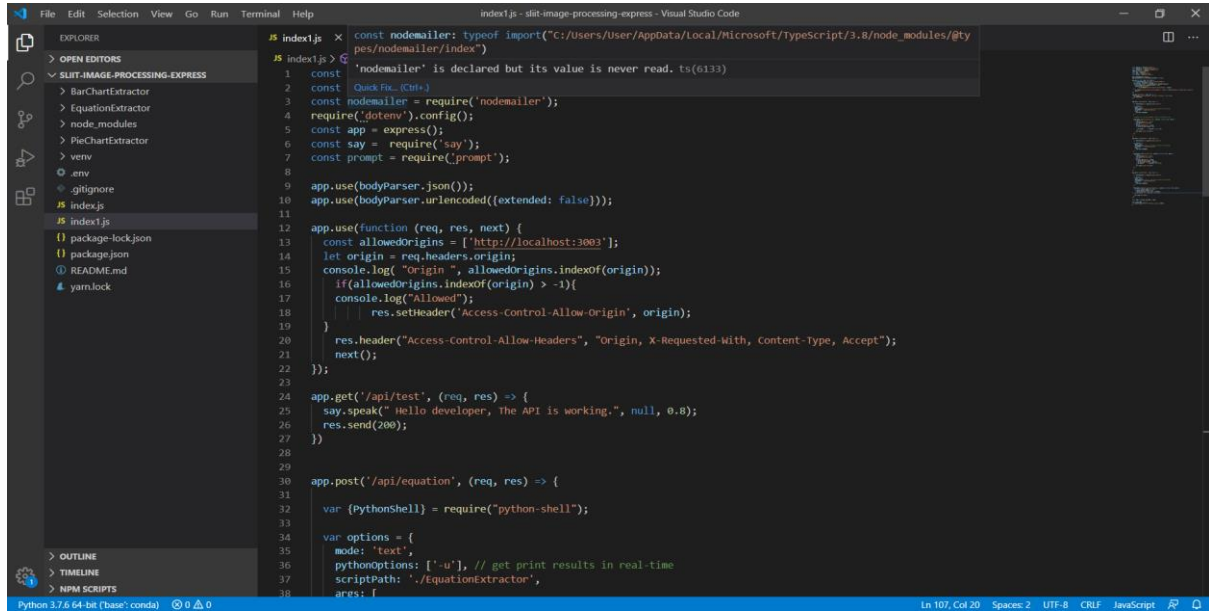
```

Figure 15: Box Detection Algorithm Applied Source Code of Bar Chart

■ Implementation of REST API

The REST API implementation was done by using express java script. It was implemented as for three routing paths for Mathematical equations, pie charts and bar charts to send the POST request.

The below figures depicts the implementation of API to use fast and accurate speech recognition to convert audio.



```
const nodemailer = require('nodemailer');
const dotenv = require('dotenv').config();
const express = require('express');
const say = require('say');
const prompt = require('prompt');

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: false}));

app.use(function (req, res, next) {
  const allowedOrigins = ['http://localhost:3003'];
  let origin = req.headers.origin;
  console.log('Origin', allowedOrigins.indexOf(origin));
  if(allowedOrigins.indexOf(origin) > -1){
    console.log("Allowed");
    res.setHeader('Access-Control-Allow-Origin', origin);
  }
  res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
  next();
});

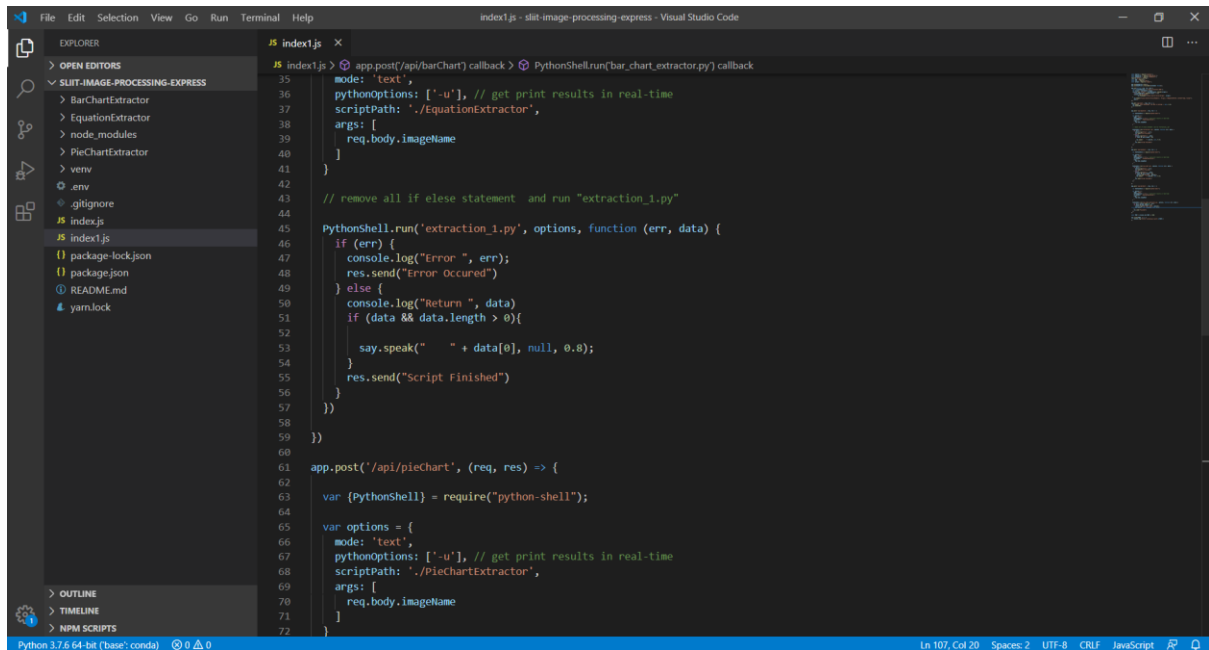
app.get('/api/test', (req, res) => {
  say.speak(" Hello developer, The API is working.", null, 0.8);
  res.send(200);
});

app.post('/api/equation', (req, res) => {
  var {PythonShell} = require("python-shell");

  var options = {
    mode: 'text',
    pythonOptions: ['-u'], // get print results in real-time
    scriptPath: './EquationExtractor',
    args: [

```

Figure 16: Source Code of Implementation of REST API Part 1



```
PythonShell.run('extraction_1.py', options, function (err, data) {
  if (err) {
    console.log("Error ", err);
    res.send("Error Occured")
  } else {
    console.log("Return ", data)
    if (data && data.length > 0){
      say.speak(" " + data[0], null, 0.8);
    }
    res.send("Script Finished")
  }
});

app.post('/api/pieChart', (req, res) => {
  var {PythonShell} = require("python-shell");

  var options = {
    mode: 'text',
    pythonOptions: ['-u'], // get print results in real-time
    scriptPath: './PieChartExtractor',
    args: [
      req.body.imageName
    ]
  }

```

Figure 17: Source Code of Implementation of REST API Part 2

2.4.3 Tools and Technologies

- **Tesseract OCR**



Tesseract is an optical character recognition engine with open-source code, this is the most popular and qualitative OCR-library. It is free software, released under the Apache License. OCR uses artificial intelligence for text search and its recognition on images.

Tesseract is finding templates in pixels, letters, words and sentences. It uses two-step approach that calls adaptive recognition. It requires one data stage for character recognition, then the second stage to fulfil any letters, it wasn't insured in, by letters that can match the word or sentence context.

- **OpenCV**



OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. Most commonly used for Image Processing. It is mainly used to do all the operation related to Images. The library is cross-platform and free for use under the open-source BSD license.

- **Postman**



Postman is a popular API client that makes it easy for developers to create, share, test and document APIs. This is done by allowing users to create and save simple and complex HTTP/s requests, as well as read their responses. The result - more efficient and less tedious work.

3. RESULTS & DISCUSSION

3.1 Results

In this section, allocates to discuss about all the results obtained from the implemented features of the CEBCRD component and accuracy of them.

▪ Accuracy of Mathematical Equations based Content Identification (MEBCI)

50 input images were tested to calculate the average accuracy of the mathematical equations-based content identification function..

- Expected output = 1
- Total number of operators and operands include in the equation = **n**
- Weighted value for one operator or operand (**Wn**) = **1/n**
- Number of accurate operators and operands in the actual output = **An**
- Accuracy of the digitization process (**Ac**) = **Wn x An**
- Maximum accuracy (**Am**) = **100 x execution number**

$\text{Average accuracy (Aa)} = [(\sum Ac) / Am] \times 100$
--

$$\text{Average accuracy (Aa) Equations Identification} = [1200 / (12 \times 100)] \times 100\% = 100\%$$

Below table contains the samples of tested experiment images with the calculation of the accuracy of the function.

Table 4: Accuracy of Mathematical Equation Based Content Identification

Sample experiment image no	Expected output of digitalized text	Actual output of digitalized text	Weighted value for one operator or operand (Wn)	Accuracy of the digitization process (Ac)	Average accuracy (Aa) of the sample experiment
1	19 + 57 = 76	19 + 57 = 76	1/5 = 0.2	0.2x5 = 1	100%
2	16 + 2t = 5t + 9	16 + 2t = 5t + 9	1/9 = 0.112	0.012x9 = 1	100%
3	3x + y + 2z = 8	3x + y + 2z = 8	1/9 = 0.112	0.012x9 = 1	100%
4	Y = mx + c	Y = mx + c	1/6 = 0.167	0.167x6 = 1	100%
5	P = F / A	P = F / A	1/5 = 0.2	0.2x5 = 1	100%
6	S = ut + 1/2at ²	S equals ut plus 1 divided by 2a t square	1/11 = 0.091	0.091x11 = 1	100%
7	S = (v - u) / 2t	S equals open brackets v minus u close brackets divided by 2t	1/10 = 0.1	0.1x10 = 1	100%
8	Y = mx + f - c (x - v) / 2	Y equals mx plus f minus c open brackets x minus v close brackets divided by 2	1/15 = 0.067	0.067x15 = 1	100%
9	a ² - b ² = (a + b) (a - b)	A square minus b square equals open brackets a plus b close brackets open brackets a minus b close brackets	1/16 = 0.0625	0.0625x16 = 1	100%
10	3 sin 30 + cos 15 = 1	3 sin 30 + cos 15 = 1	1/8 = 0.125	0.125x8 = 1	100%
11	sin A cos B + sin B cos A = 0	sin A cos B + sin B cos A = 0	1/11 = 0.091	0.091x11 = 1	100%
12	sin A / cos A = tan A	sin A / cos A = tan A	1/8 = 0.125	0.125x8 = 1	100%
Average accuracy (Aa) = [1200 / (12x100)] x 100 % = 100%					

Sample output result of Mathematical Equations based Content Identification (MEBCI) represents as follows.

```

Reading Image Velo4.jpg
Text Array [
  'S', '=', '(', 'v', '-', 'u', ')', '/', '2', 't'
]
Generating speech ... S
Generating speech ...
Generating speech ... =
Generating speech ...
Generating speech ... (
Generating speech ... v
Generating speech ... -
Generating speech ... u
Generating speech ... )
Generating speech ...
Generating speech ... /
Generating speech ...
Generating speech ... 2
Generating speech ... t
S = Open Brackets v minus u Close Brackets divided by 2 t

```

Figure 18: Sample Result of MEBCI

▪ Accuracy of Chart based Content Identification (CBCI)

Charts -based content identification function results were generated using 40 samples of input images data as 20 pie charts and 20 bar charts. The compound function of chart detection including pie charts detection and bar chart detection shows it average accuracy as 92.5%.

Calculation is based as follows.

<p>Average Accuracy = [Correct occurrences of input data sample / Total input data sample] x100% (Aa)</p>
--

Aa Pie charts = [18/20] x 100% = 90%

Aa Bar chart = [19/20] x 100% = 95%

Average accuracy (Aa) chart Identification = [(90+95) / 2] x 100% = 92.5%

Sample output result of pie chart identification and bar chart identification is display as follows.

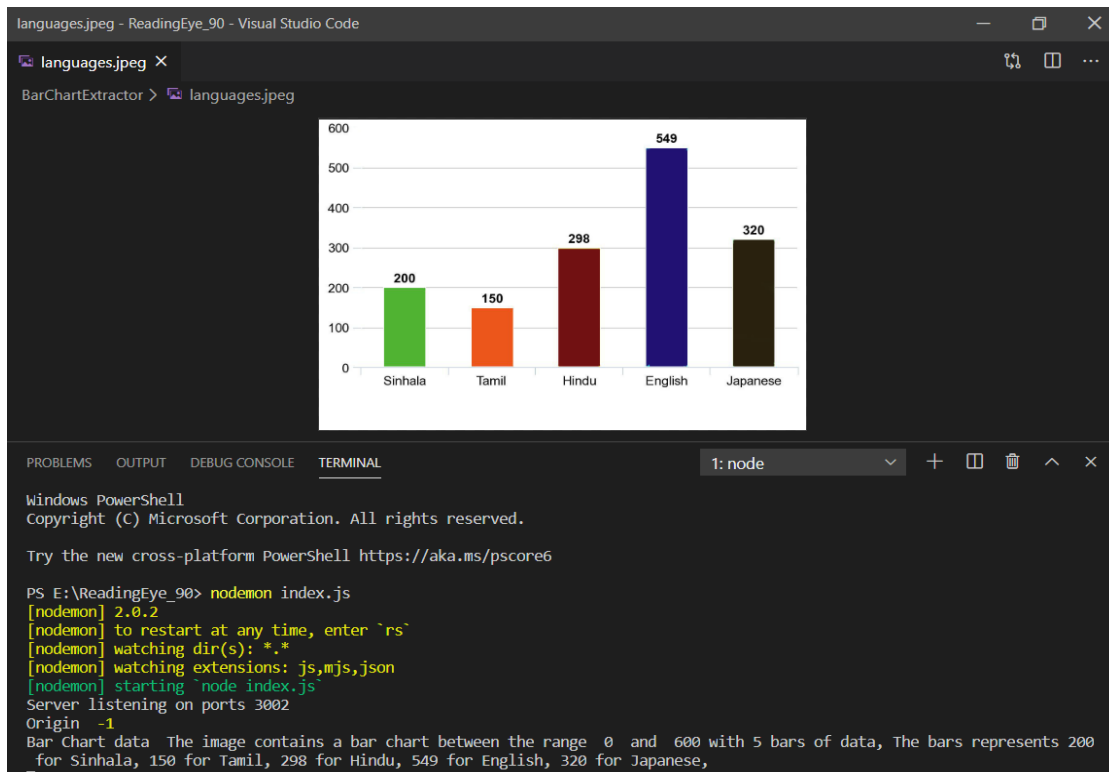


Figure 19: Sample Result of Bar Chart Identification

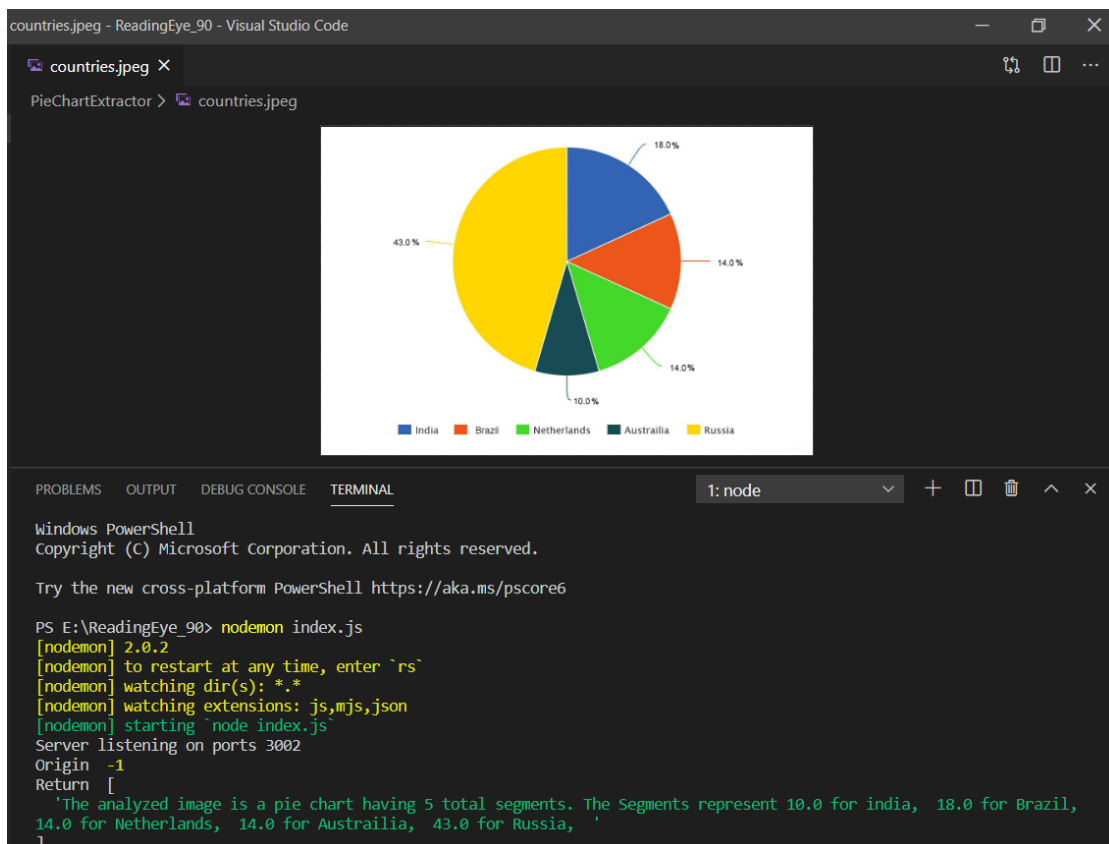


Figure 20: Sample Result of Pie Chart Identification

■ API Requests Sending Sample Results

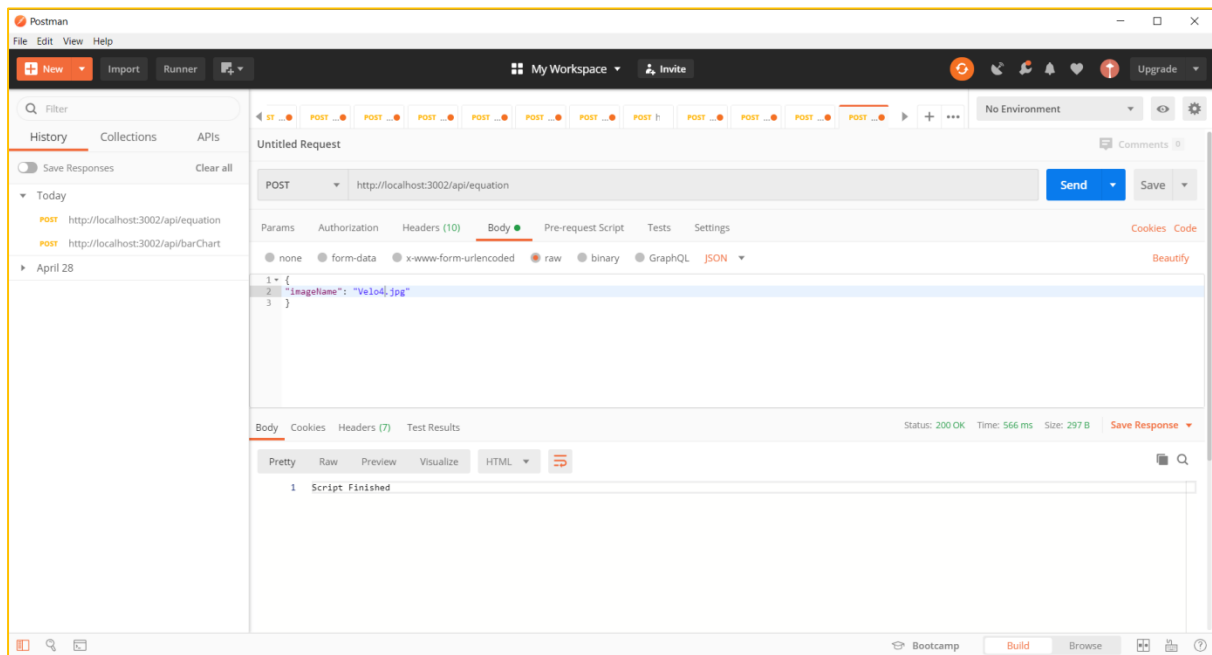


Figure 21: API Request to Get the Sample Result of MEBCI

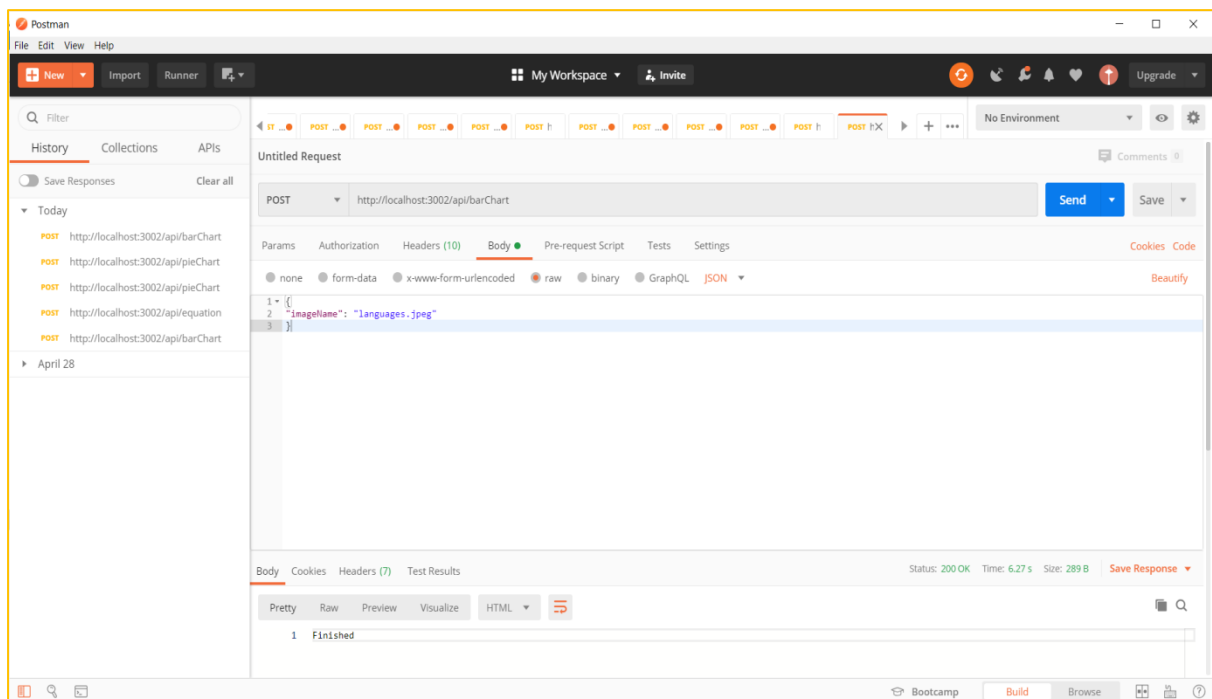


Figure 22: API Request to Get the Sample Result of Bar Chart

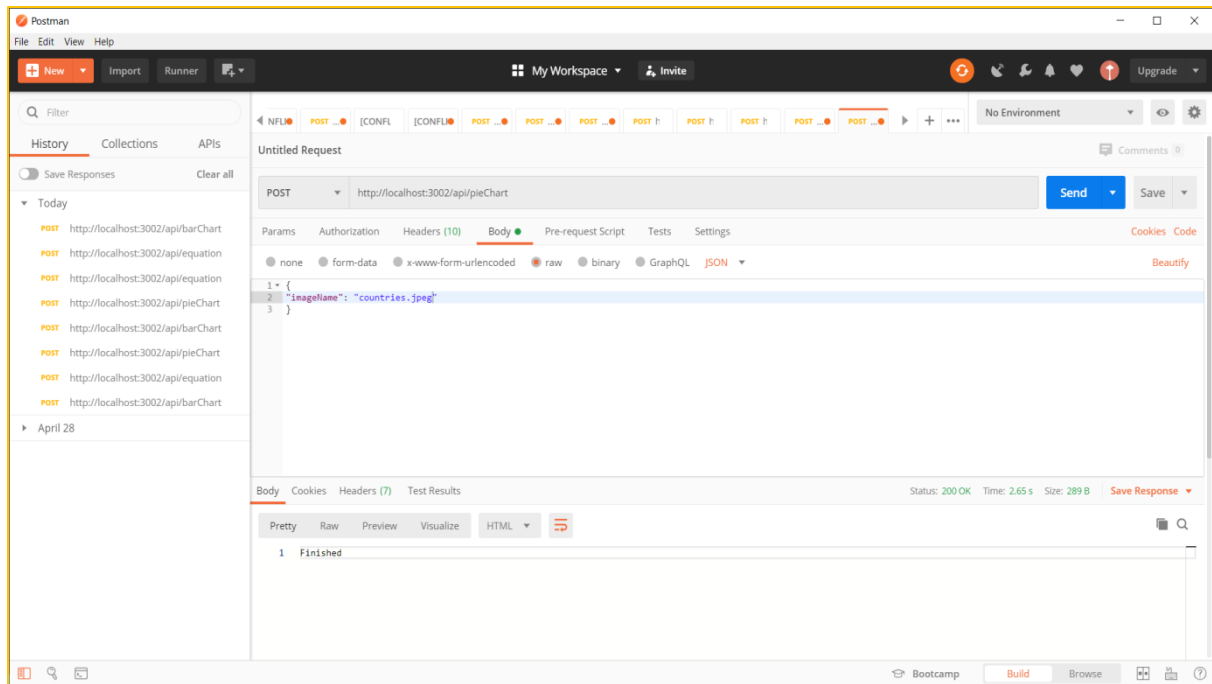


Figure 23: API Request to Get the Sample Result of Pie Chart

3.2 Research Findings

According to the previous researches on this domain, there is no any other developed system to identify all chart based contents, Image based contents, Mathematical equations based contents, Tables based contents and Text based contents with descriptive audio generation. Moreover, there were unable to find any similar mobile application with previously mentioned identifications.

The most significant findings of the research project were all the developed main functions worked with more than 90% of accuracy level. Therefore, It can be clearly seen that the final developed Reading-Eye system application is perform well and it will be major benefit for visually impaired users to get the knowledge of reading complex documents by the use of this solution in future.

3.3 Discussion

When highlighting the charts and equations based contents reading and detecting (CEBCRD) component,

Mathematical Equations based Content Identification (MEBCI) function performs well as providing 100% of accuracy with identify all mathematical operators in basic linear equations and basic trigonometric equations. Moreover this identification generates clear, descriptive detailed content of the identified mathematical equation as audio.

Furthermore, Chart based Content Identification (CBCI) function provides 92.5% of decent accuracy score as overall and for the bar chart identification it is 95% and for the pie chart identification it provides 90% of accuracy value.

Moreover, in bar charts clearly identify the X axis, Y axis values and labels with the title of the charts correctly. Pie charts also identify the legends and the segmentations of the charts with percentages and the values more accurately.

Therefore it clearly depicts in the results section the outcome of the implemented CEBCRD component performed well in the Reading-Eye system.

3.4 Summary of Student Contribution

Table 5: Summary of Student Contribution

Student ID	Student Name	Work Allocation
IT16165762	P.S.N.Kularathne Analyzing and Describing Charts and the Mathematical Equations based contents in the Documents	<ol style="list-style-type: none"> 1. Identify the charts, mathematical equations in the uploaded digital photos of the document. 2. In charts - Identify the type of the charts as, <ul style="list-style-type: none"> • Bar charts • Pie charts etc. 3. Represent the relevant data of the charts accordingly by using machine learning and image processing techniques. 4. In equations - Identify numbers, variables, and mathematical operators in, <ul style="list-style-type: none"> • Linear equations • Basic trigonometric equations Accordingly using image processing techniques. 5. Create a detailed description of the identified content of charts and equations. 6. Generate an audio file of that detailed description for the user to listening purpose by implementing REST API service.

4. CONCLUSION AND FUTURE WORKS

In this research, we proposed a solution that is capable of reading images, charts, equations and tables in a printed document using image processing, deep learning and natural language processing. The process consists of region detection and separation, secure encryption and decryption of the original document images, region analysis and content reading, conversion to textual description of the read content and lastly conversion the description to the audio narration. The system was developed mainly to help visually impaired persons to read documents easily and without someone else's support. As a further benefit, it limits the need to use Braille formatted books as it can read many types of graphical contents in a document. With all these features, Reading Eye will provide better solution for visually impaired people to gain to read the document contents of information and fulfill their knowledge in better way.

As for future works, we propose the following suggestions mentioned below,

- Implement a feature to read tables that are having no horizontal and vertical border lines.
- Extend the chart reading function to read other chart types such as line charts, area charts and scatter plots.
- Include a feature to the existing system that can analyze graphs.
- Develop an iOS application to attract more users.

5. REFERENCES

- [1] Who.int. (2018). GLOBAL DATA ON VISUAL IMPAIRMENTS 2010. [online] Available at: <https://www.who.int/blindness/GLOBALDATAFINALforweb.pdf> [Accessed: 06-Aug-2019].
- [2] World Health Organization, 15-Jan-2019. [Online]. Available: <https://www.who.int/blindness/en/>. [Accessed: 06-Aug-2019].
- [3] "Assistive technology", En.wikipedia.org. [Online]. Available: https://en.wikipedia.org/wiki/Assistive_technology#Visual_impairments. [Accessed: 16-Aug-2019]
- [4] A. Hellem, "Resources for the Visually Impaired," All About Vision. [Online]. Available: <https://www.allaboutvision.com/lowvision/resources.htm> [Accessed: 06-Aug-2019].
- [5] "Canny edge detector", En.wikipedia.org, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Canny_edge_detector [Accessed: 20-Aug-2019].
- [6] N. Gamage, K. Jayadewa, S. Senanayake, K. Udesitha and A. Jayakody, "Audio Assistance for Vision Impaired Individual To Recognize Graphical Content on Print Disable Documents", Annual conference 2017 - IET - Sri Lanka, 2017. [Online]. Available: <http://theiet.lk/wp-content/uploads/2017/10/24-p9.pdf.%20%5bAccessed:%2023-%20Aug-%202019%5d>
- [7] "A Simple Equation Region Detector for Printed Document Images in Tesseract - IEEE Conference Publication", Ieeexplore.ieee.org, 2019. [Online]. Available: <https://ieeexplore.ieee.org/iel7/6627713/6628563/06628621.pdf> [Accessed: 22-Aug-2019].
- [8] "Accessible Mobile Apps | American Foundation for the Blind", Afb.org, 2019. [Online]. Available: <https://www.afb.org/blindness-and-low-vision/using-technology/assistive-technology-products/mobile-apps> [Accessed: 20-Aug-2019].
- [9] "An In-depth Evaluation of the BARD Mobile App from the National Library Service for the Blind and Physically Handicapped | AccessWorld | American Foundation for the Blind", Afb.org. [Online]. Available: <https://www.afb.org/aw/15/2/15607> [Accessed: 20-Aug-2019].
- [10] Play.google.com. [Online]. Available: <https://play.google.com/store>. [Accessed: 21-Aug-2019].
- [11] "Read text aloud. PDF reading books speak text to voice, read it out loud", Captivoice.com.15 [Online]. Available: <https://www.captivoice.com/capti-site/public/entry/explore>. [Accessed: 22-Aug-2019].
- [12] J. Amara, P. Kaur, M. Owonibi and B. Bouaziz, "Convolutional Neural Network Based Chart Image Classification..", Pdfs.semanticscholar.org, 2019. [Online]. Available: <https://pdfs.semanticscholar.org/5fe7/c360e851f648095bb630eec742c7b809cf2b.pdf>. [Accessed: 23-Aug-2019].

- [13] "A Simple Equation Region Detector for Printed Document Images in Tesseract - IEEE ConferencePublication", Ieeexplore.ieee.org, 2019. [Online]. Available: <https://ieeexplore.ieee.org/iel7/6627713/6628563/06628621.pdf>. [Accessed: 02- Sep- 2019].
- [14] J. Amara, P. Kaur, M. Owonibi and B. Bouaziz, "Convolutional Neural Network Based Chart Image Classification." Pdfs.semanticscholar.org, 2019. [Online]. Available: <https://pdfs.semanticscholar.org/5fe7/c360e851f648095bb630eec742c7b809cf2b.pdf>. [Accessed: 23- Aug- 2019].
- [15] Nelli, F. (2017). OpenCV & Python - The Otsu's Binarization for thresholding. [online] Meccanismo Complesso. Available at: <https://www.meccanismocomplesso.org/en/opencv-python-the-otsus-binarization-for-thresholding/> [Accessed 1 Mar. 2020].
- [16] "Morphological Image Processing", Cs.auckland.ac.nz, 2020. [Online]. Available: <https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm>. [Accessed: 13- Jan- 2020].
- [17] Scikit-image.org. (n.d.). Circular and Elliptical Hough Transforms — skimage v0.12.2 docs. [online] Available at: https://scikit-image.org/docs/0.12.x/auto_examples/edges/plot_circular_elliptical_hough_transform.html#algorithm-overview [Accessed 2 Feb. 2020]
- [18] Scikit-image.org. (n.d.). Circular and Elliptical Hough Transforms — skimage v0.12.2 docs. [online] Available at: https://scikit-image.org/docs/0.12.x/auto_examples/edges/plot_circular_elliptical_hough_transform.html [Accessed 2 Feb. 2020].
- [19] Scikit-learn.org. (n.d.). scikit-learn: machine learning in Python — scikit-learn 0.22.2 documentation. [online] Available at: <https://scikit-learn.org/stable/> [Accessed 3 Feb. 2020].
- [20] Medium. (n.d.). A Box detection algorithm for any image containing boxes.. [online] Available at: <https://medium.com/coinmonks/a-box-detection-algorithm-for-any-image-containing-boxes-756c15d7ed26> [Accessed 4 Feb. 2019].

6. APPENDICES

Appendix A - Use Case Diagram of CEBCRD Component

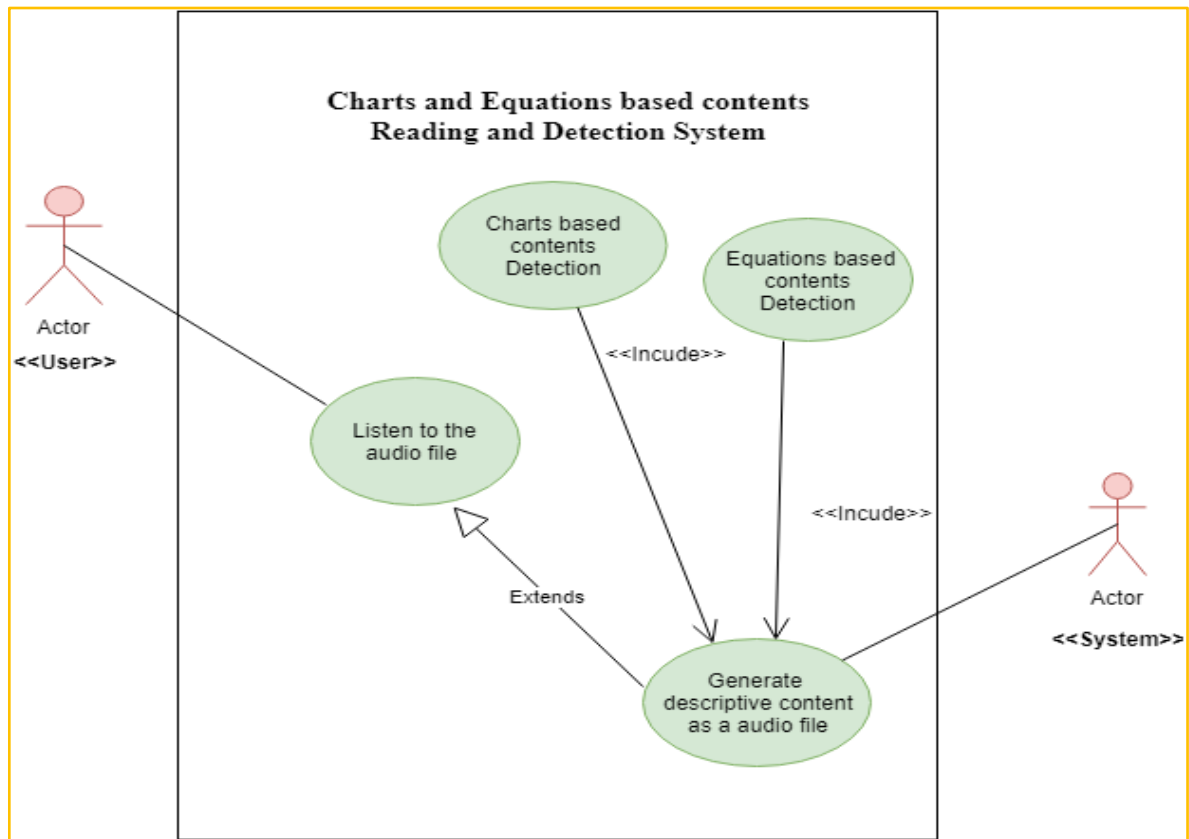


Figure 24: Use Case Diagram of CEBCRD Component

Appendix B - Usecase Scenarios of CEBCRD Component

Table 6: Use case Scenario 1

Use case ID	CE_UC1
Use case Name	Retrieving files from the folders
Usecase Description	The system has to retrieve Charts and Equations from the respective folders.
Primary actors	System
Pre-conditions	1. User should capture the photograph. 2. The system should be connected to cloud services.
Main flow	1. The system should check the repository where the files are saved. 2. System checks Charts and Equations folders. 3. The system returns the relevant files for analyzing. 4. End of the Use case.

Table 7: Use case Scenario 2

Use case ID	CE_UC2
Use case Name	Charts based on contents detection.
Usecase Description	The system detects objects and areas in the chart based image content.
Primary actors	System
Pre-conditions	<ol style="list-style-type: none"> 1. The user should capture the photograph. 2. The system should be connected to cloud services. 3. The system should previously retrieve Charts from the respective folder.
Main flow	<ol style="list-style-type: none"> 1. The system analyzes the chart based image content. 2. The system compares the chart based image content with the data sets for object detection. 3. The system detects objects (size, type, etc) and areas separately. 4. The system saves the identified objects and areas temporary. 5. End of the Usecase.
Post-conditions	The chart based content output will be saved as a .txt file format for text generation.

Table 8: Use case Scenario 3

Use case ID	CE_UC3
Use case Name	Equations based on contents detection.
Usecase Description	The system detects objects and areas in the equations based image content.
Primary actors	System
Pre-conditions	<ol style="list-style-type: none"> 1. The user should capture the photograph. 2. The system should be connected to cloud services. 3. The system should previously retrieve Equations from the respective folder.
Main flow	<ol style="list-style-type: none"> 1. The system analyzes the equations based image content. 2. The system compares the equations based image content with the data sets for object detection. 3. The system detects objects (numbers, operators, characters, etc) and areas separately. 4. The system saves the identified objects and areas temporary. 5. End of the Usecase.
Post-conditions	The equations based content output will be saved as a .txt file format for text generation.

Table 9: Use case Scenario 4

Use case ID	CE_UC4
Use case Name	Generate Descriptions
Usecase Description	The system generates detailed descriptions for chart based and equations based contents.
Primary actors	System.
Pre-conditions	<ol style="list-style-type: none"> 1. The user should capture the photograph. 2. The system should be connected to cloud services. 3. The system should previously retrieve Charts and Equations from the respective folders. 4. The system should previously detect the objects of Charts and Equations.
Main flow	<ol style="list-style-type: none"> 1. The system retrieves temporary text files that were saved earlier when the object detection. 2. A detailed description will be generated by taking text files as the input data by using NLP techniques. 3. The generated detailed description of files will be saved in the JSON file format. 4. End of the Usecase.

Table 10: Use case Scenario 5

Use case ID	CE_UC5
Use case Name	Generate Audio Files
Usecase Description	The system generates audio files by using the generated description of files.
Primary actors	System
Pre-conditions	<ol style="list-style-type: none"> 1. The user should capture the photograph. 2. The system should be connected to cloud services. 3. The system should previously retrieve Charts and Equations from the respective folders. 4. The system should previously detect the objects of Charts and Equations. 5. The system should previously generate a detailed description of the files.
Main flow	<ol style="list-style-type: none"> 1. The system concatenates all the JSON files that were generated earlier. 2. Using the Microsoft Azure text to speech API generates an audio file. 3. End of the Usecase.

Table 11: Use case Scenario 6

Use case ID	CE_UC6
Use case Name	Listen to the audio file.
Usecase Description	The user will listen to the audio file narration.
Primary actors	User
Pre-conditions	1. User should turn on the mobile data.
Main flow	1. The Usecase starts when the user captures the photograph of the document. 2. After processing the system will generate an audio file. 3. The user will listen to the audio narration. 4. End of the Usecase.
Post-conditions	The app will be ready to play the audio.

Appendix C - Class Diagram of CEBCRD Component

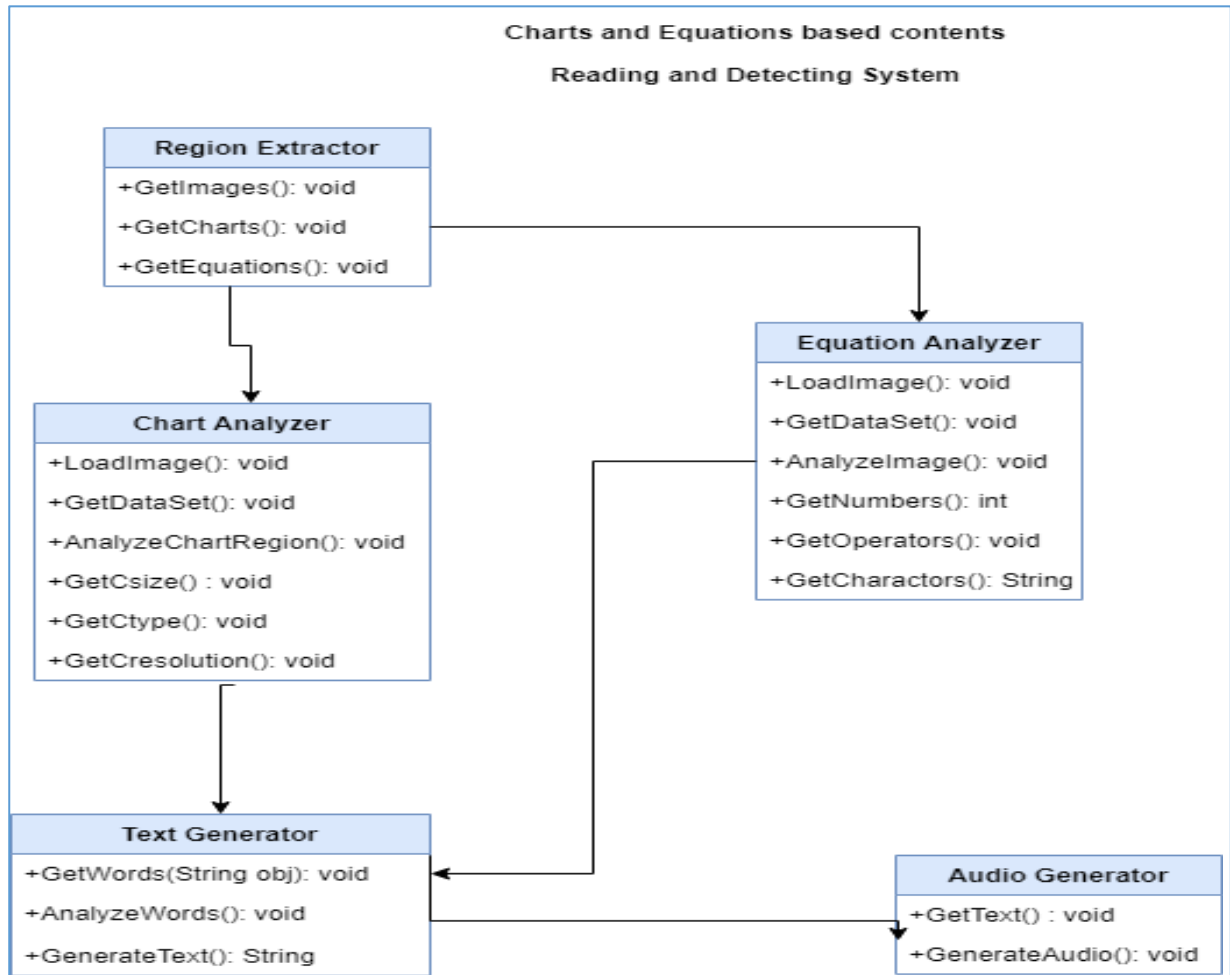


Figure 25: Class Diagram of CEBCRD Component

Appendix D - Works Breakdown Structure (WBS) of Reading-Eye System

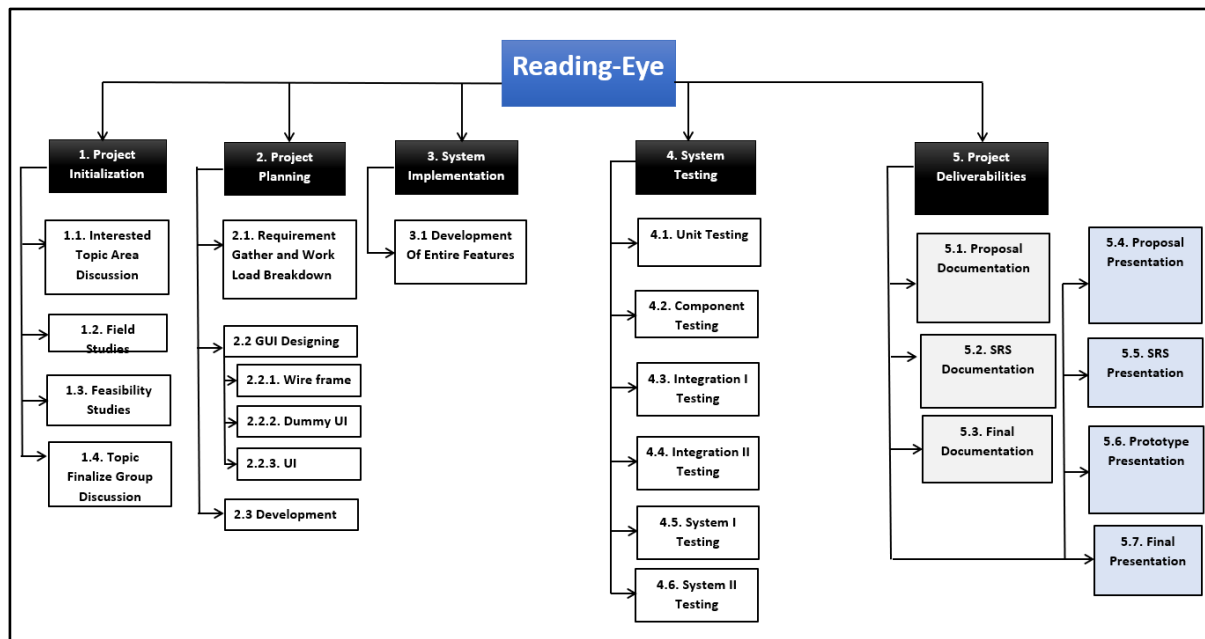


Figure 26: Work Breakdown Structure Diagram