Bilkent University

Department of Computer Engineering

CS 492 Senior Design Project

Final Report

Sum of Sounds

Team members

Aliyu Vandana Saifullah

Bilal Siraj

Kasymbek Tashbaev

Supervisor

Prof. Dr. Uğur Doğrusöz

Jury Members

Prof. Dr. Varol Akman

Prof. Dr. Özcan Öztürk

May 25, 2020

Table of contents

1. Introduction

Nowadays speech synthesis technologies are very popular and there are lots of text-to-speech applications on the market that read books, articles, docs in an almost human voice. Some of these applications are Intelligent Speaker and Natural Reader that read plain texts without any problems, but when there are mathematical formulas in the text almost all of such text-to-speech applications have trouble with voicing the formulas. For example, when Natural Reader meets mathematical formulas it just ignores them and continues reading the next plain text [1].

However, there are people with visual impairments who do research in mathematics and need to read scientific articles and books, which usually contain lots of formulas and cannot be read by these applications [2]. This motivated our team to develop an application that could voice the mathematical formulas and equations. Such a project can be useful not only for people with vision problems but also for developers of text-to-speech applications, who can integrate our application to theirs.

2. Requirements Details

2.1 Functional Requirements

- The application can read formulas from documents or images.
- The user can upload files in PDF, images in png and jpg formats.
- The user can enter the source LaTeX into a text box on the web app to be voiced rather than uploading a file.
- The user can control the audio playback using play, pause and stop buttons.
- The user can choose between two reading voices, a male or female.
- The user can adjust the reading pace to suit their requirements.
- The user can download the audio output as an MP3 file.

2.2 Non-Functional Requirements

2.2.1 Usability

The application has a user-friendly interface that allows users to navigate through it easily without difficulty. The user interface is not flooded with a lot of functionalities that may overwhelm the user.

2.2.2 Extensibility

The application is made such that new features can be added easily as there is always room for improvement. Our design ensures that minimal changes can be added to the application. These changes will mostly be about applying new technologies rather than changing the functionality of the app.

2.2.3 Reliability

The application does not alter any files or images uploaded to voice out formulas. The application will not store users' information or track their activities in the browser.

2.2.4 Portability

SumOfSound is implemented as a web-based application and can be accessed by anyone with access to the URL. It maintains the same outlook regardless of the browser or operating system.

2.2.5 Efficiency

In order to have optimized run-time performance, our application has a threshold of one page that can be read when the user uploads a pdf file. The text part of the pdf could have been eliminated completely but we realize reading some text before or after the formula could give more context to the formula.

3. Final Architecture and Design Details

3.1. Architecture Style

SumOfSounds follows a Client/Server software architecture style, in which the server delivers the data to be displayed by the client [3]. So, by following this architecture, our program can support concurrent requests of many clients without having any conflicts.

3.2. Client

The Client package contains six classes. These classes are responsible for the graphical user interface and user experience. App class will be the main class, thereby it will have instances of all other client classes and will communicate with the server package.

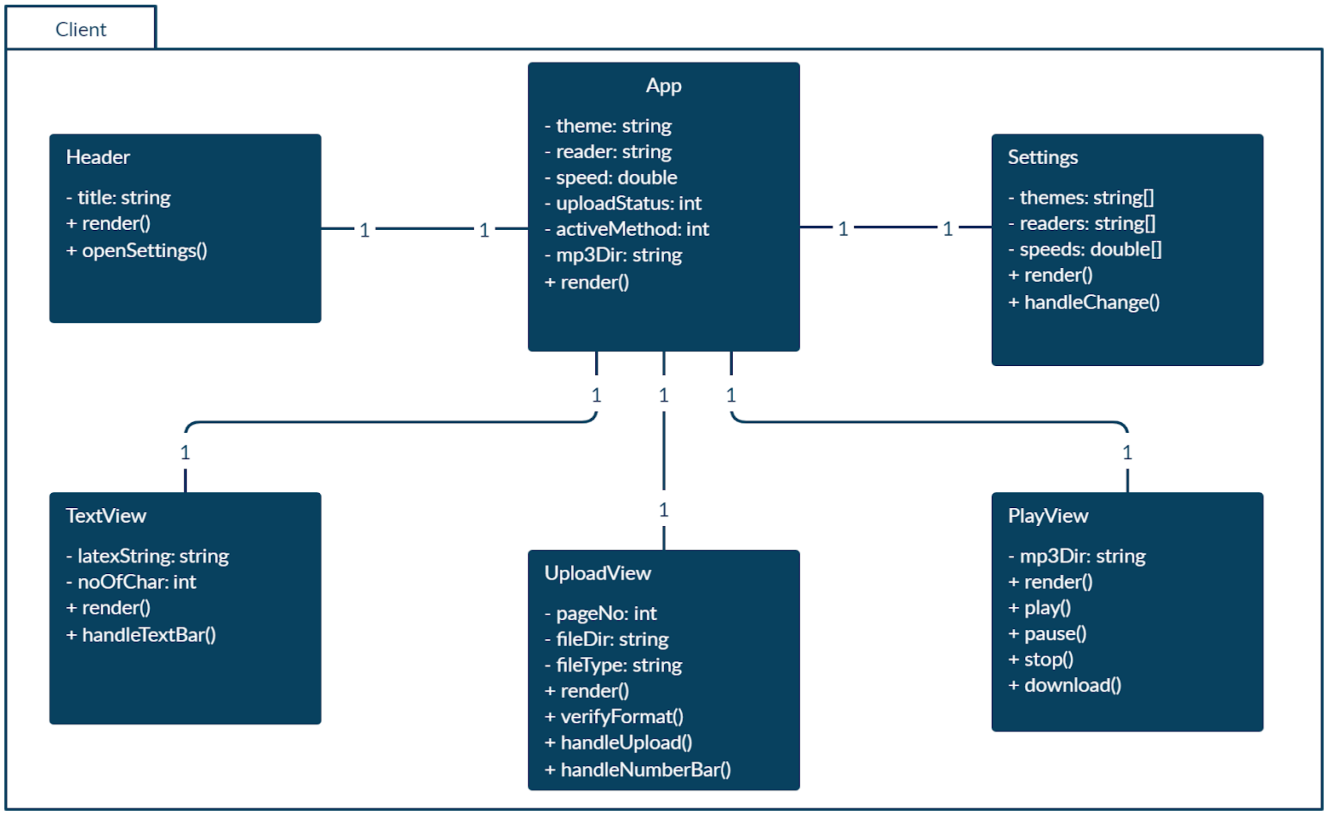

Figure 1. Class diagram of the Web Client

PlayView: Provides the UI elements and functions that allow users to play, pause, stop and download the sound file.

TextView: Provides a text bar, where users can enter a LaTeX string, and the function to handle the text bar.

UploadView: Provides an upload screen, where the user can upload the file, and the functions to handle the upload.

Header: Displays the title and provides a button, which opens the Settings page, and functions to handle this operation.

Settings: Provides a new screen, where the user can change the theme and set the speed of mp3 play.

App: Contains all other frontend components and provides communication between them. Moreover, it communicates with the server package by using the API provided by the server.

3.3. Server

The Server package contains four classes, which are responsible for generating mp3 files from the uploaded image, PDF, or LaTeX files. The Server class will be a Facade class, so it will provide an API to the client.

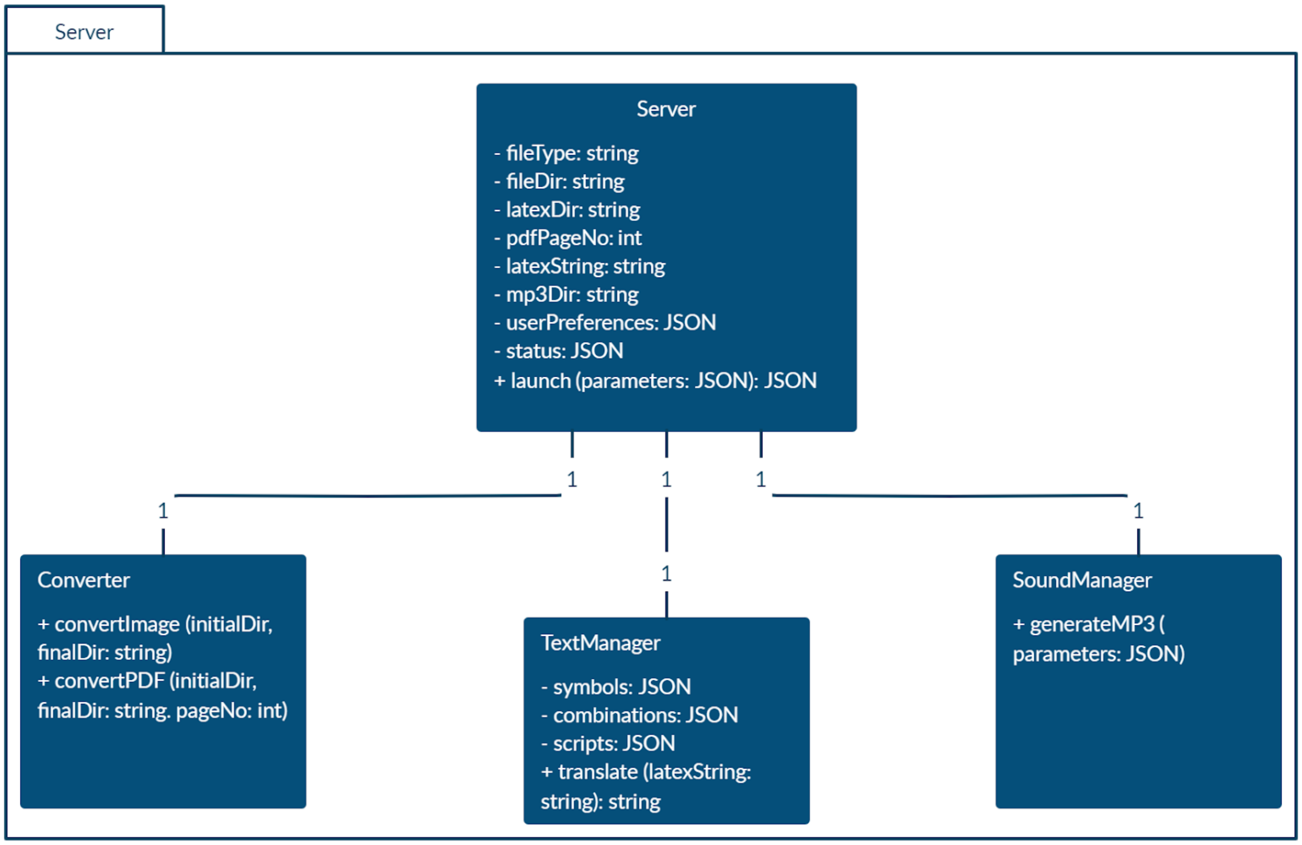

Figure 2. Class diagram of the Web Server

Converter: Manages the conversion of uploaded files to LaTeX. If the uploaded file is PDF, firstly it converts the chosen page to the image, then by using MathPix API converts the image to LaTeX.

TextManager: Manages translation of LaTeX script to plain text. The class contains JSON files with symbols, scripts, and their transcription. By using this data, it generates plain text from the LaTeX script. Moreover, our system is context-dependent, so according to context, a symbol can be translated differently.

SoundManager: Manages the creation of an mp3 file from plain text. It gives the user preference settings and the plain text as a parameter to Google text-to-speech API and gets an mp3 file.

Server: Contains a single function that runs other classes' functions in the right sequence. In addition, it provides simple APIs to the client application.

4. Development/Implementation Details

4.1. Tools and Technologies Used

Node.js

Node.js is a JavaScript runtime environment that lets developers use JavaScript to write command-line tools and for server-side scripting to produce dynamic web page content [4]. It was used for implementing the server-side of SumOfSounds.

React

React is a JavaScript library for building user interfaces. It was used for implementing the client-side of SumOfSounds. Node.js and React unify web-application development around a single programming language. So, we could use only one programming language for the whole application.

npm

npm is a package manager for the JavaScript programming language and default package manager for Node.js. It contains multiple packages, which were used for both client and server development.

Semantic UI

Semantic UI is a modern front-end development framework that provides multiple UI components, such as menu, text bar, form, etc. [5]. The framework was used for client development.

Google Cloud

Google Cloud is a platform that provides Google APIs and many other services. In our application, Google Cloud Text-to-Speech was used to convert the text to an audio file.

MathpixOCR API

The MathpixOCR API is a JSON API for extracting text from images. It has first-class support for scientific notation [6]. This API is used on the client-side to convert images to the LaTeX script.

Github

GitHub is a web-based hosting service for version control using Git. We used it for version control and to host our source code [7].

Heroku

Heroku is a cloud platform as a service (PaaS) based on a managed container system for deploying and running modern apps [8]. Heroku supports Node.js and React.js and was used to deploy our application.

4.2. Client Implementation

The client application is implemented in JavaScript programming language using React libraries and some libraries provided by npm, as well as Semantic UI framework. The application has the main page and five big components: header, player, settings, text view, and upload view.

Header

Header is the top section of the web page. It is implemented using the Menu component from Semantic UI. The header only displays the name of the application and provides one button that switches to the settings page.

Player

Player is the component that is responsible for playing the audio file and providing a download function. It is implemented using the AudioPlayer component from the "react-h5-audio-player" library provided by npm.

Text view

Text view is responsible for providing a text area where users can enter LaTeX string or plain text, which will be converted to audio. It uses three Semantic UI components: Message displays instruction for use, Form provides text area, Button to submit the text.

Upload view

Upload view is responsible for providing an input form where users can upload a document, which will be converted to audio. It uses four Semantic UI components: Message, Button, Form provides upload bar, Input provides a field to read page number.

Settings page

The settings page lets the user change the speed of audio, the speaker, and the theme of the website. It has a Slider component to adjust speed and two Dropdown components to select speaker and theme, as well as Button components to save or discard changes.

App page

This is the main page and contains all components above. It communicates with the server using APIs provided by the server. When a file is uploaded it sends a post request to "/api/generate" with all inputs and sends the get request to "/output.mp3" to get the audio file.

4.3. Server Implementation

The server application is implemented in JavaScript programming language on Node.js runtime environment using libraries provided by npm.

The main goal of the server application is to extract the conceptual information of formulae in image or pdf file and transform it into an audio file. The application contains four steps: two main and two optional steps.

Step 0: converting pdf to image

If a user uploaded a pdf file, he or she is required to enter a specific page of pdf as well. So in this step, the program converts the page of the given pdf to an image file. This is done in the Converter class using npm package "pdf2pic", which is a utility for converting pdf to image and base64 format [4]. This is an optional step and only executed if a pdf file is uploaded.

Step 1: converting an image to LaTeX

The next step is to convert the image of the math equation to a format of markup language, LaTeX. This step is done in the Converter class using the MathpixOCR API. The API converts simple math equations that have integrals or limits, with 100% confidence, however, for complex equations that have matrices, it may generate a script that is hardly readable. This is an optional step and only executed if a pdf or image file is uploaded.

Step 2: converting LaTeX to plain text

The main step of the application is to convert the LaTeX script to the plain text. This is handled in the TextManager class that uses three different JSON files, which have over 200 LaTeX commands and over 30 math expressions. TextManager firstly removes from LaTeX script the commands that give only visual information but no conceptual information, then it tokenizes the script, lastly, it replaces the LaTeX commands and expressions with readable text format. The algorithm is context-dependent, so some commands depending on other commands nearby may be translated differently. For example in LaTeX form, the character '_' is usually used for presenting "index" meaning, however, if this character appears in the expression containing "\sum", "\prod" and "\int" commands it would mean "from", with "\log" it would symbolize "to the base of". Also the '^' character, which generally means "to the power of", however when it is used after "\sum", "\prod", and "\int", it means "to".

Math formula	**LaTeX**	**After conversion**
$a_1 + a_1^2$	a_1 + a_1^2	a index 1 plus a index 1 power 2
$\int_a^b \! f(x) \, \mathrm{d}x.$	\int_a^b \! f(x) \, \mathrm{d}x.	integral FROM a TO b OF f of x d x.
$\log_{10} 3x^3$	\log _{10} 3x^3	LOGARITHM BASE 10 of 3x power 3

Table 1. Formula conversion

In some cases, recognizing the exact concept of a formula can be difficult. As shown in Table 2, the output of a simple linear equation could be ambiguous. Our algorithm takes this into account and puts brackets at the right place to make the output unambiguous.

Math formula	**LaTeX**	**After conversion**
$\frac{a}{b} + 1$	\frac{a}{b} + 1	a OVER b plus 1

$\frac{a}{b+1}$	\frac{a}{b+1}	a OVER open bracket b plus 1 close bracket
$\frac{\frac{a}{b+1}}{b-1}$	\frac{\frac{a}{b+1}}{b-1}	open bracket a OVER open bracket b plus 1 close bracket close bracket OVER open bracket b minus 1 close bracket

Table 2. Formula conversion

The result of output produced by the TextManager class depends on how much of the document's logical structure is recognized and converted to the markup language. Therefore, if the LaTeX script contains commands that are not recognized by the application, TextManager may generate a text, which cannot be fully understandable by humans.

Step 3: converting plain text to audio

The last step is to convert the plain text generated by the TextManager to the audio file. This is handled by the SoundManager class using Google text-to-speech API.

All steps together

The Server class is responsible for establishing a connection between all other server classes, as well as connections between client and server. Firstly, it gets input from the client and if it is a pdf or image file, Server calls Converter class, passes pdf or image file as a parameter, and gets the LaTeX script. Then, it calls the TextManager class, passes LaTeX script as a parameter, and gets a plain text. Lastly, it calls SoundManager class, passes plain text, and gets an audio file, which is later returned to the client.

5. Testing Details

5.1. Unit Testing

We did not do automated tests but manually tested the software. We did unit testing on three classes on the server-side.

TextManager

We found about 30 different LaTeX scripts on the internet, each of which contains different LaTeX commands and expressions. We run the TextManger passing these scripts as input and if the output is unsatisfactory, we change the part of code that causes the problem. If our program could not recognize given commands and expressions, we added them to the program.

SoundManager

We used plain texts produced by TextManager and some text from the internet, which does not necessarily contain math elements, to test the SoundManager. These texts were given as input to the class and we check whether the audio file generated is correct or not. In addition, we used different parameters: sound speed and reader, to test the SoundManager.

Converter

We found about 10 different pictures with math formulas on the internet and 3 pdf files, which contain multiple math equations. Firstly, we tested the function that handles pdf to image conversion using pdf files we found. When we were sure that the function was working properly, we started testing the other function which is responsible for an image to LaTeX conversion. For some test inputs, the function was returning an error that it could not read the image, however since we used a third-party API, it could not affect it. So for some files, our software may display a message that it could read files.

5.2. System Testing

After the implementation of the client, we manually tested the UI. We checked whether the message components were displaying the right messages, as well as when we clicked on the buttons whether they were doing the right operation. Then after finishing the implementation of the server, we tested the whole application by uploading different files and checking whether the generated audio file was correct.

6. Maintenance Plan and Details

As part of our preventive maintenance after before we roll out the product, we will let random users utilize the functionality of our system in edge cases to see how our system responds. Combinations of user actions that may lead to the error will be corrected before the app is available on the market. For the adaptive maintenance of our system we will constantly update the system by considering the user feedback and the software market. These may include the addition of new features, improving the UI, and integrating with different API that may be much better than the ones we use now. In case another product wants to integrate with ours, the required modification will be done to ensure proper integration. We will allocate a budget that will sustain the support of APIs we are currently using.

7. Other Project Elements

7.1.Consideration of Various Factors

- We are using paid third-party APIs, so their costs affect our project, however, APIs are cheap, so the effect of economic factors is not high.
- The design of the system is driven by a social need, the need for better accessibility tools, in this case, text to speech. The design of the user interface and user experience, such as allowing users to choose a reading voice and customize the display, caters to the users' personal preferences.

Factors	**Effect level (out of 10)**	**Effect**
Public health	0	-
Safety	0	-
Welfare	0	-

Global	2	People that read English documents are the target users
Social	9	Personal preferences of the users affect the design
Environmental	0	-
Economic	2	The price of APIs

Table 3. Factors that can affect analysis and design

7.2.Ethics and Professional Responsibilities

The application converts user-uploaded text to speech including formulas supported in LaTeX. The raw text, images, or files uploaded to the application are not recorded or shared without user consent. The developers are not responsible for the nature of the information users upload and cannot be held liable for the use of the application on any items of media that are copyrighted and not in the public domain. This includes any sensitive information that users may upload such as credit card details and passwords.

7.3 Judgements and Impacts to Various Contexts

Judgement Description: Web-based application		
	Impact Level	Impact Description
Impact in Global Context	high	Making our application a web-based application means the users can access from multiple platforms which widen the scope of our users.
Impact in Economic Context	Nominal	We would not need to have a version for different OS platforms
Impact in Environmental	_	_

Context		
Impact in Societal Context	Nominal	Most people use phones of either Android or IOS and developing a mobile rather than the web-based application will alienate some of these people.

Table 4. Judgements and impacts

Judgement Description: Voicing LateX recognized formulas		
	Impact Level	Impact Description
Impact in Global Context	Nominal	LateX is widely used by people in the academic field which represents the bulk of our users.
Impact in Economic Context	Nominal	Focusing on LateX recognized formulas narrows our scope of formulas to encapsulate.
Impact in Environmental Context	_	_
Impact in Societal Context	_	_

Table 5. Judgements and impacts

Judgement Description: Using Google's Text-to-Speech API		
	Impact Level	Impact Description
Impact in Global Context	Nominal	Google's API supports many applications and its text-to-speech API is the most popular and widely used
Impact in Economic Context	High	The cost of API is largely reduced as Google is rich enough to support non-company affiliated projects.

Impact in Environmental Context	_	_
Impact in Societal Context	Nominal	Google's text to speech api is among the best in terms of realistic speech as it is powered by machine learning modules.

Table 6. Judgements and impacts

7.4 Teamwork and Peer Contribution

We divided the project into work packages and assigned leaders to each work package who will manage that package and those who are working with them. The table below shows the division of work amongst the team members.

WP#	Work package title	Leader	Members involved
WP1	Deliver High-level project report	Bilal	All
WP2	Deliver Low-level project report	Kasymbek	All
WP3	Purchase APIs	Saifullah	All
WP4	Develop backend service to convert the input to LaTeX	Saifullah	Saifullah and Kasymbek
WP5	Develop backend service to convert LaTeX to plain text	Kasymbek	All

WP6	Develop React.js components	Bilal	Bilal and Saifullah
WP7	Integrate the frontend and backend	Bilal	Bilal and Kasymbek
WP8	Deliver Final report	Saifullah	All

Table 7. Work Distribution

7.5 Project Plan Observed and Objectives Met

Milestones	**Tasks**	**Status**
Deliver High-level project report	● Outline the report ● Do necessary research	Complete
Deliver low-level report	● Outline the report ● Do necessary research	Complete
Purchase APIs	● Get Mathpix API ● Get Google API ● Test APIs	Complete
Develop backend service to convert the input to LaTeX	● Create backend server.js file ● Implement function using MathPix API to convert the file to LaTeX ● Test the service	Complete
Develop backend service to convert LaTeX to plain text	● Write the plain text for each symbol ● Implement function to convert LaTeX string to plain text ● Integrate Google API to the service	Complete

	● Test the service	
Develop React.js components	● Implement React components without backend services ● Get feedback on UI and UX of the website ● Update React component according to feedback	Complete
Integrate frontend and backend	● Integrate back-end services into React components ● Test the final product	Complete
Deliver final report	● Prepare final ● Prepare project demo	Complete

Table 8. Project Milestones

7.6 New Knowledge Acquired and Learning Strategies Used

Since our program is a web application, we learned about React.js and Node.js. We also read tutorials and practiced the exercises at w3schools.com and tutorialspoint.com, as well as read the official documentation of these frameworks. In addition, we learned the syntax of LaTeX through tutorial videos and instruction manuals. We did exercise at overleaf.com and generated LateX based documents to test our skills. Moreover, we read the official documentation of Google and MathPix APIs in order to understand how to use them in our code. Throughout the process of development we were able to apply these things that we learned which further improved our understanding.

8. Conclusion and Future Work

The application we developed has shown that it is possible to present mathematical structures to people with visual impairments. At this stage, SumOfSound can take as an input pdf and image files, as well as LaTeX files which are in the format that we used in our application, and reliably produce comprehensible audio renderings of mathematical expressions.

The problems associated with mathematical access for people with visual impairments are not only related to reading formulae but also involve mathematical graphical information, such as line graphs, pie charts, etc. The graphical information is common and pervasive and access to graphs, drawings, and figures is an important component in the math learning process. However, presenting these forms of mathematical representation audibly is more complicated. Hearing the graphed math formulae alone could not be sufficient to transfer and describe complex mathematical constructs. Therefore, finding a solution to make graphs and charts accessible to visually impaired people is necessary. In the future, we could extend our program, so it can extract information from graphs and transfer it to audio.

9. Glossary

API:	Application Programming Interface
UI:	User Interface
UX:	User Experience

10. References

[1] “Natural Reader” *Free Text to Speech: Online, App, Software & Commercial license with Natural Sounding Voices.* [Online]. Available: https://www.naturalreaders.com/. [Accessed: 10-Nov-2019].

[2] V. Vertogradov, “ Looking for volunteers,” *VKontakte*. [Online]. Available: https://vk.com/id11510315?w=wall11510315_11361/all. [Accessed: 13-Oct-2019].

[3] “What is Client/Server Architecture? - Definition from Techopedia,” *Techopedia.com*. [Online]. Available: https://www.techopedia.com/definition/438/clientserver-architecture. [Accessed: 29-Dec-2019].

[4] Node.js, “About Node.js,” *Node.js*. [Online]. Available: https://nodejs.org/en/about/. [Accessed: 26-May-2020].

[5] I. Gerchev, “Introducing: Semantic UI Component Library,” *SitePoint*, 28-Jan-2014. [Online]. Available: https://www.sitepoint.com/introducing-semantic-ui-component-library/. [Accessed: 26-May-2020].

[6] “Introduction to MathPix,” *Mathpix API v3 Reference*. [Online]. Available: https://docs.mathpix.com/. [Accessed: 26-May-2020].

[7] “About GitHub,” *GitHub*. [Online]. Available: https://github.com/about/. [Accessed: 26-May-2020].

[8] “Heroku Platform,” *Heroku*. [Online]. Available: https://www.heroku.com/platform. [Accessed: 26-May-2020].

[9] “pdf2pic,” *npm*. [Online]. Available: https://www.npmjs.com/package/pdf2pic. [Accessed: 26-May-2020].