Bilkent University

Department of Computer Engineering

CS 491 Senior Design Project

Analysis 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

November 8, 2019

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. But almost all of the text-to-speech applications have trouble with voicing texts that contain formulas. However, there are people with visual impairments who do research in mathematics or physics and need to read scientific articles and books, which usually contain lots of formulas and cannot be read by these applications [1]. This motivated our team to develop an application that could voice the mathematical formulas and equations. Such a project can be useful for both developers of text-to-speech applications and people with vision problems.

2. Current System

There are lots of text-to-speech applications on the market now, such as an 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 [2]. Apart from this deficiency in reading formulas, the existing systems have user-friendly interfaces, give users the option to change reading voices and more. However, most have limitations and charge a premium to use the majority of their features [2].

3. Proposed System

3.1 Overview

The application will be a web app that can be accessed from a modern browser with its URL. Users will be greeted with a simple interface which allows them to enter source LaTeX, or upload a file to the system. If the user chooses to enter source LaTeX, and are ready to have it read out, they simply have to press the play button to have the program begin reading. In the case

that the text entered exceeds the character limit, the program will read everything up to the end of the character limit. If the user chooses to upload a document, the document will be displayed within the web app. If then the user clicks the play button, once again the program will read everything from the beginning up to the end of the character limit. Otherwise, the user will have the option to select the part of the document they want to have read, before clicking play. The user will also be able to click on parts of the selected text in a document to have repeated. Once again the character limit applies to this case as well. In all of these cases, the user will be able to click the pause button to pause the audio or the stop button for the program to stop reading. The user will have to choose between two reading voices and select their reading pace. In the case of an uploaded document being displayed the user can choose a display theme to customize the size, font and other attributes of the text being displayed. Once a piece of text has been read, the user will have the option to download an MP3 file of the reading.

3.2 Functional Requirements

- The application will be able to read formulas from documents or images.
- The user will be able to upload files in PDF, LaTeX, and supported image formats.
- The user will be able to enter source LaTeX into a text box on the web app rather than uploading a file.
- The user will be able to control the audio playback using play, pause and stop buttons.
- The user will be able to select the part of the document they want to be voiced out.
- The user will be able to choose between two reading voices, one male and one female.
- The user will be able to adjust the reading pace to suit their requirements.
- The user will be able to change themes to customize the size, font, and color of the display of the text being read.
- The user will be able to download the audio output as an MP3 file.

- The application will limit the number of characters that a user can have voiced out. This limit excludes characters generated to voice out formulas.

3.3 Nonfunctional Requirements

3.3.1 Usability

- The application will have a user-friendly interface that will allow users to navigate through it easily without difficulty.
- The application will not be flooded with a lot of functionalities that may overwhelm the user.

3.3.2 Extensibility

- The application will be made such that new features can be added easily.
- As there is always room for improvement, the application will be updatable.

3.3.3 Reliability

- The application will not alter any files uploaded to voice out formulas.
- The application will not store users' information or track their activities in the browser.
- The formulas in users' documents will be identified and converted to LaTeX format, if not already in LaTeX, which will then be voiced out accurately.

3.3.4 Availability

- The application can be accessed on the web by anyone with access to the link.

3.3.5 Efficiency

- The application will be able to identify formulas in texts in a few seconds and voice them out without intermittent delay or breaks.

3.4 Pseudo Requirements

3.4.1 Version Control

We will use the Git mechanism to control the development of the project and the contributions of the group members. Github pages will be used to present the project.

3.4.2 External Tools

We will use Mathpix API to obtain the LateX representation of images uploaded by the user. We will also use Google API to voice out the text representation of the LateX string

3.4.3 Deployment Platform

The application will be developed as a web application using the React framework for the front-end and Node.js for the backend, so it can run in all operating systems.

3.5 System Models

3.5.1 Scenarios

Scenario 1

Use Case Name: Upload document

Actor: User

Entry Conditions:

- User must be on homepage screen of the app

Exit Conditions:

- File is uploaded

The main flow of events:

1. The user selects the upload button from the screen
2. User is redirected to choose a file to upload

Scenario 2

Use Case Name: Select part to voice

Actor: User

Entry Conditions:

- User uploads a document and selects the part to voice

Exit Conditions:

- The selected part has been voiced

The main flow of events:

1. The user highlights part of the document to voice
2. The highlighted part is voiced out

Scenario 3

Use Case Name: Enter LateX string

Actor: User

Entry Conditions:

- User is on the homepage screen

Exit Conditions:

- User has entered the LateX string

The main flow of events:

1. User clicks on ‘Enter LateX string’ button from homepage screen
2. The user enters the LateX string

Scenario 4

Use Case Name: Customize settings

Actor: User

Entry Conditions:

- User is on the homepage screen

Exit Conditions:

- The user goes back to homepage screen

The main flow of events:

- User clicks on the ‘customize settings’ button from homepage screen

- User is redirected to the settings menu to change settings

Scenario 5

Use Case Name: Download mp3

Actor: User

Entry Conditions:

- User has uploaded a document and selected the part to voice
- User has entered a LateX string

Exit Conditions:

- User downloads mp3 file

The main flow of events:

1. User clicks on the ‘Download mp3’ button
2. The mp3 file is downloaded

Scenario 6

Use Case Name: Play mp3

Actor: User

Entry Conditions:

- User has uploaded a document and selected the part to voice
- User has entered a LateX string

Exit Conditions:

- The mp3 file is played

The main flow of events:

1. User clicks on the ‘play mp3’ button
2. The mp3 audio is played

Scenario 7

Use Case Name: Change the play speed

Actor: User

Entry Conditions:

- User has started playing the selected part to voice

Exit Conditions:

- User chooses speed
- The selected part which is being voiced reaches the end

The main flow of events:

1. User clicks on ‘change play speed’ button
2. The user chooses the speed to change

Scenario 8

Use Case Name: Change reader

Actor: User

Entry Conditions:

- User has entered ‘Customize settings’ menu

Exit Conditions:

- The user chooses the reader

The main flow of events:

1. User clicks on ‘Change reader’ button
2. The user chooses the reader

Scenario 9

Use Case Name: Change theme

Actor: User

Entry Conditions:

- User has entered ‘Customize settings’ menu

Exit Conditions:

- User chooses theme

The main flow of events:

1. User clicks on ‘Change theme’ button

2. User chooses theme

3.5.2 Use Case Model

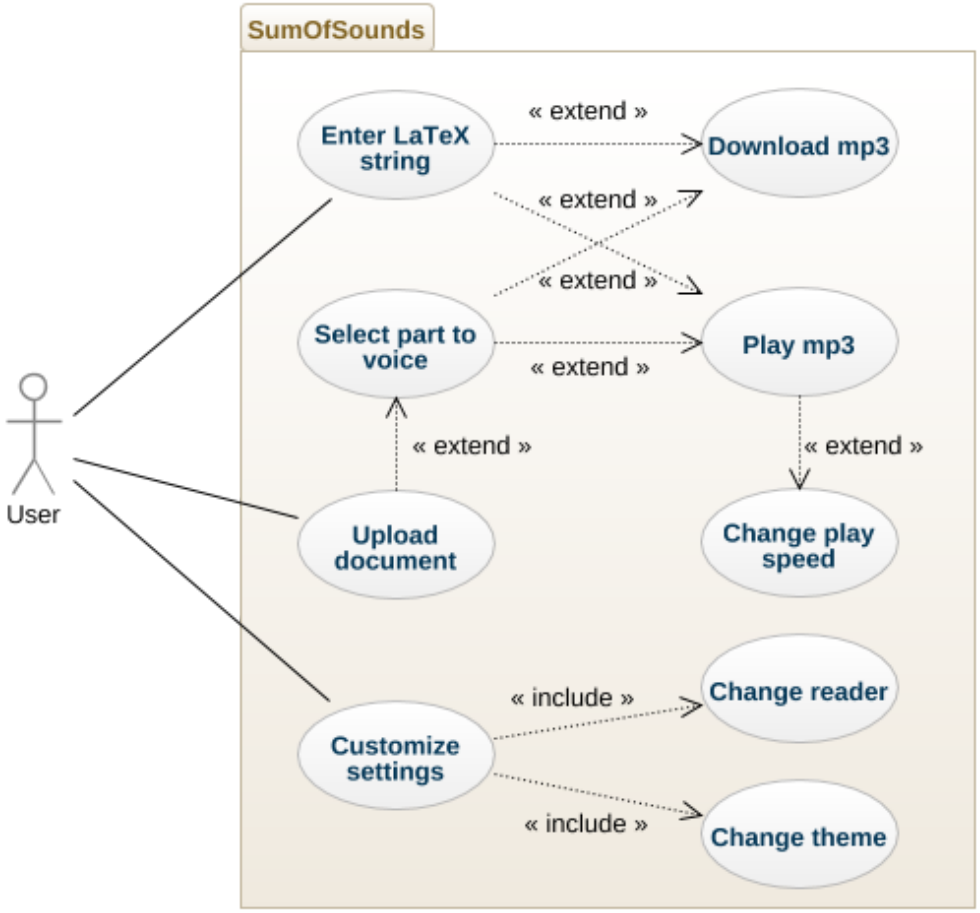

Figure 1. Use case diagram

3.5.3 Deployment Diagram

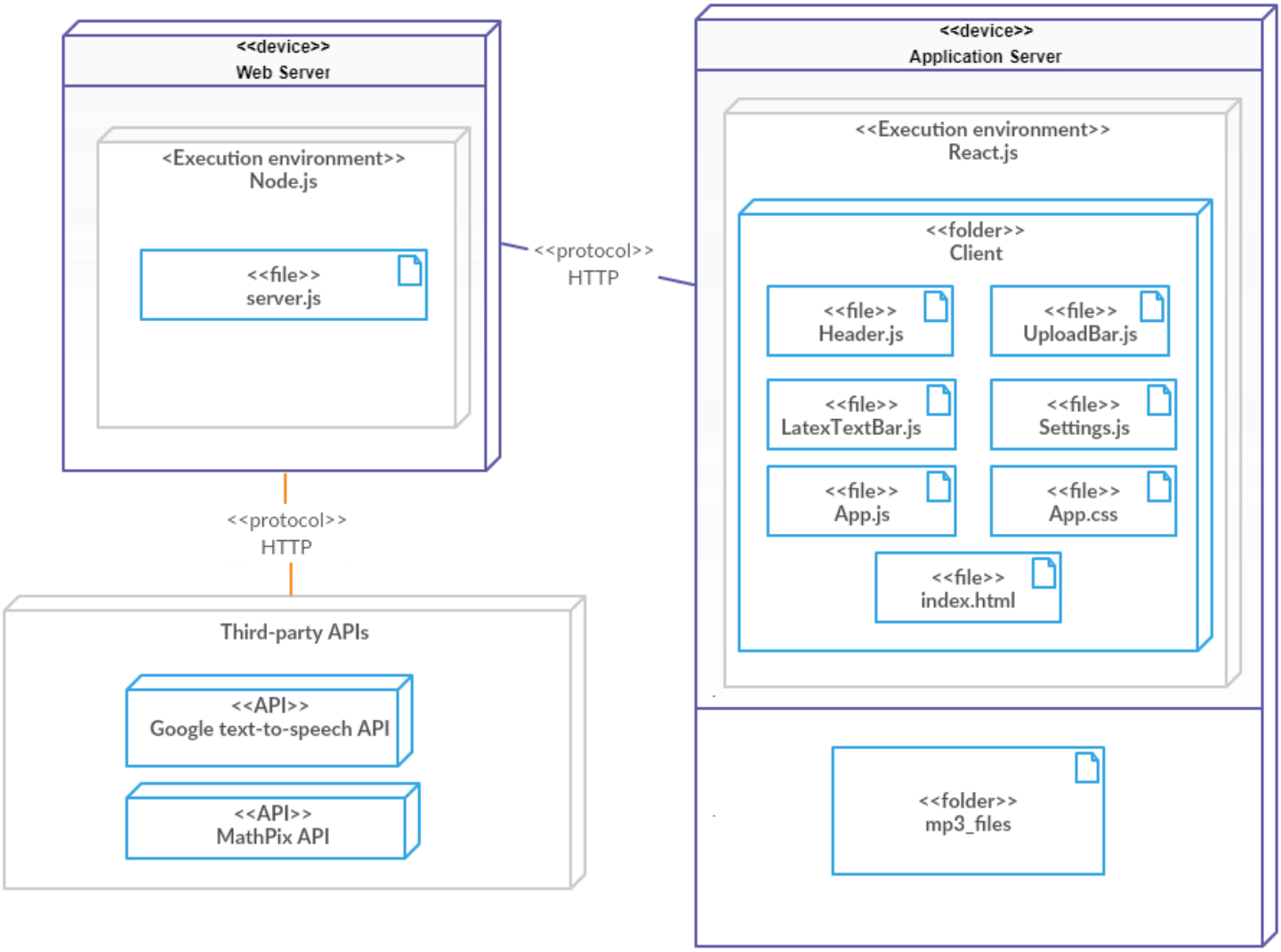

Figure 2. Deployment diagram

3.5.4 Dynamic Models

3.5.4.1 Sequence diagram

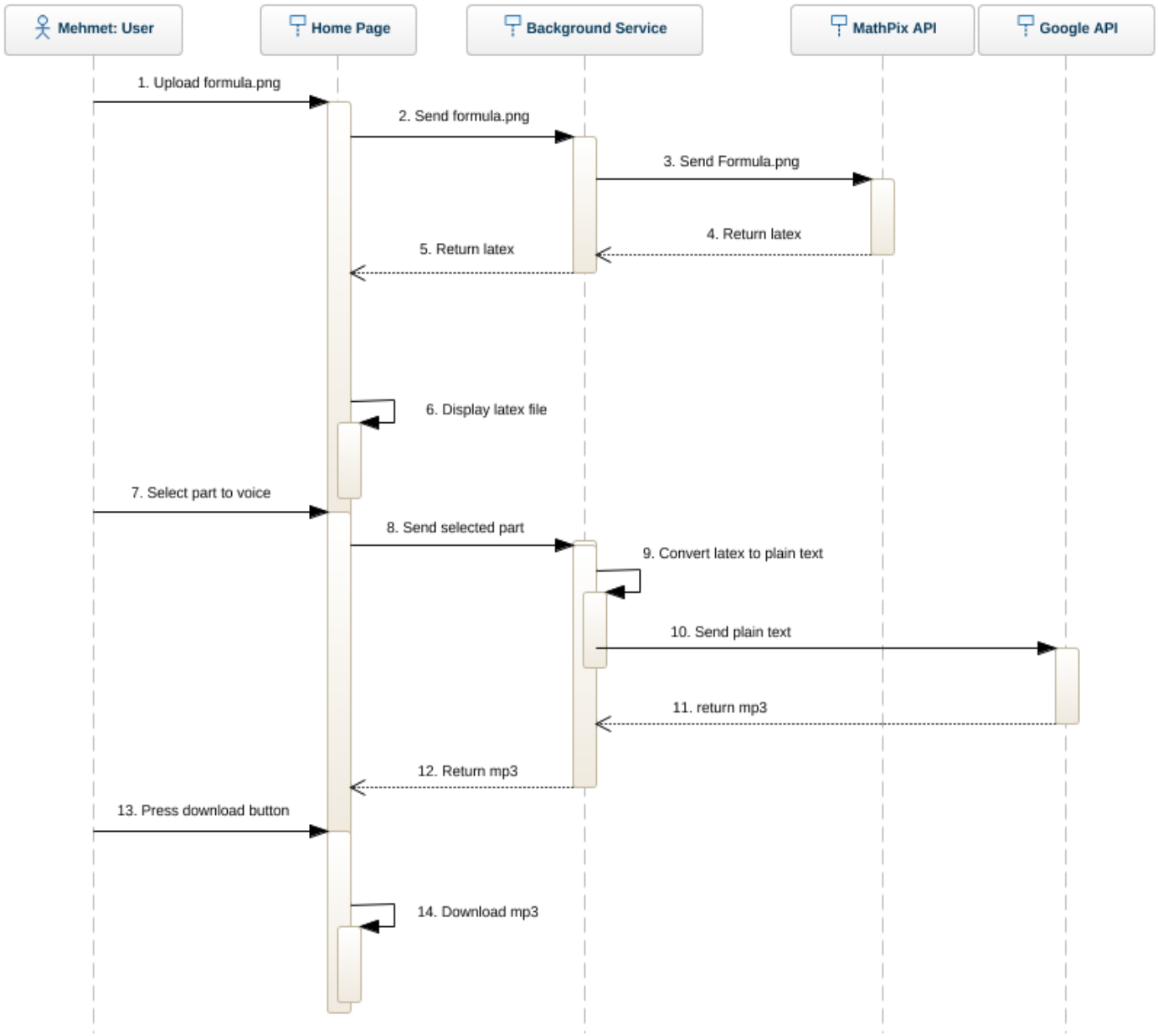

Figure 3. Sequence diagram

The diagram above shows the interaction of a user and the program when the user wants to upload a document and download its mp3.

3.5.4.2 Activity diagram

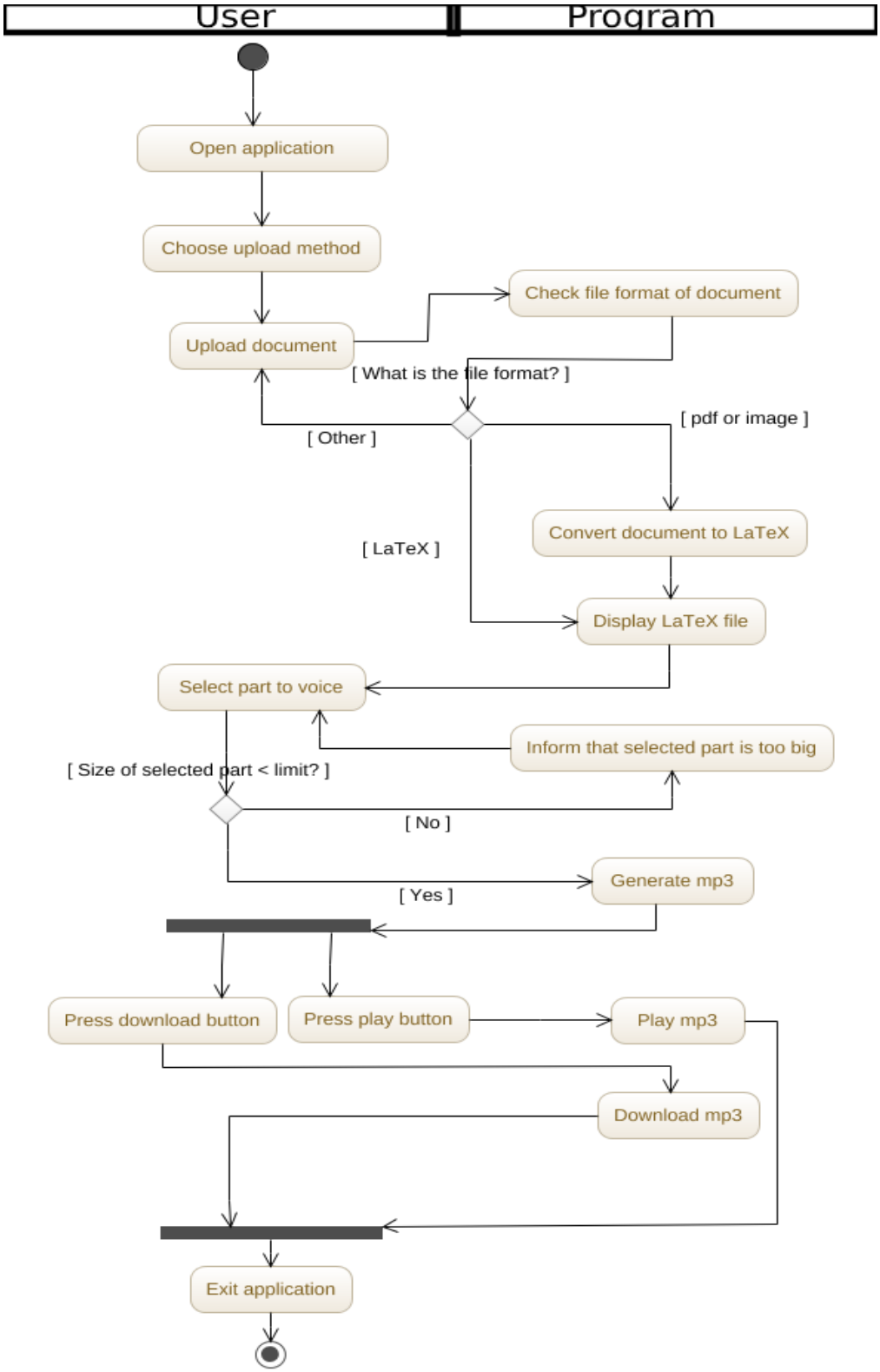

Figure 4. Activity diagram

The diagram above shows the control flow of the program and its decision paths.

3.5.5 User Interface - Navigational Paths and Screen Mock-ups

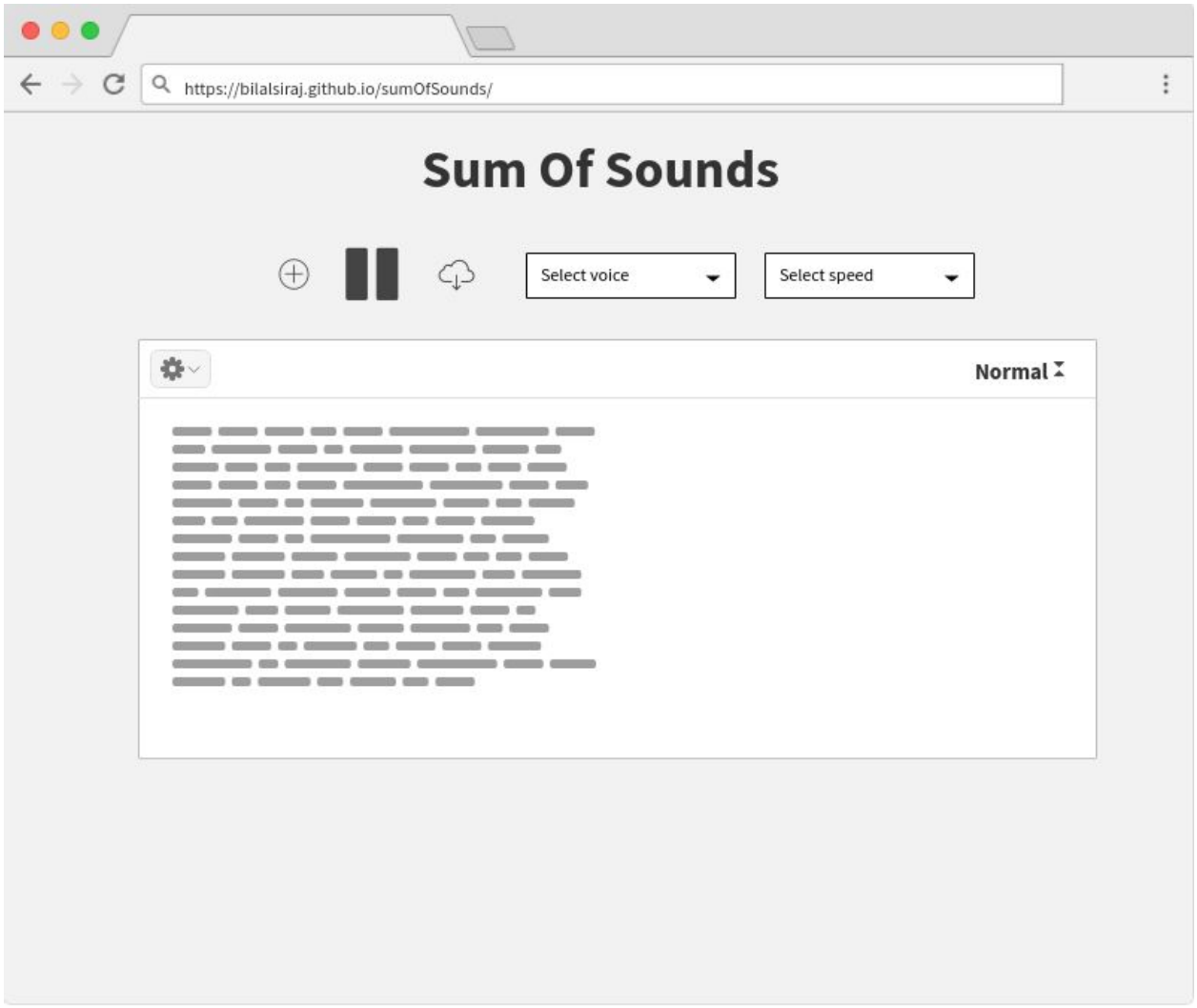

Figure 5. Mockup of main page

This is the main page of the web app. It will be clean and minimal to make it easy to navigate and make the user experience straightforward. It has a few buttons to upload a file, pause/play the audio and download the mp3 file. It also has two drop-down menus for the user to choose a reading voice and the reading pace. Below that is a box for the user to enter source LaTeX,

which has the option to change font size and a few other settings hidden under the gear icon button.

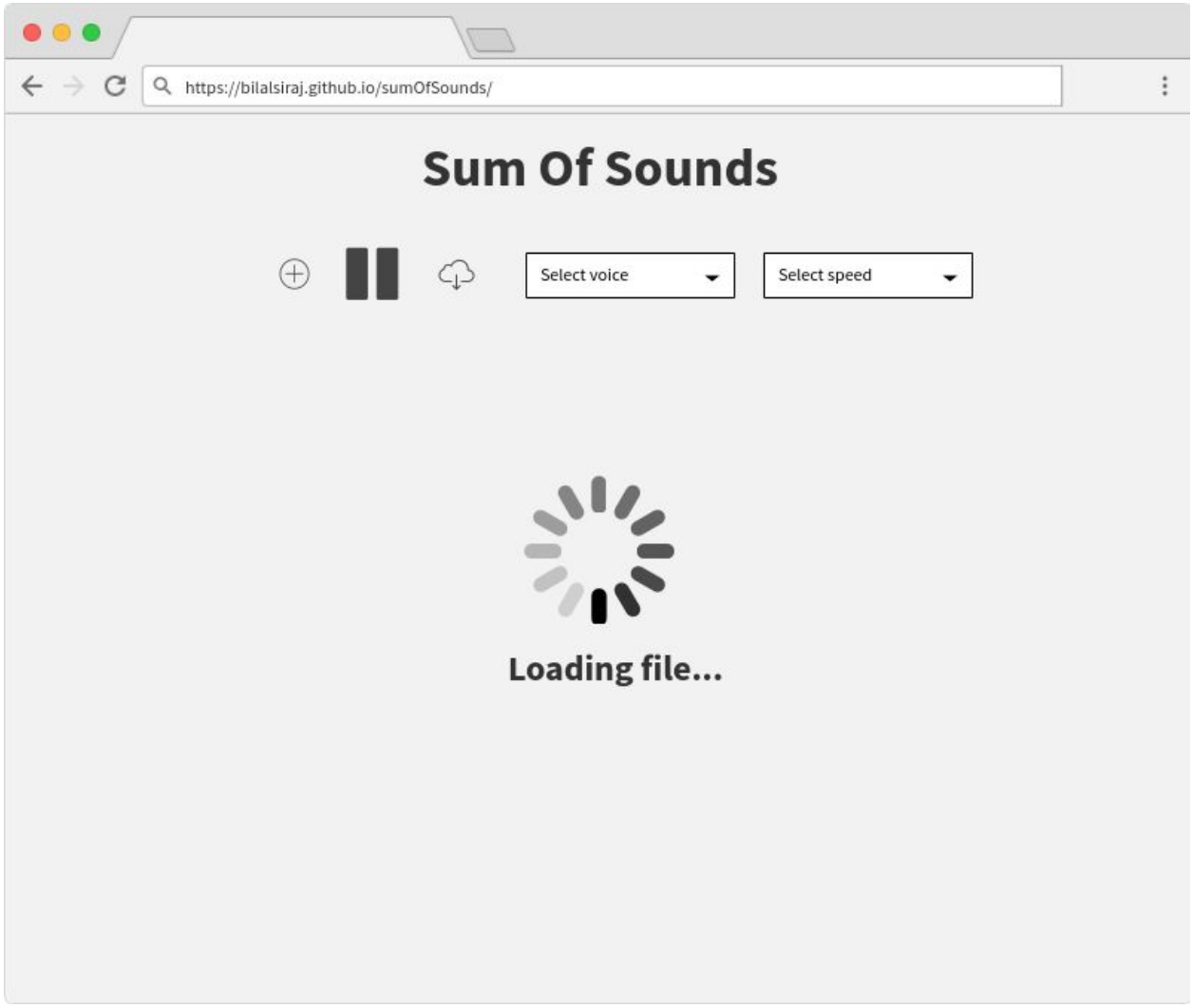

Figure 6. Mockup of page loading the file user uploads

When the user uploads a file to be read, it might take a few seconds to load and be displayed. A simple loading screen will show up at this time.

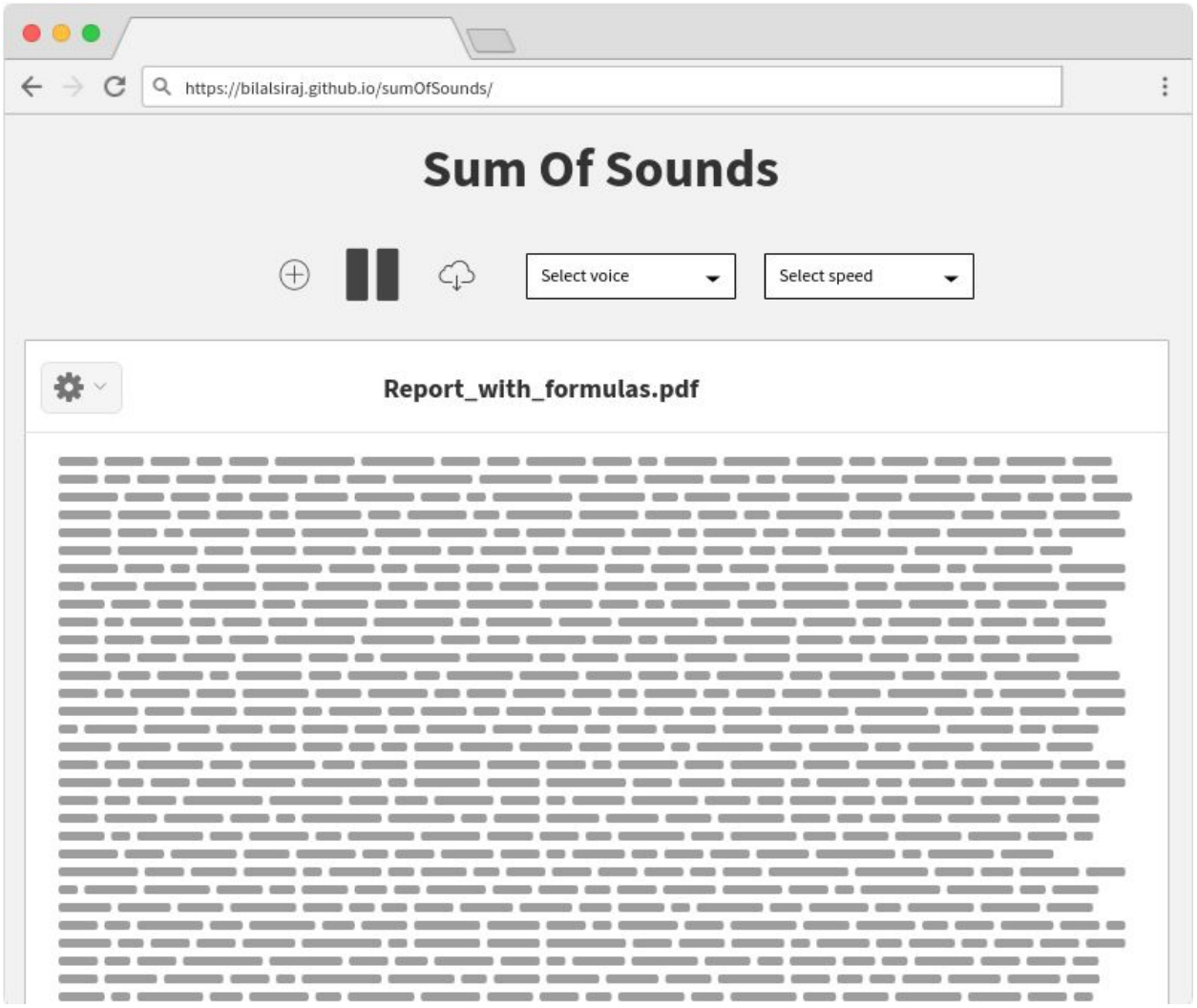

Figure 7. Mockup of the page displaying the file user uploads

Once the file is loaded, it will be displayed and upon the user clicking the play button, the reading will begin.

4. Other Analysis Elements

4.1. 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 1. Factors that can affect analysis and design

4.2. Risks and Alternatives

- One possible risk in our project is the cost of using the API's increasing due to an unforeseen amount of use of the third party services. It is possible that during implementation we find the need for other APIs, or using the ones we already have planned becomes too expensive. In such a case, we would have to find other cheaper alternative API's, although we have already chosen to work with the most cost-effective ones.
- Another possible risk is if the third party API's we are choosing to use, prove to be too inaccurate or unreliable to produce the results we need. Although a lot of thought goes

into choosing a reliable API, in case they become unusable we will find replacements that are relatively better, even with a compromise on cost or speed.

- Another possible risk is if the developers of the API's were to stop supporting it. For at least one of them, the Google Cloud Text-to-Speech API this is almost impossible. However, for the others, it is a very unlikely possibility. Once again, our only choice would be to find another suitable API to replace it.
- The last possible risk is if the processing done by our program is too resource-intensive for the developers to run on their local machines. In such a case, we would need to use third-party servers to do the processing.

	Likelihood	Effect on the project	B Plan Summary
Cost of the APIs	Very low	Our program highly relies on paid third-party APIs, so without them, it will not work	Find similar APIs, which are cheaper
The correctness of the APIs	Low	The program will not work, as intended.	Find similar APIs, which are more accurate
Developers stop supporting APIs	Very low	The program will not work.	Find APIs with similar functionalities
Processing becomes too resource-intensi	Low	Testing during development becomes too time-consuming	Use third-party servers to do the processing

ve to run locally			

Table 2. Risks

4.3. Project Plan

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

WP 1: Deliver High-level project report			
Start date: *12.10.2019* **End date:** *21.12.2019*			
Leader:	*Bilal*	**Members involved:**	*Kasymbek* *Saifullah*
Objectives: *To plan, discuss, delegate work, and produce the final draft of the High-level design report.*			
Tasks: ***Task 1.1 Outline the report*:** *Develop the outline of the report, this includes subtopics and important updates since the last report* ***Task 1.2 Do relevant research:*** *Complete the necessary research to inform our content in the report*			
Deliverables ***D1.1:*** *High-level project report*			

WP 2: *Deliver Low-level report*			
Start date: *3.02.2020* **End date:** *17.02.2020*			
Leader:	*Kasymbek*	**Members involved:**	*Bilal* *Saifullah*
Objectives: *To plan, discuss, delegate work, and produce the final draft of the Low-level design report.*			
Tasks: ***Task 2.1 Outline the report*:** *Develop the outline of the report, this includes subtopics and important updates since the last report* ***Task 2.2 Do relevant research:*** *Complete the necessary research to inform our content in the report*			
Deliverables ***D2.1:*** *Low-level design report*			

WP 3: Purchase APIs
Start date: *22.10.2019* **End date:** *25.12.2019*

Leader:	*Saifullah*	**Members involved:**	*Kasymbek* *Bilal*
Objectives: To get the group access to the necessary API's and pay for them if necessary			
Tasks: ***Task 3.1 Get MathPix API on multiple accounts:*** *Sign up for group or individual access to use the API and purchase, if necessary, the quota required during development* ***Task 3.2 Get Google API on multiple accounts:*** *Sign up for group or individual access to use the API and purchase, if necessary, the quota required during development* ***Task 3.3 Test purchased APIs:*** *Ensure that all team members' accounts or group account is working and all team members have access on their personal machine**s***			
Deliverables ***D3.1:*** *Google and MathPix API access credentials*			
WP 4: *Develop backend service to convert the input to LaTeX*			
Start date: *26.02.2020* **End date:** *26.03.2020*			
Leader:	*Saifullah*	**Members involved:**	*Kasymbek*

Objectives: *To complete the service which converts all user input file formats to LaTeX using MathPix API*
Tasks: ***Task 4.1 Create a backend server.js file:*** *Set up the backend server* ***Task 4.2 Implement function using MathPix API to convert the file to LaTeX:*** *Write code to convert user input formats of PDF and images to source LaTeX using the MathPix API* ***Task 4.3 Test the service:*** *Ensure the service is stable, fast and reliable by testing with multiple document formats and image formats. Ensure all potential errors in this service are handled.*
Deliverables ***D2.1:*** server.js with one service

WP 5: Develop backend service to convert LaTeX to plain text			
Start date: *26.02.2020* **End date:** *8.04.2020*			
Leader:	*Kasymbek*	**Members involved:**	*Bilal* *Saifullah*
Objectives: *To develop the backend service that converts LateX string to plain text*			

Tasks:

Task 5.1. Write the plain text for each symbol**:** *Develop context-dependent English word equivalent for all possible symbols in LaTeX*

Task 5.2. Implement function to convert LaTeX string to plain text**:** *Set up the class responsible for converting LateX string to plain text*

Task 5.3. Integrate Google API to the service**:** *Implement the function that gives the plain text to Google API and return mp3 file.*

Task 5.4. Test the service**:** *Prepare the test cases for the service and test it after implementation.*

Deliverables

D5.1: the final version of *server.js*

WP 6: Develop React.js components

Start date: *26.02.2020* **End date:** *8.04.2020*

Leader:	*Bilal*	**Members involved:**	*Saifullah*

Objectives: Develop frontend of the application without integration with backend.

Tasks:

Task 6.1 Implement React components without backend services: *Implement all React components, which are Header.js, LatexTextBar.js, UploadBar.js, Settings.js, and App.js. However, the component will not have a function that does the operation with backend but only user interface.*

Task 6.2 Get feedback on UI and UX of the website: *Show the interface of the website to the friends and professor to get feedback.*

Task 6.3 Update React component according to feedback: *According to feedback we got, we will change the user interface of the website***.**

Deliverables

D6.1: Header.js

D6.2: LatexTextBar.js

D6.3: UploadBar.js

D6.4: Settings.js

D6.5: App.js

WP7: Integrate frontend and backend

Start date: *8.04.2020* **End date:** *01.05.2020*

Leader:	*Bilal*	**Members involved:**	*Kasymbek*
Objectives: *Implement the function in frontend that will communicate with backend services.*			
Tasks: ***Task 7.1. Integrate back-end services into React components*:** Implement functions in React components that will send inputs to backend services using POST method and will display the output received from backend services**.** ***Task 7.2. Test the final product*:** Prepare the test cases that test the functions above. After the implementation of function test them and fix possible bugs.			
Deliverables ***D7.1:*** *The final product*			
WP 8: *Deliver final report*			
Start date: *2.05.2020* **End date:** *14.05.2020*			
Leader:	*Saifullah*	**Members involved:**	*Kasymbek* *Bilal*
Objectives: *Write the final report and prepare a presentation for juries and for CS fair.*			

Tasks: ***Task 8.1 Write final report:*** *write* ***Task 8.2 Prepare the project presentation:*** *prepare* ***Task 8.3 Make presentation*****:** *prepare*
Deliverables ***D8.1:*** *Final report* ***D8.2:*** *Project presentation*

Table 3. List of work packages

4.4. Ensuring Proper Team-work

Each week we are going to meet at least once and maintain a record of what we have done that week. We will also assign work till the next week for each team member and record that. We have assigned leaders to each work package who will manage that package and those who are working with them. Everything will be recorded in a document that will be shared with the supervisor.

4.5. Ethics and Professional Responsibilities

The application will convert user-uploaded text to speech including formulas supported in LaTeX. The raw text, images, or files uploaded to the application will not be 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.

4.6. New Knowledge and Learning Strategies

Since our program is going to be a web application, we should learn React.js and Node.js. We are going to read the tutorials and practicing the exercises at w3schools.com and tutorialspoint.com, as well as read the official documentation of the frameworks. In addition, we are going to learn syntax of LaTeX by reading tutorials, doing exercise at overleaf.com and possibly generating our following reports in LaTeX. Moreover, we will read the official documentation of Google and MathPix APIs in order to understand how to use them in our code.

5. Glossary

API: Application Programming Interface

UI: User Interface

UX: User Experience

React.js: JavaScript library for building user interfaces.

Node.js: Cross-platform JavaScript run-time environment and library.

6. References

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

[2] "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].