**Chapter 4: Implementation and Discussion of Results**

**4.1 Introduction**

This chapter covers the implementation details of the **PDF to Audio Converter Application**. It describes how the application was developed, the tools and technologies used, the challenges encountered during implementation, and how they were addressed. This chapter also includes the results of the testing conducted on the application, showcasing the functionality of the system and its performance in converting PDF documents to speech.

**4.2 Development Environment and Tools**

The development of the **PDF to Audio Converter Application** involved both client-side and server-side technologies. Below is an overview of the environment and tools used during the implementation:

**4.2.1 Backend (Server-Side) Development**

* **Programming Language**: PHP
* **Text Extraction Library**: PDF2Text, a PHP-based library for extracting text from PDF files.
* **Database**: MySQL for storing metadata related to uploaded PDF files.
* **Web Server**: Apache (XAMPP) for local development and testing.

**4.2.2 Frontend (Client-Side) Development**

* **HTML/CSS**: Used for building the user interface and ensuring responsiveness using Bootstrap.
* **JavaScript**: For handling speech synthesis using the Web Speech API.
* **Web Speech API**: A browser-native API used to convert extracted text into speech, providing speech synthesis functionality.

**4.2.3 Development Tools**

* **Integrated Development Environment (IDE)**: Visual Studio Code.
* **Version Control**: Git for managing the project's source code.
* **Browser**: Google Chrome and Mozilla Firefox for testing Web Speech API compatibility.

**4.3 Implementation Details**

**4.3.1 Backend Implementation**

1. **File Upload Functionality** The backend provides a file upload handler that accepts PDF files from the user via an HTML form. The uploaded file is processed by PHP, which checks the file’s type and size to ensure it is a valid PDF. The file is then stored in a designated directory on the server.

php

Copy code

if (isset($\_POST['convert'])) {

$namephoto = $\_FILES["img"]["name"];

$tmp\_location\_photo = $\_FILES["img"]["tmp\_name"];

$target\_dir = "pdf/";

$target\_file = $target\_dir . rand() . basename($namephoto);

// Move uploaded file to server directory

if (move\_uploaded\_file($tmp\_location\_photo, $target\_file)) {

// Process the file

echo "<script>alert('File Uploaded Successfully');</script>";

} else {

echo "<script>alert('File Upload Failed');</script>";

}

}

1. **Text Extraction using PDF2Text** Once the file is uploaded, the **PDF2Text** library is used to extract the text from the PDF file. This library parses the PDF and retrieves any text it can find, converting it into a plain-text format that can be processed for speech synthesis.

php

Copy code

include('class.pdf2text.php');

$a = new PDF2Text();

$a->setFilename($target\_file);

$a->decodePDF();

$extractedText = $a->output(); // Extracted text

1. **Database Integration** MySQL is used to store metadata about the uploaded PDF files, such as the file name, file path, and upload timestamp. This allows for record-keeping and potential future use of the stored PDF files.

php

Copy code

$sql = "INSERT INTO pdf\_tbl (file, upload\_time) VALUES ('$target\_file', NOW())";

$result = mysqli\_query($conn, $sql);

**4.3.2 Frontend Implementation**

1. **User Interface** The frontend is built using HTML and Bootstrap to provide a clean and responsive interface. The user can upload PDF files using a simple form, view the extracted text, and interact with the playback controls (play, pause, resume).

html

Copy code

<form action="convert.php" method="post" enctype="multipart/form-data">

<input type="file" class="form-control" name="img" required="">

<button type="submit" name="convert" class="btn btn-primary">Convert to Audio</button>

</form>

1. **Speech Synthesis using Web Speech API** Once the text is extracted from the PDF, it is displayed on the frontend. JavaScript is used to interact with the Web Speech API, which handles the conversion of text to speech. Users can play, pause, and resume the speech using the following code:

javascript

Copy code

var synth = window.speechSynthesis;

var inputTxt = document.querySelector('.txt');

var voices = [];

function speak() {

if (synth.speaking) {

console.error('Speech already in progress');

return;

}

if (inputTxt.value !== '') {

var utterThis = new SpeechSynthesisUtterance(inputTxt.value);

utterThis.voice = voices[0]; // Choose the first available voice

utterThis.pitch = 1;

utterThis.rate = 1;

synth.speak(utterThis);

}

}

document.getElementById('playBtn').onclick = function() {

speak();

};

document.getElementById('pauseBtn').onclick = function() {

synth.pause();

};

document.getElementById('resumeBtn').onclick = function() {

synth.resume();

};

1. **Customization of Speech** Users are given the ability to adjust the rate and pitch of the speech using sliders. These controls are linked to the Web Speech API's rate and pitch properties:

javascript

Copy code

var pitchControl = document.getElementById('pitch');

var rateControl = document.getElementById('rate');

pitchControl.onchange = function() {

utterThis.pitch = pitchControl.value;

};

rateControl.onchange = function() {

utterThis.rate = rateControl.value;

};

**4.4 Testing and Results**

**4.4.1 Testing Procedure**

The system was tested on various browsers and with different types of PDF documents to ensure its functionality and performance. The primary objectives of the testing were:

1. **File Upload and Validation**: Ensuring that only valid PDF files could be uploaded and processed.
2. **Text Extraction**: Verifying that the system correctly extracts text from a variety of PDFs, including those with simple text, formatted text, and non-text elements like images.
3. **Speech Synthesis**: Testing the speech synthesis functionality, including the play, pause, and resume features, as well as customization of speech rate and pitch.
4. **Cross-Browser Compatibility**: Ensuring that the Web Speech API works on different browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge.

**4.4.2 Test Results**

* **File Upload**: The file upload functionality worked as expected, with proper validation of file type (only PDFs) and file size. Invalid files (such as images or text documents) were rejected with appropriate error messages.
* **Text Extraction**: The text extraction worked well for PDFs containing simple text. However, for complex PDFs with images, tables, or multiple columns, the extraction was not always perfect, as the PDF2Text library struggled with layout and formatting.
* **Speech Synthesis**: The speech synthesis was successful across all tested browsers. The play, pause, and resume functionality worked as intended, and users were able to adjust the speech rate and pitch using the provided sliders. The audio output was clear, and customization options allowed users to tailor the experience to their preferences.
* **Cross-Browser Compatibility**: The Web Speech API was fully functional on Google Chrome and Mozilla Firefox. However, support for the Web Speech API on other browsers, such as Safari and Internet Explorer, was limited.

**4.4.3 Challenges Encountered**

* **Complex PDFs**: Extracting text from PDFs with complex layouts (e.g., multi-column text, images) was a challenge, as the text extraction was not always accurate. Future improvements may involve integrating more advanced PDF parsing libraries that can handle complex layouts better.
* **Browser Compatibility**: While the Web Speech API worked well on modern browsers like Chrome and Firefox, there were limitations on other browsers like Safari and Internet Explorer, where the API either performed poorly or was unsupported.

**4.5 Discussion of Results**

The **PDF to Audio Converter Application** performed well in most cases, meeting the primary objectives of converting PDF text into audio. The file upload and text extraction process was efficient, and the speech synthesis provided a seamless experience for users. The ability to pause, resume, and customize speech playback enhanced the user experience.

However, there are still areas for improvement. The PDF text extraction module could be further enhanced to handle more complex PDF layouts, including those with images and tables. Additionally, addressing browser compatibility issues would make the application more universally accessible.

**4.6 Summary**

In this chapter, the implementation details of the **PDF to Audio Converter Application** were discussed, highlighting the tools and technologies used in both the backend and frontend components. The system successfully achieves its goal of converting PDF text into speech using PHP and the Web Speech API. The testing process demonstrated the functionality of the system, though certain challenges were encountered, particularly with complex PDFs and browser compatibility. The next chapter will provide conclusions and recommendations for future improvements to the system.