A Real Time Project report

on

TYPE2TONE

Submitted to

Jawaharlal Nehru Technological University, Hyderabad

Submitted in partial fulfillment of requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

MANGALI KEERTHI	**236P1A0589**
ORSU RAJESHWARI	**236P1A05A8**
PALAVALASA KUSUMA	**236P1A05B0**
PINNASI AKSHITHA	**236P1A05B7**

Under the guidance of

Dr. K. Ramakrishna, Ph.D.

Assistant professor, Department of CSE

Department of Computer Science and Engineering

Rishi MS Institute of Engineering & Technology for Women

Approved by AICTE, Affiliated to JNTUH

Nizampet Cross Road, near JNTUH, Kukatpally, Hyderabad, Telangana 500085

2024-25

RISHI MS INSTITUTE OF ENGINEERING AND TECHNOLOGY FOR WOMEN

(Affiliated to JNTUH University, Approved by AICTE)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled "TYPE2TONE" is being submitted by

MANGALI KEERTHI	**236P1A0589**
ORSU RAJESHWARI	**236P1A05A8**
PALAVALASA KUSUMA	**236P1A05B0**
PINNASI AKSHITHA	**236P1A05B7**

In partial fulfilment of the award of degree of **BACHELOR OF TECHONOLOGY** in **COMPUTER SCIENCE & ENGINEERING** in Rishi MS Institute of Engineering & Technology for women, affiliated to the Jawaharlal Nehru Technological University, Hyderabad during the year 2024-2025 is a record of bonafied piece of work, undertaken by the supervision of the undersigned.

Internal Guide	Head of the Department
Dr. K.Ramakrishna Ph.D	Dr.Archana Patil M.Tech,Ph.D
Assistant Professor	Assistant Professor, HOD

Submitted for Viva Voice Examination held on______________________________

External Examiner

ACKNOWLEDGEMENT

This report will certainly not be completed without due acknowledgement to all who have helped us with our Major Project work.

We would like to express our sincere thanks to our Secretary, **Ms. Rajasree**, for her constant supervision and encouragement, which helped us in completing this work successfully.

We are expressing our sincere gratitude to our Principal, **Dr. K R N Kiran Kumar**, for his timely suggestions, which helped us to complete this work successfully.

It's our privilege to thank **Dr. Archana Patil**, Head of the Department, for her encouragement during the progress of this work.

We derive great pleasure in expressing our sincere gratitude to our Project Coordinator **Ms.Madhavi** and **Ms.Swetha** for their encouragement and timely suggestions during the progress of this work.

We express our sincere thanks to our Guide, **Dr K. Ramakrishna,** for giving us moral support, kind attention, and valuable guidance throughout this work.

We are thankful to our **Rishi MS Institute of Engineering and Technology for Women** for providing the required facilities during the Major Project work.

We would like to thank our parents and our friends for being supportive all the time, and we are very much obliged to them.

MANGALI KEERTHI	**236P1A0589**
ORSU RAJESHWARI	**236P1A05A8**
PALAVALASA KUSUMA	**236P1A05B0**
PINNASI AKSHITHA	**236P1A05B7**

RISHI MS INSTITUTE OF ENGINEERING & TECHNOLOGY FOR WOMEN

(Affiliated to JNTUH University, Approved by AICTE)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vision & Mission of the CSE Department

Vision of the Department

To promote a center of excellence in a sustainable academic environment to impart technical skills and a strong research platform to prepare global leaders.

Mission of the Department

1.To create a good academic platform with modern teaching and learning methodologies.

2.To enhance leadership qualities with ethics, values, and a sense of teamwork for professional excellence.

3.To inculcate research culture by encouraging projects in advanced technologies through industry interactions.

TABLE OF CONTENTS

ABSTRACT

The Text-to-Speech Converter is a simple web-based application that allows users to convert written text into audible speech. Developed using HTML, CSS, and JavaScript, this program leverages the browser's SpeechSynthesis API to transform user-inputted text into speech. The user interface consists of a text area for inputting text, a language selection dropdown to choose from available voices, and a "Listen" button to trigger the speech synthesis. When the user clicks the button, the program reads aloud the text entered in the text area, utilizing the selected voice. The application provides an interactive and user-friendly way to hear text read aloud, making it useful for accessibility purposes, language learning, and other scenarios where text needs to be audibly communicated.

CSS ensures a visually appealing and responsive design, optimizing the user experience across various devices. JavaScript serves as the core programming language, implementing the Web Speech API to facilitate speech synthesis, enabling users to modify voice, pitch, and speech rate settings to personalize the output. The converter is designed to be lightweight, requiring no external installations, ensuring cross-browser compatibility and a seamless experience. This project has significant implications for accessibility, particularly for individuals with visual impairments or reading difficulties. Additionally, it serves as a valuable tool for language learners, educators, and professionals seeking hands-free interaction with text-based content. By leveraging native browser functionalities, this implementation demonstrates the potential of web technologies to create practical and inclusive applications, showcasing how simple yet powerful solutions can contribute to improved user engagement and accessibility.

LIST OF ACRONYMS

SNo.	**Acronyms**	**Description**
1	HTML	Hyper Text Markup Language
2	CSS	Cascading Styling Sheet
3	JS	Java Script
4	TTS	Test To Speech

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Text-to-Speech (TTS) technology is a type of assistive system that converts written text into spoken words. It allows computers, smartphones, and web applications to "speak" the text displayed on the screen. This technology has become increasingly common in modern applications, ranging from digital assistants like Siri and Google Assistant to educational and accessibility tools. TTS bridges the gap between written communication and audio output. It is widely used in applications that help users with reading disabilities, vision impairments, or those who prefer listening over reading. This capability improves digital accessibility and user interaction, making information consumption more convenient and inclusive. The main objective of a text-to-speech system is to make digital content more accessible and user-friendly. It provides an alternative way for users to consume information, especially when reading is not feasible, such as while driving, multitasking, or for individuals with learning challenges.

Web applications often use TTS to enhance user experience. Integrating TTS into a website allows users to hear selected content using the browser's built-in features. Modern browsers support the Web Speech API, which provides TTS functionality directly through JavaScript. This eliminates the need for external libraries or plugins and makes implementation lightweight and efficient. This project aims to develop a simple text-to-speech Speech Converter using HTML, CSS, and JavaScript. The application will take user input in the form of text, and upon clicking a button, convert that text into speech using the browser's speech synthesis capabilities. The focus of this project is not just on converting text to speech, but also on understanding how different web technologies can be combined to create an interaction and functional web tool. This will help in gaining practical knowledge of web development and accessibility enhancement.

Some key benefits include:

Accessibility: Assists visually impaired users by reading text aloud. Multitasking: Allows users to listen while performing other tasks. Education: Helps in language learning, pronunciation, and comprehension.

Convenience: Enables hands-free access to information.

Key Features of the Project: User-Friendly Interface: Clean and simple input area for entering text. Cross-Browser Compatibility: Works in most modern browsers. No External Dependencies: Uses built-in browser APIs. Real-Time Feedback: Instant audio output on user interaction.

Text-to-Speech is a powerful and practical feature in modern computing. By converting written words into spoken output, it enhances accessibility and user interaction. This project demonstrates how basic web technologies—HTML for structure, CSS for design, and JavaScript for functionality—can be used to build a useful TTS tool. This introduction lays the foundation for understanding the significance, application, and technical basis of the Text-to-Speech Converter and highlights its relevance in today's technology-driven environment.

1.2 PROBLEM STATEMENT

In today's digital world, accessibility and user engagement are critical aspects of web and software development. A significant number of users, including those with visual impairments, reading difficulties, or multitasking needs, often face challenges in accessing and consuming textual information on digital platforms.

Although a large portion of content is delivered in written form, not all users are able to easily read or comprehend this text due to various physical or situational limitations. Additionally, in educational and assistive technology domains, there is a growing demand for tools that convert text to audio in real-time to aid learning and improve user experience. The absence of simple, user-friendly, and browser-based Text to Speech (TTS) solutions creates a gap in accessibility and content usability.

Therefore, the problem is:

How can we develop a lightweight, browser-compatible web application that effectively converts user-inputted text into clear and natural-sounding speech using only front-end web technologies (HTML, CSS, and JavaScript)? This project aims to address this problem by leveraging the Web Speech API to create a responsive, accessible, and functional Text-to-Speech Converter that can be used across various devices without additional software or installations.

1.3 SCOPE OF RESEARCH

The research for this project focuses on developing and implementing a Text-to-Speech Converter using web technologies—primarily HTML, CSS, and JavaScript. The project aims to explore the capabilities, limitations, and practical applications of the Web Speech API for speech synthesis within a browser environment.

Key Areas of Research:

1. <u>Web Speech API Functionality</u>: Investigating the browser-based speech synthesis features provided by the Web Speech API, including supported languages, voice modulation, speech rate, pitch, and compatibility across browsers.

2. <u>Front-End Web Development</u>: Exploring how HTML and CSS can be used to create a clean and accessible user interface for TTS tools. Focus will be placed on creating an intuitive layout that allows users to easily input and manage text.

3. <u>User Accessibility and Experience</u>: Researching the role of TTS in enhancing digital accessibility, especially for users with visual impairments, reading disorders like dyslexia, and for language learners. This includes studying guidelines from WCAG (Web Content Accessibility Guidelines).

4. <u>Performance and Responsiveness:</u> Analyzing how efficiently the application can convert and play speech without lag or interruption. This includes evaluating different browser performances and optimization techniques for smoother speech delivery.

5. <u>Application Scenarios</u>: Identifying real-world use cases for TTS tools in education, assistive technology, e-learning, customer service bots, and content accessibility. The research will explore how TTS solutions can be customized for different user needs.

6. <u>Limitations and Improvements</u>: Exploring the current limitations of browser-based TTS (such as offline functionality, voice variety, or language support) and identifying potential areas for future enhancement, such as adding voice selection, text file input, or speech-to-text integration.

1.4 EXISTING SYSTEM

Text-to-Speech (TTS) technology has been widely adopted and integrated into various platforms, applications, and operating systems. Several existing systems provide TTS functionality, both online and offline, using advanced natural language processing and speech synthesis techniques. These systems have been developed for accessibility, education, virtual assistance, and general user convenience.

1. Operating System-Based TTS

Windows Narrator: Built into Windows OS, Narrator is a screen reader that reads text and UI elements aloud. It supports basic navigation and speech customization.
macOS Voice Over: A screen reader in Apple's macOS that reads aloud text and allows users to interact with their device using keyboard shortcuts and gestures.
Android Talk Back / iOS Voice Over: Built-in accessibility features in smartphones that read on-screen content for visually impaired users.

2. Online TTS Services and Applications

Google Text-to-Speech: Integrated into Android and Google services, this API powers speech features in apps like Google Translate and Google Assistant.

Amazon Polly: A cloud-based TTS service by AWS that converts text into lifelike speech using deep learning. It supports multiple languages and voice styles.

IBM Watson Text to Speech: IBM's AI-based service that provides natural-sounding voice output with high customizability and language support.

Responsive Voice & I Speech: These are browser-based TTS services that can be embedded into websites to read content aloud. They require internet access and often come with licensing fees for full access.

3. Open-Source Libraries

eSpeak: A compact, open-source speech synthesizer for English and other languages, available for multiple platforms.

Festival: A free, multilingual speech synthesis system developed by the University of Edinburgh.

Flite (Festival Lite):A small, fast run-time speech synthesis engine suitable for embedded systems and mobile devices.

4. Browser-Based Solutions

Web Speech API (SpeechSynthesis): Supported in most modern browsers, this API allows developers to create lightweight TTS tools using JavaScript. It supports multiple voices and basic speech controls like rate and pitch, making it ideal for simple applications.

1.4.1 DISADVANTAGES OF THE EXISTING SYSTEM

While current text-to-speech systems are widely used and offer various features, they still come with several drawbacks that can limit their usability, accessibility, or efficiency in certain contexts.

1. Dependence on Internet Connectivity: Many cloud-based TTS services (like Google Cloud TTS, Amazon Polly, IBM Watson) require a constant internet connection to function. This can be a limitation for users in remote areas or where reliable internet is not available.

2. Cost and Licensing Issues: Most advanced and high-quality TTS APIs are paid services with subscription-based pricing. This makes them less suitable for small projects, student use, or low-budget applications. Even browser-based third-party TTS tools may require licensing for commercial use.

3. Limited Customization in Built-in Systems: Default operating system TTS (e.g., Windows Narrator, macOS VoiceOver) often offer minimal customization options in terms of voice tone, pitch, language variety, or emotional expression. They are not always flexible for developers or users seeking personalized experiences.

4. Complexity in Integration: APIs provided by major platforms can be technically complex to integrate, especially for beginners or in projects where simplicity and lightweight functionality are preferred. Developers often need API keys, handle authentication, and manage server interactions.

5. Inconsistent Browser Support: While the Web Speech API is a promising browser-native solution, it still has:

Inconsistent behavior across different browsers. Limited support for offline usage. Fewer voices and options on certain platforms (e.g., mobile vs. desktop).

6. Privacy Concerns: Cloud-based TTS services may transmit user input to external servers for processing, raising concerns about data privacy and security, especially when dealing with sensitive information.

7. Lack of Multilingual and Natural Speech Support: Some TTS systems offer limited language support or robotic-sounding voices, which affect the naturalness and effectiveness of communication, especially in applications meant for education or customer interaction.

8. Device and Platform Dependency: Some systems are tightly coupled with specific devices or platforms (e.g., Android, Windows), making them inaccessible or inconsistent across different environments.

1.5 PROPOSED SYSTEM

To overcome the limitations of existing Text-to-Speech (TTS) solutions, this project proposes the development of a lightweight, browser-based Text-to-Speech Converter using only HTML, CSS, and JavaScript. The system utilizes the Web Speech API, which is supported by most modern web browsers, to convert user-inputted text into spoken audio without the need for internet-based services or external APIs.

Key Features of the Proposed System :

1. Browser-Based Functionality: The application will work entirely within the browser, eliminating the need for server-side processing or internet connectivity once loaded.

2. No External Dependencies: Unlike many existing TTS services, this system will not rely on third-party APIs, making it free to use and easy to deploy in any environment.

3. User-Friendly Interface: A clean and simple UI will be created using HTML and CSS, allowing users to input text easily and control speech playback with the click of a button.
4. Customizable Speech Output: Using JavaScript, users can adjust parameters such as: Voice selection (if supported by the browser), Speaking rate, Pitch, and volume.
5. Accessibility Focused: The system will be designed with accessibility in mind, making it a valuable tool for users with visual impairments, reading difficulties, or learning disabilities.
6. Cross-Platform Compatibility: As it is web-based, the tool can run on various devices and platforms (PC, tablet, smartphone) without any additional installations.

1.5.1 ADVANTAGES OF PROPOSED SYSTEM

The proposed Text to Speech Converter system, built using HTML, CSS, and JavaScript, offers several advantages over existing solutions, especially in terms of simplicity, accessibility, and platform independence.

1. No Internet Required: Once the web page is loaded, the application can function offline, as it uses the browser's built-in Web Speech API. This makes it ideal for users with limited or no internet access.

2. Cost-Effective and Open Source: The entire system is developed using free and open-source technologies. There are no subscription fees, licensing requirements, or API usage limits, making it highly suitable for students, small projects, and educational institutions.

3. Easy to Use: With a minimalist and intuitive user interface, users can easily input text and generate speech with a single click. No technical knowledge is required to operate the application.

4. No Installation Needed: Since the application runs entirely in a web browser, it does not require any installation. It is compatible with most modern browsers, including Chrome, Firefox, Edge, and Safari.

5. Highly Portable and Cross-Platform: Being web-based, the TTS converter can be accessed from any device with a browser—desktop, laptop, tablet, or smartphone—making it highly flexible and portable.

6. Enhanced Accessibility: The system improves accessibility for users with: Visual impairments, Reading disorders (like dyslexia), Learning difficulties It allows them to consume text content in an audio format, promoting digital inclusion.

7. Customizable Speech Settings: The JavaScript-based implementation allows users to control: Voice selection (if multiple voices are supported by the browser), Speech rate, Pitch, and volume. This ensures a more natural and personalized listening experience.

8. Lightweight and Fast: The system is lightweight, requiring minimal resources and loading quickly in any browser. It does not depend on heavy libraries or back-end servers.

9. Privacy Friendly: All text processing and speech synthesis occur locally in the user's browser, ensuring that no data is transmitted to external servers. This makes it safer for handling sensitive or personal content.

10. Easy Integration: The system can be easily embedded into any website or learning platform. It's perfect for adding voice features to educational tools, e-books, and assistive web applications.

CHAPTER 2
LITERATURE REVIEW

Books:

1. "Speech and Language Processing" by Daniel Jurafsky and James H. Martin

This book is considered a cornerstone in the field of computational linguistics and speech processing. It covers a wide range of topics such as speech synthesis, automatic speech recognition, natural language processing, and machine translation. The sections on TTS delve into phonetics, prosody, and various synthesis techniques. Its blend of theory and practical applications makes it ideal for understanding foundational concepts.

2. "Text-to-Speech Synthesis" by Paul Taylor

Dedicated entirely to the field of TTS, this book is an excellent resource for learning about the design and implementation of TTS systems. It goes into depth about concatenative synthesis, parametric synthesis, and statistical models. The book also addresses the challenges of prosody and the importance of linguistic and phonetic rules.

3. "Deep Learning for Natural Language Processing" by Palash Goyal, Sumit Pandey, and Karan Jain

As neural network-based synthesis becomes a dominant approach in TTS, this book is a valuable resource for learning deep learning techniques. It covers topics like recurrent neural networks (RNNs), attention mechanisms, and their applications in natural language tasks, including text-to-speech.

4. "Foundations of Voice Technologies" by Juergen Schroeter and Alexander I. Rudnicky

- This book offers insights into voice technologies like TTS and voice recognition. It is well-suited for understanding the interdisciplinary aspects of speech systems, combining elements from electrical engineering, computer science, and linguistics.

5. "Artificial Intelligence: A Guide to Intelligent Systems" by Michael Negnevitsky

While not exclusively focused on TTS, this book provides an overview of AI principles that are crucial for understanding the computational models used in speech synthesis, including neural networks and fuzzy logic.

Articles:

1. Journal Articles from IEEE Transactions on Audio, Speech, and Language Processing:

These journals frequently publish state-of-the-art research on TTS systems. Some notable topics include neural network-based synthesis methods (like WaveNet), prosody modeling, and low-resource language adaptation.

2. Research Papers on ACM Transactions on Speech and Language Processing:

ACM papers explore innovative speech synthesis technologies and their applications. Look for articles on multilingual TTS systems, context-aware synthesis, and emotional expressiveness in speech.

3. Papers on ArXiv.org:

ArXiv hosts many cutting-edge papers on deep learning models for TTS, including advancements in Tacotron, WaveNet, and recent AI innovations in the field.

4. "WaveNet: A Generative Model for Raw Audio" by Äron van den Oord et al.

This seminal paper introduces WaveNet, which revolutionized TTS with high-quality, natural speech synthesis.

5. "Tacotron: Towards End-to-End Speech Synthesis" by Jonathan Shen et al.

Tacotron papers discuss the shift towards end-to-end models, which bypass traditional pipeline methods to achieve efficient and natural speech generation.

How to Access These Resources:

- Google Scholar: Use this tool to find academic papers on TTS systems, linguistic analysis, and neural synthesis methods.
- ResearchGate: Many authors share their work here, offering accessible insights into their methodologies and findings.
- Institutional Libraries: If you are affiliated with a university or institution, their library may offer free access to journals and books.
- Book Retailers and Platforms: Websites like Amazon and Springer often carry the books mentioned above.

CHAPTER 3

METHODOLOGIES

3.1 System Architecture

The system architecture is designed as a client-side web application that integrates user interface elements with browser-native speech synthesis capabilities. It is composed of three key layers:

1. User Interface (UI Layer)

This is the frontend that users interact with:

Text area: For inputting the text that will be converted to speech.

Select Dropdown: Lists available system voices for the user to choose from.

Button: Triggers the speech output when clicked.

2. JavaScript Logic (Application Layer)

This layer handles dynamic behavior and user interaction:

Voice Loading: Captures the list of available voices once the browser provides them.

Voice Selection: Updates the selected voice dynamically when the user changes the dropdown.

Speech Synthesis Control: Prepares and sends the speech request with the selected voice and user-input text.

3. Web Speech API (Speech Engine Layer)

This is the browser-provided system responsible for actual text-to-speech conversion:

Speech Synthesis Utterance: An Object that holds speech data like voice and text.

Speech Synthesis: Interface to control speaking, pausing, or canceling speech. speak () method: Executes the speech request.

Together, these layers form a responsive and interactive text-to-speech system where the frontend and browser API work together in real-time to provide speech output.

Benefits of system architecture:

1. Modular Design: The architecture separates the application into distinct components: the user interface (HTML), the application logic (JavaScript), and the speech processing (Web Speech API). This modularity improves maintainability, as each component can be developed, tested, and modified independently without affecting the others.

2. Dynamic Voice Loading: The use of the ‘onvoiceschanged’ event allows the application to load system voices dynamically. This is crucial because different browsers and devices load voices at different times. By waiting for the voices to be ready, the system ensures a smooth and reliable user experience across platforms.

3. Scalability and Extensibility: This architecture makes it easy to scale the system by adding more features. For example, developers can add controls for speech rate, pitch, and volume, or even allow saving speech as audio. The separation of responsibilities means new features can be added with minimal impact on existing functionality.

4. Real-Time User Interaction: The system responds to user actions (like selecting a voice or clicking the speak button) immediately, offering a dynamic and interactive experience. This responsiveness is key to user satisfaction and aligns with modern web application standards.

5. Utilizes Built-in Browser Capabilities:

The architecture leverages the Web Speech API, which is natively supported in most modern browsers. This eliminates the need for third-party libraries or external APIs, reducing complexity, increasing performance, and ensuring long-term support and stability.

6. Cross-Platform and Device Compatibility: Since the system is built entirely with standard web technologies (HTML, CSS, JavaScript), it works seamlessly across desktops, laptops, tablets, and smartphones on any platform that supports the Web Speech API.

7. Lightweight and Fast: The application runs entirely on the client side, meaning no server interaction is required. This makes the system fast, with minimal latency, and reduces infrastructure costs since there's no need for backend processing.

8. Low Learning Curve: The architecture is simple and easy to understand, making it accessible to beginner developers. This is beneficial for educational purposes or for integrating speech capabilities into larger projects.

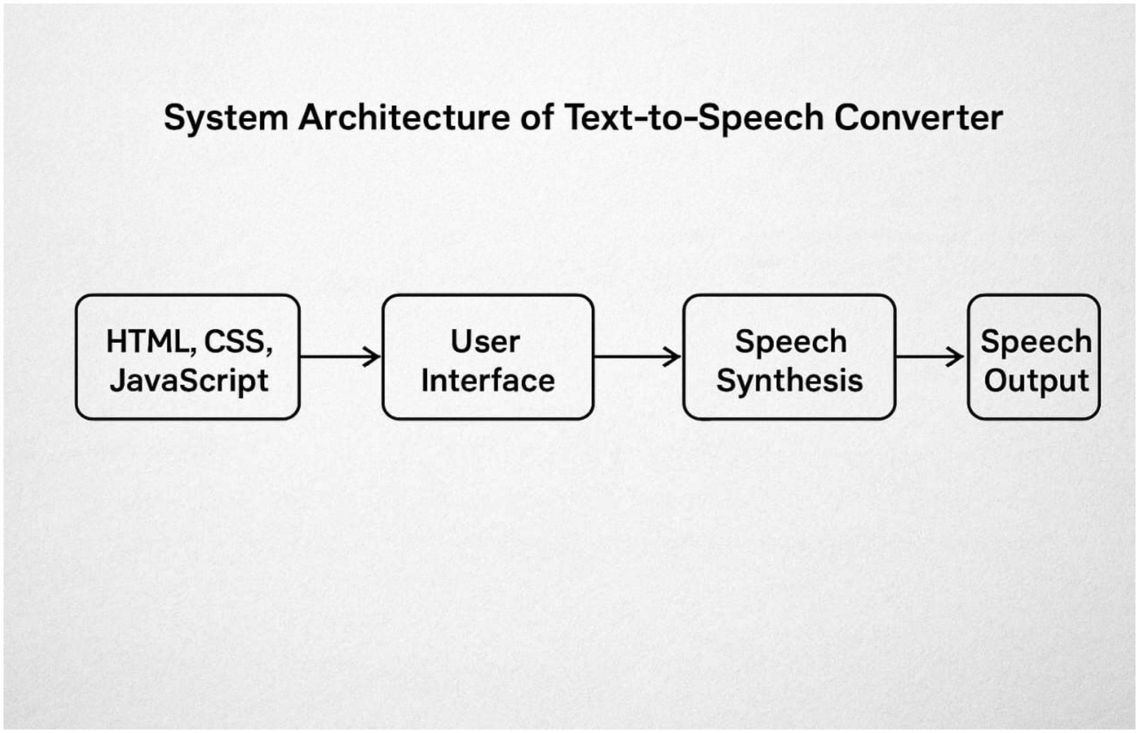


Fig3.1.1: System Architecture of Type2tone

3.2 ALGORITHM

1. Initialize components

- Create a new SpeechSynthesisUtterance instance.
- Create an empty array to store voice options.
- Get references to HTML elements (textarea, select dropdown, and button).

2. Load voices

When the voices changed, the events fired:

- Retrieve all available voices using getVoices().
- Store them in the voices array.
- Set the default voice for the utterance.
- Populate the dropdown menu with the voice names.

3. Handle voice selection

When the user selects a voice from the dropdown:

- Set the voice property of the utterance to the selected voice.

4. Handle speech

When the button is clicked:

- Get the text from the textarea.
- Set it as the text property of the utterance.
- Use speechSynthesis.speak() to speak the utterance.

CHAPTER 4

SYSTEM REQUIREMENTS SPECIFICATION

4.1 SOFTWARE REQUIREMENTS:

A text-to-speech (TTS) converter project typically involves software requirements that cover functionality, performance, and compatibility. Here's a brief explanation:

1. Input Processing:

- Text Input: The system must accept text input in various formats (e.g., plain text, UTF-8, or files like .txt, .docx).
- Language Support: Support for multiple languages and character sets (e.g., English, Spanish, Unicode for non-Latin scripts).
- Preprocessing: Ability to clean and normalize text (e.g., handling punctuation, abbreviations, or special characters).

2. Speech Synthesis:

- TTS Engine: A core engine like Google TTS, Amazon Polly, Microsoft Azure TTS, or open-source options (e.g., eSpeak, Festival) to convert text to audio.
- Voice Options: Multiple voice profiles (male, female, accents) with customizable pitch, speed, and volume.
- Naturalness: Support for prosody (intonation, stress) and context-aware pronunciation for natural-sounding output.

3. Output Requirements:

- Audio Formats: Export audio in common formats (e.g., MP3, WAV) with configurable quality (bitrate, sample rate).
- Real-Time Processing: Low-latency conversion for real-time applications (e.g., virtual assistants).
- Accessibility: Compatibility with screen readers or assistive technologies.

4. Platform and Integration:

- Operating System: Cross-platform support (Windows, macOS, Linux, Android, iOS) or specific platform targeting.
- APIs/SDKs: Integration with APIs (e.g., Web Speech API, cloud-based TTS services) or libraries (e.g., gTTS, pyttsx3 for Python).

- User Interface: A simple GUI or CLI for user interaction, with options to input text, select voices, and save output.

5. Performance and Scalability:

- Speed: Fast processing for large text inputs or batch conversions.
- Resource Usage: Optimized for low CPU/memory usage, especially for mobile or embedded systems.
- Scalability: Cloud-based systems should handle multiple concurrent requests.

6. Development Tools:

- Programming Languages: Python, JavaScript, or C++ for flexibility and library support.
- Libraries/Frameworks: Libraries like SpeechSynthesis (browser-based), Tacotron, or DeepSpeech for advanced synthesis.
- Dependencies: External APIs or pre-trained models for neural TTS (e.g., WaveNet).

7. Non-Functional Requirements:

- Reliability: Consistent output quality across inputs.
- Security: Secure handling of user data, especially for cloud-based TTS services.
- Maintainability: Modular code for easy updates or voice additions.

8.Optional Features:

- Offline Mode: Local TTS engine for offline use.
- Customization: User-defined dictionaries for specific pronunciations.
- Multimodal Support: Integration with other systems (e.g., chatbots, IoT devices).

4.2 HARDWARE REQUIREMENTS

Hardware requirements refer to the specific physical components and capabilities a computer system must possess to effectively run a particular software application. These specifications ensure that the software operates smoothly, efficiently, and without errors. Meeting the minimum hardware requirements is essential for basic functionality, while adhering to the recommended specifications can enhance performance, especially when handling larger datasets or more complex tasks.

For a Text-to-Speech (TTS) Converter Project, the hardware requirements are generally very simple because most of the heavy work is done in software. But depending on how advanced your project is (basic web app vs. industrial-level system), requirements can vary.

The hardware requirements for a text-to-speech (TTS) converter project depend on the project's scope. Here's a brief overview:

1. Processing Power (CPU/GPU):

- Basic Projects: A standard CPU is sufficient for lightweight TTS engines or API-based solutions (e.g., Google TTS).
- Advanced Projects: Neural TTS models require more powerful CPUs or GPUs (e.g., NVIDIA GPUs with CUDA support) for real-time processing or training custom models.
- Embedded Systems: Low-power processors for offline TTS with optimized engines.

2. Memory (RAM):

- Minimum: 4-8 GB RAM for basic TTS applications or cloud-based API integrations.
- Recommended: 16 GB or more for neural TTS models or handling large text inputs, especially during development or training.
- Embedded Devices: 512 MB to 2 GB for lightweight offline TTS engines.

3. Storage:

- Basic Needs: 1-5 GB for TTS software, libraries, and pre-trained models (e.g., open-source engines like Festival).
- Neural Models: 10-50 GB for large models or voice datasets, especially for custom voice synthesis.
- Cloud-Based: Minimal local storage (just for the app), as models are hosted remotely.

4. Audio Output:

- Speakers/Headphones: Standard audio output devices for testing and playback.

- Sound Card: Basic integrated sound cards are sufficient for most projects, though high-quality DACs may be needed for professional-grade audio output.

5. Network (for Cloud-Based TTS):

- Internet Connection: Stable broadband (5-10 Mbps minimum) for real-time API calls to cloud services.
- Latency: Low-latency networks for applications requiring instant audio output (e.g., virtual assistants).

6. Platform-Specific Hardware:

- Desktop/Laptop: Standard systems with Windows, macOS, or Linux for development and testing.
- Mobile Devices: Mid-range smartphones/tablets (e.g., 2-4 GB RAM, quad-core processors) for mobile TTS apps.

7. Optional Hardware:

- Microphone: For testing voice-related features or integrating speech-to-text with TTS.
- Edge Devices: For on-device neural TTS (e.g., Google Coral) to reduce reliance on cloud services.

Notes:

- Offline vs. Online: Offline TTS requires more local processing power and storage for models, while cloud-based TTS shifts demands to servers, needing only minimal client hardware.
- Scalability: For enterprise-grade systems, server-grade hardware (e.g., multi-core CPUs, high RAM, GPU clusters) may be needed for handling concurrent requests.
- Power Efficiency: Critical for mobile or IoT devices to ensure low battery consumption.

4.3 LANGUAGES/TECHNOLOGIES USED

A web-based TTS converter uses HTML for structure, CSS for styling, and JavaScript for functionality, leveraging the browser's built-in Web Speech API for text-to-speech conversion. This approach is lightweight, cross-platform, and doesn’t require external libraries or servers for basic functionality.

1. HTML (HyperText Markup Language):

Role: Provides the structure of the TTS web application.

Usage:

- Create input fields for users to enter text (e.g., `<textarea>` or `<input>`).
- Add buttons for triggering speech synthesis (e.g., "Speak", "Stop").
- Include dropdowns or selectors for choosing voices, pitch, or speed.

Example: A `<textarea>` for text input and a `<button>` to initiate speech.

2. CSS (Cascading Style Sheets):

Role: Styles the user interface for a clean, responsive, and user-friendly experience.

Usage:

- Style the text input area, buttons, and voice selection dropdowns.
- Ensure responsiveness for mobile and desktop devices using media queries.
- Apply visual feedback (e.g., button hover effects, loading animations).

3. JavaScript:

Role: Handles the logic for text-to-speech conversion using the Web Speech API.

Usage:

- Access the `SpeechSynthesis` interface to convert text to speech.
- Retrieve available voices (via `speechSynthesis.getVoices()`) and populate a dropdown.
- Configure speech parameters like voice, pitch, rate, and volume.
- Handle user interactions (e.g., clicking "Speak" to start synthesis or "Stop" to cancel).

Key Features:

- Text Input Processing: Read text from the `<textarea>`.
- Speech Synthesis: Create a `SpeechSynthesisUtterance` object to define the text and settings.
- Dynamic Voice Selection: Allow users to choose from available voices (browser-dependent, e.g., Google or Microsoft voices).
- Event Handling: Manage start, stop, or error events during speech synthesis.

4. Web Speech API:

Role: Browser-native API for TTS, eliminating the need for external libraries.

Features:

- SpeechSynthesis: Core interface for controlling speech synthesis.
- SpeechSynthesisUtterance: Object to define text, voice, and parameters (pitch, rate, volume).
- speechSynthesis.getVoices(): Lists available voices (varies by browser and OS).
- Supports multiple languages and voices (e.g., Chrome includes Google voices, Edge includes Microsoft voices).

Limitations:

- Voice availability depends on the browser and platform.
- No offline support in some browsers.
- Limited customization compared to neural TTS APIs (e.g., Google Cloud TTS).

5. Optional Technologies:

- External APIs: For advanced features, integrate cloud-based TTS APIs (e.g., Google Cloud TTS, Amazon Polly) via JavaScript's `fetch` or `axios` for HTTP requests. This requires API keys and server-side handling for security.
- Audio Processing: Use the Web Audio API to manipulate or enhance audio output (e.g., adding effects).
- Frameworks: Use React, Vue, or Angular for scalable UI development, though plain JavaScript is sufficient for simple projects.

Sample Workflow

1. HTML: Create a form with a `<textarea>` for text, a `<select>` for voice selection, and buttons for "Speak" and "Stop".

2. CSS: Style the interface to be responsive and visually appealing.

3. JavaScript:

- Use the Web Speech API to read text and synthesize speech.
- Dynamically populate voice options and handle user inputs.
- Add event listeners for buttons to trigger or cancel speech.

4. Testing: Test across browsers (Chrome, Firefox, Edge) to ensure voice compatibility and performance.

Key Considerations

- Browser Support: Web Speech API is supported in most modern browsers (Chrome, Edge, Safari, Firefox), but voice quality and availability vary.
- Performance: Minimal hardware requirements since processing is handled by the browser.
- Limitations: Basic TTS quality compared to neural APIs; no advanced prosody control.
- Scalability: For advanced features (e.g., custom voices), integrate cloud APIs, requiring additional JavaScript logic for API calls.

4.4 UML Diagrams

Use Case Diagram

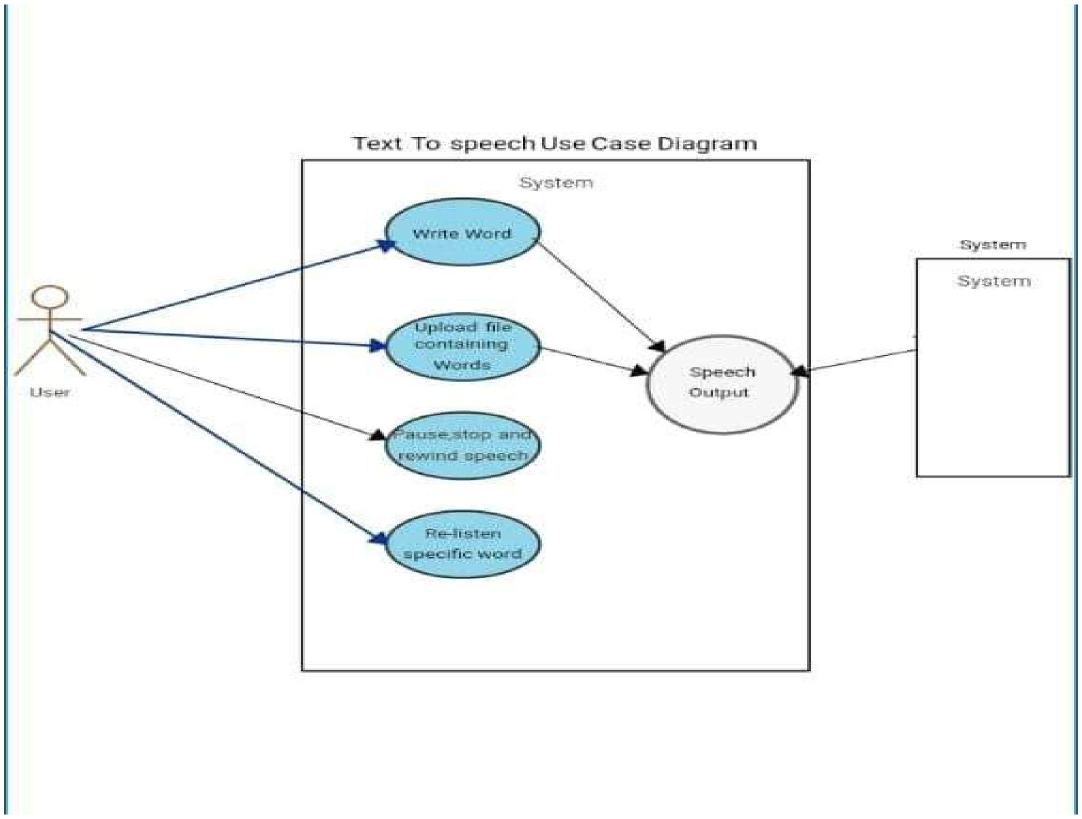

Text To speech Use Case Diagram
System
Write Word
Upload file containing Words
Pause,stop and rewind speech
Re-listen specific word
Speech Output
User
System
System

Fig 4.4.1: Use Case diagram of Type2Tone

A use case diagram for a **text-to-speech converter** built with HTML, CSS, and JavaScript represents the interactions between users and the system's functionalities. In this case, the primary **actor** is the **End User**, who interacts with the converter to input text, generate speech, and listen to the output. Additionally, an **Administrator** may exist to configure settings or enhance the system. The key **use cases** include entering text, converting it into speech, playing the audio, adjusting voice settings such as speed and pitch, downloading the generated speech, and supporting multiple languages if integrated. The **relationships** within the system show that the **End User** interacts with all core functions, while the **Administrator**, if applicable, handles system configurations.

Sequence Diagram

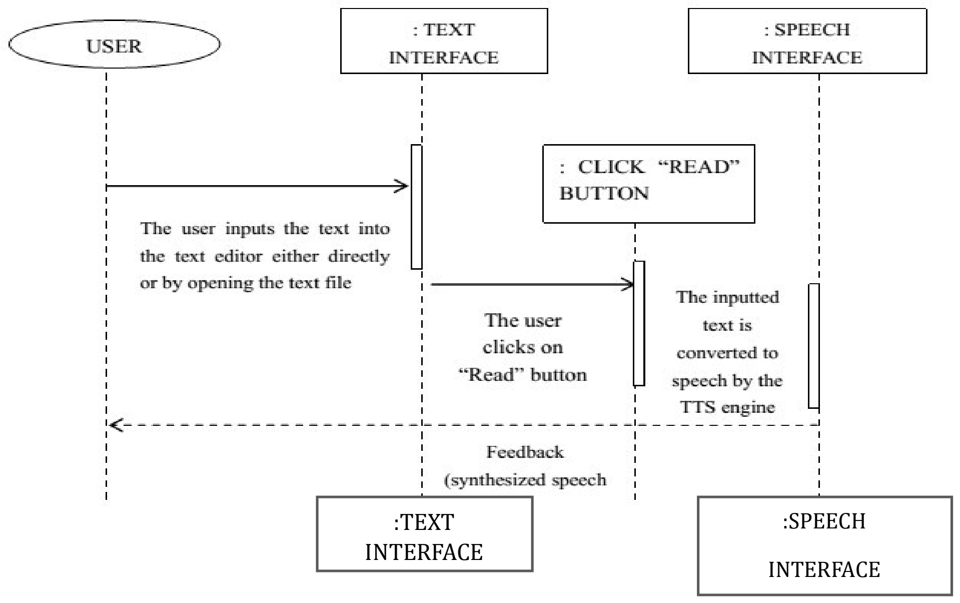

USER
: TEXT INTERFACE
: SPEECH INTERFACE
: CLICK "READ" BUTTON
The user inputs the text into the text editor either directly or by opening the text file
The user clicks on "Read" button
The inputted text is converted to speech by the TTS engine
Feedback (synthesized speech
:TEXT INTERFACE
:SPEECH INTERFACE

Fig4.4.2: Sequence diagram of Type2Tone

A sequence diagram for a text-to-speech converter built using HTML, CSS, and JavaScript illustrates the step-by-step interactions between the user, system components, and external processing elements over time. The sequence begins when the user enters text into the converter's interface, triggering the system to process the input. The system then forwards the text to a speech engine or API, which converts it into audio data. Once the audio is generated, the system retrieves the output and plays the speech for the user. If the converter supports customization, the user may adjust settings such as speed, pitch, or language preferences, which the system processes accordingly before applying changes to the speech synthesis. Additionally, if a download feature is available, the system allows the user to save the generated speech as an audio file. Throughout this process, the interaction follows a structured flow, ensuring seamless communication between user actions and system responses. By representing these interactions visually in a sequence diagram, developers can better understand how components interact, optimizing functionality and improving accessibility for users.

Activity Diagram

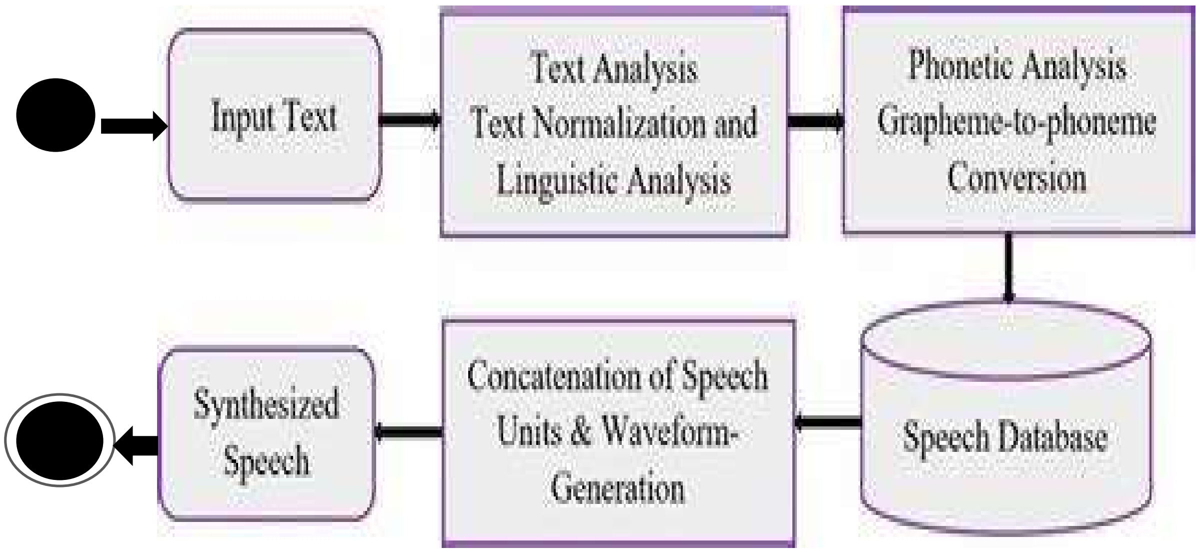


Fig4.4.3: Activity diagram of Type2Tone

An activity diagram for a text-to-speech converter built using HTML, CSS, and JavaScript represents the flow of operations, illustrating how different processes are executed step by step. The diagram starts when the user enters text into the system, triggering the first activity—text input validation—where the system checks for any unsupported characters or formatting errors. Once validated, the system sends the text to the speech synthesis engine or API, initiating the conversion process. The next step is audio generation, where the engine processes the input and creates corresponding speech output.

After the speech is generated, the system provides options for the user. The user may play the generated speech, listen to the output, or adjust voice settings such as pitch, speed, and language preferences. If modifications are applied, the system updates the speech synthesis accordingly before playing the refined version. Additionally, if the user wishes to keep the output for later use, they can download the generated audio file, completing the process. The activity diagram visually maps these interactions, ensuring clarity in understanding the system's workflow, making it useful for refining functionality and enhancing accessibility.

CHAPTER 5

IMPLEMENTATION AND RESULTS

1. Project Setup:

Objective: Create a web-based TTS converter where users input text, select a voice, and hear the text spoken aloud.

Tools

- HTML: Structures the user interface (text input, voice selector, buttons).
- CSS: Styles the interface for a clean, responsive design.
- JavaScript: Handles TTS logic using the Web Speech API.
- Environment: Any modern browser (e.g., Chrome, Edge, Firefox) with Web Speech API support.
- Files: A single `index.html` file containing HTML, inline CSS, and JavaScript (or separate 'css' and 'js' files for modularity).

2. Development Steps:

Step 1: Create the HTML structure with input fields and buttons.

Step 2: Style the interface using CSS for usability and aesthetics.

Step 3: Implement JavaScript to:

- Initialize the Web Speech API.
- Load voices dynamically when the browser makes them available.
- Handle user interactions (text input, voice selection, speak/stop actions).

Step 4: Test the application in multiple browsers to ensure compatibility.

Step 5: Deploy the project (e.g., host on GitHub Pages or a web server) or run locally via a browser.

3. Optional Enhancements:

- Add sliders for pitch, rate, and volume control.
- Include error handling (e.g., alert if no text is entered).
- Integrate a cloud-based TTS API (e.g., Google Cloud TTS) for advanced voices, requiring `fetch` requests and API key management.

5.1.2 WEB APPLICATION CODE

1. HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Type2Tone</title>
  <link rel="stylesheet" href="project1.css">
  <link rel="icon" href="icon.jpg"
    type="image/x-icon" />
</head>
<body>
  <div class="hero">


    <h1>Type2Tone</h1>
    <textarea placeholder="write anything here...."></textarea>
    <div class="row">
     <select></select>
     <button>Listen</button>
     </div>
  </div>
<script src="project1.js"></script>
</body>
</html>
```

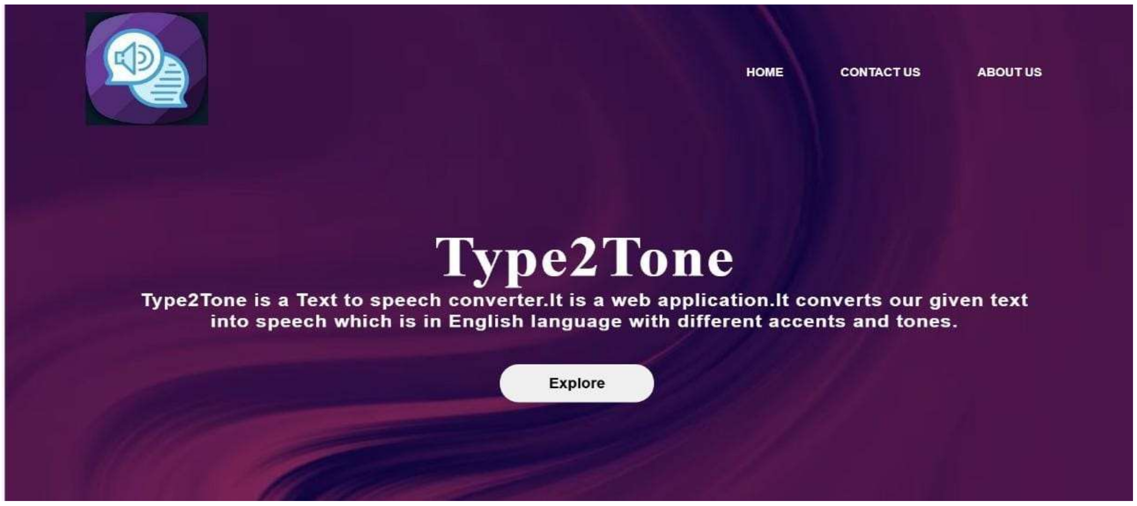
2. CSS

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Type2Tone</title>
  <link rel="stylesheet" href="project1.css">
  <link rel="icon" href="icon.jpg"
    type="image/x-icon" />
</head>
<body>
  <div class="hero">
    <h1>Type2Tone</h1>
   <textarea placeholder="write anything here....."></textarea>
   <div class="row">
    <select></select>
    <button>Listen</button>
    </div>
  </div>
<script src="project1.js"></script>
</body>
</html>
```

3. JAVASCRIPT

```
let speech = new SpeechSynthesisUtterance();
let voices = [];
let voiceSelect = document.querySelector("select");
window.speechSynthesis.onvoiceschanged =() => {
  voices = window.speechSynthesis.getVoices();
  speech.voice = voices[0];
 voices.forEach((voice, i) => (voiceSelect.options[i] = new Option(voice.name, i)));
};
voiceSelect.addEventListener("change", () =>{
   speech.voice = voices[voiceSelect.value];
});
document.querySelector("button").addEventListener("click", () =>{
   speech.text = document.querySelector("textarea").value;
   window.speechSynthesis.speak(speech);
});
```

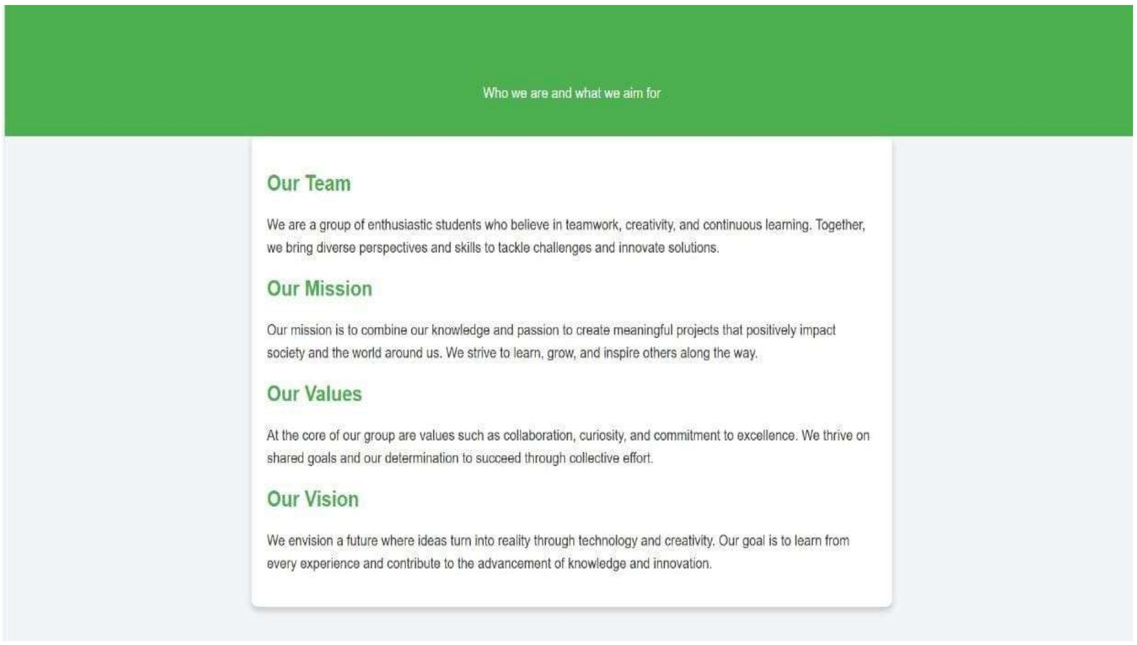
5.2 OUTPUT



Type2Tone

Type2Tone is a Text to speech converter.It is a web application.It converts our given text into speech which is in English language with different accents and tones.

Fig5.2.1: Homepage, where you access other pages in this website.



Who we are and what we aim for

Our Team

We are a group of enthusiastic students who believe in teamwork, creativity, and continuous learning. Together, we bring diverse perspectives and skills to tackle challenges and innovate solutions.

Our Mission

Our mission is to combine our knowledge and passion to create meaningful projects that positively impact society and the world around us. We strive to learn, grow, and inspire others along the way.

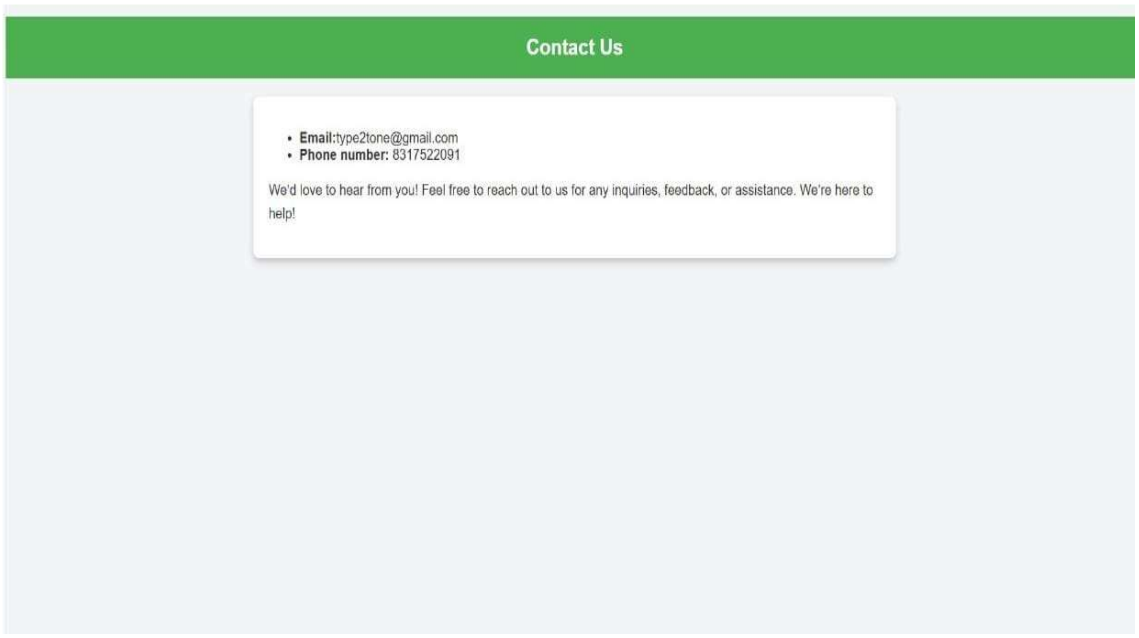
Our Values

At the core of our group are values such as collaboration, curiosity, and commitment to excellence. We thrive on shared goals and our determination to succeed through collective effort.

Our Vision

We envision a future where ideas turn into reality through technology and creativity. Our goal is to learn from every experience and contribute to the advancement of knowledge and innovation.

Fig5.2.2: About Us page.



Contact Us

- **Email:**type2tone@gmail.com
- **Phone number:** 8317522091

We'd love to hear from you! Feel free to reach out to us for any inquiries, feedback, or assistance. We're here to help!

Fig5.2.3: Contact Us page, where you can contact us if there is any issue.

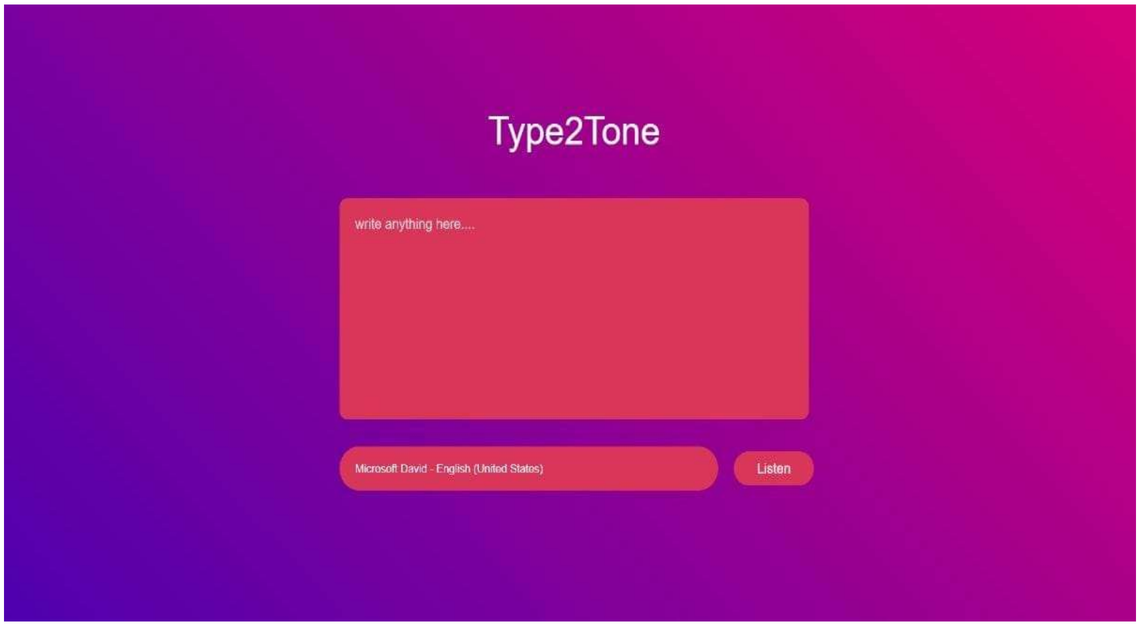
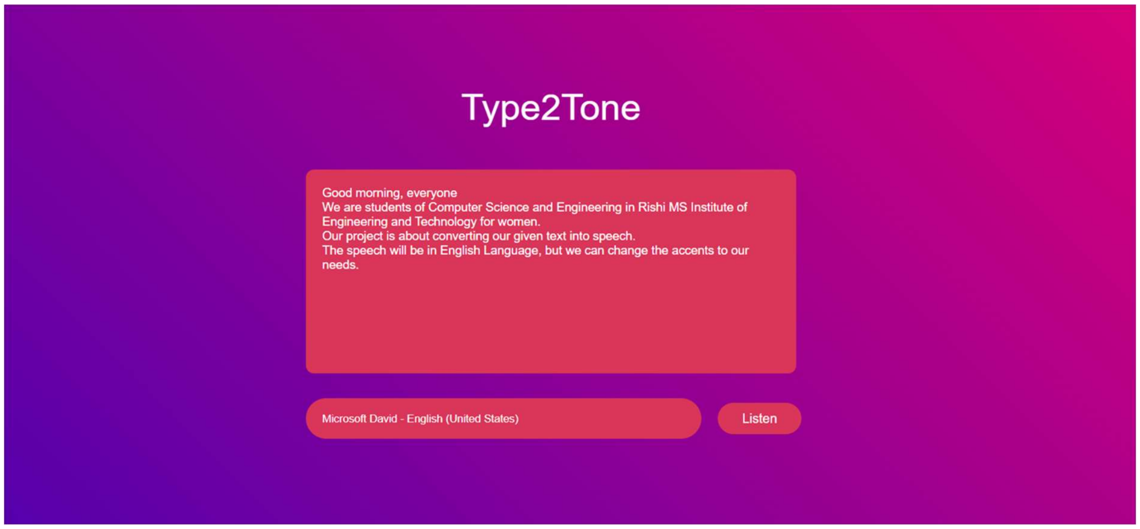


Fig5.2.4: Main page, where we can enter our text and we will get speech as output.



Good morning, everyone
We are students of Computer Science and Engineering in Rishi MS Institute of Engineering and Technology for women.
Our project is about converting our given text into speech.
The speech will be in English Language, but we can change the accents to our needs.

Fig5.2.5: After entering the test into the textarea.

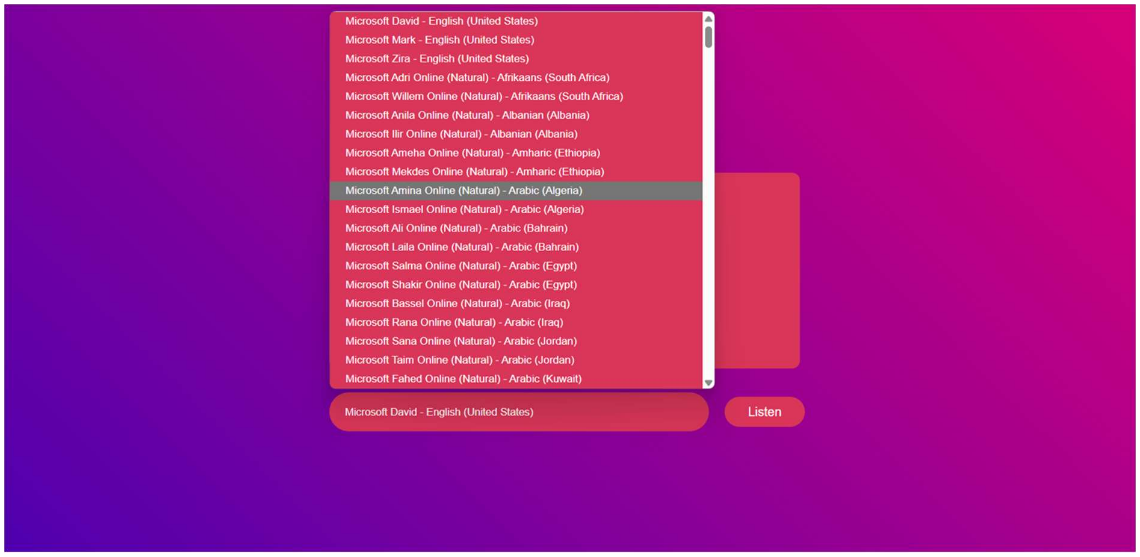

Microsoft David - English (United States)
Microsoft Mark - English (United States)
Microsoft Zira - English (United States)
Microsoft Adri Online (Natural) - Afrikaans (South Africa)
Microsoft Willem Online (Natural) - Afrikaans (South Africa)
Microsoft Anila Online (Natural) - Albanian (Albania)
Microsoft Ilir Online (Natural) - Albanian (Albania)
Microsoft Ameha Online (Natural) - Amharic (Ethiopia)
Microsoft Mekdes Online (Natural) - Amharic (Ethiopia)
Microsoft Amina Online (Natural) - Arabic (Algeria)
Microsoft Ismael Online (Natural) - Arabic (Algeria)
Microsoft Ali Online (Natural) - Arabic (Bahrain)
Microsoft Laila Online (Natural) - Arabic (Bahrain)
Microsoft Salma Online (Natural) - Arabic (Egypt)
Microsoft Shakir Online (Natural) - Arabic (Egypt)
Microsoft Bassel Online (Natural) - Arabic (Iraq)
Microsoft Rana Online (Natural) - Arabic (Iraq)
Microsoft Sana Online (Natural) - Arabic (Jordan)
Microsoft Taim Online (Natural) - Arabic (Jordan)
Microsoft Fahed Online (Natural) - Arabic (Kuwait)
Microsoft David - English (United States)
Listen

Fig5.2.6: Selecting required tone.

CHAPTER 6

TESTING

Testing the Text-to-Speech (TTS) Converter is crucial to ensure its functionality, accessibility, and compatibility across different devices and browsers. Functional testing is the first step, where the core features such as text input, voice selection, pitch and rate adjustment, and playback controls (play, pause, stop) are thoroughly verified. Each interaction should produce accurate speech output without glitches or unexpected behavior. Next, compatibility testing ensures the converter operates smoothly on major browsers like Chrome, Firefox, Edge, and Safari, as well as on both desktop and mobile devices.

Responsive UI design must be maintained to accommodate varying screen sizes. Accessibility testing focuses on making the tool user-friendly for individuals with disabilities, ensuring support for keyboard navigation, screen readers, and clear speech synthesis. Performance testing evaluates the speed and efficiency of the speech synthesis process, checking for any delays in playback and optimizing voice transitions for a smooth experience. Additionally, edge case testing is essential for handling various scenarios, such as long text inputs, special characters, and multilingual support, as well as ensuring the tool gracefully manages empty or erroneous inputs without crashing. A well-rounded approach to testing improves reliability, enhances user experience, and identifies potential areas for future enhancements such as more natural voice synthesis or expanded language support.

CHAPTER 7
RESULT

1. Functional Outcomes:

•Text-to-Speech Conversion: Users can input any text, select a voice, and hear it spoken through the browser's audio output.

•Voice Selection: The dropdown lists available voices (e.g., Google US English, Microsoft David) based on the browser and OS.

•Platform: Works on any device with a modern browser (desktop, mobile, tablet).

2. Performance:

•Speed: Near-instant speech synthesis for short texts, with minimal latency (browser-dependent).

•Resource Usage: Lightweight, requiring no significant CPU or memory since the Web Speech API leverages browser capabilities.

•Scalability: Suitable for individual use; for multi-user scenarios, a cloud API would be needed.

3. User Experience:

•Interface: Clean, intuitive UI with a responsive design that adapts to different screen sizes.

•Accessibility: Supports users with visual impairments by enabling text-to-speech functionality.

•Customization: Limited to browser-provided voices and basic parameters (pitch, rate), but sufficient for basic TTS needs.

4. Limitations Observed:

•Voice Quality: Varies by browser (e.g., Chrome offers better voices than Firefox).

•Language Support: Dependent on browser voices, which may not cover all languages or accents.

•Advanced Features: Lacks neural TTS quality (e.g., natural prosody) unless extended with cloud APIs.

•Offline Support: Limited in some browsers if voices require internet access.

5. Testing Results:

•Browsers Tested: Chrome, Edge, and Safari provide the best voice options; Firefox has fewer voices.

•Devices: Works on laptops, smartphones, and tablets with consistent functionality.

6. Potential Improvements:

•Cloud Integration: Adding Google Cloud TTS or Amazon Polly for higher-quality voices

•UI Enhancements: Adding visual feedback (e.g., progress bar during speech).

•Multilingual Support: Dynamically adjust ‘utterance.lang’ based on user input or detected text language.

CHAPTER 8
CONCLUSION

The text-to-speech (TTS) converter project, built using HTML, CSS, and JavaScript with the Web Speech API, successfully delivers a lightweight, browser-based solution for converting text into spoken audio. By leveraging the Web Speech API, the project achieves core functionality—allowing users to input text, select from available voices, and control speech playback—without requiring external dependencies or complex setups.

Key Achievements

Functionality: The application provides seamless text-to-speech conversion with voice selection and basic controls (start/stop), meeting the primary objective of enabling accessible audio output.

User Experience: The clean, responsive UI ensures usability across devices (desktop, mobile, tablet), with intuitive controls and accessibility considerations (e.g., keyboard navigation, ARIA attributes).

Performance: The project is resource-efficient, relying on browser-native capabilities, resulting in minimal latency and no significant hardware demands.

Cross-Platform Compatibility: Thorough testing across browsers (Chrome, Edge, Firefox, Safari) confirms reliable performance, though voice quality and availability vary.

CHAPTER 9

FUTURE ENHANCEMENT

Possible Future Enhancements:

1. Voice Preview Feature: Let users preview a voice before selecting it by speaking a sample sentence when they select a voice.

2. Pitch and Rate Control: Add sliders to adjust the pitch and rate (speed) of the voice.

(‘speech.pitch’ and ‘speech. rate’ properties exist for this.)

3. Save Settings: Save the user’s preferred voice, pitch, and rate using localStorage, so when they come back, it remembers.

4. Pause, Resume, and Stop: Add buttons for pause, resume, and cancel during speech

(‘speechSynthesis.pause()’‘speechSynthesis.resume()’,

‘speechSynthesis.cancel()’).

5. Error Handling: Add checks for empty text areas and handle cases where no voices are loaded or the speech fails.

6. Multi-language Support: Allow users to select different languages, not just voices, depending on their needs.

7. Responsive Design: Improve the user interface for mobile and tablet devices, making it feel like a real app.

8. Text Uploading: Allow users to upload ‘.txt’ files and have the app read them aloud.

9. Highlighting Words as They Are Spoken: Visually highlight the words on the screen while they are being spoken for better accessibility.

10. Download as Audio File:

Advanced: Let users generate and download the speech as an audio file (this requires a server-side solution because browsers alone can't record ‘SpeechSynthesis’ output yet).

CHAPTER 10

REFERENCES

1. Web Speech API Documentation (MDN Web Docs)

- This is the official reference for the `SpeechSynthesis` and `SpeechSynthesisUtterance` objects you are using.
- Link: [https://developer.mozilla.org/en US/docs/ Web/API/ Web_Speech_API] (https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API]

2. W3C Web Speech API Specification

- The technical specification for browser support and how speech synthesis is structured.
- Link: https://wicg.github.io/speech-api/

3. Google Text-to-Speech (for Android)

- If you want to reference mobile TTS, Google's TTS system is widely used.
- Link:https://developers.google.com/assistant/sdk/reference/rpc/google.cloud.texttospeech.v1

4. ResponsiveVoice (Commercial Web TTS Service)

- Shows how cloud services integrate TTS for web apps.
- Link: https://responsivevoice.org/

5. Github Projects

- Example repositories of TTS implementations using JavaScript and other languages.
- Example: https://github.com/mdn/web-speech-api