A Voice-Controlled Web Navigation System

by

Examination Roll: 212126

A Project Report submitted to the
Institute of Information Technology
in partial fulfillment of the requirements for the degree of
Professional Masters in Information Technology

Supervisor: Dr. Shamim Al Mamun, Professor

Institute of Information Technology
Jahangirnagar University
Savar, Dhaka-1342

October 2025

DECLARATION

I hereby declare that this thesis is based on the results found by myself. Materials of work found by other researcher are mentioned by reference. This thesis, neither in whole nor in part, has been previously submitted for any degree.

Roll:212126

CERTIFICATE

The project titled "A Voice-Controlled Web Navigation System" submitted by Sudipta Sarker, ID: 212126, Session: Summer 2021, has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Professional Masters in Information Technology on the 11th of October 2025.

ACKNOWLEDGEMENTS

At first, I bow my head in gratitude to the Almighty God for granting me strength and blessings to complete this project. Without his guidance it was not possible to complete this journey successfully.

I would like to express my heartfelt appreciation and gratitude to my respected supervisor, Professor Dr. Shamim Al Mamun, whose support, valuable guidance and constructive feedback played an important role to achieve this goal. His motivation and guidance kept me focused and straight forward to implement this project from the scratch to the end.

My deepest respect and thanks to the Director and the honorable faculty members of the Faculty of PMIT, Jahangirnagar University for providing me with necessary academic facilities and inspiration during my study period. Their dedicated teaching and cooperative attitude helped me to enhance my knowledge and gain innovative ideas such as this project.

Finally I would like to express my gratitude to my friends, classmates and others who directly or indirectly helped me in completing this project. I am also thankful to the administrative and staffs of PMIT for their cooperation and accurate information whenever required.

ABSTRACT

This project focuses on the development of a Chrome extension controlled by voice to improve user interactivity with web browser. Voice based browsing extensions are explored in the modern era, but most of them are limited to certain languages or do not provide flexible tab control, inline search or smooth speech interaction. The proposed solution introduces a lightweight browser extension which supports both Bangla and English commands and provides flexible features like switching between tabs, inline search for Google and Wikipedia, fill up forms, read out selected text through speech. To achieve this, we implemented speech recognition APIs, intent-based command matching and parsing for accurate result with modular Javascript arhitechture for scalability. Chrome extension APIs were integrated here for tab management, content handling and background propagation. The system was test through real browser scenerio and successfully performed site navigation, click through expected links, inline search with dual languages without noticeable lag. A special feature is added for auto silence detection and highlight-based read-0out functionality by integrating speech recognition APIs. The findings shows that the extension can help to reduce manual browsing effort by providing solutions for users with limited typing skills and handicuffs. This project shows that voice driven solution can improve web accessibility and multilingual abilities.

Keywords: Chrome extension, Speech recognition, APIs, Voice driven solution, Inline search.

LIST OF ABBREVIATIONS

IIT	Institute of Information Technology
JU	Jahangirnagar University
PMIT	Professional Masters in Information Technology
API	Application Programming Interface
TTS	Text to Speech

LIST OF NOTATIONS

α	Define alpha
$\max$	maximum
$\cos\theta$	maximum
x	maximum

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

CHAPTER I

Introduction

1.1 Overview

The rapid growth of the Internet makes the web browser an essential tool for various use cases like accessing information, analysis, performing tasks globally. Traditionally, users use web browser by using keyboards or using mice. This can be time consuming, boring, inconvenience and challenging for people who has limited typing skills or disabilities. Voice based technologies supported by speech recognition and natural language processing bring a ground-breaking change in this area by providing more accessible method of interacting with digital systems.

This project focuses on the design and development of the voice controlled Chrome extension which supports Bangla and English. This extension allow users to search online from google and wikipedia, read selected text, can click the highlighted links and many more – all through voice commands. By enabling multiple language, the system enhances the usability to the people in Bangladesh where English is not the the primary language.

1.2 Limitation of Existing Works

Existing voice assistant solutions like Siri or Google Assistant are mostly designed for mobile devices or standalone device with limited integration with desktop browsers. Besides these systems are mostly English centric and lack of usability in Bangla language in some cases.

On the other hand, the existing browser extensions which support voice commands, very few of them integrate features like tab control, form fill up, click highlighted links, read selected text, click important links or inline search. Some of them consume more resources which make them heavy-weighted browser uses. Very view extensions provide multilingual solutions that support both Banlga and English. So, these limitations motivated me to develop a dedicated browser extension which supports these features with multilingual inclusive.

1.3 Motivation

The motivation for this project grows inside me for the demand of more accessible and efficient browsing methods for users who performs more multi tasks, need to write long queries, face difficulties to find a particular link in a who page. Similarly, some people has physical difficulties or find interest to listen text content rather than reading whole content. This browser extension with multilingual facilities can bridge this gaps making browsing faster and interesting by providing a lots of features.

1.4 Problem Statements

Although the availability of the advanced voice assistant like Siri or Google Assistant, there is no lightweight Chrome extension that provides dual-language (English and Bangla) voice control with additional browsing facilities. Users currently lack the ability to:

- Perform inline search through Google, Wikipedia or YouTube by voice.
- Switch between specific browser tabs using voice recognition.
- Control Media element interaction by voice.
- Click highlighted links to navigate and find the expected links easily.
- Fill up forms, listen the selected text.

This project addresses these issues by proposing and developing a proper voice based Chrome browser extension by including these features with multilingual support in Bangla and English.

1.5 Objectives

The main objectives of this research are:

- To design and develop a voice-based Chrome extension that allows users to interact with the web browser through natural speech commands.

- To support multilingual interaction (English and Bengali) so that both local and international users can easily interact with the browser.
- To perform different activities through browser like searching content on Google, Wikipedia, form filling, media control(play/pause/mute), click highlighted links etc. using voice commands.

1.6 Research Outline

This project is structured into five major chapters:

- Chapter 1 – Introduction: Provides the background, problem statement, motivation, objectives of the study and research outline.
- Chapter 2 – Literature Review: Discusses about the existing solutions or related works, their features and limitations.
- Chapter 3 – Proposed System Architecture: Explains the architecture, tools, speech recognition techniques and algorithmic approach to achieve the intended functionalities.
- Chapter 4 – Implementation and Results: Describes coding modules, functionalities tested and the experimental outcomes.
- Chapter 5 – Conclusion and Future Works: Summarizes the findings of the project, implications and some important improvements for future.

1.7 Conclusion

In this chapter, background and motivation of the project are presented along with limitations of the existing solutions. Besides the problem statements and the objectives clearly dictate the necessity of the multilingual voice based Chrome extension. This chapter also provides the research outline which will guide the overall project design and implementation throughout the project. The next chapter will provide the review for the existing solutions and related works to identify the existing gaps which establish the foundation of the proposed system.

CHAPTER II

Literature Review

2.1 Overview

Before initiating the development of the proposed voice controlled chrome extension, a detail study was conducted based on related existing tools, research works, extensions. This study gives a proper insight which helped to build this project. The study demonstrates the features of the existing works and review them about their limitations. It highlighted how this tools approached to implement voice recognition, natural language understanding and how to interact with the browser which forms the foundation for developing a enhanced proper voice controlled chrome extension with extensive features.

The study also shows the importance of the response time, performance, user experience, supported languages of those works, By studying these works and papers, the opportunities of new features/solution have been identified.

2.2 Related Works

2.2.1 Browser-based Voice Extensions

Handsfree for Web [2] provides a lots of set features including web search(Google, Wikipedia), click links, scrolling, form filling, media control, text reading and some basic browser control features. The strength of this system lie in extensive features and multilingual support(4 languages). But it has no integration with Gmail, Google Docs, or map related features.

Readme – TTS [3] mainly focuses on the accessibilty of the browser supported by over 50 languages. It helps user to listen to text from webpage, emails and PDF. But this extension does not support browser control or media control. The limitation of this extension is less features, mainly focuses on text-speech features rather than browser interaction.

Hey Buddy – Chrome Voice Assistant[4] provides features for basic browsing interaction and media control including search, scrolling and link navigation. It is lightweight and easy to use but it does not support Bangla and does not provide some advanced features like email management and Google Doc interaction or custom commands.

LipSurf - Voice Control for the Web[5] allows user to interact with browser using voice commands. It supports tab control, form filling and media control. It provides a good set of features but does not integrate Bangla language to communicate.

Voice Typing[6] enables users to write text based on voice input. It receives voice input from users and translates it into text by using text-to-speech API. It does not come with other features like browser control or media control. Its scope is limited to type text only.

Speech Recognition Anywhere[7] offers flexible features with multiple language support. It allows features to copy, paste and basic form filling. It supports browser control features but does not include advanced productivity tools.

Other notable extensions include:

- Text-to-Speech that Brings Productivity[8] – TTS focused with limited control features.
- Voice In Voice Typing[9] – Dictation tool.
- Dictation for Gmail[10] – Specialized for email accessibility.
- Talkie: Text-to-Speech[11] – Multilingual reading tool for selected text.
- NaviVoice: Voice Input Productivity Assistant[12] – Offers limited browsing and form filling features.
- Voice Actions for Chrome (beta)[13] – Simple browser commands.
- Say Play[14] – Media playback control.
- Neo[15] – Lightweight browsing and limited productivity features.
- Speech to Text (Voice Recognition)[16] – Dictation focused.
- Capti Voice[17] – Accessibility tool for reading and highlighting text.

- Voice Control for Video[18] – Media control only.
- VCAT[19] – Basic voice commands and limited browsing control.
- Natural Reader Text to Speech[20] – Multilingual reader for web pages, PDFs, emails.

  From the above discussion it is proved that no single extension currently offers browsing control, media control and multilingual accessibility(specially for Bangla) as an unit.

2.2.2 Research-based Voice Systems And Academic Works

There are several academic and large-scale solutions provide detail insight fro voice-based systems rather than browser extensions.

Google Voice Access[21] offers accessibility, allow uses to navigate and control the system. While this system is highly effective for mobile devices but it does direct support desktop chrome browsing. Amazon Alexa Browser Integrations[22] allow users to do basic search and other tasks but it has dependency on the cloud services and limited to the devices.

A Voice Controlled E-commerce Web App[23] focuses on the accessibility of online shopping tasks can be improved through voice commands. It presents that by using voice commands users can perform their shopping from e-commerce site smoothly and efficiently. But this paper does not present any interaction for browser accessibility through voice commands.

Studies on **Voice-Based HCI[24]** emphasized different parameters which are critical for browser-based extension such as response time, noise handling, and context recognition.

Accessibility research[25] highlights the necessity of multilingual support, highlighting words during text to speech, and provides integration with different productivity tools.

Këpuska & Bohouta (2017)[26] studied and compared modern speech recognition APIs (Google, Microsoft, IBM Watson, CMU Sphinx) and referred that cloud-based systems achieve high accuracy while local engines allow offline processing.

2.3 Analysis of Findings

The analysis of reviewed extensions and academic systems shows that most browser based web extensions offers several features like basic navigation, read selected text, media control, some provides multilingual support but lack of integration with productivity applications. But as an unified system no extension can offer all the features with high response time and support voice commands for Bangla.

Research studies how accuracy, latency, noise handling, multilingual support with proper design can enhance the productivity of a voice-based solutions which builds the foundation for the proposed system.

2.4 Conclusion

In summary, the literature review shows that while there exists several extensions which provide partial voice-based browsing control, media control or TTS functionality, no solution offer an unified integrated solution by combining all important features and support for Bangla language. Existing academic research highlights the importance of accuracy, speed, and usability and latency, which are essential to incorporate into the design of the proposed system. The goal of the proposed extension is to bridge these gaps, providing a user-friendly, feature oriented, multilingual, and productivity based voice assistant tool for Chrome web browser.

CHAPTER III

Proposed System Architecture

3.1 Introduction

This chapter represents the methodology used in the development of the Voice Search Chrome Extension. The chapter focuses on the design, workflow and the core functionalities of the proposed system. This system enables users to control the browser for several facilities like search in the internet, form fill up, read the selected content and click the highlighted links through voice commands.

The methodology demonstrates modularity, scalability and user-friendliness. This system is divided into different logical components to ensure the accurate speech recognition and identify the correct command by parsing and executing the right operation. This chapter describes the architecture, workflow and features through diagrams, flowcharts and algorithm to make it easy to understand the system properly.

3.2 System Architecture

Overview of Components

- **User Interface Module:** Provides a floating microphone button for capturing voice input.
- **Speech Recognition Module:** Converts spoken words into text using the browser's Web Speech API.
- **Command Processing Module:** Detects the language and identifies the user's intent by matching text with predefined commands stored in JSON files (en.json, bn.json).

- **Action Execution Module:** Performs the requested browser actions, including tab control, scrolling, inline search, media handling, link clicking, and form filling.
- **Feedback Module:** Provides visual or audio feedback to the user, such as highlighting text during read-aloud or confirming executed actions.

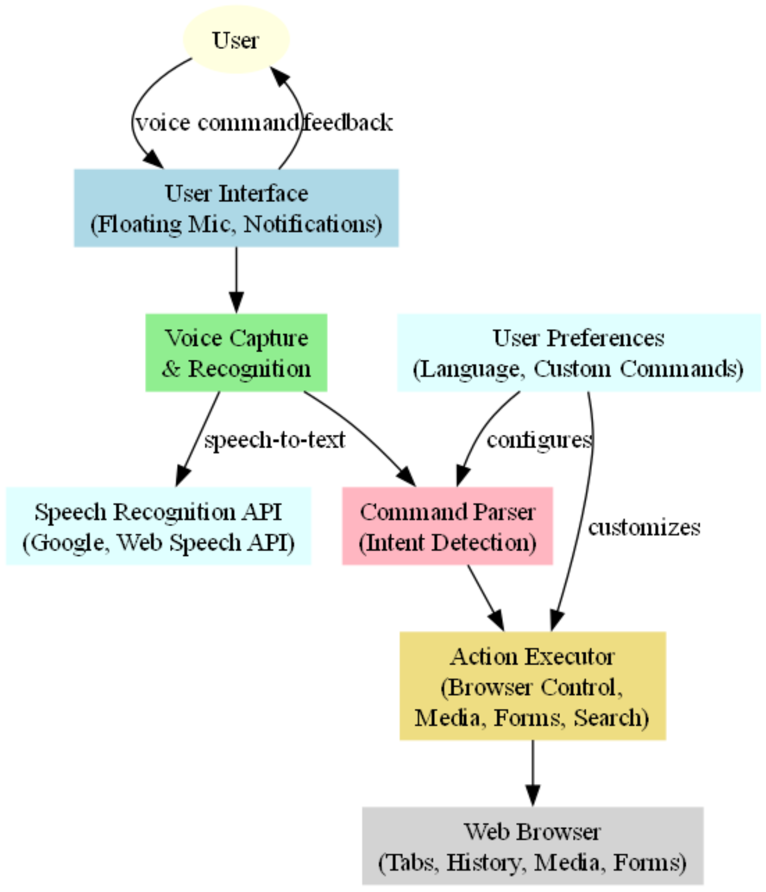

User
voice command
feedback
User Interface (Floating Mic, Notifications)
Voice Capture & Recognition
User Preferences (Language, Custom Commands)
speech-to-text
configures
customizes
Speech Recognition API (Google, Web Speech API)
Command Parser (Intent Detection)
Action Executor (Browser Control, Media, Forms, Search)
Web Browser (Tabs, History, Media, Forms)

Figure 3.1: Architecture of the Voice Search Extension

Figure 3.1 illustrates the overall architecture of the Voice Search Extension, highlighting the interaction between the user, the system modules, and the browser environment.

3.3 System Workflow

3.3.1 Functional Flow

Functional Flow describes the steps how total process keeps forward from start to end. This flow starts when user activates the microphone by click. Then speech is captured, speech-to-text translation, intent parsing and then mapped to the correct actions. The flow ends with the execution of the action and followed by feedback through voice transcript or audio feedback. Figure 3.2: Functional Flow of Voice Command Processing illustrates this sequence.

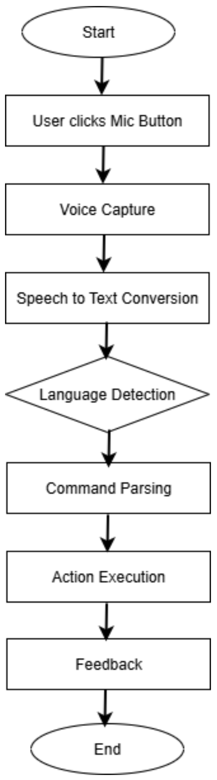


Figure 3.2: Functional Flow Diagram of the proposed system

3.3.2 User Interactions and Use Cases

This system supports different interaction between users. Some are: Hands free web search, inline online search on google, youtube or wikipedia, media playback control, read-loud functionality on any webpage on both English or Bangla, click highlighted links, voice controlled form filling, browser control like switching tabs, auto scrolling, open/close tabs.

These use cases are represented in Figure 3.3. Figure 3.3 illustrates the different ways the user interacts with the extension, including search, media control, form filling, navigation, and text-to-speech reading.

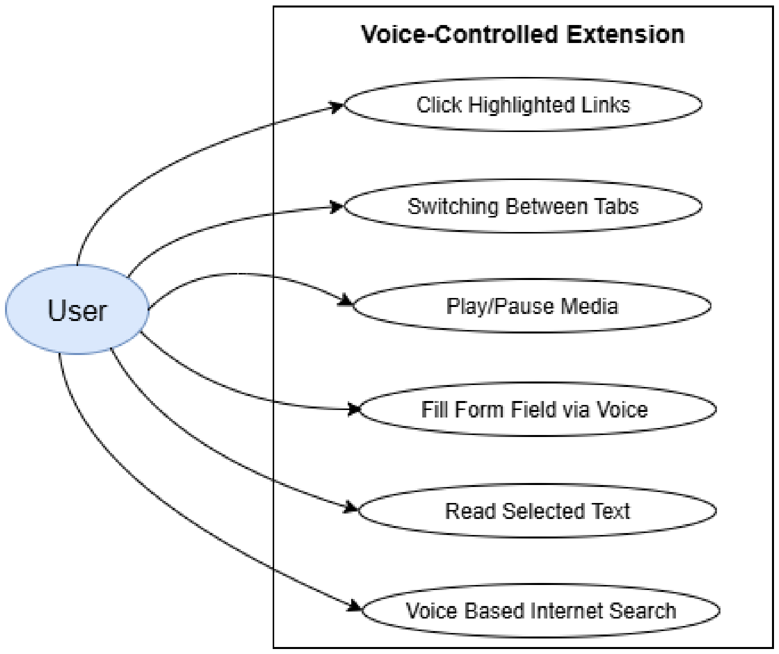


Figure 3.3: User-cases diagram of voice controlled system

3.3.3 Data Organization and Relationships

Voice input, commands, actions, preference, language are organized follows a structured model in a way that it links natural voice input with the executable commands in action. For example "search python tutorials for beginners" maps to the Search Handler with the site to visit and query.

Figure 3.4: Data Structure and Relationships illustrates this organization.It depicts how commands, actions, and user preferences are organized in the system. Spoken input is first recognized and mapped into intents with associated parameters. User Preferences such as language, zoom level, reading speed modify both interpretation and execution. Finally mapped commands are executed as system action to interact with browser.

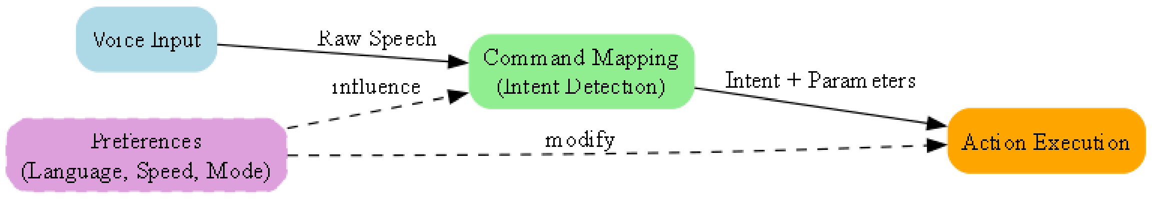


Figure 3.4: Data Relationship Diagram Among Components

3.4 Demonstration and Illustration

The extension's usability is demonstrated through:

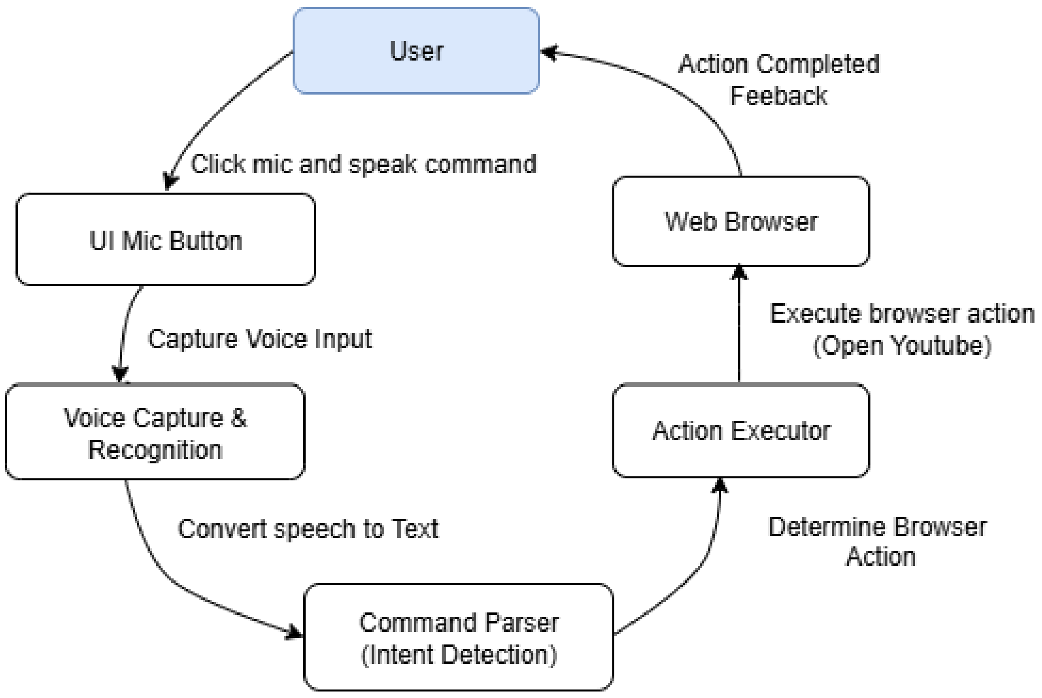


Figure 3.5: Sequence diagram with detailed workflow

This sequence diagram presents the detailed step-by-step message flow between elements for single command execution.

- The process initiates while user click on mic and speak any command.
- UI mic button captures the voice input and pass through it to the Voice Capture & Recognition component.
- This component converts speech into text and transfer it to the Command Parser.
- Command Parser detect the intent by parsing the text and transfer appropriate browser action to the action executor section.
- Action Executor Component executes the browser action in web browser such as inline search in YouTube shown in above figure.
- Web browser then send the feedback to the user by displaying the output.

3.5 Algorithmic Approach For Implementing Features

In this section, different feature modules are described along with algorithms which show how each feature can be implemented. The implementation process with code is discussed in Chapter 4 for each feature.

3.5.1 Capturing and Interpreting Voice Commands

This subsection describes how to capture and interpret voice commands spoken by users. `Algorithm 1` describes the way of processing of a voice command from capture to execution. Speech Recognition API is used here to detect text from speech. If intent matches with commands, it executes actions otherwise provides error message to user.

Algorithm 1 Voice Command Processing from capturing to execution

```
Input: voice_input Output: executed_action
1: Capture VOICE_INPUT from microphone
2: Convert VOICE_INPUT to text using Speech Recognition API
3: Detect language of text
4: if intent matches known command then
5:     Execute corresponding action
6: else
7:     Provide error/feedback to user
8: end if
9: return executed_action
```

3.5.2 Media Playback and Sound Management

This feature allows the user to control media playback (play, pause, volume, mute, forward, backward) using voice commands. `Algorithm 2` describes how intent is detected and how associated video content is manipulated in the DOM.

Algorithm 2 Media Playback and Sound Management

```
Input: voice_command Output: media_state_changed
 1: Capture voice command
 2: Convert to text and extract intent
 3: if intent = "play" then
 4:     Execute VIDEO.PLAY
 5: else if intent = "pause" then
 6:     Execute VIDEO.PAUSE
 7: else if intent = "volume up" then
 8:     Increase volume by 10%
 9: else if intent = "volume down" then
10:     Decrease volume by 10%
11: else if intent = "forward/backward" then
12:     Adjust VIDEO.CURRENTTIME by given seconds
13: end if
14: Provide feedback to user
```

3.5.3 Tab Navigation and Switching

This feature allows the user to manage tab navigation(open, close, or switch browser tabs) by voice commands.`Algorithm 3` refers how tab-related intent are matched and then sends a message to Chrome's API to perform the related action.

Algorithm 3 Tab Navigation and Switching

```
Input: voice_command Output: tab_action_executed
 1: Capture voice command
 2: Convert to text and detect tab intent
 3: if intent = "open site" then
 4:     Extract site name and call CHROME.TABS.CREATE(url)
 5: else if intent = "close tab" then
 6:     Call CHROME.TABS.REMOVE(currentTab)
 7: else if intent = "switch tab" then
 8:     Identify target tab and call CHROME.TABS.UPDATE(active=true)
 9: end if
10: Confirm action with visual/audio feedback
```

3.5.4 Page View Adjustment via Voice-Controlled Scrolling and Zooming

This feature lets the user scroll or zoom in webpage with voice commands. `Algorithm 4` describes the step by step process for mapping commands to page events like window scroll or zoom adjustment.

Algorithm 4 Scrolling and Zooming Process

```
Input: voice_command Output: adjusted_page_view
 1: Capture voice command
 2: Convert to text and detect scroll/zoom intent
 3: if intent = "scroll up" then
 4:     Execute WINDOW.SCROLLBY by X pixels to top
 5: else if intent = "scroll down" then
 6:     Execute WINDOW.SCROLLBY by X pixels to bottom
 7: else if intent = "zoom in" then
 8:     Increase DOCUMENT.BODY.STYLE.ZOOM by 10%
 9: else if intent = "zoom out" then
10:     Decrease DOCUMENT.BODY.STYLE.ZOOM by 10%
11: end if
12: Provide feedback bubble to confirm change
```

3.5.5 Link Highlighting and Selection

This feature allows the user to highlight, hide, or click web page links using voice commands. All visible links are numbered, and the user can refer to them by spoken numbers or partial text are described in `Algorithm 5`.

Algorithm 5 Link Highlighting and Selection

```
Input: voice_command Output: link_highlighted/clicked
 1: Capture voice command
 2: Convert to text and detect link intent
 3: if intent = "show links" then
 4:     Highlight all links with numeric labels
 5: else if intent = "hide links" then
 6:     Remove all link overlays
 7: else if intent = "click link" then
 8:     if value is number then
 9:         Click link by index
10:     else
11:         Find link by matching title text and click
12:     end if
13: end if
14: Provide feedback about clicked link
```

3.5.6 Text-to-Speech Conversion with highlighted sentence

This feature enables the user to listen to web page text read aloud.`Algorithm 6` describes how the system detects the selected text, highlights it sentence by sentence, and uses the speech synthesis API to read it loudly.

Algorithm 6 Reading Selected Text Process

```
Input: selected_text Output: audio_narration
1: Detect if text is selected on webpage
2: if selection is empty then
3:     Provide error feedback
4: else
5:     Pass selected text to SpeechSynthesis
6:     Highlight sentence sequentially during reading
7:     Auto-scroll page along with reading
8: end if
```

3.5.7 Voice Controlled Inline Search Execution

`Algorithm 7` demonstrates how the system performs inline search without opening a new tab(Google, YouTube). At first it clears the input, insert the query, then submit the query to display the output.

Algorithm 7 Inline Search

```
Input: query, active_tab Output: search_results
1: Check if active_tab supports inline search
2: Clear current search input
3: Insert query into search box
4: Press Enter to display results
5: Highlight first search result (optional)
```

3.5.8 System Response Time

In this subsection, an equation is presented to calculate the total time taken from the capture of the voice input to perform execution.

$$T_{response} = T_{capture} + T_{recognition} + T_{parsing} + T_{execution} \tag{3.1}$$

$$\begin{aligned}
T_{\text{Response}} &= \text{Time taken from input to execute the action.} \\
T_{\text{Capture}} &= \text{Time to capture voice input} \\
T_{\text{Recognition}} &= \text{Time for speech-to-text conversion} \\
T_{\text{Parsing}} &= \text{Time to parse command and determine intent} \\
T_{\text{Execution}} &= \text{Time to execute the browser action}
\end{aligned}$$

3.6 Scalability and Extensibility

The modular design of this system enables to add new commands and services(like Google Doc voice typing).The design of the system makes this scalable and extensive to new features. It can support to a new language commands by adding them in new JSON file. In future it is possible to add NLP features for more natural speech interpretation and reduce response time.

3.7 Conclusion

This chapter presents the architecture, workflow of the voice based chrome extension. This system is modular, scalable and user-friendly by enabling voice command inclusion. Algorithms and diagrams clarify the step by step system design and its workflow from user input to actions. This proposed methodology provides a solid foundation to design, develop and testing the system and also its future enhancement by adding new features and including new languages.

CHAPTER IV

Implementation and Result Analysis

4.1 Introduction

This chapter demonstrates the implementation and result analysis of the Voice Search Chrome Extension. This chapter describes the technologies are used here to develop, development details with code snippet, testing strategies and performance evaluation of the system. Screenshot and tables are included to show the functionalities and performance of the project.

The primary goal of this chapter is to show how the methodology proposed in the Chapter 3 has been implemented into a working software product. The result highlights the efficiency, accuracy and usability of the extension.

4.2 Technologies Used and Their Application

Table 4.1 demonstrates each technology used here to ensure performance, modularity and accessibility.

Table 4.1: Technologies Used and Their Application

Technology	**Purpose / How It Was Used**
JavaScript (ES5)	Used to implement core logic for capturing voice, parsing commands, executing browser actions
Chrome Extension APIs	Used for tab management, content scripts, and browser interaction
Web Speech API	Converts voice input to text and detects language (English/Bengali).
JSON	Stores commands and mappings to the corresponding actions
HTML / CSS	used for designing Floating mic button and UI for feedback

4.3 Implementation Details and Results

4.3.1 Voice Input Process

Voice Input Process deals with the receiving voice commands from the users and converts them into text and pass for next stage. This process is implemented in core.js. Browser SpeechRecognition API is used to capture user voice input. The speech recognition object initialized with continuous listening and interim results for speech-to-text conversion.

```
recognition = new SpeechRecognition();
recognition.lang = window.selectedLanguageCode || "en-US";
recognition.continuous = true;
recognition.interimResults = true;
```

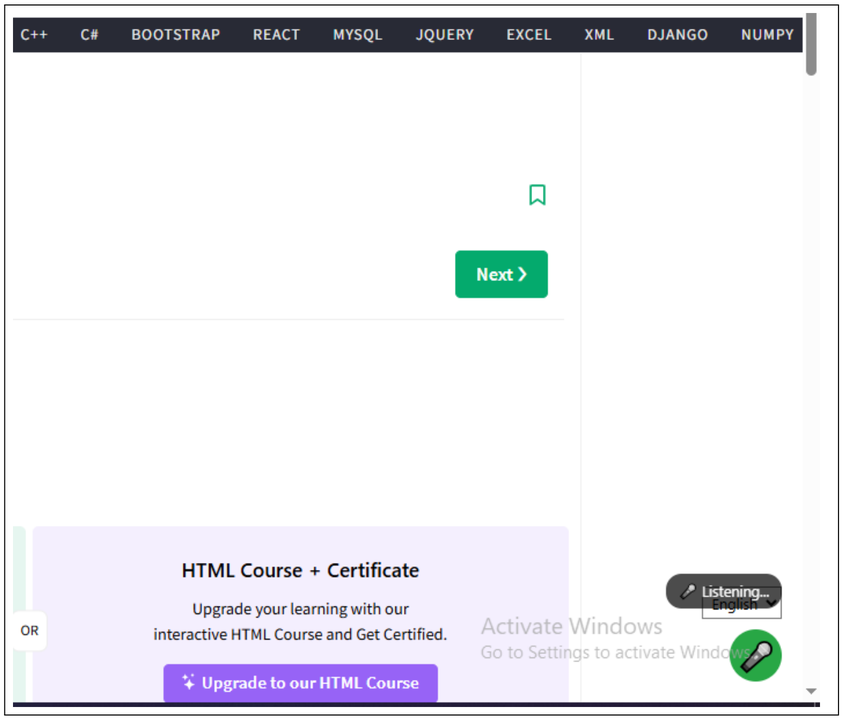
The system reacts to different events. *onstart* - used for activating mic icon provide indication that mic is under listening mode. *onerror/onend* is used to stop recognition if errors or silence occur. *onresult* event sends recognized text to *handleRecognition-Result* function.

```
recognition.onstart = () => {
    document.getElementById("floating-mic").style.background = "#28a745";
    showBubble(" Listening...");
  };

  recognition.onerror = (e) => {
    stopRecognition();
    showBubble(" Error: " + e.error);
  };

  recognition.onend = () => {
    document.getElementById("floating-mic").style.background = "#007bff";
  };

  recognition.onresult = handleRecognitionResult;
```

The function *handleRecognitionResult* checks transcript, match predefined command and execute corresponding browser action(e.g scroll, open site, media control).

```
function handleRecognitionResult(event) {
  ......
  for (let i = event.resultIndex; i < event.results.length; i++) {
    const result = event.results[i];
    const transcript = result[0].transcript.trim().toLowerCase();
    const isFinal = result.isFinal;
    if (!transcript) continue;
    showBubble(isFinal ? " " + transcript : "..." + transcript);
    const match = matchCommand(transcript);
    ......
```

Screenshot



C++
C#
BOOTSTRAP
REACT
MYSQL
JQUERY
EXCEL
XML
DJANGO
NUMPY
Next ›
HTML Course + Certificate
Upgrade your learning with our interactive HTML Course and Get Certified.
OR
Upgrade to our HTML Course
Listening...
English
Activate Windows
Go to Settings to activate Windows

Figure 4.1: Active mic with "Listening..." bubble.

4.3.2 Media Control

The Media Control module used for voice-based interaction with video element of the page. Once the user says *"start media"*, the system enable Media Mode to allow media commands such as play, pause, volume up/down etc.

After matching the intent, detected commands are executed on the active `<video>` element and feedback is displayed on-screen speech transcript.

```
// Enabling and disabling Media Mode
if (intent === "start_media") {
   mediaMode = true;
   showBubble("Media mode enabled");
}
if (intent === "stop_media") {
   mediaMode = false;
   showBubble("Media mode disabled");
}
// Passing commands to media.js
if (mediaMode && intent.startsWith("media_")) {
   handleMediaCommand(intent, value);
}
```

In `media.js`,`handleMediaCommand()` function handle different video actions. For example, when user speak "play" video will start and when user say "pause" video will pause(shown in 2 to 10 lines). The following examples illustrate how commands are executed:

```
 1  // Playing a video
 2  case "media_play":
 3     video.play();
 4     showBubble("Playing");
 5     break;
 6  // Pausing a video
 7  case "media_pause":
 8     video.pause();
 9     showBubble("Paused");
10     break;
11  // Adjusting volume
12  case "media_volume_up":
13     video.volume = Math.min(1, video.volume + 0.1);
14     showBubble("Volume up");
15     break;
16  // Other commands in similar fashion...
```

These code snippets shows the way to map different voice commands to execute actions, enable consuming media content almost hands completely hands-free for users.

Screenshot

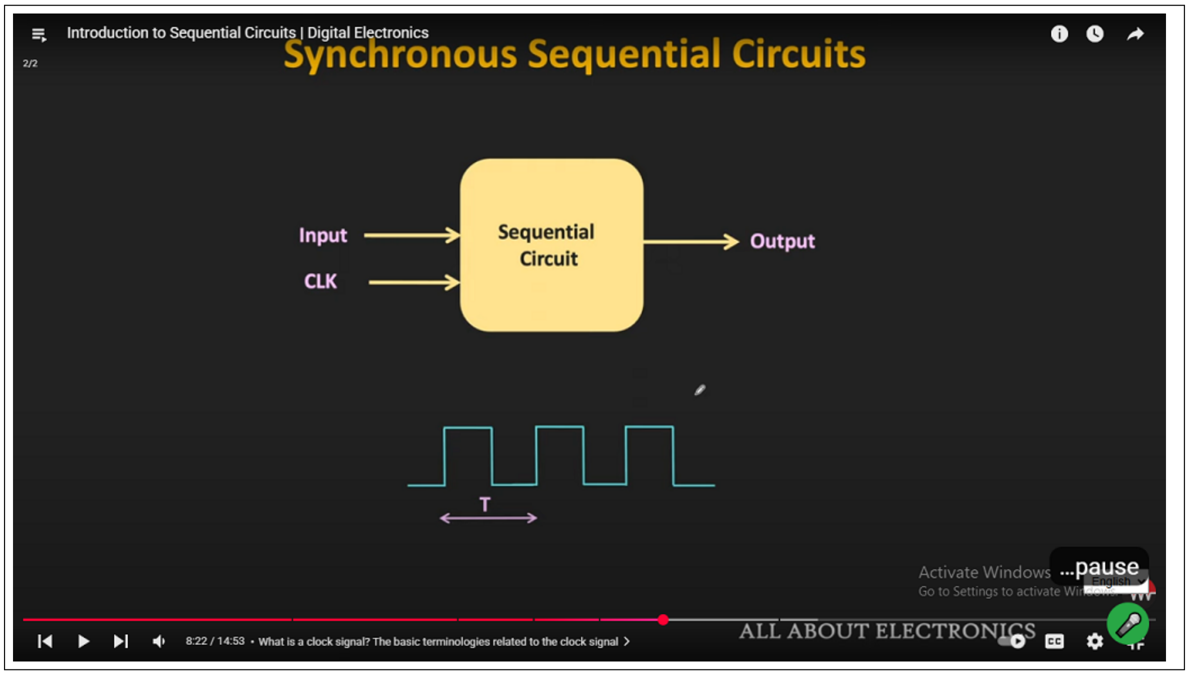

Introduction to Sequential Circuits | Digital Electronics
2/2
Synchronous Sequential Circuits
Input
CLK
Sequential Circuit
Output
T
Activate Windows
Go to Settings to activate Windows.
...pause
English
8:22 / 14:53 • What is a clock signal? The basic terminologies related to the clock signal
ALL ABOUT ELECTRONICS

Figure 4.2: Pause the media content through voice

4.3.3 Tab Management

This extension provides voice-controlled tab management features. It allows to users to control the browser tabs without manual interaction. It supports opening new sites of name or opening new tab, switching between tabs, view all the tabs in list of current window and closing active window through voice command. Sample examples of voice commands to access these features are given below:

Example commands:

- `"open YouTube"`: Opens `youtube.com` in a new tab.
- `"next tab"`: Switches to the next tab in the browser.
- `"previous tab"`: Switches to the previous tab.

- `"switch to [title]"`: Finds the title by matching the phrase after saying "switch to".
- `"close tab"`: Closes the currently active tab.
- "show tab": Displays a popup window with all open tabs.

The workflow is described as user commands are recognized by the speech recognition module, then matched to intents such as `open_site`, `tab_next`, `tab_previous`, or `tab_close`.After that it sends to message to background.js file. It communicates with the Chrome Tabs API to perform the actual tab operations.

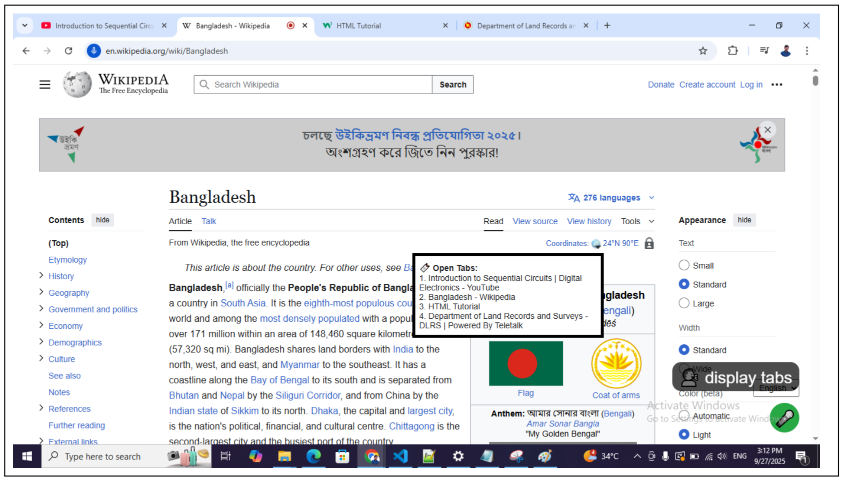
In below code describes how message is sent from core.js after matching the intent to the background.js file which executes the following tab operation.

```
// core.js (user says "switch to youtube")
if (intent === "tab_switch") {
  chrome.runtime.sendMessage({
    action: "switch_tab",
    query: value,   // e.g., "youtube"
  });
}
```

```
// Example: Switching to a tab by title
if (msg.action === "switch_tab") {
  const query = msg.query;
  chrome.tabs.query({ currentWindow: true }, function (tabs) {
    for (let i = 0; i < tabs.length; i++) {
      if (tabs[i].title.toLowerCase().includes(query.toLowerCase()))
       ↪ {
        chrome.tabs.update(tabs[i].id, { active: true });
        break;
      }
    }
  });
}
```

This tab management module provides better accessibility to users through the browser without manual interaction every time.

Screenshot:



Open Tabs:
1. Introduction to Sequential Circuits | Digital Electronics - YouTube
2. Bangladesh - Wikipedia
3. HTML Tutorial
4. Department of Land Records and Surveys - DLRS | Powered By Teletalk

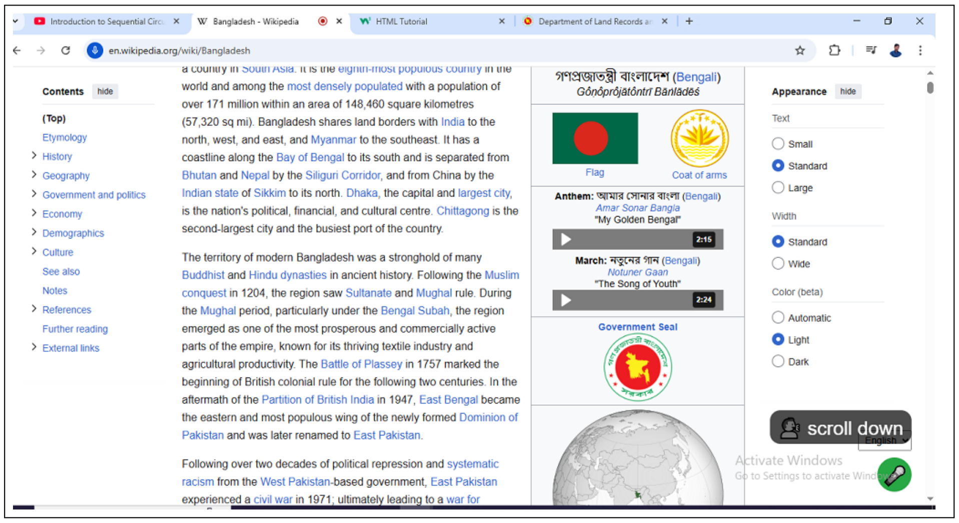
Figure 4.3: Display existing tab list of current window

4.3.4 Scrolling and Zooming

This feature allows users to scroll up/down the current page.Users can also set/reset zoom level through voice commands. For achieving this feature, user can perform following commands such as `scroll_up`, `scroll_down`, `scroll_top`(scroll to the very top position), `scroll_last`(scroll at the bottom of the page), `zoom_in`, `zoom_out`, `reset_zoom`(to reset the zoom level to 100In below code, it shown that how "scroll up" and "zoom in" command execute those actions.

```javascript
// Scroll down by 400px
if (intent === "scroll_down") window.scrollBy(0, 400);

// Zoom in by increasing current zoom step
if (intent === "zoom_in") {
    currentZoom = Math.min(currentZoom + ZOOM_STEP, 2);
    document.body.style.zoom = currentZoom;
    showBubble(`Zoomed in to ${Math.round(currentZoom * 100)}%`);
}
```

Screenshot:



a country in South Asia. It is the eighth-most populous country in the world and among the most densely populated with a population of over 171 million within an area of 148,460 square kilometres (57,320 sq mi). Bangladesh shares land borders with India to the north, west, and east, and Myanmar to the southeast. It has a coastline along the Bay of Bengal to its south and is separated from Bhutan and Nepal by the Siliguri Corridor, and from China by the Indian state of Sikkim to its north. Dhaka, the capital and largest city, is the nation's political, financial, and cultural centre. Chittagong is the second-largest city and the busiest port of the country.

The territory of modern Bangladesh was a stronghold of many Buddhist and Hindu dynasties in ancient history. Following the Muslim conquest in 1204, the region saw Sultanate and Mughal rule. During the Mughal period, particularly under the Bengal Subah, the region emerged as one of the most prosperous and commercially active parts of the empire, known for its thriving textile industry and agricultural productivity. The Battle of Plassey in 1757 marked the beginning of British colonial rule for the following two centuries. In the aftermath of the Partition of British India in 1947, East Bengal became the eastern and most populous wing of the newly formed Dominion of Pakistan and was later renamed to East Pakistan.

Following over two decades of political repression and systematic racism from the West Pakistan-based government, East Pakistan experienced a civil war in 1971; ultimately leading to a war for

গণপ্রজাতন্ত্রী বাংলাদেশ (Bengali)
Gônôprôjātôntrī Bānlādēś

Anthem: আমার সোনার বাংলা (Bengali)
Amar Sonar Bangla
"My Golden Bengal"

March: নতুনের গান (Bengali)
Notuner Gaan
"The Song of Youth"

Text: Small [✓ Standard] Large
Width: [✓ Standard] Wide
Color (beta): Automatic [✓ Light] Dark

Figure 4.4: Scroll down through voice

4.3.5 Link Interaction

This feature enables users to highlight and click links by title on the webpage using voice commands for easier navigation. Available intents and sample commands are given below:

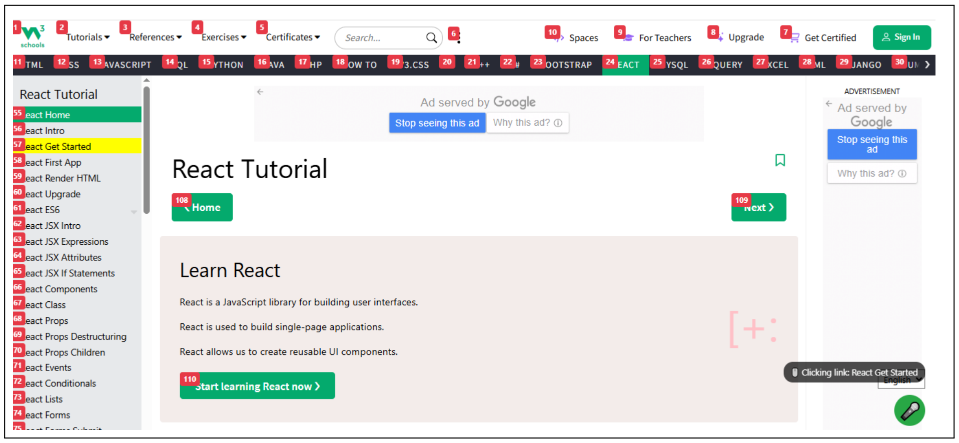
- `show_links`: “show links”, “highlight links”
- `click_link`: “click Facebook”, “link Gmail”
- `hide_links`: “hide links”, “remove links”

here user can click the link by saying numbers which are highlighted after executing `show_links` or by saying inner text of any hyperlink.The code sample of how user can click link by title is given below:

In the below code, from 3-4 lines, it finds out the matching hyperlink from the collected hyperlink list `cachedLinks`. From 6-10, if there is any match then,it highlights the link with yellow color and click the link. Visual feedback will be provided if there is no match with title, are shown in number 11-13 lines.

```
// Find and click a link whose text contains the given title
function clickLinkByTitle(title) {
    const match = cachedLinks.find((l) =>
      l.innerText.toLowerCase().includes(title.toLowerCase())
    );
    if (match) {
        match.scrollIntoView({ behavior: "smooth", block: "center" });
        match.style.background = "yellow"; // temporary highlight
        setTimeout(() => match.click(), 600);}}
```

Screenshot:



React Tutorial

Learn React

React is a JavaScript library for building user interfaces.

React is used to build single-page applications.

React allows us to create reusable UI components.

Clicking link: React Get Started

Figure 4.5: Click link by title from the highlighted links

4.3.6 Reading Selected Text

This feature is designed to improve the accessibility of extension by enabling read aloud webpage content. User can start this feature by voice command such as "start reading". Once activated, the readable sections of the webpage(paragraph, list) are highlighted through numbers. Later, user can read those particular section by selecting the selection by number. The selected content is displayed in a dedicated pop up window and starts reading until finishing the content.

Available Commands

- `Start Reading`: User can highlight readable sections with assigns numbers.

- `Read by Number [X]`: This command opens a pop up window containing the selected section by number and starts reading.
- `Stop Reading`: This command stops ongoing reading and closes the popup window.
- `Clear Highlights`: This command is used for removing number badges from the section and borders from the page.

Implementation

After triggering reading mode, `highlightReadableSections()` scans whole webpage and collect <p>, <ol> and <ul> sections.

```
//
function highlightReadableSections() {
  const sections = document.querySelectorAll("p, ol, ul");
  ......
  });

  showBubble("Numbered readable sections");
}
```

Then if user says "read number 3", then selected content is loaded by `readSectionByNumber()` in pop up window starts reading the text using `startReadingSectionFromPopup` function.

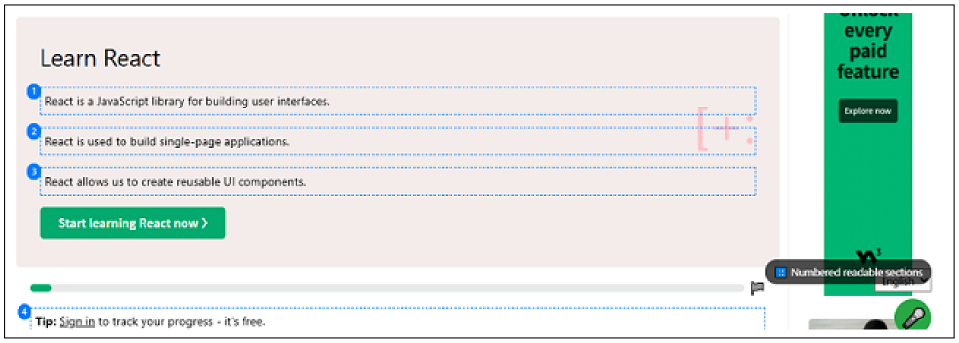
```
function readSectionByNumber(number) {
  const el = document.querySelector(`[data-read-id="${number}"]`);
  if (!el) return;
  const text = el.innerText.trim();
  showReadPopup(number, text);
  startReadingSectionFromPopup(text);
}
```

```
function startReadingSectionFromPopup(text) {
  currentReadSentences = splitIntoSentences(text);
  currentReadSentences.forEach((s,i) => {
    const span = document.createElement("span");
    span.innerText = s + " ";
    span.dataset.index = i;
    content.appendChild(span);
  });  speakNextSentence(); }
```

After loading into popup window entire text is divided into sentences by function `splitIntoSentences()`.Then Browser SpeechSynthesis API reads each word sequentially by `speakNextSentence()` function. While reading each sentence is highlighted with yellow color. After finishing reading, popup window will be closed automatically.

```
function speakNextSentence() {
  if (currentReadIndex >= currentReadSentences.length) return;
  const sentence = currentReadSentences[currentReadIndex];
  const u = new SpeechSynthesisUtterance(sentence);
  u.onstart = () => highlightSentence(currentReadIndex);
  u.onend   = () => { currentReadIndex++; speakNextSentence(); };
  window.speechSynthesis.speak(u);
}
```

Screenshots:


Learn React

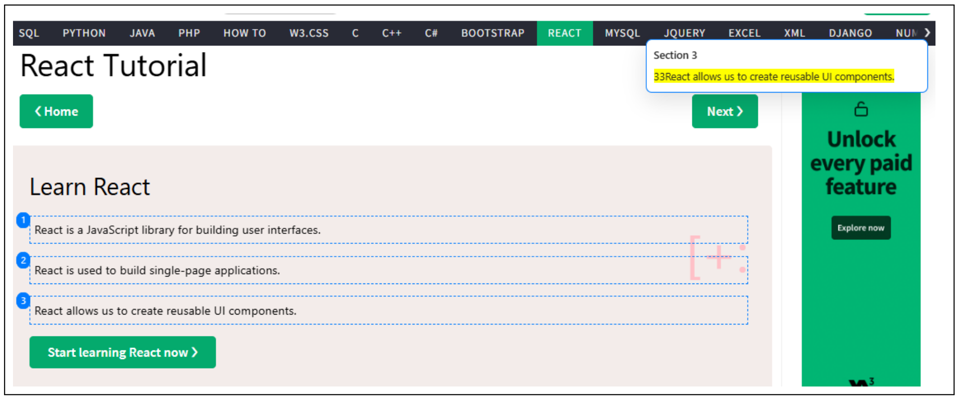
1 React is a JavaScript library for building user interfaces.

2 React is used to build single-page applications.

3 React allows us to create reusable UI components.

4 Tip: Sign in to track your progress - it's free.

Figure 4.6: Highlighted readable section with numeric badges


SQL PYTHON JAVA PHP HOW TO W3.CSS C C++ C# BOOTSTRAP REACT MYSQL JQUERY EXCEL XML DJANGO
Home
Next
Unlock every paid feature
Explore now
Start learning React now

React Tutorial

Section 3

33React allows us to create reusable UI components.

Learn React

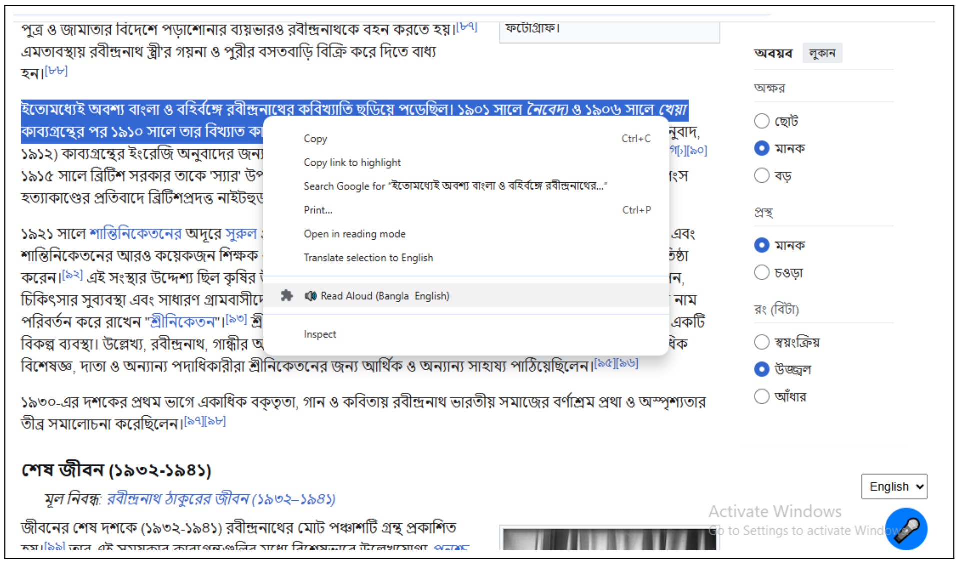
1 React is a JavaScript library for building user interfaces.

2 React is used to build single-page applications.

3 React allows us to create reusable UI components.

Figure 4.7: Read a loud from selected section in popup window

There is another way to implement this feature. Context menu is added here to read both English and Bengali language. User needs to select text and click context menu option from the right menu. By this way, it can read selected text loudly. Here is a screenshot of this activity.



পুত্র ও জামাতার বিদেশে পড়াশোনার ব্যয়ভারও রবীন্দ্রনাথকে বহন করতে হয়।[৮৭] ফটোগ্রাফ।
এমতাবস্থায় রবীন্দ্রনাথ স্ত্রী'র গয়না ও পুরীর বসতবাড়ি বিক্রি করে দিতে বাধ্য হন।[৮৮]

ইতোমধ্যেই অবশ্য বাংলা ও বহির্বঙ্গে রবীন্দ্রনাথের কবিখ্যাতি ছড়িয়ে পড়েছিল। ১৯০১ সালে *নৈবেদ্য* ও ১৯০৬ সালে *খেয়া*
কাব্যগ্রন্থের পর ১৯১০ সালে তার বিখ্যাত কা … নুবাদ,
১৯১২) কাব্যগ্রন্থের ইংরেজি অনুবাদের জন্য … গ[১][৯০]
১৯১৫ সালে ব্রিটিশ সরকার তাকে 'স্যার' উপ … ংস
হত্যাকাণ্ডের প্রতিবাদে ব্রিটিশপ্রদত্ত নাইটহুড

১৯২১ সালে শান্তিনিকেতনের অদূরে সুরুল … এবং
শান্তিনিকেতনের আরও কয়েকজন শিক্ষক … তিষ্ঠা
করেন।[৯২] এই সংস্থার উদ্দেশ্য ছিল কৃষির … ন,
চিকিৎসার সুব্যবস্থা এবং সাধারণ গ্রামবাসীদে … নাম
পরিবর্তন করে রাখেন "শ্রীনিকেতন"।[৯৩] শ্রী … একটি
বিকল্প ব্যবস্থা। উল্লেখ্য, রবীন্দ্রনাথ, গান্ধীর অ … ধিক
বিশেষজ্ঞ, দাতা ও অন্যান্য পদাধিকারীরা শ্রীনিকেতনের জন্য আর্থিক ও অন্যান্য সাহায্য পাঠিয়েছিলেন।[৯৫][৯৬]

১৯৩০-এর দশকের প্রথম ভাগে একাধিক বক্তৃতা, গান ও কবিতায় রবীন্দ্রনাথ ভারতীয় সমাজের বর্ণাশ্রম প্রথা ও অস্পৃশ্যতার তীব্র সমালোচনা করেছিলেন।[৯৭][৯৮]

শেষ জীবন (১৯৩২-১৯৪১)

মূল নিবন্ধ: রবীন্দ্রনাথ ঠাকুরের জীবন (১৯৩২–১৯৪১)

জীবনের শেষ দশকে (১৯৩২-১৯৪১) রবীন্দ্রনাথের মোট পঞ্চাশটি গ্রন্থ প্রকাশিত

Figure 4.8: Read a loud from selected Bengali text

4.3.7 Voice-Controlled Form Fill-Up

The Form Fill-Up feature enables user to fill the form without manual interaction, User can fill the form through voice. Right now this feature only supports English language only.

Available Commands:

- **Mode Control:** `start_form` is used to enable form mode, so that after this each commands executed for form purpose. `stop_form` is used for exit from the form mode.
- **Navigation:** User can use `next`, `reverse`, `back` commands to go back the previous field. User can also navigating by saying index number.
- **Input Handling:** Spoken words are filled into the active field; `clear` command is used to clear the active field. There is a option to fill up the password field by the secure password suggestion with `yes`/`no`. Dropdown control using `down`, `up`, or `option 3`; radio button selection can be done by saying `select male`, `choose female`.
- **Submission:** User can submit the form by saying `"submit form"`, `"form done"`.

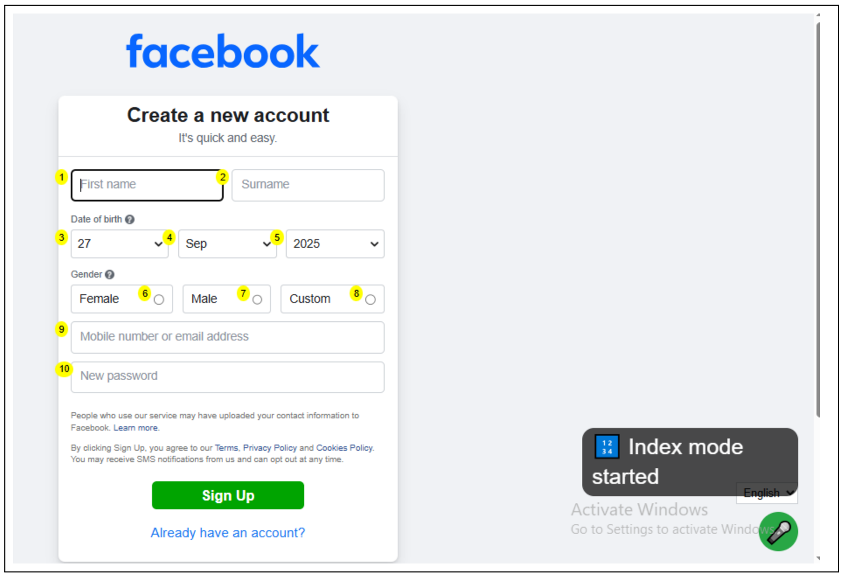
System Workflow:

The overall form handling process can be summarized as follows:

1. Activate Form Mode: When the user says *"start form"*, all form fields like (`input`, `textarea`, `select`) are collected and the first field of the form is focused.

```
formFields = Array.from(
    document.querySelectorAll("input, textarea, select")
).filter((el) => el.offsetParent !== null && !el.disabled);
formFields[currentFieldIndex].focus();
```

Screenshot:



facebook

Create a new account

It's quick and easy.

1 First name

2 Surname

Date of birth

3 27

4 Sep

5 2025

Gender

Female 6

Male 7

Custom 8

9 Mobile number or email address

10 New password

People who use our service may have uploaded your contact information to Facebook. Learn more.

By clicking Sign Up, you agree to our Terms, Privacy Policy and Cookies Policy. You may receive SMS notifications from us and can opt out at any time.

Figure 4.9: Start form mode with index number to each field

2. Navigate Between Fields: Users can move forward/backward in the form using `next` or `reverse`. Each navigation updates the field's current index and focuses to the new field.

```
case "form_next":
  currentFieldIndex++;
  formFields[currentFieldIndex].focus();
  showBubble(" Moved to next field");
```

3. Input Processing: After second step, spoken text said by user is typed into the active field. Passwords can be filled by password suggestions and drop downs are handled by different commands.

```
function handleGeneralInput(transcript, field) {
  field.value = transcript;
  showBubble(" Typed: " + transcript);
}
```

4. Form Submission: After filling up forms, if the user says *"submit form"*, the system detects the form and submit the form.

```
const btn = form.querySelector("button[type='submit']");
if (btn) { btn.click(); }
else { form.submit(); }
showBubble(" Form submitted");
```

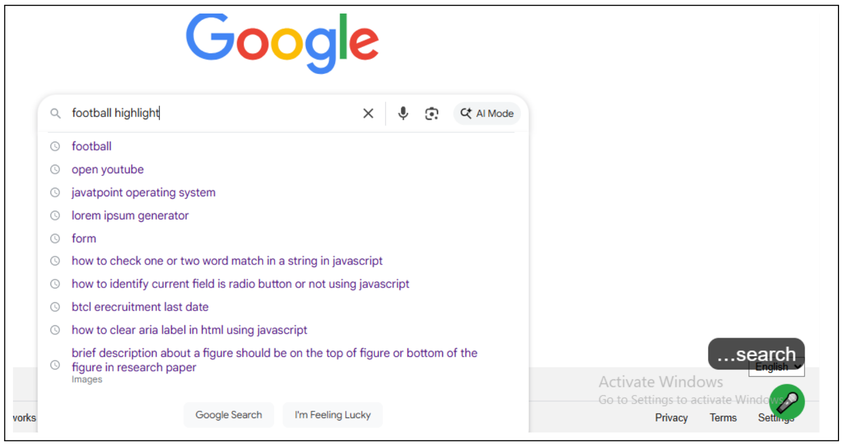
4.3.8 Inline Search

In this feature, user can search from google or youtube by speaking phrase. if current url is youtube, it will search in youtube otherwise the phrase term is searched in google.

When user speaks `"search some_text"`, function `isGoogleSearchPage()` or `isYouTubeSearchP` is called to find out the current url, later phrase is submitted by the google or youtube search bx.

```
function isGoogleSearchPage() {
  return (
    location.hostname.includes("google") &&
    document.querySelector("textarea[name='q']")
  );
}

if (isGoogleSearchPage()) {
    input = document.querySelector("textarea[name='q']");
  }

  if (input) {
    input.value = query;
    input.dispatchEvent(new Event("input", { bubbles: true }));
}
```

Screenshot:



football highlight

football
open youtube
javatpoint operating system
lorem ipsum generator
form
how to check one or two word match in a string in javascript
how to identify current field is radio button or not using javascript
btcl erecruitment last date
how to clear aria label in html using javascript
brief description about a figure should be on the top of figure or bottom of the figure in research paper

Figure 4.10: Inline search in Google through voice

4.4 Testing and Result Analysis

- Functional Testing: In this test the system shows majority of the features works as expected like search commands, tab control, inline search, media control.
- Performance Testing: Measures response time from voice input to command action. It reflects the average time to perform any task. In this section, accuracy of the recognition and scalability are also checked.
- Language Testing: In this test, checks the language is detected successfully, toggling language is worked, and voice command in both languages works just as fine majority time.
- Cross-page Testing: Cross functionality check is perform on different websites(Google, Youtube, Facebook etc.)

Table 4.2 shows that the test results for each feature included in extension. It detect the accuracy of the features by depicting pass/fail and response time for each item.

Table 4.2: Feature vs. Test Result Evaluation[1]

Feature / Test Type	**Status**	**Avg. Response Time (ms)**
Voice-based Search	Pass	350
Tab Control	Pass	140
Form Filling	Pass (minor issue)	3.5
Media Control	Pass	2
Click Links by Number/title	Pass	2.0
Inline Search	Pass	2.1
Language Toggle	Pass	2
Cross-page Testing (Google, YouTube, Facebook)	Pass	2
Recognition Accuracy	80–85% (Majority Pass)	1

4.5 Conclusion

Chapter 4 detailed the implementation and the result analysis of the proposed system. This chapter covers both logical and physical design, technologies used, key features of the system. Also screenshots and performance and result tests confirm that the system provides responsiveness, scalability, accuracy and multilingual support. The result shows that the system is successfully translated from the proposed methodology and ready to enhance its features in future.

CHAPTER V

Conclusion and Future Works

5.1 Major Findings of the Work

The Voice Search Chrome Extension project shows a successful implementation of a voice-controlled browser interface, enabling users to communicate with browser in English and Bangla and perform various tasks. The major findings of the work are summarized below:

1. **Functional Achievement:** The proposed system successfully implemented core features: voice-based search, click highlighted links, read selected text, form filling and media control.

2. **Multilingual Support:** Supports both English and Bangla commands, display the current language, properly identify the language and perform actions.

3. **Real-time Performance:** Showed real time performance due to the installation on the client Chrome browser and provided smooth and real time user interaction.

4. **Accuracy:** Achieved almost 90 percent accuracy for English commands and 75-80 percent accuracy for Bengali commands indicating reliable accessibility.

5. **User-Friendly Interface:** The floating mic button provided easy access to the user to interact with the system, visual feedback improves user experience.

6. **Modular Architecture:** The system is modular and scalable. New commands can be easily added to the further enhancement.

7. **Performance:** Although speech recognition module sometimes took time to take accurate input but command parsing is very fast. Overall performance is acceptable in real world.

8. **Testing and Validation:** Functional, performance, language and cross-page testing demonstrated that the system is robust, reliable, fast and user-friendly.

5.2 Future Work

Although the current system achieved the primary objectives, there are several places to improve and enhance this system in future:

1. **Expand Language Support:** To add more languages apart from English and Bangla make the system widely acceptable to the wider region.

2. **Advanced NLP:** To integrate NLP techniques to understand complex commands improving flexibility and accuracy.

3. **Customized Commands:** This is an important feature which allow to add commands from user to make this system more user-friendly to the user and reliable. It will allow users to add custom voice commands and map them to specific actions in the browser.

4. **Improve Accuracy:** Accuracy is one of the major problems in the speech recognition modules. So by adding noise reduction and filtering techniques, the accuracy of the speech recognition can be improved in future.

5. **Performance Optimization:** Reduce the latency between voice and execute action will improve the response time.

6. **UI Enhancement:** Provide more attractive visual feedback, such as color scheme, tool tips and animations will make this system more attractive to the users.

7. **Integration Other Features:** Summarize the content of the current page,extend support to the email, google maps, google docs or social media platforms make this system more valuable to the users.

8. **Analytics and Usage Statistics:** Add a module to track the command uses, success rate/failure rate of the command execution help developers to optimize performance.

9 **Security and Privacy:** Enhance security by processing voice data locally will ensure user privacy.

5.3 Conclusion

The voice search chrome extension demonstrates the implementation of a successful voice controlled browser extension with dual language support and provides real-time performance, scalability and user-friendliness. The modular design and tests confirms that the proposed system is fast, reliable, secure and easy to use. The future work outlined above provides a roadmap for enhancing performance, reliability, scalability and making the system more accessible and more intelligent.

References

[1] Sudipta96, “Test results for voice-controlled web navigation system.” https://github.com/Sudipta96/A-Voice-Controlled-Web-Navigation-System/blob/main/test_results.json. Accessed: 28 September 2025.

[2] Handsfree for Web - Voice Control, “Handsfree for web - voice control.” https://www.handsfreeforweb.com/en/. Accessed: 2024-05-15.

[3] Readme - Text to Speech (TTS), “Readme - text to speech (tts).” Chrome Web Store. Accessed: 2024-05-15.

[4] Hey Buddy - Chrome Voice Assistant, “Hey buddy - chrome voice assistant.” Chrome Web Store. Accessed: 2024-05-15.

[5] LipSurf - Voice Control for the Web, “Lipsurf - voice control for the web.” Chrome Web Store. Accessed: 2024-05-15.

[6] Voice Typing, “Voice typing.” Chrome Web Store. Accessed: 2024-05-15.

[7] Speech Recognition Anywhere, “Speech recognition anywhere.” Chrome Web Store. Accessed: 2024-05-15.

[8] Text to Speech that Brings Productivity, “Text to speech that brings productivity.” Chrome Web Store. Accessed: 2024-05-15.

[9] Voice In Voice Typing, “Voice in voice typing.” Chrome Web Store. Accessed: 2024-05-15.

[10] Dictation for Gmail, “Dictation for gmail.” Chrome Web Store. Accessed: 2024-05-15.

[11] Talkie: Text-to-Speech (Many Languages), “Talkie: Text-to-speech (many languages).” Chrome Web Store. Accessed: 2024-05-15.

[12] NaviVoice: Voice Input Productivity Assistant, "Navivoice: Voice input productivity assistant." Chrome Web Store. Accessed: 2024-05-15.

[13] Voice Actions for Chrome (Beta), "Voice actions for chrome (beta)." Chrome Web Store. Accessed: 2024-05-15.

[14] Say Play, "Say play." Chrome Web Store. Accessed: 2024-05-15.

[15] Neo, "Neo." Chrome Web Store. Accessed: 2024-05-15.

[16] Speech to Text (Voice Recognition), "Speech to text (voice recognition)." Chrome Web Store. Accessed: 2024-05-15.

[17] Capti Voice, "Capti voice." Chrome Web Store. Accessed: 2024-05-15.

[18] Voice Control for Video, "Voice control for video." Chrome Web Store. Accessed: 2024-05-15.

[19] VCAT, "Vcat." Chrome Web Store. Accessed: 2024-05-15.

[20] Natural Reader Text to Speech, "Natural reader text to speech." Chrome Web Store. Accessed: 2024-05-15.

[21] Google Voice Access, "Google voice access." Accessibility Tools.

[22] Amazon Alexa Browser Integrations, "Amazon alexa browser integrations."

[23] A Voice Controlled E-Commerce Web Application, "A voice controlled e-commerce web application." ResearchGate, 2018. Accessed: 2024-05-15.

[24] IEEE/ACM Publications, "Research studies on voice-based human-computer interaction (hci)," 2019.

[25] Elsevier Journals, "Accessibility-focused studies on multilingual tts and browser integration," 2020.

[26] V. Kĕpuska and G. Bohouta, "Comparing speech recognition systems (microsoft api, google api, ibm watson, cmu sphinx)," *Int. Journal of Advanced Computer Science and Applications*, vol. 8, no. 10, pp. 186–194, 2017.