Acceleration of Web Interface Generation Through Voice Commands

Himanshu Rattan, Shantnu Agarwal and Poovammal E

Department of Computer Science and Engineering, SRM Institute of Science and Technology, Kattankulathur, India

hv5471@srmist.edu.in

Abstract. Advancement in technology is useless unless it is accompanied by a greater access to technology. Be it lack of technical knowledge or a dearth of ability to comprehend or reason, technology must allow low-code or no-code access to all. This work tries to bridge this accessibility barrier by bringing Natural Language Processing (NLP) to the website development process. Using NLP and Azure Cognitive Services, we will be able to understand what elements the user wants to add to the website by extracting the required information. Dynamic generation of web pages using voice commands from the user will not only increase low-code accessibility to naive users, but will also speed up work for experienced users by allowing them to easily add elements to the website.

1. Introduction

With the advent of modern web technologies, we are looking to automate the development work flow. We can trim down the time and resources that are needed to set up basic elements. These elements do not require much skills or specialization. Examples include generating boilerplate to test backend services, making simple websites or prototyping. Skipping the initial process of development helps focus time and resources on points of actual concern. These may be client-specific requirements or the need to add sophisticated business logic to web services. Several solutions are being developed with collaborations between industry experts and researchers to accomplish just this. Endeavours such as developing web designs and using artificial intelligence to read and write static code, or web interface prototyping aims to achieve similar goals. Researchers have attempted bit-by-bit to solve the challenge of accessibility towards web page generation. However, no real progress has been made thus far. Researchers have tried to work with offline methods for speech recognition in order to provide alternative methods for user input. However, they have failed to achieve satisfactory results with offline methods [1]. Hence, authors have used switched to cloud based solutions [1], [2] like Microsoft Azure or Google Cloud Platform.

There is no single existing way that will allow developers to get a quick head start in their projects, nor there exists any viable solution which will enable people with low-code or no-code access to implement their own web-based interface. Developers are forced to start from the absolute basic learning process about web technologies. If a solution could be developed that solves this problem, it would revolutionise the software industry. All web-oriented projects begin with the development of a basic layout. This is an expensive and time-consuming task which does not require highly skilled professionals. Hence, expertise of skilled people can be better put to use doing other tasks. This work
is also a way to increase efficiency in the work place. Saving important man-hours of skilled professionals is a big cost-saver for companies working on tight budgets.

Outside of professional development, existing web page development techniques deny access to non-trained individuals. Widely, web development is restricted to skilled professionals alone. Traditional ways to generate webpages and web interfaces are not easy to access and also take up to a few hours just to generate a basic layout. The outcomes of such efforts then have to be made responsive so that it can adjust to different screen sizes. The issue for responsiveness is solved by availability of CSS frameworks like Bootstrap, which makes it convenient for software engineers to get started with easy-to-use syntax and also provide a good documentation. Even though, these frameworks still cannot be used by amateurs and their use is still limited to that by professionals.

2. Related Works

We have surveyed few papers published internationally and made an attempt to overcome some of the limitations that other researchers have been unable to overcome.

2.1. HTML Voice

This paper [3] helps user to generate web pages with voice input. Python has been used for most part of their work and PyQt5 has been used for GUI generation. Voice command is converted to text with Google Speech API. The recognised command is then tokenised and Part of Speech Tagging is applied. The understood input is then processed and tokenised words are looked for in the code dictionary created using sqlite3. The extracted knowledge is mapped with corresponding HTML and written and saved in a HTML file. Hence this work allows the user to create static pages with some styling and is available in an executable file. This limits its use to Windows-based computer systems. The software implementation needs to be downloaded and installed on the local system. As a future scope, CSS frameworks can be added and the system can be made language independent.

2.2. Voice Driven Dynamic Generation of Web Pages

In this paper [1], user’s input is taken as voice commands. Google’s Speech text API has been used as it offers a better speech recognition model. The converted text is shown to the user to confirm that the speech input has been correctly understood. After user’s approval, text is broken down and phrases are tagged using a pre-defined dictionary. Then the sentence is broken down and Part of Speech Tagging is applied. Based on this, the code is written by the system. The system offers multi-input functionality with support to add and remove elements at any stage. As a future scope, multi-language support can be added to the program. Suggesting and predicting templates using Artificial Intelligence can also be implemented in the future. The generated output is a static web page which seriously limits its application in the real world. Moreover, the output is also not compatible with mobile devices which comprise the majority of internet users.

2.3. VoiceEye: A Multimodal Inclusive Development Environment

The authors [2] have focused on providing greatly increased accessibility to development for people with disabilities. However, this comes at a great cost that requires specialized hardware switches. With the help of Microsoft’s Azure, they have implemented voice inputs. A gaze input is also provided that allows users to provide input letter-by-letter by gazing at keys on an on-screen keyboard. Further, they have evaluated the performance of their system by testing it with disabled as well as non-disabled people. The authors have proposed that improvements can be made to support the debugging process and provide auto completion.

2.4. Traditional Code and Run Techniques

Basic web development process begins with the code being written from scratch in HTML, CSS and JavaScript and serving them to clients over a web server. This way of development requires moderate understanding of underlying principles and is easy to understand. However, this also means that the
complexity and functionality of the resultant website are limited to what can be achieved within the project timeline. Since the code is written from scratch, there is often no time left to add advanced functionalities or introduce new concepts.

2.5. Web Framework and Libraries
Professionals make use of web frameworks and libraries such as React to speed up the development process. These provide a pre-written and well documented code base that is easy to build upon. Advanced functionalities are easier to implement and more time is available for deep customization of work. However, this also means that people with low-code or no-code access cannot build their websites using these technologies without hiring professional web developers.

2.6. Website Builders
There also exist website builders such as Wix and Microsoft’s Sketch2Code. Wix allow users to create a website using drag and drop functionalities. On the other hand, Sketch2Code transforms hand-drawn designs into HTML code with AI. The picture is also used to improve the underlying AI. These tools are useful for people who have absolutely no idea about programming but most of these services are paid and don’t offer access to the files generated at the backend.

One of the major challenges that has been faced by these works is making the generated web page responsive. Also, their final products are available as an executable file which can run on a desktop only. This also adds extra security clearance requirements that need to be met before using in a professional environment and thus limits the use cases. There are better alternatives available to Google’s Speech API which could have been used to accomplish healthier results. Limited functionality, lack of scalability and non-compatibility of generated output with mobile devices are also some of the limitations in these works. They need to be worked upon and solutions or enhancements needs to be made.

3. Inferences from the Related Works
3.1. Understanding the Importance of Speech
Humans express themselves in the most effective way by communicating through speech. It is their natural way of sharing thoughts with other humans. This knowledge can be incorporated to develop a solution which uses voice as input to develop exactly what the user desires. Considering speech as a system input command will enable easy interaction with the computer. This does not mean that the user would speak each and every word that has to be written in code. Instead, understanding the user’s natural language, building a system which understands it and then writes the appropriate code intelligently are the main goals. A neat interface that proactively shows the user what the system is comprehending from the voice input and simultaneously shows the results will achieve the goals described above. Such a solution could be used to shorten project timelines, create rapid web pages and also for learning through reverse engineering of concepts. By making use of Microsoft Azure Cognitive Speech Services instead of Google Speech to Text API [1], [2], we are able to achieve better results in understanding voice commands than custom trained voice recognition models.

3.2. Understanding the Impact on IT Industry
Since the basic tasks of developing static frameworks would be eliminated, professionals currently working in these fields would need to be reskilled. Businesses and individuals looking to take their businesses online would save expenses on hiring IT professionals and there would be less demand for freelancing of static websites and interfaces. Designers and developers work in cohesion to bring out the best possible product for the end-user, but this ideal case harmony is rarely achieved in real life. Allowing the designer, who has the idea in their mind, to experiment with web page generation will ultimately help in delivering the best possible product. The responsiveness that lacked earlier implementations [1], [2], [3], [4], [5] has now been achieved by our solution thus eliminating the need
to have responsive designs being made before work can start on a project. This will further help shorten the project timelines.

4. Advances Done in This Work / Methodology Adopted
Work by various authors [1], [2], [3], [4], [5] relied on locally installing software which hindered the ability of enterprise users to access such services on workplace system due to possible security issues. By ensuring that this work runs on the Cloud, we are enabling professionals to readily make use of this work on the corporate internet.

Microsoft Azure is a top-rated service for cognitive speech services and offers services at a lower price than competition. It allows us to incorporate speech inputs in regional dialects, while also allowing us to develop custom models for regional dialects when needed. This improves transcriptional accuracy. This would not have been possible by making use of other vendors such as Google Cloud which have been used by other authors [1], [3].

This work also adds a new feature which permits the user to download the source code from the cloud. The user is then free to edit the code as per their needs if they wish to do so and if they possess the technical expertise that is needed for this task. This work also trims down on the need to use specialised hardware. Such hardware was used to allow accessibility to disabled people [2], [6], but this work overcomes this requirement by making use of voice commands.

By making use of Flask in this work, we are also ensuring that the system remains scalable when deployed to the cloud for public use. Cloud deployment also has the added advantage of being modular and providing a Unicode based development, thus allowing us to add support for non-ASCII languages in the future. A fast debugger and a built-in server enhance the flask development as well.

4.1. Features and Provisions
This work empowers the user to get started with helpful features that would otherwise be time-consuming or challenging. Provision is made for user to understand which commands can be used and how to proceed with it. The home page provides helper commands to get the user started. Below are some functions that are supported by this work. It should be noted that the voice input is in natural language and thus there is no need to speak the exact statements as given below. The available commands are:

4.1.1. Create a table. When the user asks for a table to be coded, the system will prompt back to determine required parameters like number of rows and columns. As per the specified input, a responsive table layout with desired rows and columns is generated. A modular approach has been followed to break down complex inputs in to simpler formats. Example, ‘create a table with x rows and y columns’ has been broken down in to 3 inputs ‘create a table’, ‘x rows’ and ‘y columns’. This makes it easier for novice users to make use of this work.

4.1.2. Add navigation. Allows the user to create a navigation bar on the page with customised headers. The words to be written as headers are user definable. They can also be provided as follow-up inputs through voice commands.

4.1.3. Change background colour. By default, the background colour is set to white. By making use of a similar command, the user can change the CSS background property. The change is affected in the inline CSS of the body tag in the HTML.

4.1.4. Add heading. Headings can be added to separate divisions in the web page. These are Section-level headings, marked in HTML by the <h3> tag, can be created using this command.

4.1.5. Add paragraph. Voice dictated paragraphs can be added in the file. The exact words to be written in the paragraph are transcribed using speech-to-text conversion.

4.1.6. Add input field. If the user wishes to create a form, they can create input fields for it with the help of this command.

4.1.7. Add button. Button elements containing custom text can be added as well.
On a user experience enrichment note, there are no restrictions on the order in which the commands are delivered. This also increases the scope of this work and brings it closer to practical everyday use.

5. Data Flow

![Flow Chart showing logical architecture of the work](image)

Figure 1. Flow Chart showing logical architecture of the work

Figure 1 explains the flow of control in a typical use case scenario. This visualizes the flow when a user triggers the application to develop a web page. It describes the end-to-end flow starting from the trigger to the end of execution of the logic.

**Step 1** Natural Language Input: The user starts with a voice input stating which web elements he wants to add on the webpage.

**Step 2** Convert input to temporary audio file: Voice input is accepted through the default system microphone and stored in a temporary audio file.

**Step 3** Decode audio with Azure Cognitive Services: This file is transmitted to Azure Speech APIs where the transcription takes place.

**Step 4** Identify words of interest: The transcript text is then used to identify the user’s needs. This is done by looking for supported keywords and web elements.

**Step 5** Check for existing file: If a work-file already exists, it is used and the requested elements are added to it. Otherwise, a new file is created and is used as the default work-file for future commands.

**Step 6** Reference bank: A repository of boilerplates for supported elements is referenced to add the code for corresponding user input. At this point, the system also checks if the desired boilerplate requires additional input from the user.

**Step 7** Additional Input: If the boilerplate for the desired element is such that additional input is required, then user is provided with appropriate prompts.
6. Results and Discussions

In our tests, the user takes 122 seconds to develop a web page using each and every feature available in the work. This is done totally through voice commands. In contrast, an average experienced user took 210 seconds to develop the same web page, but making use of Bootstrap to write the code themselves. Here is how a user will be able to create a static web page using voice commands.

Initially the user is present at the home page as seen in figure 2, where some sample voice commands are shown. The user can try any command at any time. The input command does not have to be exactly the same but must contain the keyword which tells which element to insert in the web page. Consider the user clicks on the button “Speak” and says the phrase “Create a table”.

![Acceleration of Web Interface Generation Through Voice Commands](image)

**Figure 2.** Home Page

The Microsoft Azure Cognitive Speech Services then transcribes the input command. The code base realizes that the user wants to create a table and redirects it to the associated page which in this case asks the user to mention number of rows for the table as shown in figure 3.

![How many rows?](image)

**Figure 3.** Asking the user how many rows to add in the Table.

![How many Columns?](image)

**Figure 4.** Asking the user how many columns to insert.

After the user mentions the number of rows, it will be asked to provide the number of columns as well as shown in figure 4.

Then the user will be redirected to the result page as exemplified in figure 5 where they can either give another command or view the generated HTML file. Complex commands have been broken down to simple ones to increase user friendliness and help naïve users generate web pages. The table is filled with dummy data which can later be changed in the HTML file.
In case the user input command is not recognized by the system, an error is displayed to the user as shown in figure 6 which asks the user to provide the voice input again. Error handling is an important feature that proactively alerts the users about issues at hand. Thus, allowing them to correct their inputs or trying again with a different one. Once satisfied, the user can download the web page with inline CSS. This will empower the user to customise and improve upon the output of the work as per their needs. This is especially useful when the user wishes to fast track their development process in a professional environment.

Other voice commands can be given one by one. Each command would then result in redirection to the associated page. All the commands except “Add a navbar” can be used any number of times. The final result could look like what is shown in the figure 7. Making use of Bootstrap v4.5 has an added advantage that allows us to efficiently generate the HTML file and the resultant web page is responsive for all the screen sizes as shown in figure 8 with a smartphone device.
7. Limitations

Our success is limited by various constraints that currently include voice input, which is limited to clearly pronounced English language. While web pages elements like navigation bar, table, input fields, labels and headings can be added and background colour changed, other tasks still have to be done manually which requires technical know-how of web development. Advanced customisation of elements, for example removing is yet to be achieved. Although the user can add elements, there is currently no way to uniquely identify and remove an element once it has been added to the web page. Any such intention has to be dealt with by starting from scratch. Editing of existing elements is also not currently supported but can be built upon this system using the same principle as used to delete elements.

8. Future scope for feature enhancement

Efforts can be made to improve the results by optimising the voice model to capture regional dialects of the same language and by adding support for multi-lingual input. This would allow additional means to take voice commands and thus further spread access to technology to non-English speaking parts of the world. In order to use this solution on-premises, IT administrators would have to conduct a security audit for the app before allowing its use. A cloud-first approach will ensure that the solution is easy to use and does not have a need to install any software package, thus isolating the solution from issues of operating system compatibility and skipping the need to go through a security audit by IT administrators.

Figure 8. Responsiveness of the generated web page on a smartphone device.
9. Conclusions
At the end, the user was able to generate the webpage shown in figure 9 in 122 seconds. This amounts to a 72% faster generation of web page for the same desired output. The actual time difference will be significant for larger works containing more elements. The generated webpage was responsive and was available as a HTML file with inline CSS. This allows the user to edit and host it if they desire to do so. The goal of this paper was to automate parts of web development process. It was essential to make it easier for non-technical people to get started with web page generation. For professionals, this work is a way to jumpstart their development process through boilerplate generation. Since this work has made use of Bootstrap, the resultant HTML file is responsive, mobile-first, developer-friendly. This goal has been accomplished. Azure Cognitive services can be better utilised to support multi-dialect and multi-lingual provisions. Although attempt has been made to simplify the web development process, there is always room for improvement.

![Figure 9](image)

**Figure 9.** Comparison of average time taken by experienced users to generate webpage shown in Figure 7 by different means.

References
[1] Modak S, Vikmani S, Shah S and Kurup L 2016 Voice driven dynamic generation of webpages *Fourth International Conference on Computing Communication Control and Automation* (Pune India)
[2] Paudyal B, Creed C, Frutos-Pascual M and Williams I 2020 VoiceEye: a multimodal inclusive development environment *DIS20* (Eindhoven Netherlands)
[3] Chadha H, Mhatre S, Ganatra U and Pathak S 2018 Html voice *Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)* (Pune India)
[4] Bajwa I, Sarwar W A and Syed I H 2006. speech language engineering system for automatic generation of web based user forms *International Conference on Man Machine Systems* (Malaysia)
[5] Bajwa I S, Siddique I and Choudhary M A 2016 Automatic web layout generation using natural language processing techniques *2nd ICSCCE* (Athens)
[6] Begel A 2005 Programming by voice: a domain-specific application of speech recognition *In AVIOS speech technology symposium-SpeechTek West* (United States of America)