Volume 66, Issue 2, 2022
Journal of Scientific Research
of
The Banaras Hindu University

Acceleration of Dynamic Web Interface Generation Through Multilingual Voice Commands

Himanshu Rattan [0000-0003-4386-3511], Shantnu Agarwal[0000-0001-9377-4942] and E. Poovammal[0000-0002-4996-1377]

School of Computing, SRM Institute of Science and Technology, Kattankulathur, India

hv5471@srmist.edu.in

Abstract. Modern web solutions require skilled efforts from professionals. This is expensive to attain and does not suit the budget and necessity of everyday people. Businesses looking to develop a website, or professional website development service providers can greatly benefit from a new approach to web development. This new approach comes in the form of voice-assisted development which allows the user of a cloud-hosted service to speak in natural language and express their needs to the system. The system follows up to enquire about necessary information and generates the desired result. The user would be able to interact in a language that they can choose themselves. Combined with the ability to enable dynamic web pages, this work aims to bring a revolution in the web development industry.

Keywords: Accessibility, Dynamic Web Development, Multilingual Web pages, Speech to Code.

1 Introduction

This paper aims to propose a new system that allows humans to interact with a voice-based and cloud-hosted service and develop modern-looking responsive web pages in a timely and user-friendly manner. Decades of research and development have been done in the field of web development. Even then, existing methods of web development are mostly only accessible by those having sound technical knowledge about this field. An alternative of hiring web developers or making use of third-party drag and drop web builders are also available to those who are willing to pay for these premium services. These approaches and the problem with them are discussed later in this paper.

A survey has been carried out to validate this work's usefulness. An effort has also been put in to understand user needs and what constitutes a modern web page. This work has been presented to twenty-five people from various walks of life. They were asked to use it and experiment with it. On a broad note, it was observed that twenty-three people found this work to be a better way of developing websites as compared to traditional coding methods.

There have been attempts to modernise the way web development works and to democratise the development process, but these attempts have often focused on improving development facilities for professionals [1]. These include the introduction of web development frameworks. Frameworks can be thought of as libraries or pre-written packages of code that others can use. They allow the developer to reuse the code written by someone else and often helps to speed up the development process. Apart from web development, frameworks also exist in other fields of computer engineering such as data science, where pre-written data models are used by data science engineers. Examples of frameworks in web development include Bootstrap, which is a CSS and JavaScript framework, and others such as React. Bootstrap, React and more are indeed useful for professionals working in web development. However, they hardly make any effort to make it easy for people who do not know how to code to develop web pages.

Dynamic web pages are fed with data from a database engine. Ever since the rise of dynamic web pages in the early 1990s, they have dominated the web. Therefore, it does not make much of a difference if one can automate a process, which results in a rarely used output. That is why this work also allows the user to create a link between the front-end webpage and a database engine. With this, it is now the first time that such a link has been automatically created through voice commands.

In this work, two categories of people have been kept in mind. First, those with low-code access, commonly known as amateurs. Second, those with no-code access, which are people who are not familiar with coding or computer languages. Both require a more user-friendly approach to development and making use of natural language voice input will make it easier for these people to create simple and responsive web pages for their work. The user can speak in natural language and express their needs to the system, which will understand the needs and write code corresponding to the fulfilment of those needs. The option to choose from a multitude of languages to provide voice input can also enable greater inclusion of these people.

Apart from the above-mentioned two categories of people, there is another kind. The third category of people who will find great use and ease of development in this work are professionals themselves. The web development community is poised to evolve and would require highly skilled professionals. Software engineers engaged in web development can also use this work to achieve a head start in their development work when starting a new project. UI and UX developers who wish to test their designs can also use this work and quickly prototype. With the help of this project, it is believed that an industry-wide change may be triggered.

DOI: 10.37398/JSR.2022.660226

As professionals grow more comfortable using this project in their daily work, they will end up saving a lot of time that would have otherwise been spent in writing and testing boilerplate code. Even a small issue during this process of boilerplate generation would take exponentially more time to identify and fix. All this and more would be solved or reduced to a great extent with the adoption of this automation cum accessibility tool. Time saved in this will enable engineers to increase their productivity and avoid downtime. Reduced failure rates of code, due to automation of code generation, will help improve the mental state of the engineers as well. Further, since the task of boilerplate generation will be vastly reduced, many engineers will have to upskill to stay relevant in the industry. This is estimated to maintain high standards of professionalism as well as propel the industry forward.

2 Commercially Available Technologies

A glance at various commercially available technologies suggests that these technologies only partially reduce the complexity of web development.

Wix is one such commercially available service that allows drag and drop service to users at a given price. It offers end-to-end service. Users can create web pages using drag and drop features, but it also forces the users to ultimately host the website through Wix itself. Professionals cannot use it to speed up their work or get a head start. While Wix may help some small businesses, it is ultimately a paid web hosting service with an entirely different goal than this work. This work aims to empower people to build their websites themselves and then take full control of their products. It guarantees full freedom of hosting environment and the final output in HTML format assures that the user can host the webpage on any hosting service of their choice.

Microsoft has developed Sketch 2 Code, a project which is powered through its Azure cloud service. Users can upload their sketches of web pages and then download a suitable HTML file that uses the same design. Though it is also a great product for people to build small web pages, it does not provide any backend development support.

Before any of these above tools were made available to the public, the existence of a few popular frameworks has driven the web development industry forward for the past two decades. Bootstrap and React are two of the more popular frameworks used by professionals in the software industry. They provide extensive libraries and support for most modern web technologies and designs. Web pages generated using these frameworks can be easily optimised to run on mobile devices as well. However, given the complexity of these frameworks, they deny development access to amateur people and even to new developers. There is a steep learning curve that may be avoided unless the person is related to it via their career goals.

People looking to build simple static websites or even those who require an easy way to build a business website with dynamic data inputs must not be held up due to the above restrictions of existing technologies.

3 Related Works

Few researchers have tried to bring about a similar change in this web development over the years. Many have succeeded in developing solutions that would allow users to speak or type in natural language but are unable to cross the threshold of practical utility. Based on the way the user input works, the research works can be classified in the following categories.

3.1 Text Based, Typed User Input

Reference [2,3] provided a great new way to generate forms in HTML. This new way allows the user to provide typed input into the software, which then converts it into input fields and labels that constitute a form in HTML. It is, however, severely limited by the fact that only forms can be generated. No other element of the web can be developed using this work. The authors plan to add support for PHP and ASP.net in the future. Text-based typed input is also a limitation towards democratising web development access to all. To overcome this, the authors also plan to add support for voice-based inputs in the future.

3.2 Voice Based User Input

Chadha, H., Mhatre, S., Ganatra, U. and Pathak, S. [1] have gone a step ahead and provide the ability to use voice-based input. Enabling the user to speak in natural language provides a major uplift in accessibility. It simplifies the process from the user's perspective and allows users to directly speak to the system. Others [4] have also developed a similar approach to increase accessibility in web development. Both groups of researchers have relied on Google Speech APIs to transcribe the user input. Through tokenisation and part of speech tagging, they have been able to provide a proof of concept for a single language, basic elements oriented and voice input-based web development future.

Unfortunately, without support for dynamic web pages, these works do not provide the much-needed backend aspect of web development. They also rely on host OS-based software solutions to run and therefore need to be installed on users' computers.

3.3 Gaze Interaction

In the past, researchers have attempted to develop dedicated IDE keeping in mind similar applications [5]. These authors have worked on a multimodal IDE. Their works make significant strides in providing development access to physically challenged users. By making use of dedicated hardware switches and technologies such as gaze interaction, these authors have developed a new way to interact with the computer and write code. This works alongside the voice-based input technique. The virtual keyboard, whose input works through gazing at a particular key, enables the users to start coding without any physical movements. This comes at a cost of requiring specialised hardware that is needed, i.e., the hardware switches and camera setup. They plan to work on popular IDE features such as code completion and debugger.

Powered by the Microsoft Azure cloud, this work takes a different approach to solving the challenge of accessibility for

all. It is limited by the accuracy issues that come with gaze interaction and the need to install the IDE on the host computer. Making use of cloud technologies to develop a web app, would have allowed professionals working in secured environments to use the work.

3.4 Inadequacies and Scope of Improvement

All works published in the past on similar topics had certain common limitations that restricted their practical use in daily life. Perhaps the single biggest shortcoming was the executable-package nature of the work. This meant that each user had to separately download and install an instance of the work on their local machine before even trying it out. This severely limits its use in an office environment where installation of third-party applications is a big and tedious task that is rarely undertaken. Using a cloud-based solution would not have been possible in older works, but all modern works are expected to be cloud-hosted. This drastically increases the practical application of the work.

Another major showstopper is the absence of responsiveness in the generated outputs. A responsive page is one whose elements realign and resize to fit the aspect ratio of the client device. The availability of internet-connected devices in various shapes and forms has made it virtually mandatory for web developers to develop responsive pages. The majority of users today are using mobile devices and lack of support for a responsive web page in existing solutions prohibits many developers from using them.

Other less significant issues that were found in past works are the inclusion of only one language for input [1-7] or the use of inferior speech recognition tools. This work emphasizes these points and goes a long way to enable natural language input in up to three languages: Hindi, English, and French. However, the use of Azure Cognitive Services allows us to enable natural language input in any Azure-Cognitive-Services-supported language, without much technical effort.

4 Advances Done in This Work

For this work to be of actual use to both professionals and novice users, it needs to be closely linked with the ground reality of the web development market. This paper proposes multiple ways through which users will be able to make use off when working on websites.

There are some key functionalities that people look for when looking for ways to make a website, or discovering new ones. These are the ability of the code to re-align the HTML elements as per the display, support for up-and-coming cloud services, linking with a backend database, and user-friendliness. Keeping this in mind, this paper focuses on all these functionalities and more innovation, which has not been accomplished before.

These advances have been achieved by making use of Python 3 behind the curtains to build a functional application for the paper. Flask, a micro web framework was used to build the web application in cohesion with Bootstrap to keep the web page responsive. The multilingual support has been built by developing wrappers around the Azure Cognitive Service APIs and SDKs. To achieve dynamicity, the forms have been linked to MySQL.

4.1 Responsiveness

With the ever-expanding inventory of devices available in the consumer electronics industry, people are using displays of all shapes and sizes to access the web. The past trend of using desktops to browse websites because the mobile sites were not easy to use [8], is over. This has brought a wave of website updates and upgrade, wherein websites are now being developed to be compatible with any display size, resolution, or aspect ratio. New web technologies are enabling the different web designs to be shown in different viewports. Or in the modern scenario, websites are expected to come off as responsive on their own. This means that the elements on the webpage must reflow and resize according to the resolution and orientation of the viewport.

The web pages generated using this work, are designed to be responsive by default. Any user who creates any HTML element using this work, will by default see that the page reflows and adjusts as per the screen. Responsiveness has been achieved through the use of the open-source Bootstrap framework. Its libraries and classes have been used and configured such that they are dynamically added behind the scenes, during the generation of the webpage through voice commands.

4.2 Cloud Deployment and ZIP Export

Professionals working in the IT industry are often working on systems with restricted access to install new software. Getting software unblocked and ready to be installed in a work environment requires an IT and security clearance. For this work, such clearance is also unnecessary and should be avoided, as no data collection or file access is being done by the system. Keeping this in mind, this system has been optimised and designed to run on the cloud, more specifically as a container in Microsoft Azure [9].

It empowers users working on restricted access systems to still be able to generate web pages quickly, without compromising the security of their company by installing third-party software.

To complement this, it is imperative that the user be provided with the raw code of the generated output. This is the main reason why a developer would use this work in the first place. At any given stage during the development process through voice commands, users have the option to export the generated HTML file along with its inline CSS classes.

In situations involving dynamic content, i.e., when the user has generated a backend linked webpage, the export file will contain a server file and the SQL script needed to go along with the webpage. All form data inserted in the database is logged after the data query is dynamically generated for each entry. It ensures that at the time of download, all form data is also exported and easy to reproduce for the developer. The export is provided as a ZIP file, which contains all the three files mentioned above.

4.3 Dynamic

As is the case with the majority of professional development work, dynamic web content is in high demand. Professional web developers are constantly working on dynamic websites and often waste many precious man-hours to build basic front-end to

back-end connectivity. Using natural language input commands such as “insert a form” or “make the website dynamic”, the system understands the need to connect to a database. Instantly, the system auto-generates required server files and a database in MySQL. This database is linked to the user session and allotted name as per session-id and system time.

Upon successful web page creation, all data entered in the form fields of the webpage is dynamically coded as SQL statements [6,11]. These statements are executed, logged, and stored in a separate SQL file. If the user desires to download the generated webpage to use elsewhere, the generated server file and the SQL file are zipped along with the created webpage files and exported to the user’s system through a file download.

While such functionality was mentioned as future works in previous works, it was never brought to life. It is expected that the ability to add dynamic features, such as form data collection, simply through voice commands will empower more people to develop their websites and even help save precious man-hours at IT offices.

4.4 Multilingual

Past works have relied upon basic text inputs or single language voice input to bring accessibility and improve efficiency in a professional environment. But not all people are made the same. They speak different languages and business requirements often require the same website to be made for different regions and locales. For this reason, this work goes one step ahead and allows the user to provide natural language input in up to three languages: English, Hindi, and French.

If the user starts with Hindi, they can change to English input at any point in time. This not only helps people use the system in different languages that they know how to speak, but it also allows the creation of a single webpage containing multiple languages at the same time.

The ability to provide input in various languages also inadvertently creates an issue where the system needs to differentiate between when the input is to be translated and written into HTML and where it has to be directly written to HTML without translation. This may occur when a user creating a webpage in French wants to add a paragraph in French to the website. Since the system is translating the voice commands to English to match with the code dictionary, it is important to differentiate and understand the second command where the paragraph being spoken is not to be translated and matched with the code dictionary.

4.5 Cognitive Services

Humans can comprehend words into meaningful logic through the cognitive ability of the mind. This power is passed on to computer systems through cognitive services. Inspired by previous works [12], when the user speaks, the speech signal is digitized and some signal processing is carried out to create a template for the voice pattern and this is stored in memory. A copy of this file is flowed to the cognitive service.

When the user’s natural language input flows in the cognitive service, it is first checked whether or not translation is needed for the said input. A phonological model consisting of a large corpus of speech data from multiple speakers is available for use on Azure.

The system attempts to match the natural language input with the phonological model. Then, using the grammar and dictionary for the input language, a string of words is generated. If the input is such that it requires the system to translate it from one language to another, then a subsequent process follows. In this step, the generated string of words is taken up and translated into the intended language. This is done keeping in mind the context of the sentence since modern systems are long past the days when translation used to work on a word-by-word basis. Such a way is considered more user-friendly than directly speaking code [7].

Another application of a similar concept of recognition of user commands is done when configuring the webpage as dynamic. To retrieve information from a database, one needs to formulate a query in such a way that the computer will understand and produce the desired output. In our case, query processing is done through MySQL statements. It is not expected from non-professionals to be able to write MySQL statements and thus there is a requirement to make basic use of SQL statements and execute read/write queries, without actually interacting with the database.

The system auto-generates database queries for the user. Here, the system must differentiate between input data and database-identification information such as table and column names. To solve this issue, we are going for a simple approach where the HTML input field labels are used as column names, and the table names are generated through a combination of unique session IDs and timestamps [6]. Hence, using minimal processing of input field data, we can fulfil a critical functionality.

5 Implementation of New Approach

5.1 Architecture

The system design as seen in Fig. 1. has been created keeping in mind a modular approach. This ensures accessibility to the work by people of all experience levels. Multiple revisions were made to best suit the dynamic webpage generation needs of this work.

The session starts when the user logs on to the site. If the session doesn’t already exist, a new session is created for the user where they can save their progress. After the session has been started, we check if the file has been created for the user where they will create their dynamic page. If the session has just begun, a new file will be created and the user can start developing the page with voice commands.

After the session has begun and the file has been created, the user can select a language from the dropdown to start giving instructions. If nothing is chosen, English will be set as the default language. We can extend the support to any language and dialect supported by Microsoft Azure cognitive services. Microsoft Speech to Text can help us transcribe audio to text in more than 85 languages and variants. For now, this work only supports Hindi, English, and French.

Thereafter, the user can click on the speak button and try any of the mentioned commands. The command doesn’t necessarily have to be exactly the same as mentioned on the screen but it should contain the keyword which maps to our code dictionary.

As soon as the user gives the input, text pre-processing is applied to convert the transcribed text to the desired form. If the command isn't understood by the system, an error message is prompted to the user.

If the received input is in English, then we don't need to translate it, and hence it is directly forwarded for transcription. But, if the received input is in any of the other supported languages, then we check if a translation is required on the input or not. Some non-English inputs don't require a translation and need to be transcribed directly. For example, if a user is adding content to the webpage, then there is no need to translate that. Hence, we can directly transcribe the received input and add it to the HTML file which will be generated for the user. However, if the voice input is meant to be an instruction, rather than content input, then we need to translate the command first. This is done because an instruction can only be processed after referring from the codebase which is in English.

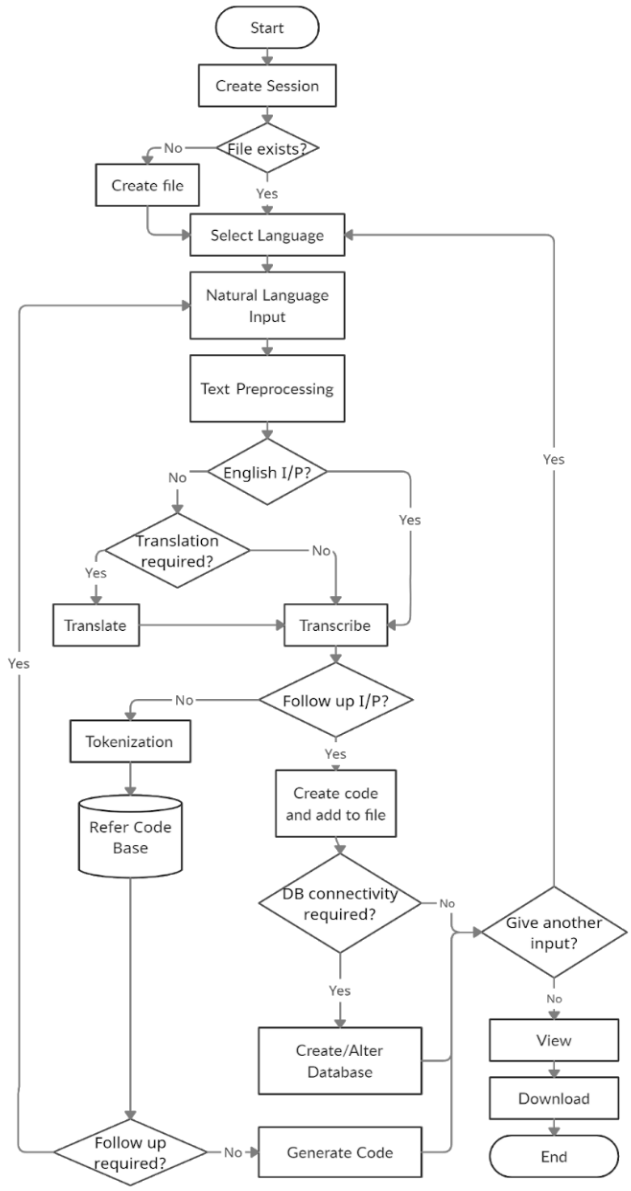

Fig. 1. Flow Diagram.

In the next step, we need to check if the input received is a follow-up input or a fresh command. If it is a follow-up input, it can be converted to code by applying appropriate processing and adding to the file directly. Due to the dynamic nature of the work, most commands require a follow-up input which need not refer to the codebase. However, all the instructions or commands need to be followed up by an appropriate second input to complete the instruction and add it to the file. For instance, if the user gives the command, "Add a heading", we need to pursue this command by asking a fitting question, which is "What is the heading". Here the follow-up question is sent to the user after tokenizing the first command and referring it from the codebase. Similarly, all the instructions have mapped follow-up inputs in the codebase which are prompted to the user after the initial command.

Such a method helps us modularize this approach and provides an easy way to give input to novice users. This can help break down complex commands into multiple shorter commands. It also assists the system and reduces the error rate as we know exactly what kind of input we would be receiving.

After the follow-up input has been converted to code, it can either directly be added to the file or lead to some other operations which are performed on specific instructions. If the user wants to make the page dynamic, database connectivity will be required with the frontend. The user can give the command "add a form" or "make the page dynamic" which will allow the user to collect form data from the page. The system will generate a server file and a SQL file. The server file processes all the input and runs on the server-side. It will connect the form to the database and insert the form data into the database. As soon as the user commands the system to make the page dynamic, a table is generated at the backend with some unique values and the timestamp as its name. All the further queries are saved in the SQL file which can be further exported by the user to run at the hosting provider of their choice. When the user adds the form, it is followed up by asking the input names which are then added as columns in the database table.

The command "Add a form" and "Make page dynamic" have been added separately because the user might want to add an input field without the backend connectivity. Such a feature helps novice and professional developers create survey or feedback pages quickly without any hassle.

Similarly, all the commands are appended in the output HTML file. The user even has the choice to change the input language after every instruction. This can help them create multilingual web pages which can attract a diverse audience. Such pages would be very difficult to create with standard coding mechanisms as it is not easy to provide such multilingual inputs. Here multilingual voice inputs have an edge over the traditional methods. The output will be displayed to the user after every command so that they can monitor the results and continue with the required changes.

5.2 File Download

After the user is satisfied with the page and is finished with the development, he/she can download all the necessary files. The output is provided as a zip file. If it's a static page, the zip file includes only the HTML file. If it's a dynamic page, the user is provided with HTML, server as well as SQL files.

Hence the user is served with all the requisite files which they can host in seconds at the hosting provider of their choice. This quickens the development process for the user and allows them

to create web pages in a feasible and adaptable way without going into the depth of what's going on inside.

Microsoft Azure Cognitive Services offers best-in-class voice recognition. It is better suited to handle native accents of various languages. In our context, accurate recognition of English (India) is a key to the working of this system and was one of the deciding factors while scouting for cloud services.

5.3 Why Azure?

The Word Error Rate (WER) is drastically reduced down to 6% when making use of Azure Cognitive Services to translate and transcribe Indian languages and accents. High-quality transcription provides us with accurate speech-to-text transcriptions with state-of-the-art speech-to-text software.

Its customisable models allow us to add specific words to the base vocabulary and also to build our speech models. It offers scalability and high availability, which is integral to this system [10]. Direct access to the same robust technology that powers speech recognition across Microsoft products is available. It is relied upon by IRCTC to handle 10 billion interactions per year.

6 Illustrative Example

To generate a complex webpage, the user can continue giving commands one after the other. Some command hints are shown on the landing page to get the user started, as seen in Fig. 2. The system will consequently process and execute the user input. These are then appended to the same HTML file for the given user.

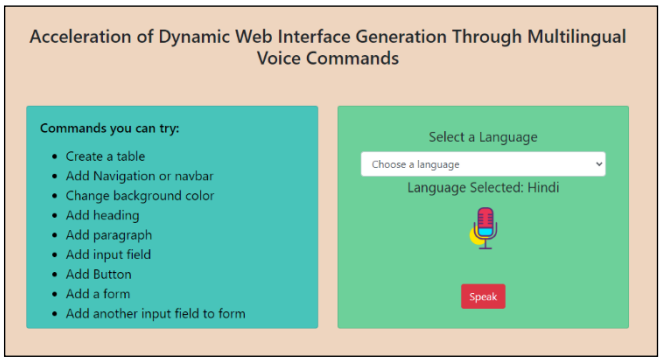

Fig. 2. Landing Page with Command Hints

Once the user reaches the home page, they'll be asked to select a language or continue with the default language which is English. The dropdown gives the user an option to choose from a wide variety of languages and offers support for multiple dialects as well. Once the user clicks on speak, they can continue to give a command in the selected language.

After an instruction is given, it will be followed up by redirecting the user to the follow-up page. For example, if the user gives the command "Add a dynamic form to the page", they will be asked to name the input fields that they want to include as shown in Fig. 3.

Fig. 3. Designated follow-up input.

Such a command will create a table in the backend to store the data that comes through this dynamic form. To avoid ambiguity between the databases of multiple users, the table name is defined using the system time. This allows a user to create multiple forms and databases unique to their website. Using similar commands, the user can keep adding input fields to their form. Whenever a new input field is added, the system will alter the same table [6]. All the SQL queries will be saved in a SQL file so that the user can just import the file and not deal with the hassle of creating a database again.

Hence the user is creating both the frontend and the backend for their website just using voice commands. The operator is even allowed to give the succeeding inputs in a different language and create multilingual web pages. This would be difficult to achieve using traditional coding methods. Such a feature will enable the website to generate traffic from a wide diversity of the audience.

7 Results and Observation

To properly analyse the results of this work, a group of twenty-five people from various walks of life was requested to be a part of a survey. This survey, conducted remotely, helped understand the practical scope of this work and test its feasibility in fulfilling the goals defined above.

7.1 Selection of Participants

For the most accurately described results, it was required that the survey be participated in by common people from different walks of life. Of a total of 25 final participants, 40% were women. 32% of participants did not have any prior experience in web development. This set of people was crucial to the survey since non-professionals are a major focal point for this work. Main highlights such as natural language input in multilingual inputs are specifically focused on the non-experienced group of people. Importance was given to the multilingual nature of this work. Even though the mother tongue of the majority of participants (56%) was Hindi, there was at least one representation each from Gujarati, Tamil, Punjabi, Kannada, French, and English. Incidentally, 48% of the participants were aged between twenty-two and thirty years of age. 20% of the participants had more than five years of experience working in the IT industry in web development. This varied set of participants helped estimate a wide variety of usage and provided important critical feedback.

7.2 Pre-survey Walkthrough

Each participant was given a demo of the system. They were provided with material to read to understand the scope of this project and to inform them about the process of the survey itself. Participants were given half an hour each to get familiar with the system and understand its working. They were asked to provide their demographics information and try out different types of web page generation using in the language of their choice, out of English, Hindi, and French.

7.3 The Survey

For the survey, a simple web page is shown in Fig. 4. This webpage included various HTML elements such as a navigation bar, different HTML heading levels, customized text input, a table where the freedom to configure the rows and columns was given to the participants. Multiple input fields and a button were also included in the webpage shared with the participants. Once the walkthrough was over, the participants were asked to develop this webpage using voice commands alone from scratch. At this stage, no help was provided to any participant and no time limit was set either. The aim was to test whether or not the participants can use the platform to develop web pages with little training. This also helped us determine the ease of use of the platform and give a proof of concept for the same.

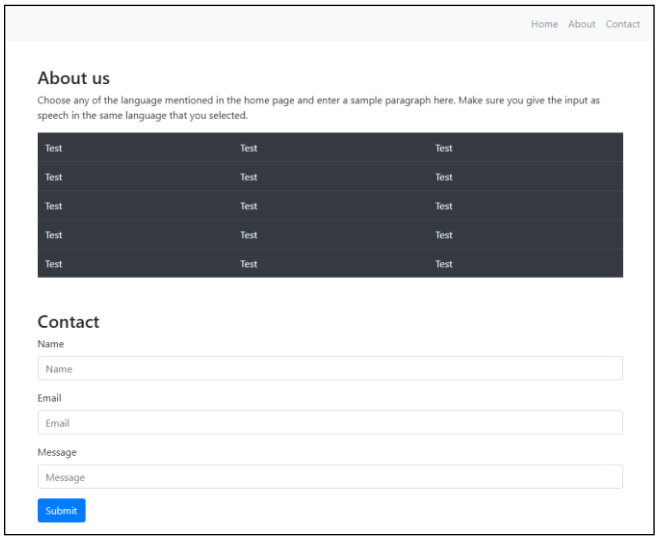

Fig. 4. Task Given to Participants

7.4 Observed Trends and Metrics

Once completed with the given task, the participants were provided a post-survey form to fill up. Quantitative questions about the functioning of the system were taken up in this survey. Even as the majority of participants were native Hindi speakers, 64% of people interacted with the system in English. 32% interacted in Hindi and only one person used French input. In a validation of this work, all 25 participants were successfully able to create the given web page, albeit in different time durations. While 56% of people took between two to five minutes to develop the webpage, 8% or 2 out of 25 were able to do so in less than two minutes. In sharp contrast, two people also took more than 10 minutes each to complete the same web page. This can be visualised in Figure 5.

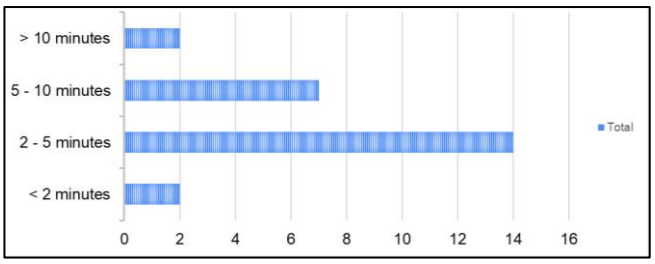

Fig. 5. Number of Participants Completing the Task in Time Interval

Of the seventeen participants who have some sort of technical experience in web development, fifteen agreed that this system helped them generate web pages faster than they could have done themselves had they chosen to code it manually. Figure 6 presents this data visually. Twenty-three participants rated the work as easy or very easy to use, and the same number of participants also said the same about the ease of use for generation of front end and back end. It must be noted that all but one, believed that this work did not bring any practical application to the table. Only two participants said that the HTML and SQL output files were not in the desired format.

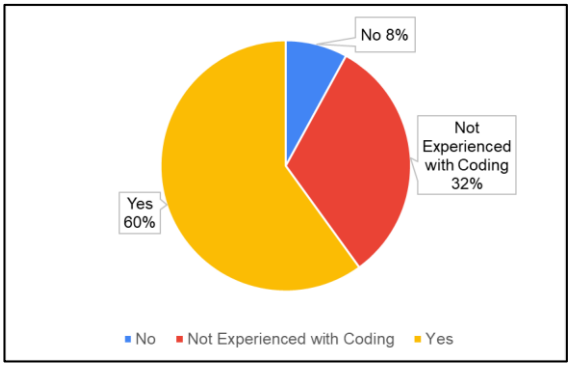

Fig. 6. Participants on Whether Development Was Faster Than Traditional Coding

Data collected and analysed as above provides us insights into this work. Additional discussions with the participants and their textual response in feedback offer many future scopes and possible enhancements that need to be researched before being implemented. The participants' desire to do more brought the system to the brink of complete utilisation. Professionals who were part of the survey wanted other HTML media elements, including images and embeds to be added as well. Editing existing elements is currently only possible by downloading and editing the code manually. An improvement in the system to enable users to edit the web page purely through voice can be added in the future. However, given that this work is aimed at bringing proof of concept, we believe that this was accomplished.

8 Conclusion

We were able to overcome many limitations of previous works like lack of responsive and dynamic output, and the inability to handle multilingual inputs. Now, the user can generate multilingual dynamic web pages. The responsiveness associated with the output provides great flexibility and real-world application of the generated web pages. Developed with a cloud-

first approach, this flexibility also extends to the user, who can now potentially make use of such a system without installing anything on their device, free of cost. Users can export the generated web page related files and host them.

Professionals will greatly benefit from functionality such as support for database queries and boilerplate generation. They will also be able to generate web pages, which consist of multiple languages, through voice rather than by using language-specific input options. This will fast-track their development process and enable the upskilling of professionals who are currently stuck working on meagre tasks.

With the help of the survey, this paper also illustrates the real-world use of dynamic generation of web pages. In addition, it showcased that most developers were able to save time using this system when compared to traditional methods. Non-professionals were also able to generate simple web pages easily and found it intuitive.

9 Acknowledgement

The authors would like to thank the School of Computing at SRM Institute of Science and Technology, Kattankulathur for motivating the students and faculty to work together and achieve a shared learning and enlightening experience.

The authors also acknowledge the contributions made by the participants of the survey. They have helped us quantitatively analyse the results of this work.

This work would not have been possible without the past research contributions made by various other authors in the fields of web development, cloud computing, natural language processing, and AI.

References

[1] Chadha, H., Mhatre, S., Ganatra, U. and Pathak, S., 2018, August. HTML Voice. In 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA) (pp. 1-4). IEEE.

[2] Bajwa, I.S., Aslam, W. and Hyder, S.I., 2006. Speech Language Engineering System for Automatic Generation of Web based User Forms. In Proceedings of the International Conference on Man-Machine Systems (ICOMMS 2006).

[3] Bajwa, I.S., Siddique, I. and Choudhary, M.A., 2006. Automatic web layout generation using Natural Language Processing Techniques. In 2nd International Conference from Scientific Computing to Computational Engineering.

[4] Modak, S., Vikmani, S., Shah, S. and Kurup, L., 2016, August. Voice driven dynamic generation of web pages. In 2016 International Conference on Computing Communication Control and automation (ICCUBEA) (pp. 1-4). IEEE.

[5] Paudyal, B., Creed, C., Frutos-Pascual, M. and Williams, I., 2020, July. Voiceye: A Multimodal Inclusive Development Environment. In Proceedings of the 2020 ACM Designing Interactive Systems Conference (pp. 21-33).

[6] Ghosh, P.K., Dey, S. and Sengupta, S., 2014. Automatic sql query formation from natural language query. International Journal of Computer Applications, 975, p.8887.

[7] Begel, A., 2005, February. Programming by voice: A domain-specific application of speech recognition. In AVIOS speech technology symposium–SpeechTek West.

[8] Maurer, M.E., Hausen, D., De Luca, A. and Hussmann, H., 2010, October. Mobile or desktop websites? Website usage on multitouch devices. In Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries (pp. 739-742).

[9] Madhurima, V., 2011. Madhulika "Windows Azure Platform: an Era for Cloud Computing". International Journal of Computer Science and Information Technologies, 2(2), pp.621-623.

[10] Verma, A., Malla, D., Choudhary, A.K. and Arora, V., 2019, February. A detailed study of azure platform & its cognitive services. In 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon) (pp. 129-134). IEEE

[11] Mahmud, T., Hasan, K.A., Ahmed, M. and Chak, T.H.C.,2015, December. A rule-based approach for NLP based query processing. In 2015 2nd International Conference on Electrical Information and Communication Technologies (EICT) (pp. 78-82). IEEE.

[12] Venayagamoorthy, G.K., Moonasar, V. and Sandrasegaran, K., 1998, September. Voice recognition using neural networks. In Proceedings of the 1998 South African Symposium on Communications and Signal Processing-COMSIG'98 (Cat. No. 98EX214) (pp. 29-32). IEEE

***