React Js Source Code Generator

[1]Y.Karthik Reddy [2]Sunandha
[1]CSE, NEC, Gudur
2Assistant Professor, CSE Department, NEC, Gudur

Abstract: *The paper describes the design, construction and working of React JS Source Code Generator. The React JS Source Code Generator is a powerful tool designed to streamline the documentation process for React developers. This innovative solution automates the generation of source code snippets, enabling developers to effortlessly create clear and concise documentation for their projects. By simply inputting relevant information such as component names, props, and functions, the generator produces ready-to-use code snippets tailored to React's syntax and best practices. This not only saves valuable time but also ensures consistency and accuracy across documentation, empowering developers to focus more on building exceptional user experiences. With its intuitive interface and customizable options, the React JS Source Code Generator revolutionizes the way developers document their React applications, promoting efficiency and collaboration in software development workflows. To provide detailed and helpful assistance for understanding easy navigation of each feature involved in React.JS Source Code Generator.*

Keywords: *React JS, source code generator, streamline documentation process, automate*

I. INTRODUCTION

In the dynamic world of web development, React.js stands out as a frontrunner, empowering developers to build interactive and responsive user interfaces with ease. However, despite its popularity and widespread adoption, developers often grapple with the challenge of efficiently documenting their React projects. Enter the React JS Source Code Generator, a revolutionary tool poised to transform the documentation process for React developers worldwide. In this comprehensive guide, we delve into the intricacies of this innovative solution, exploring its features, functionalities, and the myriad benefits it offers to the development community.

Introducing the React JS Source Code Generator, a transformative tool poised to revolutionize the documentation process for React developers. In the dynamic landscape of web development, React.js stands as a cornerstone, empowering developers to craft immersive and responsive user interfaces. However, amidst the complexity of React projects, documenting code effectively often proves challenging. Recognizing this need, the React JS Source Code Generator emerges as a beacon of efficiency, offering a streamlined approach to generating comprehensive documentation. With its intuitive interface and dynamic capabilities, this tool automates the generation of source code snippets for React components, props, and functions. By simplifying the documentation workflow, it empowers developers to create clear, consistent, and concise documentation, fostering collaboration, accelerating onboarding, and promoting code reusability.

The React JS Source Code Generator simplifies documentation for React developers, automating code snippet generation. Users input component details like names, props, and functions, and the tool produces React-syntax snippets. This streamlines documentation, saving time and ensuring consistency. With customization options and an intuitive interface, the generator promotes efficiency and collaboration in React app development. Detailed assistance facilitates easy navigation of features, enhancing user experience. Overall, the React JS Source Code Generator empowers developers to focus on creating exceptional user experiences while effortlessly documenting their React projects.

II. METHODOLOGY

1. Define Requirements:

Gather detailed requirements from the user regarding the desired output format, including file type (e.g., text file, CSV, JS), structure, and any specific data to include and all the data that is given as input in excel format only.

Determine the destination directory where the generated file will be saved.

2. Develop a User Interface (UI):

Design a user-friendly interface (GUI screen) for users to input parameters such as file name, format, and content to be included in the output file and it contains dropdown menu that select the usecase that we need.

Include validation checks to ensure the entered parameters are valid.

3. Generate Output:

Utilize appropriate programming languages or tools to generate the desired output according to the user's specifications.

Incorporate logic to process and format the data as required before writing it to the destination file.

4. Documentation:

Document the methodology, including the process of generating output files, supported formats, and any dependencies or libraries used and different types of validations while taking input format.

Provide clear instructions for users on how to utilize the system to generate and save output files in the required format.

5. File Handling:

Implement file handling mechanisms to create, write, and save the generated output file to the designated destination directory.

Ensure proper error handling to address any potential issues during file creation or writing process.

III. IMPLEMENTATION

1. CG_F001- GUI Screen:

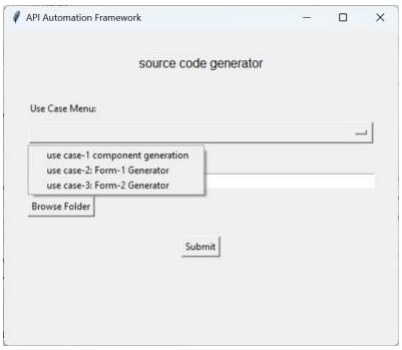
GUI stands for Graphical User Interface which is an interface between user and components like buttons, icons etc and it consists of browse button, reference folder path in text box and submit button.

As a user, I want to have a GUI screen that allows me to easily process a folder and an Excel file. This screen should provide a straightforward and intuitive interface where I can select the folder and the Excel file, initiate the processing, and view the results in a clear and organized manner. The goal is to streamline my workflow and make the handling of these files more efficient and user-friendly. With this GUI, I aim to save time and reduce the complexity involved in managing and processing data from multiple sources.

In this scenario, the acceptance criteria require that when the parent code is provided and the run button is clicked, a GUI screen should immediately pop up. This GUI screen should be user-friendly and intuitive, facilitating the subsequent tasks the user needs to perform. The process should be seamless, ensuring that upon initiation by clicking the run button, the transition to the GUI screen is smooth and immediate. This functionality is crucial to provide an efficient user experience, enabling the user to proceed with their tasks without any delays or technical issues.

2. CG_F002- Option menu for use cases:

An Option menu is a feature that is used to display the list of use cases from the available drop down menu. As a user, I want to have an option menu integrated into the system so that I can easily select the desired use case. This menu should present a list of available options in a clear and organized manner, allowing me to quickly choose the specific task or function I need to perform. By providing a straightforward way to navigate between different use cases, the option menu will enhance my overall experience, making it more efficient and user-friendly. This feature is essential for streamlining my workflow and ensuring that I can access the necessary functions with minimal effort. In this scenario, the acceptance criteria specify that when the GUI screen is active and the option menu is clicked, the use case options should be displayed promptly. The option menu must effectively showcase a list of available use cases in a clear and accessible manner.

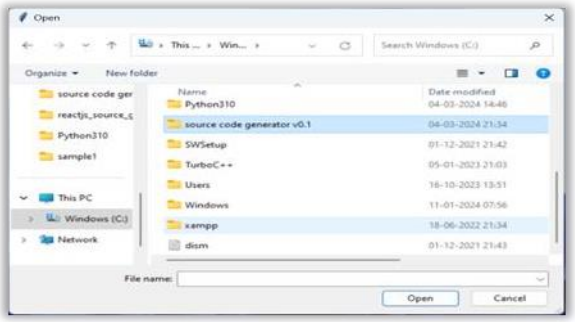


3. CG_F003- Browse and Select Folder:

- This browse button is to browse a folder in the local system
- That selected folder should contain an excel file

As a user, I want to have the ability to browse a folder from my local file system and display the folder path in a "Source" name entry label. This functionality will enable me to select and process the necessary folder and associated Excel file efficiently. By integrating a folder browsing feature with a clear and visible path display, I can ensure that I am working with the correct files. This will streamline my workflow, reduce errors, and make the task of managing and processing files more user-friendly and straightforward.

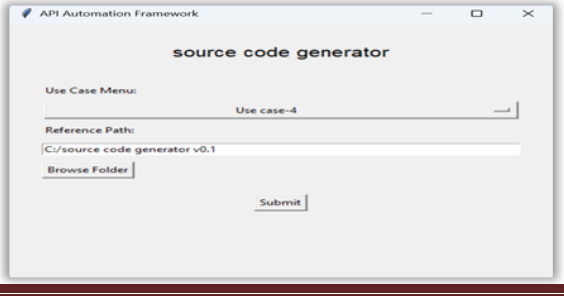
In this scenario, the acceptance criteria outline the requirement for a GUI screen that facilitates browsing the local file system through a file dialog window. The user should be able to initiate this action seamlessly. When the user wishes to browse a folder from their local file system, the interface must respond promptly by displaying a file dialog window. This window will allow the user to navigate through their directories and select the desired folder. The functionality ensures that the user can easily and efficiently locate and choose the necessary folder, which is critical for subsequent processing tasks.



- Display Selected Folder path:

In this scenario, the acceptance criteria focus on the requirement for the GUI screen to display the selected folder path once a folder has been chosen by the user. When the user provides a folder, the system should promptly and accurately display the path of the selected folder on the GUI screen. This functionality is crucial as it provides immediate visual confirmation to the user that the correct folder has been selected for subsequent processing.

The GUI screen must be designed to clearly and prominently show the selected folder path. This can be achieved by having a dedicated "Source" name entry label or a similar field where the path will be displayed. The field should be easily noticeable and readable, ensuring that users can quickly verify the path without any confusion. This feature enhances the user experience by providing transparency and confidence that the correct data source is being used.



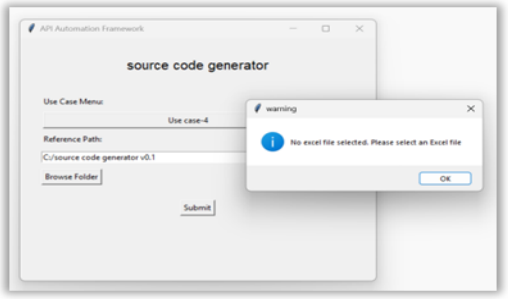
3.CG_F004- Validation for Submit Button:

Case-1:

Submit button is a feature, which helps the user for validating the input files (folder, excel file and values in the input excel, etc. In this scenario, the acceptance criteria outline the need for the GUI to handle situations where a user attempts to submit a selected folder without having chosen an Excel file. When the user provides a folder but does not select an Excel file, and then clicks on the submit button, the system should respond by displaying a pop-up message box. This message box should clearly inform the user that no Excel file has been selected, guiding them to rectify the oversight before proceeding.

The primary goal of this feature is to ensure that users are aware of the missing Excel file, which is essential for the processing task. The pop-up message should be designed to catch the user's attention immediately, preventing any confusion or errors that might arise from attempting to process incomplete data. The message within the pop-up should be concise yet informative, clearly stating that an Excel file needs to be selected to proceed with the operation.

The implementation of this feature involves several key steps. First, the system must include a validation check that triggers when the user clicks the submit button. This check should assess whether an Excel file has been selected alongside the folder. If no Excel file is detected, the system should halt the submission process and generate the pop-up message.



Case-2:

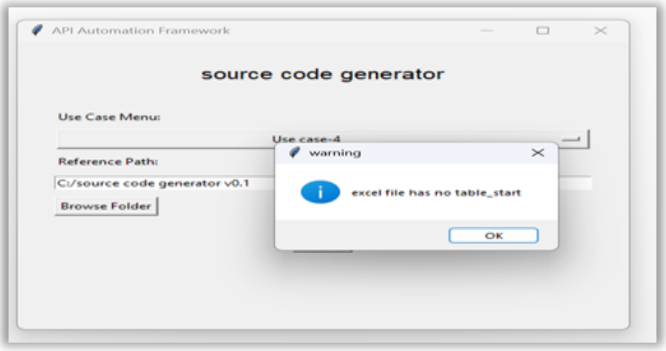
In this scenario, the acceptance criteria delineate the necessity for the GUI to handle instances where a user attempts to submit an Excel file lacking the "#table_start" marker. When a user selects an Excel file but the file does not contain the "#table_start" marker, and subsequently clicks on the submit button, the system should respond by displaying a pop-up message box. This message box should explicitly inform the user that the selected Excel file does not contain the required "#table_start" marker, prompting them to review and amend the file before proceeding.

The primary objective of this feature is to ensure that users are made aware of any discrepancies in the structure of the Excel file, particularly the absence of the "#table_start" marker, which is crucial for identifying the start of a data table. The pop-up message should be designed to immediately capture the user's attention, preventing any confusion or errors that might arise from attempting to process data without this essential marker. The message within the pop-up should be succinct yet informative, clearly indicating the issue and guiding the user on how to rectify it.

Implementing this feature involves several key steps. Firstly, the system must include a validation check that verifies the presence of the "#table_start" marker within the selected Excel file. If the marker is missing, the system should halt the submission process and generate the pop-up message.The design of the pop-up message box is critical for facilitating user understanding and interaction. It should be prominently displayed on the GUI, with a clear and unequivocal message. For instance, the message might read, "Excel file does not

contain #table_start marker. Please ensure the file structure includes the necessary marker to proceed." This direct instruction aids users in comprehending the issue and taking corrective action.

The system should also anticipate and handle edge cases, such as users repeatedly attempting to submit an Excel file without the "#table_start" marker. The pop-up message should consistently appear in these situations, reinforcing the necessity of ensuring the file structure includes the required marker until the condition is met. This persistent feedback mechanism is vital for upholding the integrity of the data processing workflow.



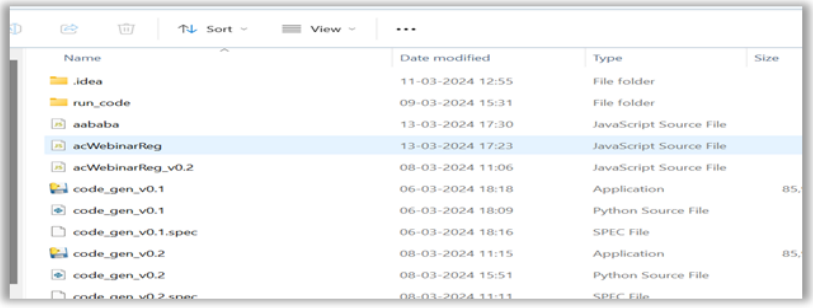
5. CG_F005- Generate Output in Required Format :

In this scenario, the acceptance criteria specify the requirement for the GUI to facilitate the creation of a new destination file with a given name when the user clicks on the submit button. When a user provides both a folder and an Excel file and subsequently clicks on the submit button, the system should respond by generating a new destination file with the specified name within the provided folder.

The primary objective of this feature is to enable users to easily create a new destination file where processed data can be stored. The system should seamlessly integrate this functionality into the submission process, automating the creation of the destination file to streamline the user's workflow. Upon clicking the submit button, the system should immediately initiate the creation process, ensuring that the new file is generated promptly.

Implementing this feature involves several key steps. Firstly, the system must capture the provided folder and Excel file from the user's input. Then, upon receiving the submission command, the system should proceed to create a new destination file within the specified folder. This file should be generated with the name provided by the user, ensuring that the file is easily identifiable and accessible for storing processed data.

The design of the destination file creation process should be intuitive and seamless for users. Upon successful creation of the new file, the system should provide feedback to the user, confirming that the destination file has been generated and is ready for use. This feedback can be in the form of a pop-up message or a status update on the GUI, informing the user that the operation was successful.



6. CG_F006- Usecase-1: Component Generation:

In this feature, we need to generate the a component in a page by taking reference to given input file and stores that new data in the given destination file with required format.

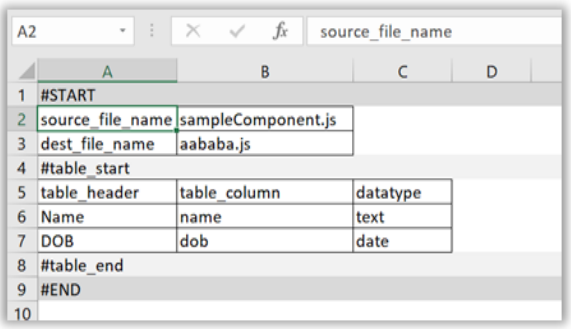
Upon clicking the "run" button, the system should generate a React component within a new file, as specified. This action ensures the seamless integration of React components into the project structure, facilitating the development of dynamic user interfaces. By automatically creating a new file containing the React component, users can efficiently manage their codebase and leverage React's component-based architecture for building interactive and responsive applications. This feature streamlines the process of incorporating React components into projects, enhancing development productivity and enabling users to create engaging user experiences with ease.

The generation of React components via a simple user action—clicking the "run" button—provides several benefits that enhance development productivity. Firstly, it allows developers to quickly create new components without having to write the boilerplate code manually. This is particularly useful in large projects where maintaining consistency and adhering to coding standards are paramount. The automated generation ensures that all components follow a uniform structure, making the codebase easier to understand and maintain.

Secondly, this feature supports the principle of separation of concerns by keeping the component logic separate from other parts of the application. Each generated file contains only the code relevant to a specific component, which promotes modularity and reusability. Developers can easily locate and update individual components without affecting the rest of the application, leading to a more organized and maintainable codebase.

Moreover, this automation enhances the development workflow by allowing developers to focus on higher-level tasks such as designing the user interface and implementing business logic. With the repetitive task of writing boilerplate code handled by the system, developers can dedicate more time to creating engaging and responsive user experiences. This shift in focus can lead to more innovative and effective application design, as developers have more bandwidth to experiment with new ideas and optimize the user interface.

The system-generated React components also adhere to best practices and industry standards, ensuring that the resulting code is not only functional but also performant and scalable. This is particularly important in today's web development landscape, where applications need to handle a growing number of users and data efficiently. By providing a solid foundation of well-structured components, the system helps developers build applications that can scale seamlessly as user demands increase.the automatic generation of React components upon clicking the "run" button is a powerful feature that streamlines the integration of components into the project structure. This automation enhances development productivity by reducing manual coding tasks, ensuring consistency, and promoting modularity.



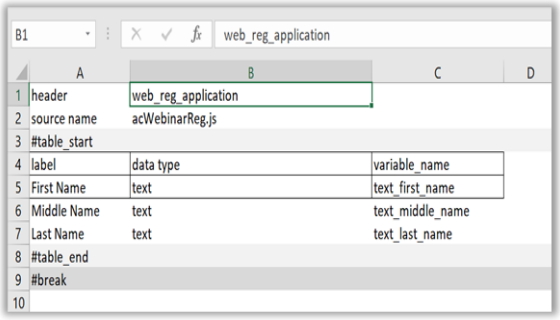
7. CG_F007- Usecase-2: Form-1,2 Generation:

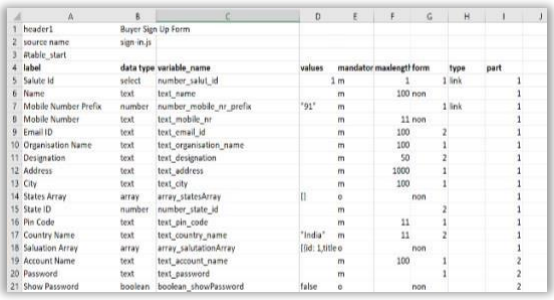
When the user triggers the execution of the parent code by clicking the "run" button, the system should respond by generating React-based form code within a new file. This feature streamlines the development process by automatically creating a dedicated file containing the React form code. By generating React-based form code upon user action, users can seamlessly incorporate form functionalities into their projects, enhancing interactivity and user engagement. This capability simplifies the integration of forms into React applications, empowering users to efficiently build dynamic and user-friendly interfaces for collecting and processing user input.

The primary advantage of this feature is its ability to generate React-based form code with a simple user action. By automating the creation of form code, the system enhances both interactivity and user engagement in applications. Forms are a critical component of most applications, as they enable the collection and processing of user input. This automation means that developers no longer need to manually code forms from scratch, which can be a time-consuming and error-prone process. Instead, they can rely on the system to produce robust, functional form code that can be easily integrated into their projects.

This capability is particularly valuable in the context of modern web development, where the ability to quickly and efficiently create dynamic, user-friendly interfaces is crucial. By simplifying the incorporation of forms into React applications, the system empowers users to focus on developing more complex functionalities and enhancing the overall user experience. This shift in focus from manual coding to higher-level development tasks can lead to more innovative and effective applications.

The automation of form code generation also contributes to a more streamlined and productive development workflow. Developers can save considerable time and effort, allowing them to allocate resources to other critical areas of their projects. This efficiency can lead to faster development cycles and quicker time-to-market for applications. Additionally, by reducing the likelihood of errors associated with manual coding, the system helps ensure that the forms are both functional and reliable.





When the user clicks the "run" button to execute the parent code, the system will automatically generate a new file containing React-based form code. This feature significantly streamlines the development process by creating a dedicated file for the React form, making it easy to integrate form functionalities into projects. By automating the generation of React form code, users can effortlessly incorporate interactive forms into their applications, enhancing user engagement and interactivity. This functionality simplifies the process of adding forms to React applications, allowing users to efficiently build dynamic, user-friendly interfaces for collecting and processing user input. Ultimately, this capability empowers developers to create sophisticated forms quickly, improving development productivity and the overall user experience.

Moreover, the generated React-based form code is likely to follow best practices and standards, ensuring that the resulting forms are not only functional but also maintainable and scalable. This adherence to best practices can be particularly beneficial for teams working on large-scale projects, where consistency and quality are paramount.

IV. RESULT

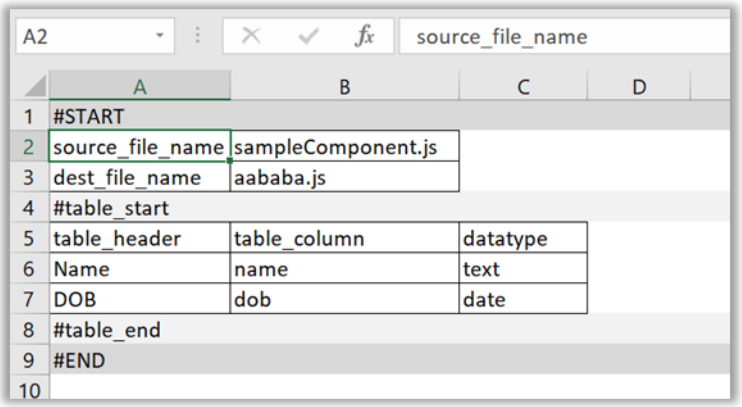
When we execute our Python script, a GUI screen appears with the following features:

- Dropdown Menu: Contains a list of use cases.
- Buttons: Includes 'Submit' and 'Browse folder' buttons.

To initiate the process, first select "Use Case-1: Component Generation" from the dropdown menu. Next, use the provided interface to navigate and select the desired folder path. Then, choose the input Excel file that contains the necessary data for component generation. Finally, click the 'Submit' button on the GUI screen to complete the process.

Component Generation Process

Input Excel Structure :- The input Excel file should consist of data structured as follows:

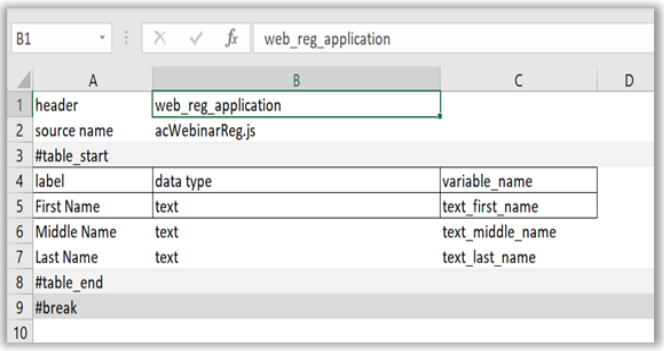


Code Generation :

Upon clicking the 'Submit' button, the Python script processes the selected Excel file and generates the code with required format with user required name from excel.

Page (AC Webinar Form) Generation Process

Input Excel Structure:- The input Excel file should consist of data structured as follows:

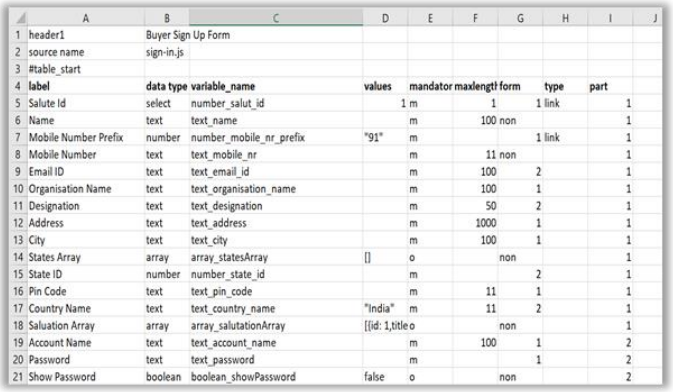


Code Generation

Upon clicking the 'Submit' button, the Python script processes the selected Excel file and generates the code with required format (here it is js)with user required name from excel.

Dynamic Form Generation Process

Input Excel Structure :The input Excel file should consist of data structured as follows:



	A	B	C	D	E	F	G	H	I	J
1	header1	Buyer Sign Up Form								
2	source name	sign-in.js								
3	#table_start									
4	**label**	**data type**	**variable_name**	**values**	**mandator**	**maxlength**	**form**	**type**	**part**	
5	Salute Id	select	number_salut_id		1 m	1		1 link	1	
6	Name	text	text_name		m	100	non		1	
7	Mobile Number Prefix	number	number_mobile_nr_prefix	"91"	m			1 link	1	
8	Mobile Number	text	text_mobile_nr		m	11	non		1	
9	Email ID	text	text_email_id		m	100		2	1	
10	Organisation Name	text	text_organisation_name		m	100		1	1	
11	Designation	text	text_designation		m	50		2	1	
12	Address	text	text_address		m	1000		1	1	
13	City	text	text_city		m	100		1	1	
14	States Array	array	array_statesArray	[]	o		non		1	
15	State ID	number	number_state_id		m			2	1	
16	Pin Code	text	text_pin_code		m	11		1	1	
17	Country Name	text	text_country_name	"India"	m	11		2	1	
18	Saluation Array	array	array_salutationArray	[{id: 1,title	o		non		1	
19	Account Name	text	text_account_name		m	100		1	2	
20	Password	text	text_password		m			1	2	
21	Show Password	boolean	boolean_showPassword	false	o		non		2	

Code Generation

Upon clicking the 'Submit' button, the Python script processes the selected Excel file and generates the code with required format (here it is js)with user required name from excel.

V. CONCLUSION

In conclusion, the React JS Source Code Generator emerges as a game-changer in the realm of React development documentation. Through its intuitive interface, customizable templates, and dynamic code generation capabilities, this tool streamlines the process of creating comprehensive documentation for React projects. By automating the generation of source code snippets for components, props, and functions, it saves developers valuable time and effort, while ensuring clarity, consistency, and accuracy in documentation. Moreover, the React JS Source Code Generator promotes collaboration, accelerates onboarding, and enhances code reusability, ultimately empowering developers to build exceptional user experiences with React. As we witness the ever-evolving landscape of web development, this innovative tool serves as a beacon of efficiency, driving progress and excellence in React development workflows. With its potential to simplify and elevate the documentation experience, the React JS Source Code Generator stands as a testament to the power of innovation in advancing the art and science of software development.

VI. REFERENCES

[1] Shari Lawrence Pfleeger. Software Engineering: Theory and Practice[M]. 2nd edition. Prentice Hall, Inc. 2001

[2] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Design Patterns: Elements of Reusable Object- Oriented Software[M]. Addison Wesley, 1995.

[3] Krzysztof Czarnecki, Hlrich W.Eisenecker. Generative Programming Methods, Tools, and Applications[M]. Addison Wesley, 2000.

[4] Stewart Baird. Teach Yourself Extreme Programming in 24 Hours[M]. SAMS, 2003.

[5] Andrew Haigh. Obejct-Oriented Analysis & Design[M]. McGraw-Hill, 2001.

[6] John Swanco. COM Programming by Example[M]. R&D Books, 2000.

[7] PradeepTapadiya. COM+ Programming[M].Prentice Hall, 2002

[8] Bruce Eckel. Thinking in C++[M]. Prentice Hall, 1995.

[9] React Native, https://facebook.github.io/react-native/, acedido em 18 de fevereiro de 2020

[10] The Best JS Frameworks for Front End, https://rubygarage.org/blog/best-javascript-frameworks-for-front-end, acedido em 18 de fevereiro de 2020

[11] React.js and Front-End Development, https://dzone.com/articles/whychoose-react-for-front-end-development, acedido em 18 de fevereiro de 2020

[12] humansoft, https://www.humansoft.pt/about-us.html, acedido em 18 de fevereiro de 2020

[13] humantrain, https://www.humansoft.pt/formacao/humantrain.html, acedido em 18 de fevereiro de 2020

[14] humanportal, https://www.humansoft.pt/formacao/humanportal.html, acedido em 18 de fevereiro de 2020