**Technical Solution Approach**

Contents

[1 Introduction 2](#_Toc127885736)

[1.1 About this document 2](#_Toc127885737)

[1.1.1 Purpose & Scope of the document 2](#_Toc127885738)

[2 Component Design 2](#_Toc127885739)

[2.1 Component Design Diagram 2](#_Toc127885740)

[2.1.1 Overall Workflow ……………………………………………………………………………………………………...2](#_Toc127885741)

2.1.2 Class Diagram…………………………………………………………………………………………………………….3

2.13 Low level diagram………………………………………………………………………………………………………4

3 Technology & Frameworks to be used……………………………………………………………………………………….6

4 Solution Approach……………………………………………………………………………………………………………………..6

5 Test Case Scenario…………………………………………………………………………………………………………………….6

# Introduction

## About this document

### Purpose & Scope of the document

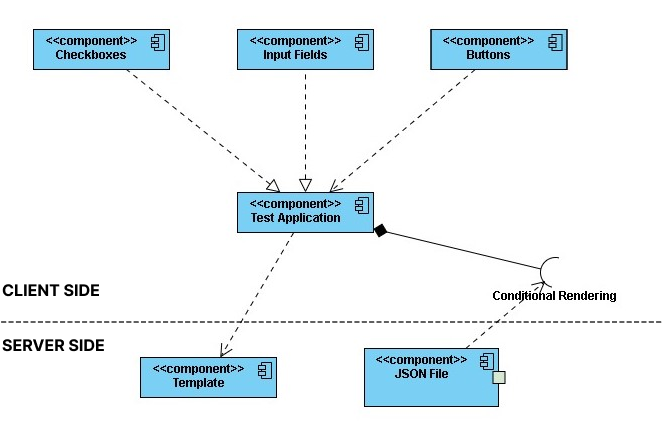
The purpose of this document is to provide a blueprint for developers who want to create dynamic UI components in Vue.js to achive highly interactive and responsive applications. These components can be easily customized and reused, providing flexibility and consistency in the UI design. This helps to streamline development and enhance user experience. The document also outlines the key features of Vue.js is to make it an ideal framework for developing dynamic UI components, such as its reactivity system, component-based architecture, and the virtual DOM .

We also use Vitest fast unit test which was powered by Vite of VueJS to reuse the configs,transformers and plugins consistent across the app And during tests. Process and we also use. Overall, the technical solution approach document serves as a guide for developers to build a robust and efficient file upload component

# Component Design

## Component Design Diagram

### Overall Workflow

****

### Class Diagram

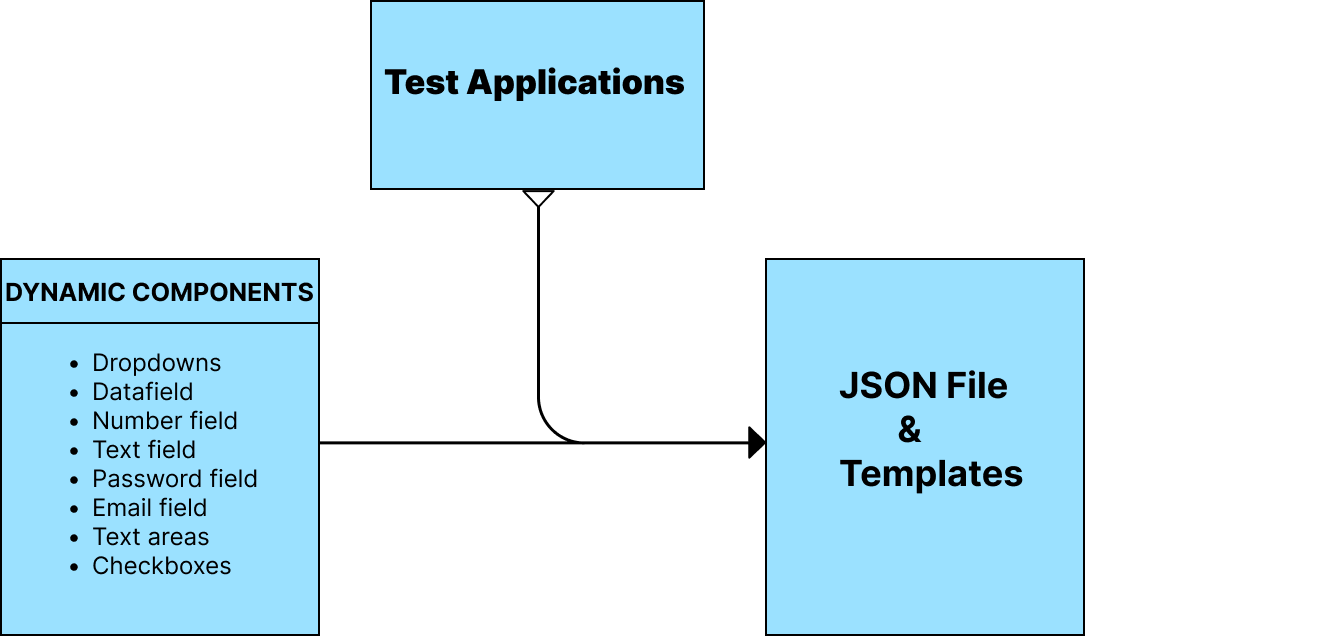


Fig 1.1

SERVER SIDE

* Conditional Rendering :A Vue 3 Directive that modifies the display style of a HTML element based on the variable passed to it.
* Template :A template provided via the  template option will be compiled on-the-fly at runtime.An alternative for the conditional rendering directive.
* Json file: For instance ,If the Content of the rendered web page is needed to modified the JSON file can be used to change the HTML,styles and objects to be rendered.

CLIENT SIDE

* TestApp.vue: The Dynamic components are to be mounted in the the test application to validate and verify the components .the test application may be any site that uses basic components that are dynamically created.It is used to check whether the components are rendered correctly in the web page.
* Dynamic Components:

1. Input Fields : The input fields are most commonly used in the login pages and user support pages.So the input field component is designed dynamically in order to use the in multiple pages.
2. Buttons : The buttons are used in web sites for multiple purposes as to redirection,toggle pages ,form submission…,so the button will be created as a template to include styles and links through the virtual DOM .
3. Page Content : The Web page must contains header ,body & footer as these components can be rendered through the Json file in order to change the values of the json object suitable for the different sites.
4. Searches :Generation of the Advanced search filters in the designated area with the required filter options.where the required fields are stored as a v-if conditional rendering statement for using only required filters.

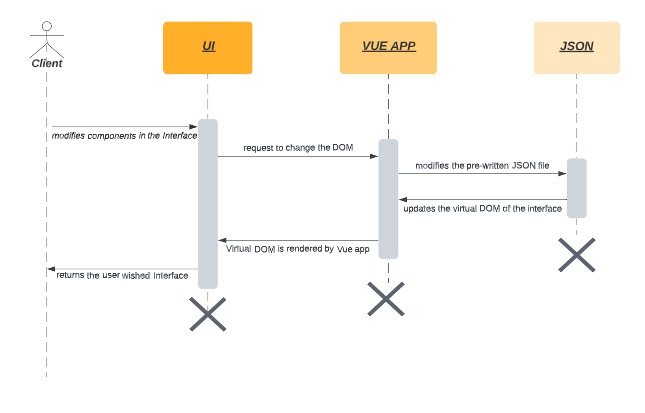
Sample Sequence diagram for dynamic UI

Fig 1.2

### Low level Design

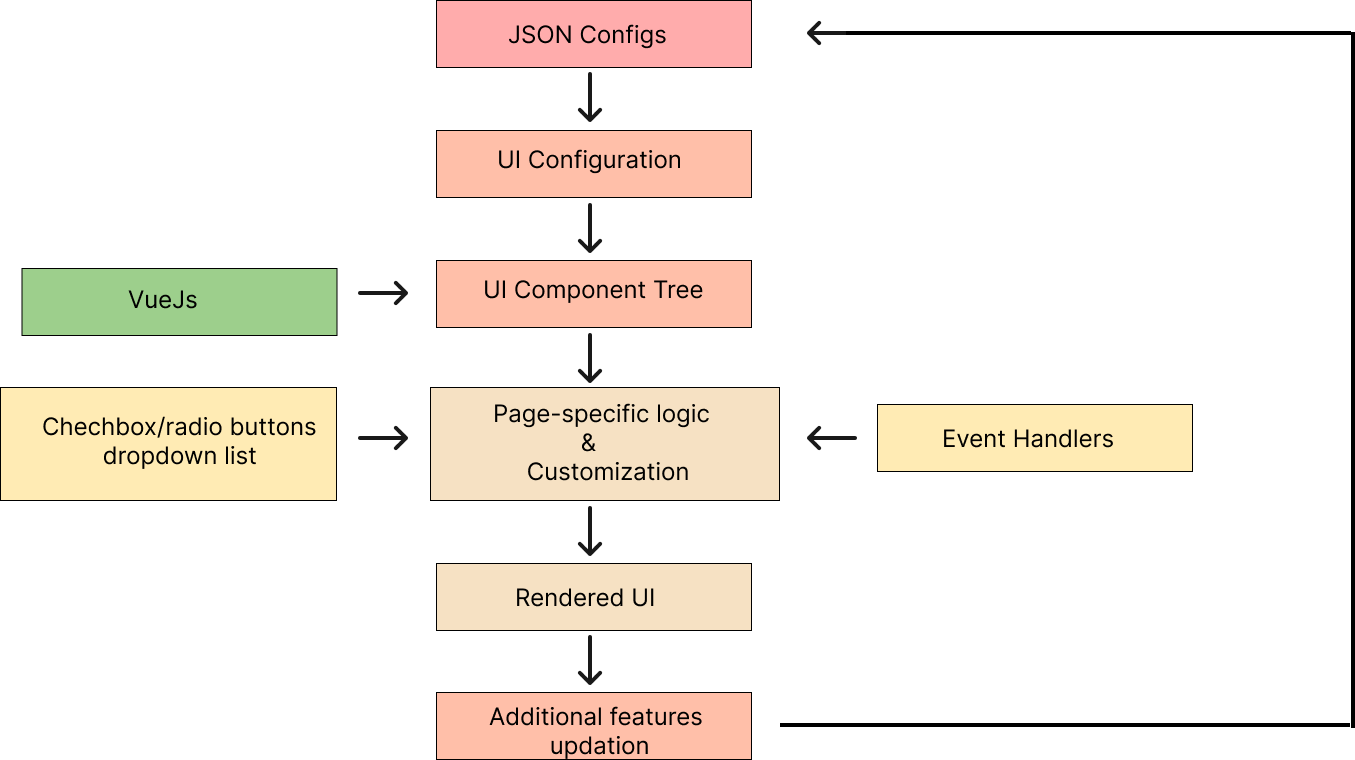


Fig 1.3

• The UI layout processing engine obtains the necessary JSON data for the dynamic UI component during page initialization.

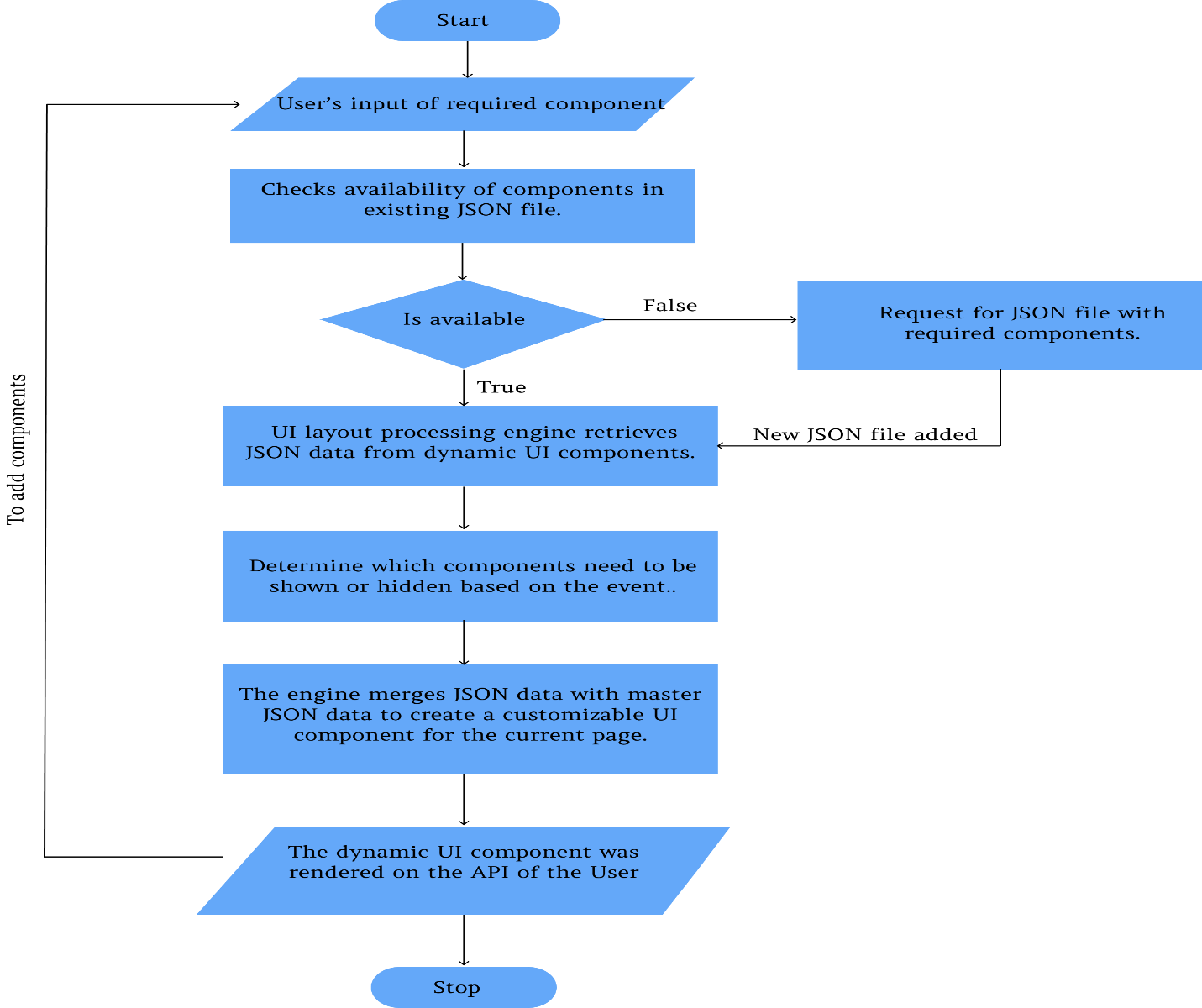
• The engine renders the component on the Interface by converting the JSON data into HTML tags.

The checkbox/radio button check, uncheck, and dropdown list change events are all monitored by the engine. The engine examines the JSON data when an event happens to determine which components should be displayed or hidden in light of the occurrence.

• To display or conceal the necessary components, the engine alters the HTML using CSS.

• The engine retrieves the JSON data for the current page together with any changes that are specific to the current page.

• To produce a unique UI component for the current page, the engine combines the master JSON data with the JSON data particular to that page.



Rendering dynamic components defined in JSON with Vue.js entails creating a Vue.js component that can render dynamic components based on the JSON object, defining the format of the JSON object, and incorporating the dynamic component into your Vue.js application.

To begin, create a Vue.js component capable of rendering dynamic components based on a JSON object. This component should be well documented, explaining the JSON object's input and output formats. Accepting a JSON object as a prop, the component should render a dynamic component based on the JSON object.

The format of the JSON object that will be used to specify the dynamic component to render must then be defined. This format should be well documented, with descriptions of the required and optional fields, as well as their formats.Finally, in your Vue.js application, you can use the dynamic component by passing the JSON object as a prop to the Vue.js component you defined in step one. The dynamic component will then be rendered by the Vue.js component based on the JSON object.

You can easily render dynamic components defined in JSON using Vue.js by following these steps and thoroughly documenting the process. This technique can be used to create reusable and dynamic components that can be easily customised based on input data.

# Technology & Frameworks to be used

* HTML & CSS
* VueJS
* Javascript
* Vuex
* Vitest

# Solution Approach

1. Based on the general designs of web pages ,Reusable web page components will be created using VueJS .

2. The designed components will be stored in a library to fetch them for various activities and converted it to JSON format.

3. The UI component will be generated and rendered by the Vue’s virtual DOM after converting the JSON file as per the user changes made on the dynamic page loading time.

4. The UI component will use the information stored in JSON format to display on the web page.

5. This component’s styles,rendering HTML tags and modified views will be processed by Vue’s two-way binding of virtual DOM during the dynamic rendering.

6. The components will be developed in VueJS and Vuex file as a self-contained module.

7. Vitest is used for fast unit test which was powered by Vite of VueJS to reuse the configs,transformers and plugins consistent across the app And during tests.

# Test Case Scenario

**Test Case 1**: Validating working of reusable input fields.

Input:Input field type(i.e text,password) to the created component.

Output:Creation of Input Field with the given type.

**Test Case 2**:Validating the Checkboxes for necessory elements to be displayed.

Input:Mark or Unmark the checkbox.

Output:Marked changes will be displayed.

**Test Case 3**: Validating working of reusable Buttons.

Input:Button name and Action to be mentioned for the created component.

Output:Creation of Button with given name.

**Test Case 4:** Validating changes for correct file type

Input:Changes in Creating new components and styles should be changed only in JSON file.

Output:After conditional rendering the changes should be viewed in testApp.vue

**Test Case 5**: We will test the features of the Counter Component:

It displays the current count. The initial value is 0 and can be set by an Input.

When the user activates the “+” button, the count increments.

When the user activates the “-” button, the count decrements.

When the user enters a number into the reset input field and activates the reset button, the count is set to the given value.

When the user changes the count, an Output emits the new count.

**Test Case 6**: Validating working of reusable password field.

Input:Field name and Action to be mentioned for the created component.

Output:Creation of password fields with given name in the label.

**Test Case 7**: Validating working of reusable search filters.

Input:A search filter template is given as input along with the required filter options needed site.

Output:Generation of the Advanced search filters in the designated area with the required filter options.

**Test Case 8**: Validating working of reusable text fields.

Input:A text field of required size and in the designated place.

Output:Creation of Text field with the field template.