**Dynamic Components – Angular 13**

**Component Specification Document**

**<Version No.>**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Prepared By / Last Updated By** | **Reviewed By** | **Approved By** |
| **Name** | Felix Titus |  |  |
| **Role** |  |  |  |
| **Signature** |  |  |  |
| **Date** |  |  |  |

**Table of Contents**

[1.0 Abstract 3](#_Toc113988054)

[1.1 Technology/Framework for Development 3](#_Toc113988055)

[2.0 Business Scenario 3](#_Toc113988056)

[Component Features 4](#_Toc113988057)

[Sample Approach 5](#_Toc113988058)

[3.0 Solution Approach 5](#_Toc113988059)

[4.0 Dependencies & Constraints 6](#_Toc113988060)

[5.0 Exception Handling 6](#_Toc113988061)

# Abstract

The dynamic component is the component that is created dynamically at the runtime. E.g to build a dynamic ad banner, using the Common Template system. New ad components are added frequently by several different teams for single application interface. This makes it impractical to use a template with a static component structure. Instead, you need a way to load a new component without a fixed reference to the component in the ad banner's template.

## Technology/Framework recommendation

This Component can be developed with java script frameworks like:

1. Angular 13

2. ReactJS

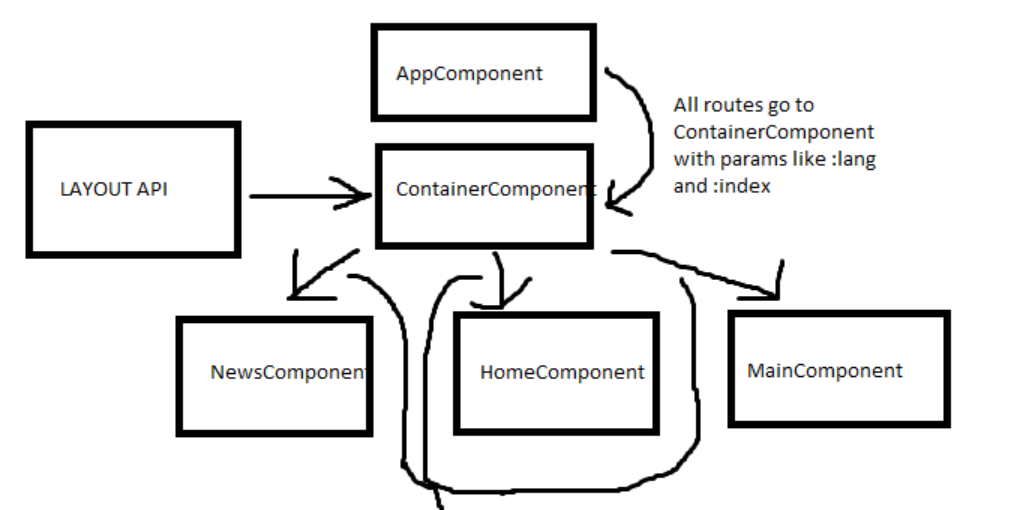
3. NodeJS

4. Vue.JS

This document outlines the example for the angular 13 but the there are similar approaches available with ReactJS as well as Node Js and Vue Js to create the dynamic loading of Ad Banner control to load multiple Advertisements/Images in the Ad Banner.

# Business Scenario

* Create Ad Banner Components to Load Dynamically
* Create service for Dynamic component loading
* Create Directive and Component to get Dynamic Component Insert Location
* Configure Dynamic Components using entry Components in module
* Create component to initiate Dynamic Loading



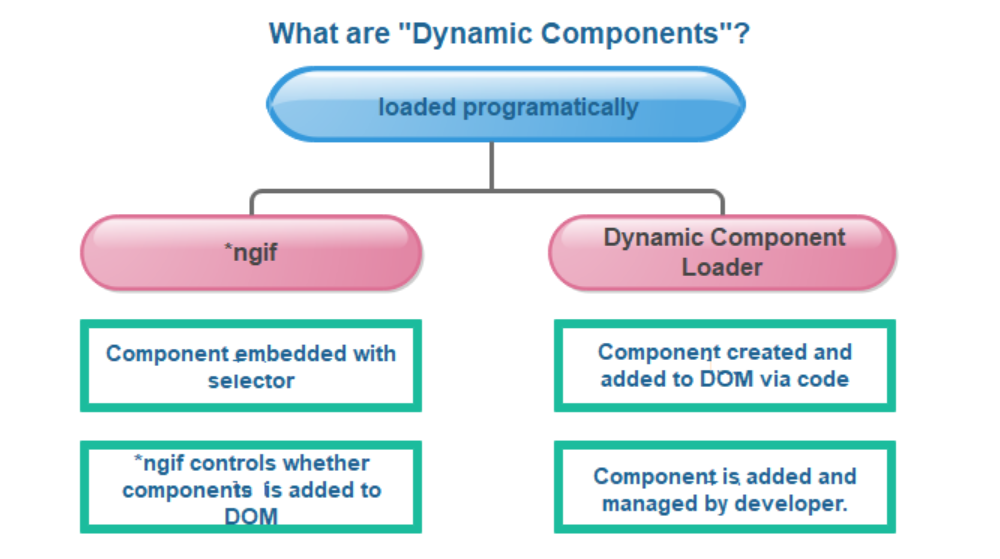
.

As per above screenshots this is sample search functionality for both basic search and advance search. In the application this will appear at multiple places as well as this may be having some addition or deletions as per the requirement into the criteria. So it will be more useful if we can create this component coming up from the standard format and should have same look and feel and once any HTML related issues fixed in 1 html then it should be fixed across all the places where this search is used.

# Component Features

We will need the following to implement dynamic component loading in our Angular app:

* Component for the Ad Banner that demand dynamic loading.
* Service to dynamically load and resolve the component.
* handling view Container Ref directive



# Solution Approach

The solution approach can be divided into several steps:

Step 1: HTML Structure

Create the HTML structure for the popup ad. This typically involves a container element for the ad content, such as a `<div>`, and appropriate styling.

Step 2: CSS Styling

Apply CSS styles to the popup ad container to control its appearance, position, and animation effects. This includes defining dimensions, positioning, background colours/images, and any desired animations.

Step 3: JavaScript Functionality

Implement a JavaScript function that handles the dynamic loading and display of the popup ad. This function should be triggered by a specific event, such as a button click or page load.

Step 4: Dynamic Loading

Within the JavaScript function, using JavaScript language to load an ad dynamically without changing the website’s domain and showing a popup ad.

Step 5: Displaying the Popup Ad

Once the content is successfully loaded, dynamically update the HTML of the popup ad container with the retrieved content.

Step 6: Showing and Hiding the Popup Ad

Implementation of buttons inn popup ad and toggling the visibility of the popup ad container accordingly.

Step 7: Testing and Refinement

Test the popup ad functionality across different browsers and devices to ensure compatibility and responsiveness.

Refine the styling, behaviour, and content of the popup ad as needed based on user feedback or design considerations.

# Dependencies & Constraints

* JavaScript Support: Dynamic loading of components for popup ads heavily relies on JavaScript. Therefore, a dependency for creating a popup ad using dynamic load components is ensuring that the target platform or browser supports JavaScript. If the platform or browser doesn't support JavaScript, the dynamic loading mechanism may not work correctly or at all, rendering the popup ad ineffective. It's important to consider the target audience and their device/browser capabilities to ensure widespread compatibility.
* Network Connectivity: Another important dependency is a reliable network connection. Dynamic loading components require fetching additional resources, such as images, stylesheets, or scripts, from remote servers. Therefore, to successfully create a popup ad using dynamic load components, a constraint is having a stable and reasonably fast network connection. If the network connection is slow, intermittent, or unreliable, it can lead to delayed or failed loading of the popup ad's components, resulting in a poor user experience or even rendering the ad unusable.

# Exception Handling

When creating a popup ad with dynamic loading components, it's important to consider exceptional handling to ensure a smooth and error-free user experience. Here's a general outline of how you can handle exceptions during the process:

* Error Detection:

Implement appropriate error detection mechanisms to identify potential issues during the popup ad creation and dynamic component loading.

Validate user input and ensure it meets the required criteria.

Check for errors in fetching or loading dynamic components.

* Graceful Error Messages:

Provide clear and user-friendly error messages when exceptions occur. These messages should inform the user about the encountered problem and suggest possible solutions.

Avoid displaying technical or cryptic error messages that might confuse or frustrate users.

* Error Logging:

Implement error logging to record exceptions that occur during the popup ad creation process.

Log relevant details such as error messages, timestamps, user actions, and any other relevant information.

Use error logs to track and troubleshoot issues efficiently.

* Graceful Degradation:

Design your popup ad in a way that gracefully degrades when dynamic components fail to load or encounter errors.

Ensure that the essential functionality and content of the popup ad remain accessible even if dynamic components are unavailable.

Consider providing fallback options or alternatives to the dynamic components, such as static content or default images.

* Testing and Monitoring:

Thoroughly test the popup ad creation and dynamic component loading processes to identify and address any potential issues.

Monitor the system and user feedback to proactively identify and resolve exceptions that occur in real-world scenarios.

Regularly review error logs and user-reported issues to improve the stability and reliability of the popup ad system.

* Continuous Improvement:

Iterate on your exceptional handling mechanisms based on the feedback, error logs, and user experience.

Implement improvements to prevent known issues from recurring and optimize the overall performance of the popup ad system.