SOLUTION APPROACH DOCUMENT

DYNAMIC LOAD COMPONENT

By

CHENNA GEETHA MADHURI	(VTU17345)
PRANEETH KASANAGOTTU	(VTU17914)
VETUKURI NIROSHA	(VTU12425)
PAILLA BINDU SRI	(VTU17363)
SHAIK ARIFULLA	(VTU18208)

Under the guidance of

Mr.S.Girirajan	(Assistant Professor)
Ms.Elakya R	(Assistant Professor)
Mrs.Nivetha B	(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN Dr. SAGUNTHALA R&D
INSTITUTE OF SCIENCE AND TECHNOLOGY

ABSTRACT

Whenever a user loads into a webpage, he will be shown multiple ads loaded at once, making the user less interested in visiting that webpage again. So, by using this dynamic load component concept the component can be loaded whenever it is required. Dynamic loading can improve the user experience by allowing navigation between different components in a single webpage. We use VUE JS, HTML, and CSS as our front end and we use API for loading multiple images. Our component loads in a specified location defined by the website developer. Our component loads the ad data dynamically. The ad banner component requests the ad service for specific ad data which fetches from the API. This ad component is not only used to load the ad data but the ad content is also based on the webpage. The ad content will vary from webpage to webpage by integrating APIs in the ad banner component to request a specific kind of ad. The ad content changes after a certain period and after reloading the web page.

KEYWORD: VUE.JS, API, Component, Directive, Dynamic Loading

LIST OF FIGURES

ABBREVIATION

HTML	HyperText Markup Language
API	Application Programming Interface
CSS	Cascading Style Sheet
NPM	Node Package Manager

TABLE OF CONTENT

CHAPTER 1
INTRODUCTION

1.1 ABOUT THIS DOCUMENT

1.1.1 PURPOSE & SCOPE OF THE DOCUMENT

- The main purpose is to dynamically load a component in a web page without a fixed reference to the component, which allows the website to be more flexible and efficient management of content.
- The scope of the solution approach includes the creation of Components, where the main component to be used is the AdBanner Component and a service component for fetching Ad content and loading it on a webpage.
- The solution approach focuses on the implementation of a modular, scalable, and maintainable system for loading components dynamically, which can be easily adapted to different websites.

1.1.2 OVERVIEW

The project involves loading advertisements dynamically on a web page using VueJS as the front-end framework. The ads are fetched from a backend API and dynamically loaded onto the web page using a service component for dynamic component loading. The ad banner component receives the ad data and dynamically loads ad components for each ad to be displayed. The position of the ad on the web page is determined by the website developer. He can use our component by using the name of the component on the developer's web page. The ad banner component also includes APIs for suggesting related ads based on the user's behavior.

CHAPTER 2

COMPONENT DESIGN

2.1 COMPONENT DESIGN DIAGRAM

In this section we describe the component design and the specific way of designing it.

2.1.1 OVERALL WORKFLOW

The component design diagram describes how the ad banner component is loaded into a website.

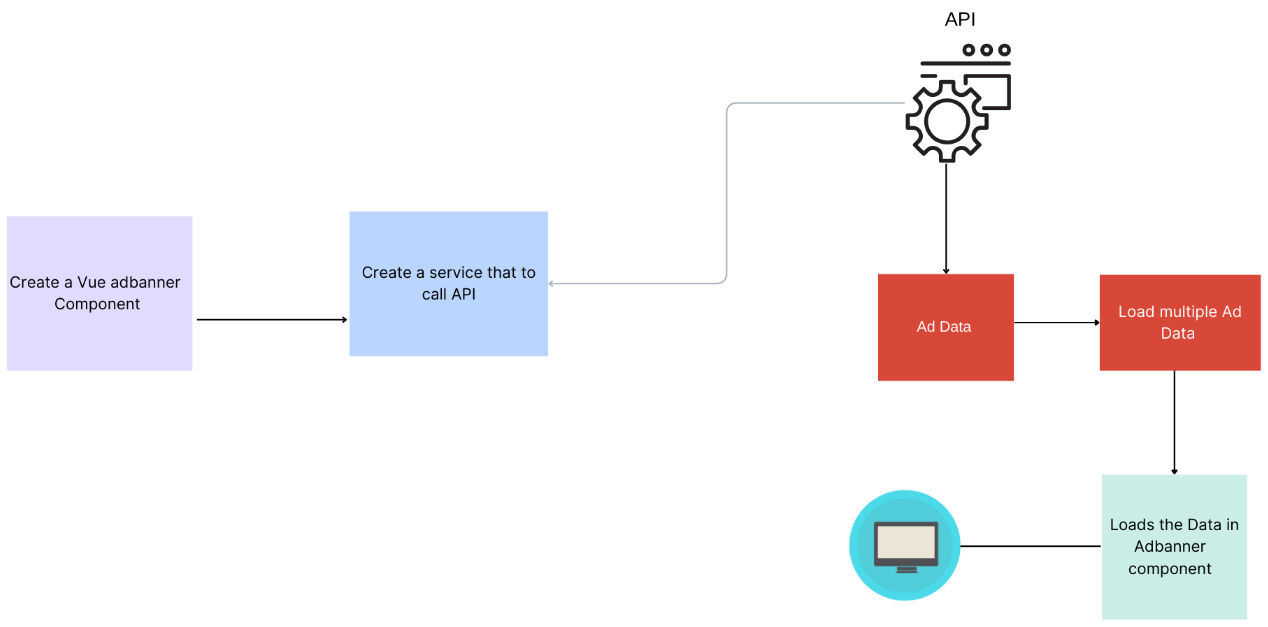
API
Create a Vue adbanner Component
Create a service that to call API
Ad Data
Load multiple Ad Data
Loads the Data in Adbanner component

Fig 2.1.1: Overall Workflow

2.1.2 SEQUENCE DIAGRAM

This sequence diagram describes the creation of the Ad Banner Component, which requests a specific kind of Ad data from the Ad Service. Ad Service then requests the Back-end API based on the ad-data requirement to the Ad Banner component and fetches the ad data to load in the ad banner component. Next, the Ad Banner Component dynamically creates an ad component for each ad in the ad data. Finally, the ad components are displayed on the website for the user.

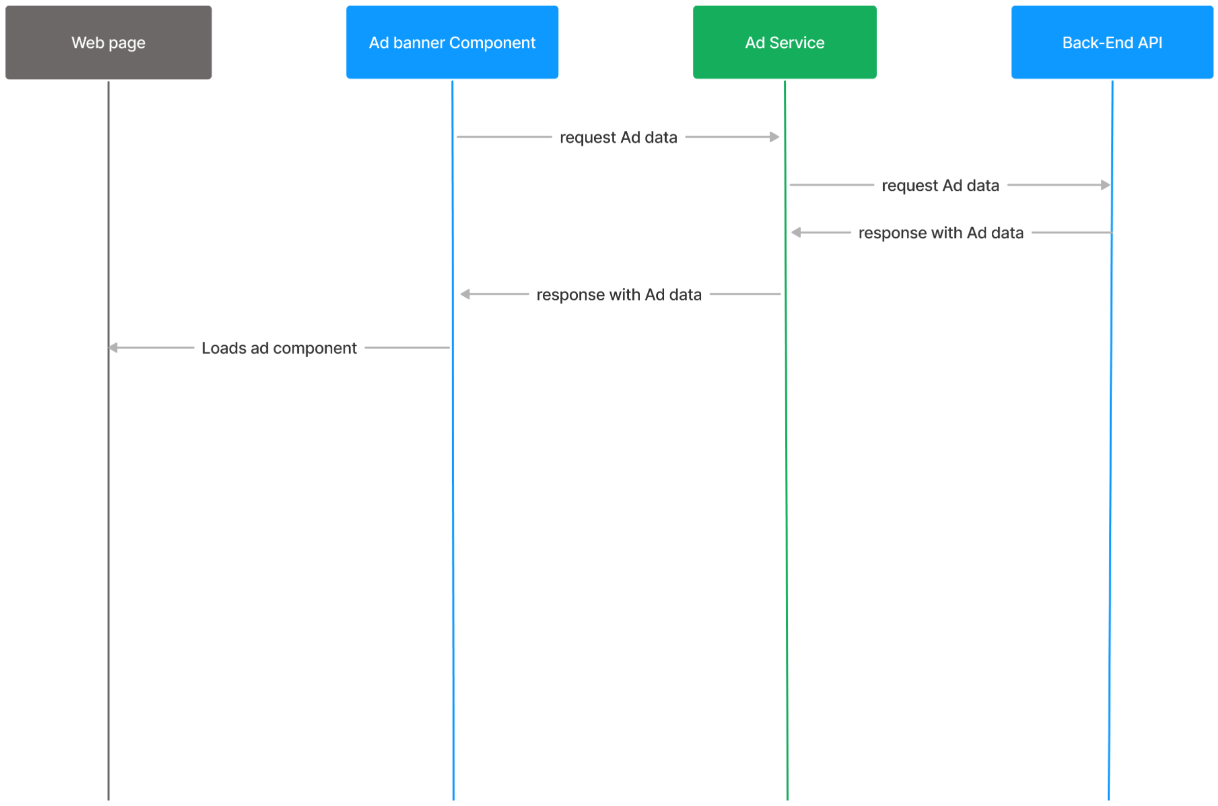

Fig 2.1.2: Sequence Diagram

2.1.3 LOW-LEVEL DESIGN:

This class diagram represents the attributes and operations/functions made by each class or component. The AdContent class is an entity with the title, imageSrc, linkUrl, getTitle(), getImageSrc(), and getLinkUrl() as the main method. Ad Service is the Service layer used to retrieve ad data from the backend API and send it to the Ad banner component using getAdLocation(), and getAdContent() methods and apiUrl attribute. The backend API class is used to provide Ad content to AD Service. This class contains the adLocations attribute and getAdLocations() method. AdBanner Directive class is used to specify the locations to display the Ad, this class contains adLocation and adContentLoaded attributes. Finally, The AdBannerComponent is a component that is responsible for loading the Ad on the web page, This class contains the adContent attribute and loadAd() method.

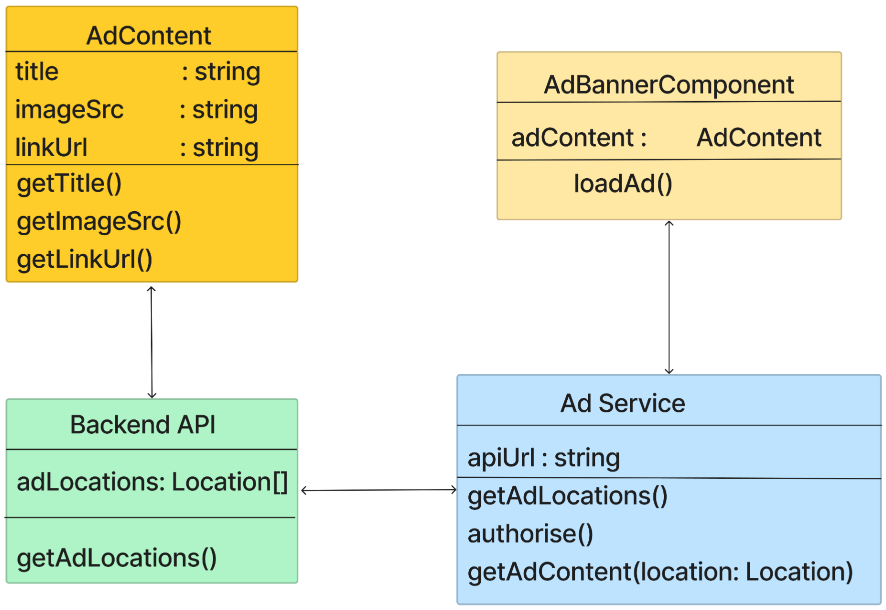

Fig 2.1.3: Low-Level Design

2.1.4 ACTIVITY DIAGRAM

The activity diagram shows how the activation process starts and ends in the web application. Activities of each module are mentioned.

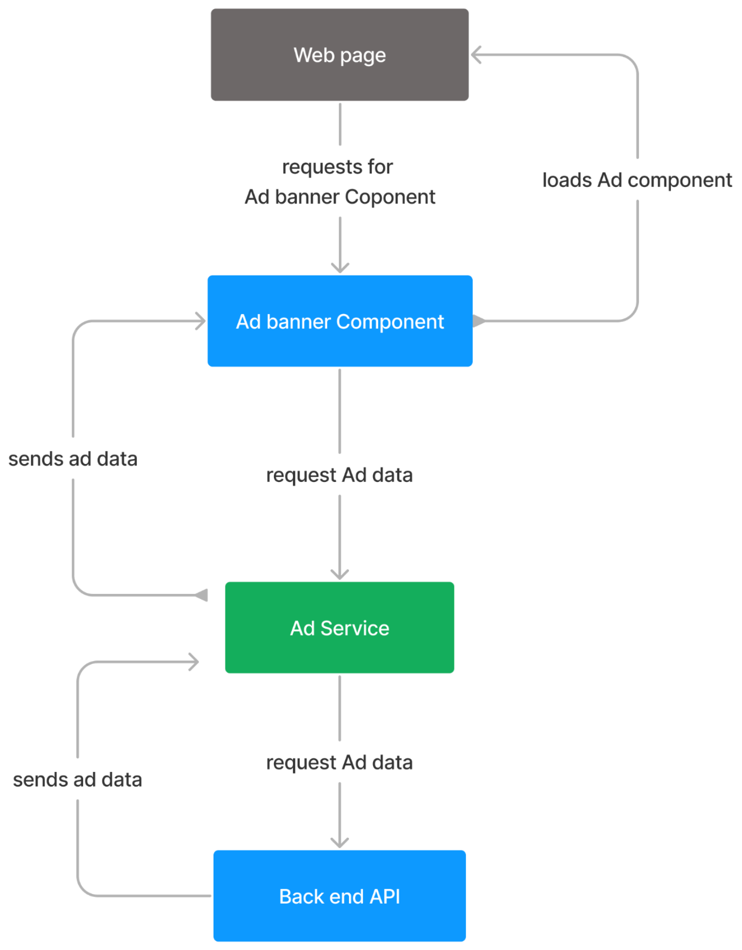

Fig 2.1.4: Activity Diagram

CHAPTER 3

TECHNOLOGY & FRAMEWORK

3.1 FRONT-END

3.1.1 Vue JS

Vue JS is a tool used for building user interfaces. It is designed to be approachable and easy to learn and can handle complex applications.

Vue.js uses a template-based syntax that makes it easy to write HTML-like code for components.

Purpose

The purpose of Vue.js is to provide an efficient way to build dynamic user interfaces for web applications.

Vue.js accomplishes this by using a component-based architecture, which allows developers to break down their UI into smaller, reusable pieces.

Prerequisite

AngularJS

Version 1.7.9

Angular offers a more modern and powerful approach to web application development and includes features such as better performance, improved dependency injection, enhanced templating, and much more.

Npm

Npm is a default package manager for javascript. Node Package Manager is installed along with node js. Npm manages the javascript runtime.

3.2 BACK-END

3.2.1 API

API stands for Application Programming Interface. APIs provide a way for applications to access and interact with each other's data and functionality, without requiring direct access to the underlying code
APIs are used in a wide range of applications, including web applications, mobile apps, and desktop applications.

Purpose

The purpose of APIs is to simplify the development process by providing a standard way for applications to communicate and share data. APIs enable developers to build complex applications more quickly and easily by leveraging the functionality of existing applications and services.

Prerequisite

HTTPS

APIs are accessed using HTTP (Hypertext Transfer Protocol), so it's important to have a basic understanding of HTTP methods such as GET, and POST.

RESTful Architecture

Many APIs are built using RESTful architecture, which involves defining resources and using HTTP methods to manipulate them.
Understanding the principles of RESTful architecture is important when designing and consuming APIs.

3.3 TOOLS USED

3.3.1 VISUAL STUDIO CODE

Visual Studio Code is a free and open-source code editor developed by Microsoft and available on Windows, Linux, and macOS. It has gained popularity among developers due to its wide range of features including support for various programming languages and syntaxes. The editor is designed to be lightweight and fast, making it suitable for both simple and complex coding tasks. It also includes an integrated terminal, which allows developers to run scripts and commands directly from the editor. Visual Studio Code has a large community of developers who have created various extensions and plugins to enhance its functionality. The versatility and ease of use make it a popular choice among developers across all industries and platforms. Debugging tools, Git integration, and the 's ability to extend its functionality via plugins.

CHAPTER 4

SOLUTION APPROACH

The solution approach for dynamically loading ad banner components in a website can be broken down into the following steps:

1. Create an main ad banner component using vue js to request recommended ads to the ad service component.
2. Create an ad service and use it to fetch the component data from a backend API.
3. The requested Ad data will be received by the ad service and then it will be sent to the ad banner component.
4. Finally, the ad is dynamically loaded on the requested web page.
5. We will create a sample test application and invoke the component for the test.
6. It undergoes unit and functional testing. Then detecting the issue and fixing the sonar, SAST, and DAST issues.

INTEGRATION

We can integrate the ad component into the web page by following certain steps as follows:

1. To incorporate the Vue.js library into your HTML file, include the CDN file of the vue page link in the head section of your HTML file.
2. To specify the component, use Vue component options to define it either in a separate file or within a <script> tag in the HTML file.
3. To generate a Vue instance, include the code in a separate JavaScript file or within a <script> tag located at the bottom of your HTML file.
4. To insert the component into the template, use its tag name within the div tag.