**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 3](#_Toc127885741)

[2.1.2 Low level Design 4](#_Toc127885742)

[3 Technology & Frameworks to be used 4](#_Toc127885743)

[4 Solution Approach 5](#_Toc127885744)

# Introduction

## About this document

The document outlines the purpose and scope of the AdDisplay component, which is responsible for displaying ads on a webpage. The component will have a simple design and allow users to dismiss ads they're not interested in.

The document also includes a component design diagram and an overall workflow sequence diagram. The AdDisplay component will be built using React JS and the document provides a solution approach that involves creating an AdDisplay component and an AdLoader component to asynchronously load ad data.

### Purpose & Scope of the document

The AdDisplay component is responsible for displaying advertisements to users. The component will be displayed in a designated space on the webpage and will show a variety of ads. The design of the banner will be simple with a clear call-to-action and minimal text. The banner will use a white background with bold, contrasting colors for the text and imagery to catch the user's attention.

The ad banner will include a button to allow the user to dismiss the ad if they are not interested in it. The AdDisplay component will be used to display ads to users as they browse the webpage. The component will be located in a designated space on the webpage and will not interfere with the user's browsing experience.

# Component Design

## Component Design Diagram

The AdDisplay component will be loaded when the user accesses the webpage. The component will request an ad from the ad server and receive a response with an ad to display. The ad will be displayed in the AdDisplay component, and the user can interact with the ad by clicking on it or dismissing it.

Data Layer

Application Layer

Presentation Layer

Ad Provider API

-Ad data

AdLoader Component

-Communicates with Ad Provider API to fetch data

React Component

### Overall Workflow

Start

Create AdDisplay Component in React

Setup AdProvider API to serve data

Setup AdLoader React Component .This component is in Charge of Loading ad

Use state variable to manage whether ad data has been loaded or not

Render the AdDisplay component depending upon on the state variable data

End

**Sequence diagram for AdDisplay:**

User React Component AdProvider API

Renders the App

Requests ad data

Responds with ad

Renders the AdDisplay component

### Low level Design

AdLoader

API Url

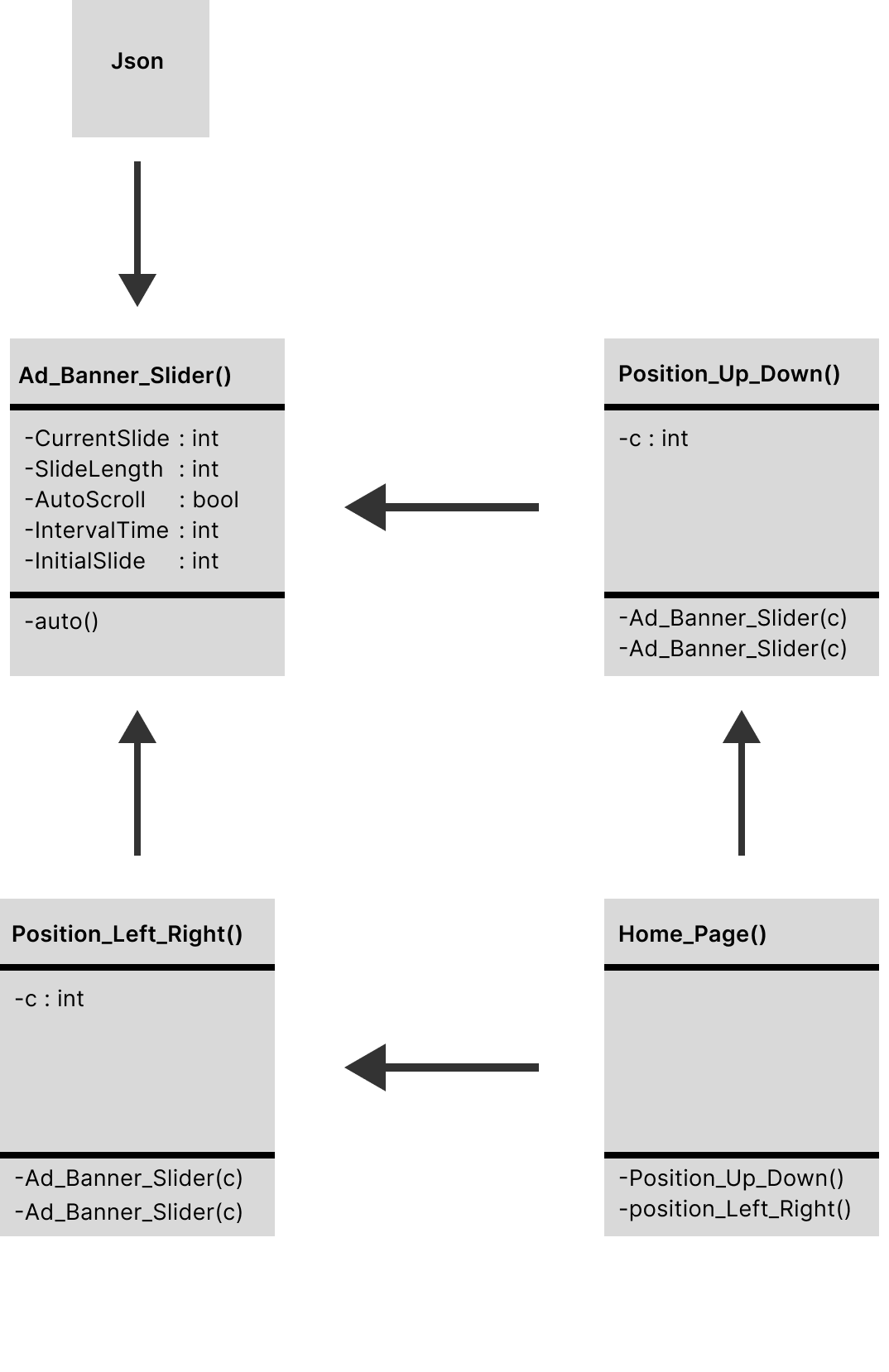
API Fetch

Main Component

Load the Ad data

Website

AdDisplay



The above is the class diagram from the implementation. The homepage is the front end of the whole process. To generate the advertisement at the top and bottom, it calls Position\_Up\_Down(), and to generate the advertisement at the left and right, it calls Position\_Left\_Right(). Both functions call the function Ad\_Banner\_Slider(), which is the slider to change the advertisements according to the initial index it is called, which is given as the parameter to both sliders, where the second slider gets the incremented initial index. The slider gets the advertisements by calling the index in the cyclic method. The slider gets the advertisements from the json file. The newly changed advertisements are updated on the webpage.

# Technology & Frameworks to be used

The Technology used is a React and the Framework used is javascript

# Solution Approach

1. Create an AdDisplay component in React. The ad banner component will be rendered by this component.

2. Create the AdLoader React component. This component is in charge of asynchronously loading the ad data and, after it has done so, rendering the AdDisplay component.

3. The adData state variable is set when the data is loaded using the useEffect hook, which retrieves the ad data from the ad provider URL. Fetched adData will be coverted to our container size. In order to catch any errors that arise during the fetch operation, we also use a try...catch block. The error state variable is then set with the error message.

4. Next, we handle three scenarios using conditional rendering:

If there is a mistake, we produce a message that details it.

We render a message stating that the ad is loading if the ad data has not yet loaded.

We render the AdDisplay component using the loaded data if the ad data has been loaded.

5. Use the URL of the ad provider's API to render the AdLoader component in your app or component.