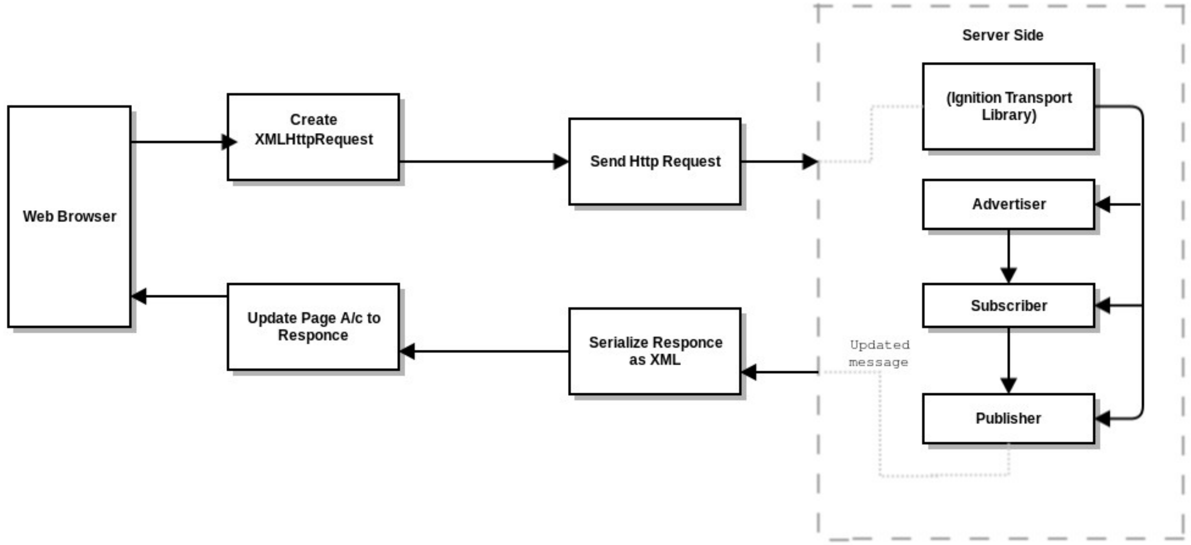
PROJECT AIM :

To create a web browser visualization tool for Ignition transport using the D3 JavaScript library.

Architecture details for AJAX communication .

Architecture for AJAX communication between the browser and the backend .



Explanation :

WEB BROWSER

Here we implement the design of view using D3js .In this part interaction of user with browser takes place .

So for now we have imagine that our backend is running in the system ,In view we have list of topic and list of service Requested presented in the nice way .

And also we are able to introspect one particular topic and after clicking on it we will be able to see the list of subscribers and some information about it .

XMLHttpRequest

Ajax interaction begins with a JavaScript object called XMLHttpRequest , this allows a client-side script to perform HTTP requests, and it will parse an XML server response. The first step in this Ajax implementation is to create an XMLHttpRequest instance. The HTTP method to use for the request (GET or POST) .

We create request using this :

variable = new XMLHttpRequest();

Send HttpRequest

We will simply send the data using this :

```
xhttp.open("GET", "Service.txt", true);
 xhttp.send();
```

When we send the HTTP request, we don't want the browser to hang around waiting for the server to respond. Instead, we want it to continue reacting to the user's interaction with the page and deal with the server's response when it eventually arrives. To accomplish this, we can register a callback function with the XMLHttpRequest and then dispatch the XMLHttpRequest asynchronously. Control then returns to the browser, but the callback function will be called when the server's response arrives.

SERVER SIDE

In server side we have ignition transport library .
On the Web server, the request arrives just like any other HttpServletRequest. After parsing the request parameters, the servlet invokes the necessary application logic (which is our ignition library) , serializes its response into XML, and writes it to the HttpServletResponse.
For example we will do this using this ,

```
document.getElementById("Topics") = xhttp.responseXML;
```

Back on the client side, the callback function registered on the XMLHttpRequest is now invoked to process the XML document returned by the server. Finally, the user interface is updated in response to the data from the server, using JavaScript without refreshing the page .